Г. Е. ЗАВОЛОДЬКО,
О. В. КАСІЛОВ,
С. В. БРОНІН

ОСНОВИ ВЕБРОЗРОБКИ

Навчально-методичний посібник
для студентів – бакалаврів та магістрів спеціальності
122 Комп'ютерні науки,
123 Комп'ютерна інженерія
денної та заочної форм навчання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Г. Е. Заволодько, О. В. Касілов, С. В. Бронін

ОСНОВИ ВЕБРОЗРОБКИ

Навчально-методичний посібник
для студентів – бакалаврів та магістрів спеціальності
122 «Комп'ютерні науки», 123 «Комп'ютерна інженерія»
денної та заочної форм навчання

Затверджено
редакційно-видавничою
радою НТУ «ХПІ»,
протокол № 2 від 26.06.2025 р.

Харків
НТУ «ХПІ»
2025

УДК 004.43
З-13

Рецензенти:
М. М. Степанов, д-р. техн. наук, с.н.с., проф. кафедри прикладної радіоелектроніки радіотехнічного факультету, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»;
О. Б. Ахієзер, канд. техн. наук, завідувачка кафедри комп'ютерної математики і аналізу даних, Харківський національний університет «Харківський політехнічний інститут»

Заволодько Г. Є.
З 13 Основи веброзробки : навчально-методичний посібник для студентів – бакалаврів та магістрів спеціальності 122 «Комп'ютерні науки», 123 «Комп'ютерна інженерія» денної та заочної форм навчання / Г. Є. Заволодько, О. В. Касілов, С. В. Бронін. – Харків : НТУ «ХПІ», 2025. – 322 с.

ISBN 978-617-05-0576-7

Навчальний посібник охоплює основи веброзробки, зокрема концепції Інтернету, вебсервісів, базових технологій створення сайтів, використання HTML, CSS, JavaScript та принципів роботи з вебхостингом. Він спрямований на формування ключових навичок вебвидавництва, проєктування, розробки, тестування та розміщення вебресурсів у мережі Інтернет. У навчальному посібнику розглядаються питання організації баз даних, основні ідеї та методи, які використовуються в сучасних системах управління базами даних.

Призначено для студентів – бакалаврів та магістрів спеціальності 123 «Комп'ютерна інженерія» денної та заочної форм навчання.

Іл. 138. Табл. 20. Бібліогр. 27.

УДК 004.43

ISBN 978-617-05-0576-7

ЗМІСТ

1. КОНЦЕПЦІЯ WEB

1.1. Основні терміни

Інтернет – це глобальна комп'ютерна система, яка: об’єднує мільйони комп’ютерів за принципом унікальних адрес. Кожен комп'ютер, якій під’єднано до Інтернету, має власну унікальну адресу. Вона здатна підтримувати обмін інформацією, тобто здійснювати комунікацію віддалених комп’ютерів, та забезпечує функціонування різноманітних інформаційних та комунікаційних служб.

Всесвітня павутина (www, Web, веб, World Web) є розподілену інформаційну систему, яка надає доступ до http гіпертекстових документів.

WWW – технологія застосування протоколу TCP/IP, що побудована на клієнтській архітектурі, яка використовує інфраструктуру Інтернет для спілкування між сервером і клієнтом.

WWW-сервери – це сховища гіпертекстової (взагалі) інформації, керованої спеціальним програмним забезпеченням.

Узагальнено в Інтернеті реалізовано два види послуг (рис. 1.1):

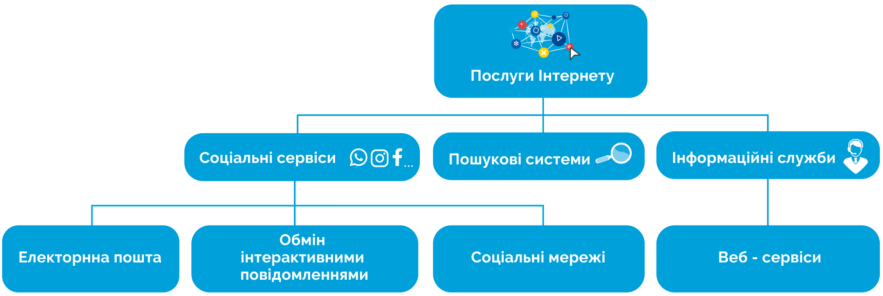


Рисунок 1.1. Послуги Інтернет

Інформаційні послуги – це послуги доступу до інформації, яку можна отримати на серверах, наприклад, сайти, документи, файли, інформацію з різних баз даних тощо. Якщо інформація розміщується на серверах для загального

користування, то будь-який користувач Інтернету може отримати необхідні дані.

Комунікаційні послуги – це послуги обміну інформацією та спілкування. Можливо обміюсатись інформацією у відтермінованому режимі. Так працює, наприклад, електронна пошта. Відправник надсилає лист до одержувача, який може його почитати у зручний час. Обмін інформацією у режимі реального часу дозволяє спілкування як між двома учасниками, так і в межах певної групи учасників. Наприклад, спілкування в мережі у текстовому, звуковому чи відео-форматі.

Послуги, які надаються користувачам Інтернету називаються **службами**, а програми, що реалізують служби – **сервісами**. Для того, щоб користувач міг користуватися певною службою, він має встановити на своєму комп'ютері програму-клієнт, що призначена саме для цієї служби. З боку Інтернету на відповідному до служби сервері має бути встановлена програма-сервер, яка спроможна обробляти запити від програми-клієнта користувача і надсилати йому результати обробки.

1.2. Основні об'єкти Інтернету (Клієнт/сервер)

Узагальнено комп'ютери у мережі можна поділити на комп'ютери-клієнти та комп'ютери-сервери (рис. 1.2).

Комп'ютер-клієнт потребує певної інформації і для її отримання надсилає повідомлення (запит) до комп'ютера-сервера, що містить дану інформацію.

Комп'ютер-сервер виконує певні дії відповідно до запиту та надсилає результат виконання назад до комп'ютера-клієнта.

Провайдери (Internet Service Provider, ISP) - компанії, надавачі будь-яких послуг.

Інтернет-провайдер – це компанія, що є основним технологічним і юридичним посередником між користувачами Інтернету та телекомунікаційним обладнанням, яке необхідне для забезпечення стабільного доступу по різних лініях зв'язку. За територіальною ознакою провайдери є міжнародними, національними і регіональними.

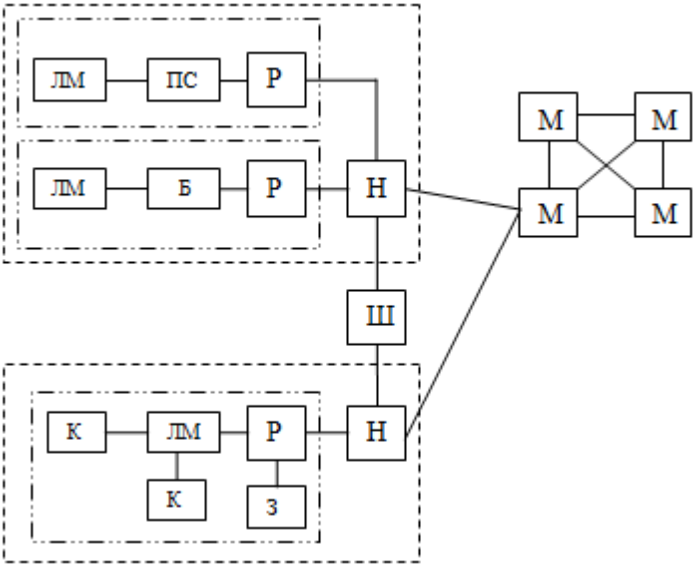


Рисунок 1.2. Загальна структура основних об’єктів в Інтернеті

М – міжнародний провайдер; ЛМ – локальна мережа; З – дзеркало; Н – національний провайдер; ПС – проксі-сервер; Б – брандмауер; Р – регіональний провайдер; Ш – шлюз; К – комп'ютер

Відповідно до наданих послуг розрізняють категорії:

Провайдер доступу – надає доступ до Інтернету по виділених, комутованих або безпровідних каналах зв'язку. Це оператор зв'язку, що має ліцензію на послуги зв'язку із наданням каналів зв'язку. Ліцензії провайдерам видаються Національною радою України з питань телебачення та радіомовлення.

Магістральні провайдери (первинні) мають у власності магістральні канали зв’язку. Вони забезпечують обмін трафіком із зарубіжними вузлами. Зазвичай, продають трафік лише у великих обсягах і надають послуги для інших провайдерів, а не для індивідуальних користувачів.

Канальні провайдери (вторинні) орендують канали зв’язку у первинних провайдерів і надають кінцеві послуги для індивідуальних користувачів. Вони

прокладають лінію зв'язку від магістралі безпосередньо до клієнта і займаються підтримкою мереж, де курсує основна частина інтернет-трафіку. Підставою для надання послуг є офіційний договір, який укладається між провайдером і абонентом.

Хостинг-провайдери – розміщують інформацію клієнта на своєму дисковому просторі і надають різноманітні послуги:

Віртуальний хостинг – надає дискового простору на сервері для зберігання і забезпечення роботи сайтів.

Хмарний хостинг – орендований сервер у віртуальному просторі, що створений з кластера фізичних серверів. Одночасне використання ресурсів кількох серверів, можливість регулювати ресурси за потребою і здійснювати оплату за послуги в залежності від навантаження на сервер.

Виділений хостинг – надає в оренду окремого сервера.

Колокація – розміщення серверів клієнта на площах провайдера.

1.3. Передача даних

Основною функцією WWW є взаємодія між вебсервером і браузерами на http (Протокол гіпертекстової передачі).

Вебсервер – це програма, яка працює на мережному комп'ютері та очікує запити клієнтів на протокол HTTP. Браузер може отримати доступ до вебсервера за доменним ім’ям або IP- адресою, передаючи необхідний ІДЕНТИФІКАТОР ресурсу в запиті. Отримавши запит від клієнта, сервер знаходить відповідний ресурс на локальному пристрої збереження даних і надсилає його як відповідь. Браузер приймає відповідь і обробляє його в залежності від типу ресурсу (відображає гіпертекст, показує зображення, зберігає отримані файли і т. д.).

Основними об’єктами Веб є ***вебсайти*** — це сукупності електронних документів (файлів), що об'єднані під однією доменною адресою і знаходяться на одному сервері за IP-адресою.

Спочатку вебсайти були лише збіркою статичних документів. На сьогодні вебдокументи окрім тексту та графіки містять анімацію, відео та звук. Але

основним у вебдокументах є гіперпосилання, які можуть бути як внутрішніми перехресними посиланнями, так і посиланнями на інші вебдокументи, що зберігаються на різних вебсерверах.

Сайти зараз в більшості є динамічними та інтерактивними. Це стало можливим завдяки втіленню вебдодатків — готових програмних комплексів для вирішення завдань вебсайту. Вебдодатки вбудовуються у склад сайту і працюють разом з ним.

У більшості випадків в Інтернеті один сайт має одну доменну адресу, за якою він ідентифікується. Можливими є інші варіанти: один сайт на кількох доменах або кілька сайтів під одним доменом.

Один і той же сайт може бути доступним за різними адресами і зберігатися на різних серверах. Копія оригінального сайту у такому разі називається дзеркалом. Зазвичай, великі сайти (вебпортали) використовують кілька доменів, щоб логічно відокремити різні види послуг (mail.google.com, news.google.com, maps.google.com). Іноді окремі домени виділяють для іншої мовної версії. Наприклад, *google.com.ua* і *google.pl* логічно є сайтами Google з різними мовами інтерфейсу, але технічно це різні сайти.

Миттєві (миттьові) повідомлення або повніше «система обміну миттєвими повідомленнями» (англ. Instant messenging, скорочено IM) — телекомунікаційна служба для обміну текстовими повідомленнями між комп'ютерами або іншими пристроями користувачів через комп'ютерні мережі (як правило через інтернет). Зазвичай, і від початку, це були невеликі текстові повідомлення. Але з розвитком у систему були додані й інші функції, такі як передавання файлів, зображень, звукових сигналів та повідомлень, відео, а також здійснення спільних дій, таких, як малювання або ігри.

Для користування цим видом комунікації необхідна клієнтська програма. Клієнтську програму системи миттєвих повідомлень часто називають інтернет-пейджером або месенджером.

Відмінність миттєвих повідомлень від, наприклад, електронної пошти полягає в тому, що обмін повідомленнями відбувається в реальному часі. При відправленні повідомлення електронною поштою, воно зберігається у поштовій скриньці на сервері. Для того, щоб отримати повідомлення,

отримувач повинен сам перевірити свою поштову скриньку і забрати його. В інтернет-пейджерах зв'язок між користувачами утримується постійно і відправлене повідомлення одразу передається користувачу.

1.4. WWW-сервіс

Служби та компоненти вебсервісу

Служба Веб є доступною в основному через Інтернет, тому користувачі часто ототожнюють поняття Веб і Інтернет. Веб можна віднести до внутрішнього наповнення, тобто, це віртуальний світ знань, тоді як Інтернет є фізичною стороною глобальної мережі у вигляді величезної кількості обладнання.

Служба Веб – світова інформаційна бібліотека, яка забезпечує засоби розміщення інформації і доступ до неї за допомогою кабелів і комп'ютерів (Інтернету). Веб є розподіленою інформаційною системою, оскільки інформація зберігається на величезній кількості вебсерверів.

Агентом користувача може бути не тільки браузер, а й бот, що видаляє вебсторінки, менеджер закачувань або інший додаток, що використовує Інтернет. Виконуючи запити до сервера, браузери, щоб була можливість їх ідентифікувати, постачають кожен запит так званим рядком призначеного для користувача-агента (UA-рядком), загорнутим в HTTP-заголовок User-Agent. Цей рядок ідентифікує браузер, повідомляє номер його версії та інформацію про операційну систему (рис. 1.3).

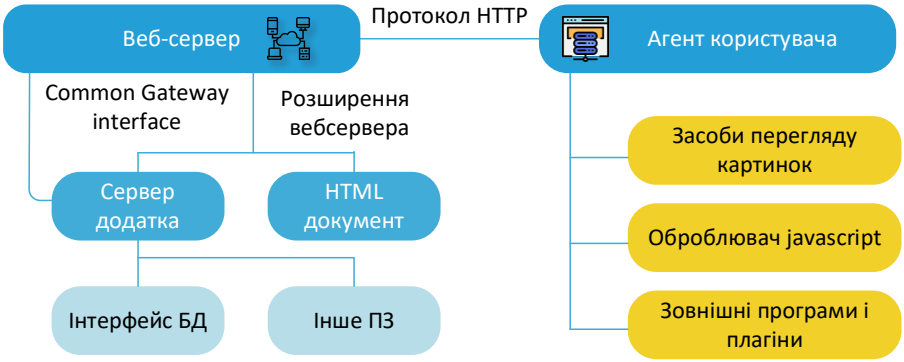


Рисунок 1.3. Взаємодія серверу та користувацького агенту

Спам-боти, менеджери закачувань і деякі браузери нерідко шлють підроблені UA-рядки, щоб видати себе за інших клієнтів. Ця ситуація відома під назвою «підміна» або «підробка призначеного для користувача агента» (user agent spoofing).

Функціонування сервісу забезпечується чотирма складовими:

URL / URI – уніфікований спосіб адресації і ідентифікації мережевих ресурсів;
HTML – мова гіпертекстової розмітки вебдокументів;
HTTP – протокол передачі гіпертексту;
CGI – загальний шлюзовий інтерфейс, який представляє доступ до серверних додатків.

Програмне забезпечення сервісу WWW

Вебсервер – це мережевий додаток, що обслуговує HTTP-запити від клієнтів, зазвичай веббраузерів. Вебсервер приймає запити і повертає відповіді, як правило, разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними. Вебсервери – основа Всесвітньої павутини. На стороні клієнта – додаток, який обслуговує HTTP-відповіді браузера. З розширенням спектра мережевих сервісів вебсервери все частіше використовуються в якості шлюзів для серверів-додатків або самі надають такі функції (наприклад, Apache Tomcat).

Браузер, вебоглядач (web-browser) – клієнтська програма для доступу до вебсерверів за протоколом HTTP і для перегляду вебсторінок. Як правило, браузери додатково підтримують і ряд інших протоколів (наприклад ftp, file, mms, pop3).

Перші HTTP-клієнти були консольними і працювали в текстовому режимі, дозволяючи читати гіпертекст і переміщатися по посиланнях. Зараз консольні браузери (такі, як lynx, w3m або links) практично не використовуються рядовими відвідувачами вебсайтів. Проте такі браузери вельми корисні для веброзробників, так як дозволяють «побачити» вебсторінку «очима» пошукового робота.

Роботи-«павуки». Поряд з браузерами, орієнтованими на користувача, існують і спеціалізовані клієнти-роботи («павуки», «боти»), що підключаються до вебсерверів і виконують різні завдання автоматичної обробки гіпертекстової інформації. Сюди відносяться, насамперед, роботи пошукових

систем, таких як google.com, yahoo.com і т. п., що виконують обхід вебсайтів для подальшої побудови пошукового індексу.

Популярні служби Інтернет

Існують універсальні і спеціалізовані служби. Спеціалізовані служби є доступними лише для вузького кола фахівців, а універсальною службою може скористатися будь-який користувач Інтернету. Загалом популярні служби Інтернет можна розділити на 8 типів.

1. Вебслужба

Універсальна служба Інтернет, що є базовою для багатьох інших служб. Програмою-клієнтом для служби Веб є браузер, який також є універсальним. Через службу Веб реалізовано:

- Вебсайти.
- Вебфоруми, блоги.
- Вікі-проєкти (Вікіпедія).
- Інтернет-магазини, інтернет-аукціони, інтернет-реклама.
- Соціальні мережі.
- Багатокористувацькі ігри.

2. Пошукові системи

Сучасні інформаційно-пошукові системи – це складний програмно-апаратний комплекс, механізми роботи якого є комерційною таємницею компанії-розробника. За допомогою спеціальних алгоритмів пошукові роботи збирають інформацію і заносять її в базу даних, де вона структурується і розташовується в певному порядку. Коли користувач вводить запит в рядок пошуку, автоматично формується звернення до бази даних. Після цього система видає найбільш відповідні до запиту документи. Інформаційно-пошукові системи постійно вдосконалюються, в механізми обробки та пошуку інформації втілюються технології штучного інтелекту, що побудовані на новітніх обчислювальних методах.

Користування пошуковими системами здійснюється через універсальний клієнт – браузер.

3. Служби комунікацій

Електронна пошта. Призначена для пересилання текстових повідомлень, до яких можна прикріплювати різноманітні файли. Програмою-клієнтом може бути як браузер, так і спеціалізовані поштові клієнти (Outlook, Mailbird, Thunderbird, The Bat!).

Служби миттєвих повідомлень, месенджери. Це клас програм, призначених для обміну повідомленнями через Інтернет у реальному часі. Передаватися можуть текстові повідомлення, голосові та відео- потоки, файли. Такі програми можуть застосовуватися для одночасного спілкування кількох осіб. Популярними месенджерами є Skype, Viber, WhatsApp, Telegram, які мають власні програми-клієнти і різні принципи організації спілкування.

Інтернет-чати. Призначені для спілкування багатьох учасників в реальному часі і, зазвичай, реалізовані в соціальних мережах чи відповідних сайтах. Програмою-клієнтом є браузер.

4. Служби новин та розсилок

Поштові та SMS-розсилки – набір повідомлень, які відправляються в потрібний момент. Обмежень у темах розсилки немає – бізнес-пропозиції, нові надходження, цикл статей тощо. Розсилки надсилаються, якщо клієнт підписався на розсилку або є зареєстрований в клієнтській базі.

Push-повідомлення – це короткі повідомлення, які вебресурс розсилає на комп'ютери і мобільні пристрої користувачів. Такі повідомлення повертають користувача на сайт, який він уже відвідував.

Технології RSS. Служба новин з обраних сайтів. RSS-потік – це простий і зручний спосіб отримати заголовки і анонси останніх новин та інших публікацій, не відвідуючи спеціально для цього сам сайт.

5. Файлові служби

FTP-служба пересилання файлів. Дозволяє емулювати на власному комп’ютері файлову структуру віддаленого комп’ютера і працювати з нею як з локальною директорією, наприклад, завантажувати файли з FTP-сервера на локальний комп’ютер і навпаки. Програмою-клієнтом можуть бути браузер, файловий менеджер чи спеціалізований FTP-клієнт.

Файлові хостинги. Файлообмінні мережі. Це сервіс, що надає користувачеві місце під файли і цілодобовий доступ до них через Інтернет. Файлові хостинги дозволяють зберігати і обмінюватися файлами. На спеціальній сторінці користувач завантажує файл на сервер файлообмінника, і натомість надає користувачеві посилання знаходження файлу, яке можна розсилати і публікувати. Перейшовши за таким посиланням, будь-який інший користувач може завантажити потрібний файл.

6. Служби мовлення

Інтернет-радіо. Служба, що дозволяє прослуховувати сотні радіостанцій, які віщають в Інтернеті.

Інтернет-телебачення. Служба, що надає змогу транслювати багато телевізійних каналів.

7. Служба віддаленого доступу

Забезпечує доступ клієнта до іншого віддаленого комп’ютера і надає можливість працювати на ньому як на власному. Віддалений доступ – функція, що дозволяє під'єднатися до віддаленого комп'ютера через Інтернет і працювати з його файлами і програмами. Програми-клієнти (TeamViewer, Віддалений робочий стіл Chrome) очікують дозволу від власника віддаленого комп’ютера і генерують пароль для доступу.

Віддалений доступ використовують системні адміністратори для управління системою і усунення поламок, а також фрілансери, компанії аутсорсингу, дистанційного навчання тощо.

8. Електронні платіжні системи

Це комплекс специфічних апаратних і програмних засобів, що дозволяє проводити електронні розрахунки. Існують різні способи і канали зв'язку для доступу, але поширеним є Інтернет з доступом через комп’ютер або мобільний телефон.

Служб в Інтернеті є багато і їх перелік постійно поповнюється. Для мобільних пристроїв створюються зручні програми-клієнти (application), які вільно чи платно пропонуються в магазинах даної платформи.

1.5. Доменна система імен DNS (Домен)

Система доменних імен

Функціонування Інтернету практично не можливе без визначення місця знаходження вебдодатку чи сервісу. У разі переглядання сайту в браузері, пересилання листа, розмови Скайпом, і в тисячі інших випадках, відбувається процес визначення місця розташування того чи іншого сервісу за допомогою розподіленої служби доменних імен **DNS** (Domain Name System – система доменних імен).

DNS сервери вирішують одне з головних завдань: для під'єднання до вебресурсу необхідно знати IP-адресу серверу, де фізично знаходяться файли сайту. Подібно до телефонного довідника, на DNS-сервері знаходиться таблиця записів відповідності, тільки замість організацій в ньому вказані вебадреси, а замість номерів телефону – IP-адреси.

Наприклад, якщо користувач хоче потрапити на головну сторінку Google, він вводить в адресний рядок браузера www.google.com, а DNS скеровує його запит до серверу з IP-адресою 74.125.19.147.

Коротко, шлях отримання адреси можна описати наступним чином (рис. 1.4):

2. Користувач вводить в адресний рядок браузера певну адресу або натискає на посилання. Сформований запит виходить з комп'ютера клієнта і перша IP-адреса, за якою він прямує, є адресою сервера провайдера, через якого клієнт має доступ до Інтернету.
3. На сервері провайдера знаходиться локальний DNS-сервер – програма, яка містить таблицю відповідності доменних адрес та IP-серверів популярних в даному регіоні вебресурсів.
4. Якщо такий запис в таблиці існує, то клієнтський запит отримує адресу, за якою потрібно прямувати далі.
5. Вебресурсів є мільйони, і таку величезну таблицю фізично неможливо тримати та обробляти на сервері провайдера. Тому, у випадку відсутності потрібного запису у таблиці, DNS-сервер провайдера звертається до спеціалізованих DNS-серверів вищих рівнів, щоб дізнатися про знаходження такого запису.

6. Спеціалізовані сервери також можуть звертатися вище, аж поки не буде знайдено інформацію. Така процедура є дещо тривалою, і користувач може побачити відповідь в браузері «Час очікування вичерпано, спробуйте ще раз». Варто оновити запит – і потрібний сайт буде завантажено.

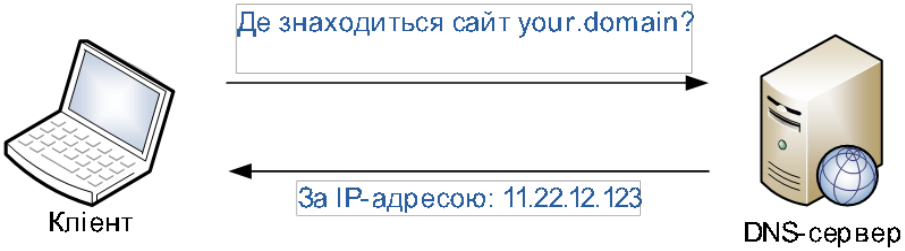


Рисунок 1.4. Звертання клієнта до локального DNS-серверу

Служба DNS – це розподілена ієрархічна структура серверів, які містять інформацію про всі доменні адреси Інтернет-ресурсів. Кожен DNS-сервер відповідає за свою зону або домен. При цьому є можливість передати (делегувати) частину DNS-імен до іншого сервера, забезпечуючи актуальність всіх даних в цілому.

Домен – складова в складному імені вебресурсу. Рівні домену рахуються справа наліво.

Кореневим доменом є «.» (крапка, яка ставиться в кінці DNS імені. Приклад: cad.lp.edu.ua.). Зазвичай, її пропускають при наборі імені, але можна її ставити, тоді це визначить ім'я, як повне FQDN (Fully Qualified Domain Name).

Домени першого рівня (приклад: ua, org, com, укр тощо) відносяться до тематичних або регіональних. Визначають країну, регіон або рід діяльності.

Домени другого рівня (приклад: mail, gmail, site) використовують для найменування сайтів в Інтернеті, їх продають реєстратори. Ціни можуть сильно різнитися, іноді в тисячі разів.

Домени третього рівня, створюються як субдомени ресурсу (forum.site.ua) або сайти географічної зони (приклад: victoria.lviv.ua). Винятком є найменування в Україні, що має свої особливості, наприклад, lp.edu.ua є іменем сайту.

Домени інших рівнів рідко коли продаються і реєструються. Зазвичай, створюються як окрема гілка сайту, наприклад, cad.lp.edu.ua і подібні імена.

Піддомен – підлеглий домен, субдомен. Наприклад: cad.lp.edu.ua. Для домену edu.ua піддоменом буде lp., для домену lp.edu.ua піддоменом буде cad.

Теоретично, кількість піддоменів може становити до 127, кожен з яких може містити до 63 символів. Але при цьому загальна довжина доменного імені не повинна перевищувати 254 символів.

Мінімальна довжина доменних імен залежить від зони і не може бути менше за 2 символи. Доменне ім'я не повинно суперечити загальноприйнятим моральним нормам. Доменне ім'я не повинно порушувати авторське право, товарні знаки та інші права інтелектуальної власності.

Зона – частина доменного імені разом з піддоменами, яка зберігається на одному сервері, а частіше – одночасно на кількох серверах. Метою виділення частини дерева в окрему зону є передача відповідальності за відповідний домен до іншої особи або організації, так зване делегування.

Делегування – передача відповідальності за частину дерева доменних імен до певної організації або особи. За рахунок делегування у службі DNS забезпечується розподіленість адміністрування та зберігання. Наприклад, за українську доменну зону .ua відповідає ТОВ «Хостмайстер». Технічно делегування виражається у виділенні цієї частини дерева в окрему зону і розміщенні цієї зони на DNS-сервері, що керується цією особою чи організацією.

Адресація вебресурсів (URI, URL, URN)

Для доступу до будь-яких мережевих ресурсів необхідно знати де вони розміщені і як до них можна звернутися.

URI: Означає ім'я та адресу ресурсу в мережі. Як правило, ділиться на URL і URN, тому URL і URN – це складові URI (рис. 1.5).

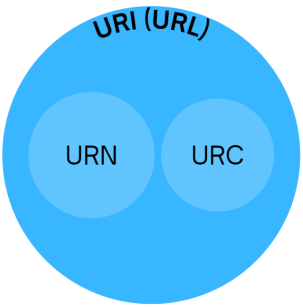


Рисунок 1.5. Схема URI

URL: Адреса деякого ресурсу в Інтернет. URL визначає місцезнаходження ресурсу і спосіб звернення до нього.

URN: Ім'я деякого ресурсу в Інтернет. Сенс URN в тому, що він визначає тільки назву конкретного предмета, який може знаходитися в безлічі конкретних місць.

У Всесвітній павутині для звернення до вебдокументів спочатку використовується стандартизована схема адресації і ідентифікації, що враховує досвід адресації і ідентифікації таких мережевих сервісів, як e-mail, telnet, ftp і т. п. – URL, Uniform Resource Locator.

<Схема>: // <логін>: <пароль> @ <хост>: <порт> / <повний-шлях-до-ресурсу>

Приклади:

URL = http://www.kpi.kharkov.ua
URN = /ukr/category/novini/index.html

Гіперпосилання та хостінг

URL – уніфікований локатор ресурсів, стандартизований спосіб запису адреси ресурсу в www і мережі Інтернет. Адреса URL має гнучку і розширювану структуру вказівки місцезнаходження ресурсів в мережі. Для запису адреси використовується обмежений набір символів ASCII. Загальний вигляд адреси можна уявити так:

<Схема>: // <логін>: <пароль> @ <хост>: <порт> / <повний-шлях-до-ресурсу>

де:

схема – схема звернення до ресурсу: http, ftp, gopher, mailto, news, telnet, file, man, info, whatis, ldap, wais і т. п.;

логін: Пароль;

ім'я користувача та його пароль, які використовуються для доступу до ресурсу

хост – доменне ім'я хоста або його IP-адреса;

порт – порт хоста для підключення;

повний-шлях-до-ресурсу – уточнююча інформація про місце знаходження ресурсу (залежить від протоколу).

Поняття «**хостинг**» (hosting, host) означає сервіс з надання обчислювальних потужностей (ресурси процесора, ОЗП, дискового простору тощо) та фізичного розміщення інформаційних ресурсів (сайтів) на серверах, що знаходяться на території провайдера і постійно під'єднані до магістралей Інтернету.

Послуга хостингу, як правило, складається з надання місця, що використовується для зберігання сайтів, а також забезпечення працездатності необхідних сервісів: електронної пошти, баз даних, зберігання файлів та послуги служби DNS, які можуть надаватися як окремі послуги або міститися в комплексі.

1.6. Використання первинних протоколів

Передача гіпертексту (протокол HTTP)

Інтернет об'єднує комп'ютери в багатьох точках земної кулі. Усі ці комп'ютери мають різне апаратне забезпечення, на них встановлені різні операційні системи та програмне забезпечення. Але всі вони мають узгоджено й швидко приймати і передавати дані. Для цього в 70-х роках минулого століття почали розроблятися правила, згідно з якими відбувався обмін даними між комп'ютерами в мережі. Збірки таких правил одержали назву «*протоколи*».

Протокол передавання даних — це правила, згідно з якими кодують і передають повідомлення й дані у мережі.

Процес передавання даних від одного комп'ютера до іншого складається з декількох етапів (рівнів). Цей процес передбачає такі операції: отримання даних від користувача, їх стиснення, шифрування, формування пакетів, на які розбиваються повідомлення, встановлення сеансу зв'язку між комп'ютером, що передає дані, та тим, що їх приймає, транспортування даних по каналах зв'язку, вибір найбільш ефективного маршруту передавання даних, формування вихідного документа з пакетів даних.

На кожному з етапів використовують окремі протоколи, сукупність яких складає набір протоколів Інтернету, що має таку назву TCP/IP, що має таке тлумачення.

TCP (англійською Transmission Control Protocol — протокол керування передаванням) *відповідає за організацію сеансу зв'язку між двома комп'ютерами у мережі.*

IP (англійською Internet Protocol — міжмережний протокол) *відповідає за маршрутизацію, тобто за те, щоб пакет було доставлено за певною адресою.* За допомогою протоколу TCP ПК перевіряє чи всі частини отримано. При отриманні всіх порцій TCP розміщує їх в потрібному порядку і збирає в одне ціле.

Найвідоміші протоколи, які використовують у мережі Інтернет

HTTP (Hyper Text Transfer Protocol) — протокол передачі гіпертексту. Використовують при пересиланні Web-сторінок з одного комп'ютера на інший.

FTP (File Transfer Protocol) — протокол передачі файлів зі спеціального файлового сервера на комп'ютер користувача. Дає можливість абоненту обмінюватися двійковими і текстовими файлами з будь-яким комп'ютером мережі.

POP (Post Office Protocol) — стандартний протокол поштового з'єднання. Сервери POP опрацьовують вхідну пошту, а протокол POP призначено для опрацювання запитів на отримання пошти від клієнтських поштових програм.

SMTP (Simple Mail Transfer Protocol) — протокол, який задає набір правил для передавання пошти. Сервер SMTP повертає або підтвердження про прийом, або повідомлення про помилку, або запитує додаткову інформацію.

Uucp (Unix to Unix Copy Protocol) – застарілий, але все ще використовуваний протокол передачі даних, у тому числі для електронної пошти. Передбачає використання пакетного способу передачі інформації, при якому спочатку встановлюється з'єднання клієнт-сервер і передається пакет даних, а потім автономно відбувається його опрацювання, перегляд або підготовка листів.

Telnet – протокол віддаленого доступу, що дає можливість працювати на будь-який ЕОМ мережі Інтернет як на своїй власній, тобто запускати програми, змінювати режим роботи тощо. На практиці можливості обмежено тим рівнем доступу, який задано адміністратором віддаленої машини.

DTN – протокол, призначений для забезпечення наддалекого космічного зв'язку.

Кожна людина має прізвище, ім'я, паспорт, ідентифікаційний код. Їх складають за певними правилами та вони є унікальними для кожної людини. Так само чинять із адресою ресурсів мережі Інтернет. Кожний ресурс Інтернету (апаратний, програмний чи інформаційний) має свою адресу. Щоб у мережі можна було обмінюватися даними, кожному з комп'ютерів надано унікальну адресу, яку називають IP-адресою.

IP-адреса – це адреса, що ідентифікує комп'ютер в Інтернеті. Її складено з чотирьох частин, розділених крапками (згідно з версією IPv4, яка діє сьогодні):

..***.***, де *** – число в діапазоні від 0 до 255. Ця адреса містить номер мережі та номер комп'ютера користувача.

Високі темпи розвитку Інтернету вже найближчим часом можуть призвести до нестачі адрес, надаваних протоколом IPv4. Для уникнення цього розроблено нову версію протоколу IP – IPv6, що надасть змогу використовувати близько 3,4 x 10^{38} адрес.

Кожен сайт в Інтернеті розміщено на комп'ютері-сервері, якому надано унікальну IP-адресу. Щоб звернутися до цього сервера, потрібно в поле адреси браузера ввести відповідну послідовність чотирьох чисел через крапки. Адресу у такому вигляді запам'ятати важко.

DNS (Domen Name System, англійською domain – область) – система доменних імен, у якій назву сервера записують як послідовність слів, розділених крапками.

Протокол HTTP

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту. Цей протокол спочатку був призначений для обміну гіпертекстовими документами, зараз його можливості істотно розширені.

HTTP – типовий клієнт-серверний протокол, обмін повідомленнями йде за схемою «запит-відповідь» у вигляді ASCII-команд (рис. 1.5). Особливістю протоколу HTTP є можливість вказати в запиті і відповіді спосіб надання одного і того ж ресурсу за різними параметрами: формату, кодуванні, мови і т. д. Саме завдяки можливості вказівки способу кодування повідомлення клієнт і сервер можуть обмінюватися кодованими даними, хоча даний протокол є символьно-орієнтованим.

HTTP – протокол прикладного рівня, але використовується також як «транспорт» для інших прикладних протоколів, насамперед, заснованих на мові XML (SOAP, XML-RPC, SiteMap, RSS і ін.).

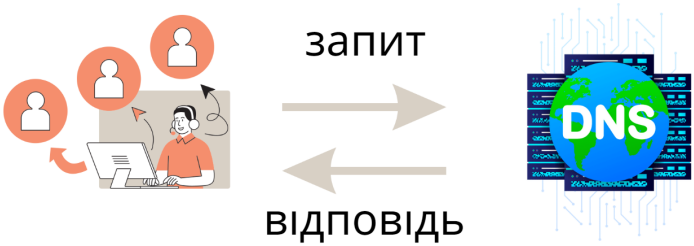


Рисунок 1.5. HTTP-запит та відповідь

Передача файлів (протокол FTP)

FTP (File Transfer Protocol – протокол передавання файлів). Сервери, що підтримують цей протокол, називаються FTP-серверами. Частина дискового простору таких серверів доступна через Інтернет. Крім того, до служб Інтернету

належать електронна пошта, служби миттєвого передавання повідомлень, служба новин User та інші.

При розгляді FTP як сервісу Інтернет мають на увазі не просто протокол, а саме сервіс – доступ до файлів, які знаходяться у файлових архівах.

FTP – стандартна програма, що працює за протоколом TCP, яка завжди поставляється із операційною системою. Її початкове призначення – передача файлів між різними комп'ютерами, які працюють у мережах TCP/IP: на одному з комп'ютерів працює програма-сервер, на іншому – програма-клієнт, запущена користувачем, яка з'єднується з сервером і передає або отримує файли через FTP-сервіс. Все це розглядається з припущенням, що користувач зареєстрований на сервері та використовує логін та пароль на цьому комп'ютері.

Ця риса послужила причиною того, що програми FTP стали частиною окремого сервісу Інтернету. Справа в тому, що доволі часто сервер FTP налаштовується таким чином, що з'єднатися з ним можна не тільки під своїм ім'ям, але й під умовним іменем anonymous – анонім. У такому випадку для користувача стає доступною не вся файлова система комп'ютера, а лише деякий набір файлів на сервері, які складають вміст серверу anonymous FTP – публічного файлового архіву. Отже, якщо користувач хоче надати у вільне користування файли з інформацією, програмами і т. і., то йому достатньо організувати на власному комп'ютері, підключеному в Інтернет, сервер anonymous FTP. Створення такого серверу – процес доволі простий, програми-клієнти FTP вельми розповсюджені, тому сьогодні публічні файлові архіви організовані в основному як сервери anonymous FTP. Перелік інформації, яка міститься на таких серверах, включає всі аспекти життя: від звичайних текстів до мультимедіа.

2. ВЕБВИДАВНИЦТВО

Розробка сайту починається з аналізу предметної галузі, обирання хостингу, та аналізу юридичних питань.

2.1. Створення та публікація вебсайту

Створення вебсайту починається з аналізу та плану розміщення вебсайту означає , що ваш ресурс (вебсайт) може бути доступний у всесвітній мережі (WWW). Зазвичай це робиться одним із двох способів. Ви можете заплатити за хостинг постачальнику послуг або самостійно розмістити його на власному сервері.

Хостинг постачальнику послуг

Файли вебсайтів, такі як HTML, зображення, відео, зберігаються на серверах, підключених до Інтернету. Коли користувачі побажають відвідати ваш вебсайт, вони введуть адресу вашого вебсайту у свій браузер, а потім їх комп'ютер підключиться до вашого сервера. Далі ваші вебсторінки будуть доставлені користувачам через Інтернет-браузер.

Використання постачальника послуг – найпростіший спосіб розміщення вебсайту. Ви можете сплатити невелику щомісячну плату і покластися на постачальника послуг, щоб подбати про все обладнання, інфраструктуру та інші пов'язані з цим потреби.

Плюси хостингу у постачальника послуг:

- зазвичай дешевше;
- підтримка доступна частіше;
- немає необхідності в технічному обслуговуванні;
- більш висока надійність.

Мінуси хостингу у постачальника послуг:

- можуть бути деякі сервісні обмеження;
- менше варіантів розміщення локацій.

Для обрання хостингу постачальнику послуг необхідно виконати 2 кроки.

Етапи вибору хостінгу

Існує два основних типи вебсайтів: статичний і динамічний.

Прості статичні вебсайти можна створити за допомогою програми «Що ви бачите, що ви отримуєте» (WYSIWYG), а потім перенести на обліковий запис хостингу.

Динамічні сайти в основному керуються додатками і використовують сценарії, бази даних та інші інструменти для генерації деяких частин сайту на ходу. WordPress та Joomla – приклади поширених програм CMS (Content Management System), які популярні сьогодні. Інші, такі як Magento і PrestaShop, використовуються для вебсайтів електронної комерції.

Порівняння типів вебхостингу

Хостинг вебсайтів також випускається із різними смаками. Наприклад, спільний хостинг – це *найдешевший і найпростіший в управлінні*.

Зі збільшенням кількості типів вебхостингу, зростають *пов'язані з цим труднощі в* управлінні обліковим записом. Наприклад, в *VPS-хостинг* вам потрібно буде керувати не лише деталями хостингу, але й середовищем, в якому він розміщений.

Тип вебхостингу, який вам знадобиться, зазвичай враховує обсяг трафіку, який ви очікуєте на своєму вебсайті, або будь-які інші конкретні потреби вашого вебсайту.

Більшість вебсайтів, які тільки починають працювати, як правило, мають низький об'єм трафіку (тобто мало відвідувачів), і спільні хостинг-акаунти будуть в нагоді для них. Спільні облікові записи також матимуть більше встановлених програм (наприклад, *Softaculous*), але щоб забезпечити задоволення своїх потреб, запитайте постачальника хостингу чи може бути встановлена в обліковому записі, який ви шукаєте, бажана програма.

Придбати план вебхостингу

Навіть у межах типів хостингу, у постачальників послуг часто доступні різноманітні плани. Ключова відмінність цих планів часто полягає у кількості

ресурсів, які кожен отримує. Чим більше ресурсів має ваш сайт, тим більше відвідувачів він може «опрацювати».

Що стосується ресурсів вебхостингу, ми зазвичай маємо на увазі три основні елементи – процесор (процесор), пам'ять (оперативна пам'ять) та накопичувач (HDD або SSD). Однак це не завжди означає хорошу роботу вебхоста.

У минулому не було простого способу оцінити ефективність роботи вебхостингу. Більшість користувачів доводиться розраховувати на огляди, які, на жаль, зазвичай роблять лише знімки продуктивності хоста та рідко їх оновлюють. Щоб розумітися на цьому, спробуйте скористатися *HostScore* – вебсайтом, який постійно оцінює ефективність роботи вебхостів на основі поточного збору даних. Це означає, що їх оцінки роботи вебхостів набагато точніші.

Також використовуйте додаткові переваги, такі, як безкоштовний SSL, доменне ім’я, рекламні кредити, які можуть допомогти вам створити або продати якісний сайт.

Деякі вебхости також пропонують інші переваги щодо більш дорогих планів, такі як спеціальні оптимізації чи вдосконалення.

Придбати доменне ім’я

Ваш вебхостинг – це фактичне місце, на якому знаходяться файли вашого вебсайту, і вам потрібно доменне ім’я, щоб користувачі могли отримати доступ до вашого сайту. Ім’я домену діє як ваша адреса на WWW. Як і справжні адреси, вебадреси *кожна унікальна*.

Багато планів вебхостингу сьогодні надають безкоштовне доменне ім’я, тому переконайтеся, що це стосується вебхостингу, який ви маєте намір придбати. Якщо це так, ви можете подбати про доменне ім’я одночасно, коли платите за свій вебхостинг.

Якщо ні, то вам потрібно буде *купувати доменне ім’я окремо*. Це можна зробити в тому самому місці, де ви придбали план хостингу, або у іншого постачальника послуг. Якщо вам потрібно придбати доменне ім’я окремо, радимо звернутися в інші місця.

Доменні імена не є фіксованими ціновими елементами і часто надходять у продаж. Деякі провайдери часто мають дешеві ціни на доменні імена, і якщо пощастить, ви можете отримати його задарма. Наприклад, часто при купівлі доменного імені пропонувалось до 98% знижки.

Виняток із цього – це випадок, коли ви стаєте власником сайту вперше. У такому випадку покупка доменного імені та хостингу в одного і того ж самого постачальника послуг може полегшити вам, початківцю, початок роботи.

Перемістити / створити свій вебсайт на сервері

Як тільки ваше доменне ім'я та план вебхостингу будуть готові, настає час міграції. Міграція сайту може бути складною, тому, якщо ви робите це вперше, зверніться по допомогу. Деякі постачальники послуг хостингу пропонують *безкоштовні міграції сайту*.

Якщо ви створили свій вебсайт локально (на своєму комп'ютері), просто перенесіть свої файли на свій вебсервер. Для цього ви можете або скористатися файловим менеджером на панелі керування вебхостингу, або здійснити передачу за допомогою FTP-клієнта.

Процес схожий на копіювання файлів з одного місця в інше на власному комп'ютері.

2.2. Юридичні питання

Аудиторії Інтернет

Різноманітність аудиторії Інтернет можна класифікувати за критерієм юридичної належності користувача (рис. 2.1).

Приватних користувачів Інтернет залежно від кількості годин, проведених в мережі Інтернет, ділять **на три підвиди**: активний користувач (surfer), звичайний користувач і початкуючий користувач.

Активний користувач Мережі: працює в Мережі не менше 100 годин в місяць. У зв'язку з цим, для нього дуже важливі гнучкі тарифні плани оплати послуг провайдера. Час, що проводиться таким користувачем в Мережі, розподіляється таким чином:

80 % — вихідні і нічний час в буденні дні (80 годин); 20 % — денний (20 годин).

Звичайний користувач Мережі: працює в Мережі не менше 50 годин в місяць. Час, що проводиться таким користувачем в Мережі, розподіляється таким чином:

60 % — вихідні і нічний час в буденні дні (30 годин); 40 % — денний (20 годин).

Рисунок 2.1. Класифікація аудиторії Інтернет

Початкуючий користувач Мережі: працює в Мережі не менше 10 годин в місяць. Це, як правило, час, що надається провайдером безкоштовно. Час, що проводиться таким користувачем в Мережі, розподіляється таким чином: 50 % — вихідні і нічний час в буденні дні (5 годин); 50 % — денний (5 годин).

У свою чергу, серед **корпоративних користувачів** залежно від обсягу циркуліруючої (отримуваною і передаваною) інформації в місяць (трафіку) можна виділити наступні групи.

Організації, що здійснюють регулярний доступ до онлайнових служб (бази даних, системи замовлення і тому подібне). Як правило, обсяг трафіку не перевищує 10 Гбайт в місяць. Для таких обсягів інформації сповна достатня виділена лінія до 1024 Кбіт/сек.

Організації, що активно використовують в своїй роботі ресурси Інтернет. Обсяг трафіку в середньому складає від 10 до 150 Гбайт в місяць. Для такого завантаження потрібна виділена лінія від 2048 до 6144 Кбіт/сек.

Організації, що надають послуги в режимі реального часу (наприклад, результати торгів на біржах). Обсяг трафіку для таких організацій в середньому складає понад 150 Гбайт в місяць. Для такого завантаження потрібна виділена лінія більше 6144 Кбіт/сек.

Правовий статус доменних імен в Україні

На даний час в Україні правовий статус доменних імен законодавчо не визначений. Закон України "Про охорону прав на знаки для товарів і послуг" містить визначення поняття доменного імені, як імені, що використовується для адресації комп'ютерів і ресурсів в Інтернеті. Доменні імена в Україні реєструються на підставі договору між реєстрантом та реєстратором, в якому зазначено права та обов'язки сторін такого договору.

Реєстрантами можуть бути:

1. Громадянин України, громадянин іноземної держави та особа без громадянства з необмеженою цивільною правоздатністю і дієздатністю.
2. Українська або іноземна юридична особа.

Домени в Україні за їх призначенням поділяються на два види:

Публічні, тобто такі, що адмініструються в інтересах певної спільноти, наприклад, домен .UA є публічним доменом.

Приватні, тобто такі, що адмініструються певною особою у своїх власних інтересах.

Доменне ім'я в Україні може бути:

- Продано.
- Здано в оренду.
- Передано у спадок.

Галузеві домени в зоні .ua

Галузеві домени призначені для реєстрації доменних імен третього рівня з врахуванням галузевої, відомчої та іншої специфіки.

UA – для зареєстрованих торгових марок.
com.ua – для комерційних організацій.
gov.ua – для урядових організацій.
net.ua – для постачальників мережних послуг.
edu.ua – для навчальних закладів, що мають ліцензію на освітню діяльність.
int.ua – для міжнародних організацій.
org.ua – для некомерційних організацій.
in.ua – для індивідуальних користувачів.
pp.ua – безкоштовний домен для приватних осіб

Кириличні доменні імена в українських доменах

19 жовтня 2010 р. – початок реєстрації IDN – кириличних доменних імен у доменних зонах .COM.UA та .KIEV.UA. Користувачі можуть реєструвати и використовувати в цих доменних зонах символи українського та російського алфавіту. Такі доменні імена мають вигляд: фірма.com.ua, торговамарка.kiev.ua.

Термін «авторське право» та його наслідки

Вебсайтом є представлена в об'єктивній формі сукупність результатів інтелектуальної діяльності, систематизованих в єдиний об'єкт таким чином, щоб він міг бути розміщений в мережі Інтернет. Відповідно до даного визначення вебсайт визначається такими ознаками:

1. Наявність результатів інтелектуальної діяльності.
2. Об'єктивна форма вираження результатів інтелектуальної діяльності.
3. Передбачає систематизованості результатів інтелектуальної діяльності.
4. Єдиний об'єкт.
5. Можливість розміщення результатів інтелектуальної діяльності в мережі Інтернет.

У вузькому розумінні, як цифровий твір, і в широкому, як майновий комплекс, включаються і матеріальні носії інформації (сервер та ін.), що виключає можливість розгляду вебсайту як комплексу виключних прав, тобто як об'єкта, регульованого тільки нормами законодавства в області інтелектуальної власності.

Переходячи безпосередньо до правової охорони вебсайтів, варто зазначити, що є ряд науковців, які розглядають їх як непоіменований об’єкт авторського права, проте є й інші, які притримуються позиції, що вебсайти – складний об’єкт, багато частин якого можна охороняти за допомогою різних видів прав інтелектуальної власності, а саме:

- програмне забезпечення, включаючи заснований на тексті код HTML, використовуваний на вебсайтах, може охоронятися авторським правом або патентним правом залежно від національного законодавства;
- дизайн вебсайту, по всій очевидності, належить охороняти авторським правом;
- системи електронної торгівлі, механізми пошуку або інші технічні інструменти Інтернету можуть охоронятися за допомогою патентів/корисних моделей;
- творчий зміст вебсайту, а саме письмовий матеріал, фотографії, графіка, музика і відео, може охоронятися авторським правом;
- бази даних можуть охоронятися також авторським правом;
- ділові назви, логотипи, назви продукції, назви доменів та інші позначення, поміщені на вашому вебсайті, можуть охоронятися в якості товарних знаків;
- зроблені за допомогою комп'ютера графічні символи, екранні дисплеї, графічні інтерфейси (GUIc) і навіть вебсторінки можуть охоронятися за допомогою законодавства про промислові зразки;
- приховані аспекти вебсайту (такі як конфіденційна графіка, код джерела, код об'єкта, алгоритми, програми та інші технічні описи, діаграми даних, логічні діаграми, інструкції користувача, структури даних і зміст баз даних) можуть охоронятися за допомогою законодавства про комерційну таємницю до тих пір, поки вони не розкриті.

Ще одним тип об'єктів авторського права на сторінці вебсайту є гіперпосилання. Гіпертекстовий документ – текстовий документ, що містить посилання на інші документи у вигляді адреси їх розташування в мережі. На відміну від звичайного документа, що має посилання, наприклад, на використовувану літературу, адресу, вказану в гіпертекстовому документі, дозволяє автоматично завантажити в пам'ять комп'ютера той документ, на який здійснено посилання. Сукупність гіпертекстових документів, розміщених, як правило, за однією адресою в мережі, складають мережевий інформаційний ресурс (англійський варіант – сайт, від англ. site – вузол, місце).

Таким чином, вебсайт може цілком відповідати всім нормам закону, але включати в себе гіперпосилання на інші сайти, де порушуються авторські права. Законодавство не дає прямих вказівок на те, чи є власник такого сайту порушником прав інтелектуальної власності.

Одним з можливих рішень проблеми правової охорони інформаційних ресурсів на вебсайтах є не лише посилення відповідальності власників серверів за розміщену інформацію, а і збільшення контролю з боку як державних правоохоронних органів, так і громадських організацій задля виявлення та недопущення подальших порушень прав інтелектуальної власності у мережі Інтернет.

Основні закони країни, яким підпорядковується, вміст вебсайту

Сьогодні інформаційні технології стрімко змінюють суспільні відносини, однак українське законодавство досі не містить визначення терміна «вебсайт». Серед науковців немає єдиної думки щодо цього, але його якнайшвидше законодавче закріплення спростить вирішення правових питань, пов'язаних з Інтернетом, зокрема у судовій практиці. Вебсайт є складним об’єктом, що може охоронятися різними видами прав інтелектуальної власності. Основний нормативний акт у цій сфері — Закон України «Про авторське право і суміжні права».

3. ЦИКЛ РОЗРОБКИ ВЕБСАЙТУ

3.1. Основні етапи розробки вебсайту

Створення сайту – це трудомісткий і тривалий процес, який відбувається в кілька етапів, у міру проходження яких ідеї замовника перетворюються в реальний функціонуючий сайт.

Розробку сайту можна порівняти з будівництвом будинку, де мають послідовно бути виконані визначені етапи: від проєктної документації і закладки фундаменту до внутрішньої і зовнішньої обробки приміщення.

Етапи розробки проєкту, як правило, виконуються послідовно, тому, вкрай важливо дотримуватися черговості етапів і розуміти, що будь-які несподівані і неузгоджені заздалегідь зміни чи правки можуть значно вплинути на ефективність роботи.

Роботу над кожним проєктом слід проводити в строгій відповідності з приведеними нижче етапами робіт розробки сайту.

1-й етап. Визначення цілей створення сайту та проведення досліджень за темою

Це найважливіший етап у створенні сайту, оскільки не можна досягти мети, якщо її немає або неправильно визначено. Від цілей буде залежати весь подальший процес створення сайту, кожен його етап. Правильно поставлена мета – це вже половина успіху.

Перш, ніж приступити до розробки, необхідно проаналізувати тему, вивчити сайти потенційних конкурентів. Надалі це допоможе у створенні власної концепції.

2-й етап. Складання технічного завдання

Розробка сайту є доволі складним процесом, який вимагає детальної інформації про майбутній проєкт. Для досягнення повного взаєморозуміння із замовником необхідно правильно складене технічне завдання, в якому мають бути відображені всі поставлені завдання. У процесі створення такого проєкту

проводиться поетапне узгодження виконуваних робіт, що дозволяє зробити сайт, який відповідає всім вимогам клієнта.

На цьому етапі розробник спільно із замовником визначаються з умовами щодо реалізації проєкту.

Складання технічного завдання (ТЗ) з попередніми вимогами до сайту.

1. Визначення рамок бюджету, оцінювання фінансових можливостей клієнта.
2. Формування команди розробників.
3. Узгодження часу на реалізацію проєкту.

ТЗ має містити такі пункти (як мінімум):

1. Тип сайту (лендинг, візитка, каталог, корпоративний, магазин тощо).
2. Структура сайту (які сторінки повинні бути на сайті).
3. Функціонал сайту (пошук, каталог, стрічка новин).
4. Стиль дизайну (витриманий, молодіжний, в темних тонах), кольорова гама, потреба у створенні нового бренду (логотип, слогани, фірмові кольори та шрифти).

3-й етап. Технічні аспекти проєктування сайту

На цьому етапі визначається архітектура сайту, файлова та логічна структура сторінок.

Архітектура сайту є основною організацією клієнтської та серверної частин сайту, їх взаємозв'язку та зв'язку з кінцевим користувачем. Створення архітектури сайту є першочерговим етапом в його проєктуванні.

Файлова структура – це чітко оформлена система організації різних файлів. Продумана і зручна файлова структура допомагає розробнику оптимізувати роботу, а також буде зрозумілою для інших фахівців, що працюють над проєктом.

Логічна структура сайту – це внутрішній устрій сайту, його «кістяк», розташування сторінок, розділів, підрозділів, додаткових матеріалів. І першочерговим завданням розробників є створення стрункого порядку з хаотичного скупчення інформації.

4-й етап. Розробка макета дизайну сайту

Цей етап поділяється на кілька підетапів:

1. Генерація ідей дизайну. Набір ідей надається замовнику у вигляді ескізів з текстовими поясненнями.
2. Розробка попереднього макета дизайну головної сторінки.
3. Виправлення зауважень замовника, доробка макета до завершеного вигляду.
4. Розробка внутрішніх сторінок за аналогічним алгоритмом.

У макеті повинні бути промальовані всі блоки, які будуть на сайті. Якщо до моменту розробки дизайну текстові та графічні матеріали ще не готові, можна використовувати довільний текст, але не можна залишати в макеті порожні місця.

5-й етап. HTML-CSS верстка

Залежно від цілей і завдань сайту, верстка повинна задовольняти деяким вимогам. Зазвичай, ці вимоги такі:

1. Кросбраузерність – сторінки повинні однаково відображатися в популярних браузерах (Mozilla Firefox, Google Chrome, Opera, Internet Explorer, Safari і т. д.).
2. Адаптивність під різні розміри екрану та мобільні пристрої.
3. Гнучкість верстки – можливість легко додавати/видаляти блоки інформації на сторінки.
4. Швидкість обробки коду браузером.
5. Валідність – відповідність до різних стандартів.
6. Семантична коректність – логічне і правильне використання елементів HTML.

6-й етап. Програмування і встановлення на систему керування контентом (CMS)

Це чисто технічний етап, на якому реалізується весь функціонал сайту. Вимоги до цього етапу визначаються у технічному завданні.

7-й етап. Заповнення сайту контентом (інформацією)

На цьому етапі відбувається наповнення сайту якісним, професійним контентом. Всі матеріали сайту, будь то тексти чи графіка, повинні вписуватися в загальну концепцію сайту, відповідати його цілям і задачам.

8-й етап. Тестування сайту і виправлення помилок

Тестування працездатності сайту може проводити як розробник, так і замовник. Найкращий варіант – це спільне тестування.

9-й етап. Публікація сайту в Інтернеті

На цьому етапі для сайту обирається доменна адреса, хостингова площадка, розгортання адміністративної частини або системи управління контентом, фізичне перенесення інформації. Здійснюються тестові перевірки в реальних умовах.

10-й етап. Просування сайту і реклама в Інтернеті

Після остаточного завершення робіт над сайтом потрібно залучати на нього відвідувачів. Для просування та популяризації сайту можна скористатися контекстною або банерною рекламою, пошуковою оптимізацією, просуванням в соціальних мережах та іншими методами.

3.2. Бриф на розробку сайту

До запуску розробки та дизайну сайту вкрай важливо, щоб ваші вимоги були викладені на папері. Так, спілкуючись в майбутньому з розробниками і дизайнерами, ви зможете ясніше пояснити, що саме вам потрібно. Не виключено, що ваші уявлення про бажаний формат, оформлення та наповнення вебресурсу можуть помінятися кілька разів в ході такого обговорення.

На допомогу замовникам (і собі, в тому числі) розробники завжди пропонують заповнити документ у формі опитувальника – бриф на сайт. «Бриф» передбачає, що інформація в опитувальнику викладається максимально коротко і чітко, без узагальнень. Якщо ви замовник – обов'язково заповніть бриф, щоб зафіксувати свої основні побажання. Якщо ви виконавець, і досі не працюєте з такою корисною штукою, як бриф сайту, – саме час почати.

Бриф на сайт є документ у письмовій формі, в якому міститься вся необхідна інформація про проєкт. Будь-який проєкт, як правило, починається зі складання короткого плану дій, щоб сформулювати основні цілі роботи і змалювати бажаний кінцевий результат. Отже, до того як ви почнете розробляти вебресурс, у вас під рукою повинна бути стратегія проєкту, в ємній і чіткій формі викладена у вигляді брифа.

У брифі фіксуються керівні принципи розробки сайту: відомості про ресурс і замовників, цілі, вимоги і технічні вказівки. А ще такий опитувальник об'єднує замовника і всіх виконавців, зайнятих у розробці сайту, в єдину команду, чітко представляє, до якого завершення повинен прийти проєкт відповідно до побажань замовника. Коротко кажучи, у брифі повинні бути відображені відповіді на головні питання існування сайту – хто, що, де, коли і навіщо.

Форма брифа може змінюватися залежно від того, чи потрібна замовнику повна розробка сайту (разом з дизайном) або, скажімо, тільки дизайн. Питання, включені у бриф на розробку і бриф на дизайн сайту, як правило, визначає виконавець проєкту залежно від завдання. Немає єдиної, загальнообов'язкової форми опитувальника, але є кілька ключових аспектів, яким просто необхідно приділити увагу.

Загальний бриф на створення сайту допомагає визначити цілі і завдання проєкту.

Пам'ятайте, що дизайн сайту визначають не тільки стиль і розташування елементів на сторінці, а й призначення вебресурсу, його мету й орієнтованість на споживача. Бриф на дизайн сайту повинен бути настільки детальним, наскільки дозволяє коротка по природі форма опитувальника. Заповнений бриф – це, по суті, гарантія того, що кінцевий продукт буде відповідати очікуванням замовника. Чим докладніше його відповіді, тим вище ймовірність того, що він отримає саме те, що йому потрібно.

Що станеться, якщо приступати до дизайну без попереднього заповнення брифу? Монополія на оформлення сайту повністю перейде до рук дизайнерів. А ймовірність того, що бачення замовника і дизайнера співпаде, катастрофічно мала. Замовник ризикує отримати готовий сайт, який зовсім не відповідає його очікуванням. Проєкт доведеться переробляти, а це може вилитися у відчутні витрати і нескінченну низку виправлень помилок.

Ось як це зазвичай відбувається: ви вирішуєте замовити розробку сайту і забуваєте (або не вважаєте за потрібне) згадати, що на сайті обов'язково повинна бути, наприклад, кнопка реєстрації особистого кабінету. Вам здається, що це очевидно – адже на всіх сайтах продавців є така кнопка, отже, дизайнер сам вмонтує її в ваш майбутній сайт. Насправді, це не так. Не існує єдиного набору елементів, які вбудовуються на сторінку «за замовчуванням». Завжди необхідно вказувати всі свої побажання, щоб потім не переплачувати за виправлення, не відкладати запуск проєкту і, в цілому, уникнути подібного негативного досвіду.

Бриф, в якому замовник описує дизайн майбутнього сайту, повинен містити наступні блоки, як описано нижче.

Цільова аудиторія

Яким ви уявляєте собі ідеального клієнта? Можливо, ви вже вивчили кон'юнктуру ринку і знаєте що потрібно споживачеві. Вкажіть конкретні характеристики цільової аудиторії і товари / послуги, які їй цікаві. Якщо цільова аудиторія розпадається на кілька груп, варто окремо відзначити кожну з них, конкретизуючи демографічні характеристики споживачів і ваші особисті очікування від їх обслуговування.

Не зайвим буде зазначити і своїх прямих конкурентів, їх методи ведення бізнесу і розподіл цільової аудиторії у сфері, в якій зайняті ви і ваші конкуренти. Виконавець зможе проаналізувати дизайнерські рішення у розробці сторінок ваших конкурентів і запропонує вам унікальні рішення для дизайну вашого сайту.

Структура сайту

Будь-сайт складається зі стандартного набору елементів сторінки – «шапка» (хедер), меню, бокова частина (сайдбар) і підвал (футер). Побажання в оформленні кожної частини структури сайту слід прописати окремим рядком. При виборі оформлення замовник також може керуватися прикладами вподобаних сайтів. Можна окремо виділити варіанти оформлення, які ви точно не хочете бачити на своєму вебресурсі. В рамках брифу така категоричність буде вам тільки у нагоді, а виконавцю буде легше орієнтуватися у загальній

масі варіантів. Часом, діючи від протилежного, вдається знайти унікальне дизайнерське рішення.

У брифі можна і потрібно відобразити специфіку наповнення сайту. Можливо, вам буде потрібно включити у вебсторінку новинний розділ, блог, інтерактивну карту, афішу заходів, форум, кнопку онлайн-бронювання? Ці та ще маса елементів визначаються виходячи з конкретної сфери діяльності і перспектив розвитку компанії.

Загальні побажання по стилю

В даному випадку, «загальне» досить швидко, з професійної подачі виконавця, переходить в «приватне». Грамотно складений бриф виносить на обговорення ключові елементи оформлення: кольорову гамму, графіку, форму блоків. Із загальних уявлень замовника виконавець формулює конкретні «так» або «ні» у дизайні, у рамках яких він повинен діяти. Як вже зазначалося вище, якщо у компанії є продуманий стиль оформлення, логотип і колірна гамма, дизайнер буде відштовхуватися від них. Продукцію одних компаній ідеально продемонструє плаский дизайн, для інших потрібно щось креативне і мудре. Якщо замовнику хочеться бачити плавні, обтічні форми кнопок і іконок, важливо уточнити це на етапі заповнення брифу – інакше на переробку піде багато часу, грошей і нервів замовника.

3.3. Приклад анкети-брифа на дизайн сайту

Щоб краще визначити цілі майбутнього сайту, потрібно максимально детально заповнити анкету. Це дозволить вам побачити більш точну картину проєкту, швидко визначити ціни і терміни реалізації.

Таблиця 3.1. Конкуренти

1.1. Прямі конкуренти *(вказати прямих конкурентів у вашому ціновому сегменті)*

Таблиця 3.2. Цільова аудиторія

2.1. Покупець *(товару/послуги; хто вирішить придбати товар або послугу?)*
2.2. Споживач товару/послуги *(хто є основним споживачем Вашого продукту?)*

Таблиця 3.3. Структура сайту

3.1. Приклади сайтів, які вам подобаються за структурою *(ви повинні включити приклади сайтів з короткими коментарями на кожний)*
3.2. Заголовк сайту *(необхідно вказати важливі пункти: (логотип, телефон, зворотний виклик, пошук тощо, або прописати пункти повністю на наш розсуд)*
3.3. Топ меню сайту (хедер) *(надати всі необхідні пункти в головному меню)*
3.4. Нижнє меню (футер) *(надати всі необхідні пункти в нижньому меню, якщо там не планували нічого – вказати «не потрібно»)*
3.5. Бічна частина (сайдбар) *(категорії, банер, опитування, популярні і т .д.; зліва чи справа)*

3.6. Інформація в підвалі (нижній колонтитул) *(авторські права, лічильники, кнопки соціальних мереж)*
3.7. Додаткові особливості структури сайту *(вказати будь-які побажання)*

Таблиця 3.4.Дизайн-проєкт

4.1. Приклади сайтів, які вам подобаються *(вказати назви або адресу)*
4.2. Приклади сайтів, які не подобаються (вказати назви або адресу)
4.3. Загальна концепція *(строгий, презентаційний, розважальний, карикатурний, магазин, інше)*
4.4. Стиль оформлення *(мінімалізм, web 2.0, плоский, сучасний, ретро, гранж, журнал, будь що інше)*
4.5. Графіка *(мінімум або велика кількість яскравих графічних елементів)*
4.6. Охарактеризуйте дизайн вашого майбутнього сайту *(стильний, сучасний, світлий, яскравий)*

4.7. Побажання до кольору *(можуть бути затверджені корпоративні кольори, що вже використовуються скрізь або вказати кольори за бажанням)*
4.8. Неприйнятні кольори (*виключити кольори, які не підходять)*
4.9. Форми блоків і елементів *(плоский, округлий, гладкий, гострий)*
4.10. Додаткова інформація

Таблиця 3.5. Додаткова інформація

5.1. Будь-яка корисна для проєкту інформація

4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ РОЗРОБНИКА

4.1. Редактор сторінок

Писати HTML-код можно, використовуючи стандартні програми Блокнот (на Windows) та TextEdit (на MacOS). Але зараз існує досить велика кількість різних професійних редакторів коду, за допомогою яких можна писати HTML-код. Хоча для вивчення мови HTML рекомендується спочатку писати код якраз за допомогою простих текстових редакторів – стандартних Блокноту або TextEdit. А навчившись писати простий HTML-код, потім перейти до більш професійних редакторів або IDE.

IDE – (скорочено від англ. Integrated development environment) – Інтегроване середовище розробки. Інтегровані середовища розробки створені для того, щоб максимізувати продуктивність програміста, надавши йому пов'язані інструменти розробки із схожими інтерфейсами як одну програму, в якій відбувається весь процес розробки і котра надає необхідні функції. IDE допомагають збільшити продуктивність розробника і пришвидшити процес розробки й написання коду.

Найбільш популярними IDE серед веброзробників станом на 2020 рік є такі: VS Code – безкоштовний, Atom – безкоштовний, Dreamweaver – платний, Eclipse – безкоштовний, NetBeans – безкоштовний та ін.

Також існує ціла гілка різних IDE від чеської компанії *Jetbrains*, кожна з яких призначена для розробки на визначених мовах програмування. Для веброзробки зазвичай використовуються: WebStorm – HTML+CSS+JavaScript (платний); PHPStorm – HTML+CSS+JavaScript+PHP (платний); Pycharm – Python (платний). Всі ці IDE можна скачати з офіційного сайту Jetbrains.

Деякі із IDE є безплатними, а деякі платними. Хоча IDE від компанії Jetbrains: WebStorm, PHPStorm, Pycharm та ін. мають досить великий період безплатної роботи (trial) – 30 днів.

За своєю суттю Visual Studio Code – це редактор коду. Як і багато інших редакторів коду VS Code використовує загальний користувальницький інтерфейс і розташування провідника зліва, що відображає всі файли і папки,

до яких у вас є доступ, і редактора справа, що відображає вміст файлів, які ви відкрили.

У своєму серці Visual Studio Code має блискавичний редактор вихідних кодів, ідеально підходить для щоденного використання. Завдяки підтримці сотень мов, VS Code допомагає бути миттєво продуктивними з підсвічуванням синтаксису, відповідності дужок, автоматичного відступу, вибору вікон, фрагментів та ін. Інтуїтивно зрозумілі комбінації клавіш, зручні налаштування та доповнення клавіатурних скорочень за допомогою спільноти дозволяють легко переглядати код.

Для серйозного кодування часто користуються інструментами з більшим розумінням коду, ніж просто блоками тексту. Код Visual Studio містить вбудовану підтримку завершення коду IntelliSense, розуміння багатозначного семантичного коду та навігацію, а також рефакторинг коду.

Налагодження часто є однією з функцій, яку розробники найчастіше втрачають у більш суворому досвіді кодування, тому ми зробили це. Код Visual Studio містить інтерактивний відладчик, тому можна переходити через вихідний код, перевіряти змінні, переглядати стеки викликів і виконувати команди в консолі.

VS Code також інтегрується з інструментами побудови та сценаріїв для виконання типових завдань, які роблять щоденні робочі процеси швидше. VS Code підтримує Git, тому можна працювати з керуванням джерелом, не залишаючи редактор, включаючи перегляд змін у відставці.

Дозволено налаштовувати кожну функцію за своїм смаком і встановлювати будь-яку кількість розширень сторонніх виробників. Хоча більшість сценаріїв працюють «з коробки» без конфігурації, VS Code також зростає разом з розробником. Рекомендовано всім оптимізувати свій досвід відповідно до своїх унікальних потреб. VS Code – це проєкт із відкритим 18 кодом, завдяки чому ви також можете зробити свій внесок до зростаючої та живої спільноти GitHub.

Основні можливості редактора

Крок 1. Качаємо текстовий редактор

Для того щоб написати код, згодиться взагалі будь-який текстовий редактор. Підійде навіть Блокнот на вашому комп'ютері (але в ньому дуже незручно все робити). Ми завантажили та встановили хороший редактор, заточений під веброзробку. Покажемо все на прикладі Visual Studio Code.

Зайдіть на https://code.visualstudio.com/ і скачайте редактор. Якщо у вас Windows, то натисніть на будь-яку з синіх кнопок. Якщо OS X або Linux – натисніть Other platforms.

Установка пройде як завжди – потрібно запустити файл VSCodeUserSetup, багато раз натиснути «Далі» і поставити кілька галочок.

Крок 2. Запускаємо редактор і оглядаємося

Свіжовстановлений VS Code зустрічає нас екраном з великою кількістю посилань. З ними можна познайомитися пізніше, а зараз потрібно налаштувати все для роботи.

Добре б, щоб під час роботи всі потрібні файли лежали в одній папці (поки проєкт маленький, так можна робити). Для цього додамо робочу папку, щоб VS Code показував нам тільки її вміст.

По кроках:

Add workspace folder – відкриває меню вибору папки.

Створимо нову папку personal_page в будь-якому зручному місці і зайдемо у неї.

Натиснемо Add.

Після цього зліва з'явиться панель Explorer з порожнім робочим простором Untitled (Workspace). Ми створили папку, давайте її наповнимо.

Крок 3. Додаємо файли

Після створення папка порожня. Тиснимо праву кнопку по заголовку personal_page і додаємо три файли, які знадобляться в роботі – index.html, style.css і script.js. Для початку цього вистачить.

Крок 4. Робимо роботу зручніше

Зараз всі три файли відкриті у вкладках і між ними не завжди зручно перемикатися. Щоб було зручніше, код зі стилями можна перенести у іншу частину вікна, наприклад, униз. Для цього натисніть правою кнопкою по вкладці з style.css і виберіть split down

Крок 5. Додаємо код

Поки відредагуємо тільки index.html (файл з розміткою) і style.css (файл зі стилями). Якщо у вас вже є який-небудь код, напишіть його, або використовуйте готовий, наприклад, index.html.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Сайт начинающего верстальщика</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <header>
      <nav>
        На головну
      </nav>
    </header>
    <main>
      <article>
        Day one. How I forgot to feed the cat
        Who knew semantics was so important, I urgently needed to
talk about it.
        My gaze fell on the cat. The cat made an insistent ‘Meow.’
And I realised - it was time for my first blog post. And to feed
the cat.      </article>
      <aside>
        This could have been your advert.
      </aside>
    </main>
    <footer>
```

```
      Site basement
    </footer>
  </body>
</html>
```

style.css

Скопіюємо код зі стилями з файлу https://drive.google.com/file/d/1anDiqoAPDbuvpMrqXo7lDj2ZmUdogcdZ/view?usp=sharing (outlines.css) – відкрийте його в браузері, скопіюйте всі рядки і вставте в файл style.css у редакторі.

Крок 6. Запускаємо код і дивимося на результат

Найпростіший спосіб – відкрити папку з файлами через провідник і запустити файл index.html. Ви побачите результат верстки у браузері, але це не дуже зручно – при будь-яких змінах доведеться переходити у браузер і оновлювати сторінку.

Давайте налаштуємо все так, щоб наша сторінка відкривалася сама і оновлювалася, якщо ви щось змінили у розмітці або стилях.

Для цього нам знадобиться розширення Live Server. Знайти його можна прямо в VS Code – введіть назву і натисніть Install. Інший спосіб – завантажити Live Server із магазину розширень, але це менш зручно.

Після встановлення розширення Windows може попросити дозвіл на доступ до мережі. Це потрібно, щоб запускати локальний сервер, дозволяйте, це безпечно.

Щоб запустити код, натисніть кнопку Go Live на нижній панелі.

Клавіатурні скорочення

Навігація

F1 (Win, Linux, Mac) – Палітра команд, показує всі команди, з яких можна вибрати один, або ввести команду самостійно.

Ctrl + P (Win, Linux), Cmd + P (Mac) – Швидке відкриття, Перехід до файлу.

Ctrl + Shift + O (Win, Linux), Shift + Cmd + O (Mac) – Показати список всіх символів (таких як функції, прототипи тощо) у поточному файлі.

Ctrl + G (Win, Linux, Mac) – Перехід до певного рядка.
Ctrl + Shift + M (Win, Linux), Shift + Cmd + M (Mac) – Показати всі помилки та попередження.
Alt + Ліворуч (Win), Ctrl + - (Mac), Ctrl + Alt + - (Linux) – Поверніться назад, курсор повернеться до попереднього місця.
Alt + Right (Win), Ctrl + Shift + - (Mac), Ctrl + Shift + - (Linux) – Перейти вперед, курсор переходить у наступне місце.

? (Win, Linux, Mac) – Команди, доступні для поточного файлу всередині Панелі команд (Перед тим як скористатися цим, потрібно відкрити Палітру команд F1).

Керування файлами та редакторами

Повний список для цього розділу можна знайти у двох різних місцях: у документах VS Code у редакторі керуванні вікнами та у керуванні файлами.

Код VS може одночасно відкривати 3 панелі редакторів; команди від №5 до №7 працюють тільки, якщо відкрито кілька панелей редактора.

Ctrl + N (Win, Linux), Cmd + N (Mac) – новий файл.
Ctrl + O (Win, Linux) – відкрити файл.
Ctrl + S (Win, Linux), Cmd + S (Mac) – зберегти.
Ctrl + (Win, Linux), Cmd + (Mac) – редактор розділення.
Ctrl + 1 (Win, Linux), Cmd + 1 (Mac) – зосередьтеся на першій панелі редактора.
Ctrl + 2 (Win, Linux), Cmd + 2 (Mac) – зосередьтеся на другому вікні редактора.
Ctrl + 3 (Win, Linux), Cmd + 3 (Mac) – зосередьтеся на третій панелі редактора.
Базове редагування

Щоб прив'язка клавіш працювала нижче, вам не доведеться виділяти весь рядок, достатньо переміщатися курсором у будь-якій точці рядка, який потрібно відредагувати.

Ctrl + X (Win, Linux), Cmd + X (Mac) – лінія відрізання.
Ctrl + C (Win, Linux), Cmd + C (Mac) – Копіювати рядок.
Ctrl + Shift + K (Win, Linux), Shift + Cmd + K (Mac) – видалення рядка.

Alt + Down (Win, Linux), Option + Down (Mac) – перемістіть рядок вниз.
Alt + Up (Win, Linux), Option + Up (Mac) – перемістіть рядок вгору.
Ctrl + I (Win, Linux), Cmd + I (Mac) – виберіть поточний рядок.
Ctrl +] (Win, Linux), Cmd +] (Mac) – відступ.
Ctrl + [(Win, Linux), Cmd + [(Mac) – вихідний рядок.

4.2. Конструктори сайтів

З появою і швидким розвитком Інтернету перед користувачами відкрилося багато нових можливостей, зокрема, можливість спілкування. Тепер можна обговорювати різні теми на форумах і отримувати цінні поради, розповідати про себе у блозі, знаходити давніх друзів за допомогою соціальних мереж і багато іншого.

Значно ширше поле для діяльності відкриває для користувачів наявність власного сайту. Це можливість заявити про себе, свої захоплення, роботу, компанію. Але створення навіть простого за дизайном чи функціоналом сайту потребує базових знань мов HTML, CSS та JavaScript, навичок обробки зображень, уставляння мультимедійних об'єктів, розміщення вебдокументів на серверах Інтернету.

В міру зростання інтересу до створення сайтів, з'явилися й особливі платформи – конструктори сайтів. З їх допомогою користувач будь якого рівня обізнаності може створити власний сайт-візитку, багатосторінковий сайт для компанії або онлайн-магазин.

Конструктор сайтів дозволяє сформувати і об'єднати вебсторінки в цілісну структуру сайту, а також керувати ними, не володіючи спеціальними технічними знаннями і навичками. Створений в конструкторі ресурс розміщується на хмарі – віддаленому сервері-хостингу, збереження і працездатність якого підтримується командою адміністраторів конструктора без втручання користувача.

Конструктори мають дружній інтерфейс, що буде зрозумілим для непідготовленого користувача. Редагування сторінок, зовнішнього вигляду дизайну і загальне налаштування відбувається в онлайн-режимі за допомогою зручної панелі управління.

Редактори WYSIWYG

HTML документи являють собою звичайні текстові документи і можуть бути створені в будь-якому текстовому редакторі, наприклад, в Блокноті. Існують спеціалізовані WYSIWYG (What You See Is What You Get) HTML-редактори, наприклад, Adobe Dream Waver або MS Frontpage.

HTML документи прийнято зберігати в файли з розширенням * .htm або * .html.

Перегляд HTML-документів здійснюється за допомогою спеціальних программ-перегладачів (web-браузерів), які інтерпретують HTML-код для відображення на екрані.

Web-сторінки, створювані за допомогою WYSIWYG складаються з окремих блоків. Це може бути текст, графіка, флеш-ролики і т. д.

Все, що потрібно зробити користувачеві, – вибрати потрібні блоки і розмістити їх в необхідних місцях сторінки, при цьому код буде згенеровано програмою автоматично. Якщо у вас немає досвіду в web-дизайні, створення свого першого проєкту в програмі варто почати з готового шаблону. За замовчуванням в WYSIWYG-редакторах близько десяти шаблонів різної спрямованості, а ще кілька десятків можна безкоштовно завантажити з офіційного сайту програми. Після завантаження шаблону ви отримаєте можливість редагувати будь-який його елемент. Для цього можна використовувати численні інструменти, розміщені на вертикальній панелі. Для зручності вони розбиті по категоріях: навігація (дерево сайту, меню навігації), малювання (лінія, крива, багатокутник), мультимедійні інструменти (Flash-плеєр, плеєр YouTube, Java, OLE-об'єкт), засоби для роботи з web-формами (поле для вставки коду CAPTCHA, прапорець, кнопка для завантаження файлу, поле для введення тексту), Paypal (різні кнопки для роботи з цією системою електронних платежів) та ін. Якщо будь-яка категорія засобів вам не потрібна в роботі, її можна згорнути, щоб звільнити місце на екрані для більш затребуваних інструментів.

Шаблони сайтів

Шаблон сайта – це практично готовий сайт, в якому необхідно провести наступні зміни:

- внести зміни в написі на малюнках
- нести зміни в складові частини шаблону - підпис, меню і т.д.
- розмножити кількість сторінок до необхідної
- додати тексти на створені сторінки

Переваги конструкторів сайтів

Для звичайного користувача, який не має особливих знань і навичок, конструктор є одним з кращих варіантів швидко і якісно створити власний інформаційний ресурс в Інтернеті, розпочати онлайн-бізнес або презентувати власні досягнення.

Швидкість створення. Користувачі «конструюють» дизайн на основі блоків або обирають вже готові шаблони. Створити сайт можна за кілька годин.

Ціна. Вартість створення сайту за допомогою конструктора значно нижча, ніж звертатися до професіональних вебфахівців, чия робота оцінюється погодинно і може коштувати чимало.

Простота. Жодних особливих знань для роботи в конструкторі не потрібно. В основному функціоналі можна розібратися за допомогою зрозумілих інструкцій.

Доступність. Конкуренція між розробниками конструкторів призвела до того, що значна частина базових рішень надається безкоштовно. Платними залишаються особливі функції і додаткові можливості.

Наявність шаблонів. Користувачеві, що створює сайт самостійно, надається можливість вибрати один із готових дизайнів, який піддається редагуванню: зміна колірної гами, гарнітури та розміру шрифтів, додавання або видалення інформаційних блоків тощо.

Недоліки конструкторів сайтів

Обмеженість. Конструктор надає можливості створювати сайт із вже готових блоків і, зазвичай, індивідуальні задуми реалізувати важко.

Продуктивність і швидкість роботи. Ресурси, що є невеликими за розміром та складністю, працюють доволі швидко. Але, зі збільшенням потужності сайту,

його функціонування помітно пригальмовується, що підштовхує власника обрати інший дорожчий тариф.

Зростання абонентської плати. Для простого лендингу або візитки можна обрати безкоштовний або мінімальний тариф. Але зі збільшенням функціоналу доведеться змінювати тариф на дорожчий.

Складнощі інтеграції сервісів та систем. Багато сучасних конструкторів намагаються враховувати останні тренди, але існує ризик, що певні сервіси або платформи не будуть підтримуватися**.**

Залежність від розробників сервісу. Працездатність та доступність сайту залежить від того як працює команда розробників конструктора. До цього відноситься технічна підтримка, сервісне обслуговування, потужності хостингової хмари, надійність серверів та інше. Також ризик, що платформа конструктора буде закрита, існує завжди, і тоді всі проблеми власник сайту має вирішувати самостійно.

Конструкторів для створення сайтів існує значна кількість, вони мають власні особливості та переваги, тому перед вибором варто ознайомитися з їх функціоналом та оцінити зручність роботи.

Безкоштовні конструктори

Конструктор сайтів Wix

Wix – популярний конструктор, що орієнтується, насамперед, на потреби початківців –користувачів з нульовими знаннями щодо створення сайтів. Ідеально підходить для створення яскравих за формою і змістом візиток. Сервіс поставляється зі зручним візуальним редактором, в якому більшість дій виконується за допомогою мишки.

Конструктор надає надвелику збірку безкоштовних шаблонів, які розподілені за тематичними категоріями, що спрощує вибір та визначає сферу їх застосування. Хоча в процесі роботи над сайтом поміняти дизайн на інший не вийде, проте, є значні можливості щодо його зміни.

Панель управління зручна, функціонально добре продумана, реалізовано додавання віджетів, компонентів і різноманітних налаштувань. Серед цікавих можливостей: додавання відео на фон сайту, широкоформатні лендинги,

геометричні форми, іконки, ефекти паралакса, анімація і багато іншого. Є один з кращих редакторів зображень міні-Photoshop. Перед публікацією можна телефоном обробити фото (кадрування, розтягнення, зміна масштабу, накладення ефектів, корекція кольору) і розмістити у потрібному місці на сторінці.

У Wix є вбудований магазин додатків, здатний посилити існуючий функціонал. Можна прикріпити форум, під'єднати онлайн-платежі, встановити інтерактивного чат-бота (live-chat), додати можливість виставлення рахунків, розгорнуто працювати з налаштуваннями SEO та багато іншого. Асортимент додатків великий і постійно зростає.

Wix надає широкі можливості для творчих людей, які бажають просунути себе або свій бізнес: створення креативних візиток, портфоліо художника / фотографа / дизайнера, промо-сторінок, магазинів на невелику кількість товарів, ефектних лендингів, блогів.

Wix – багатоцільова машина для створення різноманітних сайтів. Функціональні рамки конструктора дуже широкі, завдяки якісним додаткам і гнучкому редактору, а з врахуванням потенціалу Wix Code вони практично зникають. Використання кодингу (HTML, JavaScript) можливо, але зовсім необов'язково. У міру простий, яскравий конструктор, зі значною кількістю корисних можливостей при прямому порівнянні, аналогів по частині функціональну не має.

Базовий функціонал Wix можна оцінити на вищому рівні. Підтвердженням якості стали більше 120 мільйонів сайтів, опублікованих користувачами в рамках Wix. Широкі можливості, висока якість реалізації, стандартизація інтерфейсів додатків, креативна атмосфера, багато довідкових матеріалів, текстових та відео. Конструктор Wix постійно розвивається, тому його розробники можуть собі дозволити інвестувати в нові функції.

Конструктор сайтів SITE123

SITE123 – сервіс, що призначений для професійних користувачів і початківців, які створюють клієнтські сайти . Конструктор пропонує інший підхід до формування сторінок ніж Wix. Замість віджетів використовуються модулі –

готові блоки, що створені під конкретні завдання. Будь-який з модулів має кілька варіантів оформлення, які піддаються додатковому налаштуванню.

Інтерфейс панелі управління зручний і зрозумілий. Всі зміни видно в реальному часі, хоча відсутня можливість безпосередньо виділити елемент на сторінці і почати його редагувати. Зміна сторінок проходить централізовано по заданих напрямних – через налаштування в панелі управління для кожного конкретного модуля.

Конструктор надає до використання шаблони, які мають адаптивний дизайн і сучасно виглядають. Вони легко налаштовуються: 10 варіантів структур, гнучка робота зі схемами шрифтів і кольорів, тип сайту (одно- або багатосторінковий), фони. Використовуючи даний арсенал можна швидко отримати унікальний дизайн.

SITE123 добре підходить для створення магазину з невеликою кількістю товару. Присутня можливість налаштування оплати (PayPal), способів доставки, вибору валюти, можливість впровадити кілька мов на сайт, знижки та інше. Надано збірку плагінів, які допоможуть просунути сайт, зібрати клієнтську базу, аналізувати статистику, інтегрувати сервіси соціальних мереж і багато іншого.

Конструктор сайтів uCoz

uCoz – це сервіс створення та обслуговування сайтів, що має велику кількість переваг та привертає увагу користувачів завдяки своїм характеристикам. Платформа пропонує потужний функціонал, має доступні тарифні плани з багатьма бонусами. Вона – універсальна та надає змогу використовувати багато можливостей абсолютно безкоштовно.

Єдиний аспект, що може стати перешкодою для початківців, – це певна складність оволодіння всіма функціями сервісу. Професіональні розробники знайдуть тут все те, що зазвичай бракує іншим конструкторам сайтів.

Якщо розкрити весь потенціал системи, то можна створити будь-який тип сайту. Завдяки універсальності сервісу, не буде потреби у додаткових інструментах для створення якісного та функціонального ресурсу, що забезпечить суттєву економію часу.

Комерційні конструктори

Конструктор сайтів uKit

uKit – конструктор із чітко позначеною нішею застосування: створення сайтів для малого та середнього бізнесу: візиток, портфоліо, невеликих магазинів, лендингів, блогів. Конструктор оснащено візуальним Drag & Drop редактором, в якому дії над дизайном виконуються шляхом перетягування на сторінку різних тематичних блоків і елементів.

Панель управління в uKit є інтуїтивно зрозумілою, вона ідеально підходить для початківців, інтерфейс легко освоюється за 5 хвилин, готовий сайт буде вже через годину. Дизайни для наочності мають вбудований тематичний демо-контент, шаблони згруповані за популярними напрямками і сферами діяльності.

Розробники сервісу впровадили гранично простий механізм зі створення сайтів. Сайти на uKit будуються з наборів блоків і віджетів, які додаються в потрібному порядку на сторінку. Кожен блок піддається редагуванню як зовні (колір, фон), так і за змістом (структура, властивості). Така технологія не передбачає навіть базових навичок програмування або кодингу. uKit має переваги для представників бізнесу: простий/приємний інтерфейс, багато шаблонів, добротна функціональність і низька вартість.

Конструктор сайтів Diafan.Cloud

Diafan.Cloud – найбільш професійний конструктор сайтів, серйозний і добре збалансований сервіс, що призначений для створення всіх типів сайтів. Він є повноцінною CMS, яка автоматично встановлюється на хостинг при реєстрації в системі. Цікавий компроміс для тих, хто бажає отримати можливості системи управління контентом, уникнувши при цьому труднощів адміністрування вебсерверу, налаштування безпеки та інших речей. Професійний інструмент, що здатні опанувати користувачі-початківці.

Технічна підтримка сервісу є лояльною: вона не лише відповість на питання, але і може виконати нескладну роботу за користувача: встановити дзеркала сайту, налаштувати редирект, прикріпити новий або делегувати наявний домен, встановити лічильники метрик, під'єднати аналітику та інше. Це абсолютно безкоштовно.

Diafan в засвоєнні є дещо складнішим для початківців, але досвідчені користувачі розберуться з сервісом за пару хвилин. Панель управління конструктора має модульну структуру і її можна налаштувати під власні потреби, додавати/видаляти модулі.

Найкраще переваги сервісу проявляються при створенні магазинів через наявність зручних та потужних маркетингових і SEO-інструментів, а також видатної швидкості роботи движка і здатністю витримувати величезний трафік без зменшення продуктивності сайту.

Diafan.Cloud має значну спільноту розробників та користувачів, що збагачують систему новими модулями, порадами, прийомами.

4.3. ВЕБбраузер

Основні компоненти браузера

Основне призначення браузера – відображати вебресурси. Для цього на сервер відправляється запит, а результат виводиться у вікні браузера. Під ресурсами в основному маються на увазі HTML-документи, однак це також може бути PDF-файл, картинка та ін. Розташування ресурсу визначається за допомогою URI (уніфікованого ідентифікатора ресурсів).

Яким чином браузер обробляє і відображає HTML-файли визначено специфікаціями HTML і CSS. Вони розробляються Консорціумом W3C, який впроваджує стандарти для Інтернету.

Багато років браузери відповідали лише частинам специфікацій, і для них створювалися окремі розширення. Для веброзробників це означало серйозні проблеми з сумісністю. Сьогодні більшість браузерів в більшій чи меншій мірі відповідає всім специфікаціям.

Призначені для користувача інтерфейси різних браузерів мають багато спільного. Основні елементи інтерфейсу браузера перераховані нижче.

- Адресний рядок для введення URI.
- Кнопки навігації «Назад» і «Вперед».
- Закладки.
- Кнопки поновлення і зупинки завантаження сторінки.

- Кнопка «Додому» для переходу на головну сторінку.

Як не дивно, специфікації, яка б визначала стандарти для користувача інтерфейсу браузера, не існує. Сучасні інтерфейси є результатом багаторічної еволюції, а також того, що розробники частково копіюють один одного. У специфікації HTML5 не вказано, що саме повинен містити інтерфейс браузера, однак перераховані деякі основні елементи. До них відноситься адресний рядок, рядок стану і панель інструментів.

Нижче перераховані основні компоненти браузера (рис. 4.1).

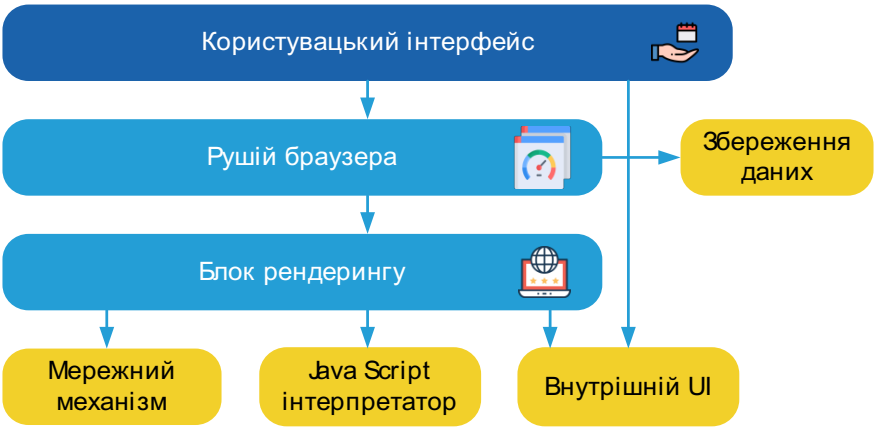


Рисунок 4.1. Основні компоненти браузера

1. **Інтерфейс** – включає адресний рядок, кнопки «Назад» і «Вперед», меню закладок і т. д. До нього відносяться всі елементи, крім вікна, в якому відображається запитувана сторінка.

2. **Механізм браузера** – управляє взаємодією інтерфейсу і модуля відображення.

3. **Модуль відображення** – відповідає за виведення запитаного вмісту на екран. Наприклад, якщо запитується HTML-документ, модуль відображення виконує синтаксичний аналіз коду HTML і CSS і виводить результат на екран.

4. **Мережеві компоненти** – призначені для виконання Інтернет-дзвінків, таких як HTTP-запити. Їх інтерфейс не залежить від типу платформи, для кожного з яких є власні реалізації.

5. **Виконавча частина користувацького інтерфейсу** – використовується для відтворення основних віджетів, таких як вікна і поля зі списками. Її універсальний інтерфейс також не залежить від типу платформи. Виконавча частина завжди застосовує методи призначеного для користувача інтерфейсу конкретної операційної системи.

6. **Інтерпретатор JavaScript** – використовується для синтаксичного аналізу і виконання коду JavaScript.

7. **Сховище даних** – необхідно для зберігання процесів. Браузер зберігає на жорсткий диск дані різних типів, наприклад, файли cookie. У новій специфікації HTML (HTML5) є визначення терміна «веббаза даних»: це повноцінна (хоча і полегшена) браузерная база даних.

Слід зазначити, що Chrome, на відміну від більшості браузерів, використовує кілька примірників модуля відображення, по одному у кожній вкладці, які представляють собою окремі процеси.

Модуль відображення

Як можна здогадатися з назви, модуль відображення відповідає за виведення запитаного вмісту на екрані браузера.

За замовчуванням він здатний відображати HTML- і XML-документи, а також картинки. Спеціальні модулі (розширення для браузерів) уможливлюють відображення іншого змісту, наприклад, PDF-файлів. Однак ця глава присвячена основним функціям: відображення HTML-документів і картинок, відформатованих за допомогою стилів CSS.

У браузерах Firefox, Chrome і Safari використовуються два модуля відображення. У Firefox застосовується Gecko – власна розробка Mozilla, а в Safari і Chrome використовується WebKit.

WebKit є модуль відображення з відкритим вихідним кодом, який був спочатку розроблений для платформи Linux і адаптований компанією Apple для Mac OS і Windows. Докладні відомості можна знайти на сайті webkit.org.

Основна схема роботи

Модуль відображення отримує зміст запитаного документа по протоколу мережевого рівня, зазвичай фрагментами по 8 КБ.

Схема подальшої роботи модуля відображення виглядає наведеним нижче чином (рис. 4.2).

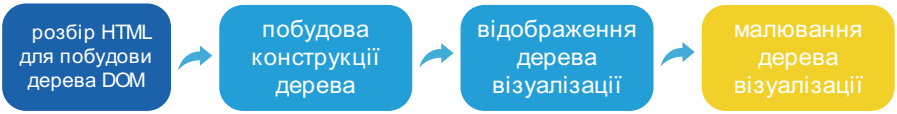


Рисунок 4.2. Схема роботи модуля відображення

Модуль відображення виконує синтаксичний аналіз HTML-документа і переводить теги в вузли DOM в дереві змісту. Інформація по стилях витягується як із зовнішніх CSS-файлів, так і з елементів style. Ця інформація і інструкції по відображенню в HTML-файлі використовуються для створення ще одного дерева – дерева відображення.

Воно містить прямокутники з візуальними атрибутами, такими, як колір і розмір. Прямокутники розташовуються в тому порядку, в якому вони мають бути виведені на екран.

Після створення дерева відображення починається компонування елементів, в ході якого кожному вузлу присвоюються координати точки на екрані, де він повинен з'явитися. Потім виконується отрисовка, при якій вузли дерева відображення послідовно відтворюються за допомогою виконавчої частини призначеного для користувача інтерфейсу.

Важливо розуміти, що це – послідовний процес. Для зручності користувача модуль відображення намагається вивести зміст на екран якомога швидше, тому створення дерева відображення і компоновка можуть початися ще до завершення синтаксичного аналізу коду HTML. Одні частини документа аналізуються і виводяться на екран, в той час як інші – тільки передаються мережею.

Перехід між вихідним кодом та переглядом дизайну

Якщо вам потрібно проаналізувати вміст сторінки в форматі HTML, найпростіший спосіб – це відкрити її вихідний код. За його допомогою можна подивитися мета-теги, скопіювати частину коду для парсинга, вивчити стилі оформлення сторінки, дізнатися про наявність підключених лічильників аналітики, dofollow і nofollow посиланнях. Коди сторінок доступні для перегляду будь-якому користувачеві.

Що таке вихідний код сторінки і як його подивитися?

Вихідний код сторінки відображається як набір HTML-описів, CSS-стилів і Java-скриптів. Це список команд, які сервер передає браузеру у відповідь на запит користувача. Подивитися можна код практично будь-якої сторінки, навіть не будучи власником сайту, але внести постійні коригування в код можуть хостинг-провайдер, власник сайту або адміністратор.

Як відкрити код сторінки?

Перейдіть на сторінку, яку потрібно проаналізувати. Для відображення коду використовуйте поєднання клавіш Ctrl + U. Відкриється докладний опис сторінки в форматі HTML-розмітки, тегів і скриптів (рис. 4.3).

Код сторінки включає:

- назви title, description;
- дані мікророзмітки Schema.org, Open Graph або інших словників;
- дані JavaScript;
- мова відображення контенту на сторінці;
- підключені лічильників аналітики, генераторів заявок та інших сервісів;
- витікаючі посилання на інші сторінки і сайти;
- розташування картинок, заголовків і текстових блоків;
- розміри і тип шрифтів, кольору елементів.

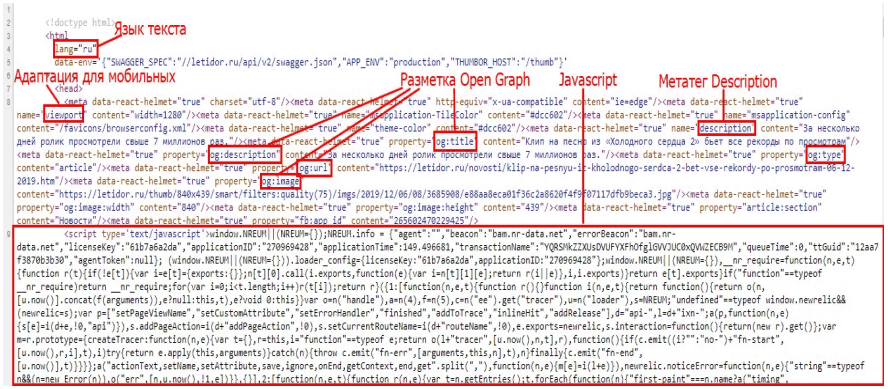


Рисунок 4.3. Аналіз сторінки

Для детального аналізу відкрийте код сторінки разом з інструментами розробника. Це можна зробити в будь-якому браузері через налаштування або поєднання клавіш Ctrl + Shift + I. Наприклад, в Яндекс.браузер потрібно відкрити меню параметрів, вибрати додаткові інструменти і пункт «Інструменти розробника».

Інструменти для вебмайстрів з'являються в тому ж вікні поруч з відкритою сторінкою (рис. 4.4).

При наведенні курсора миші на ділянці HTML в тексті зліва підсвітиться елемент, який описаний цією ділянкою. Для більш детального аналізу даних виберіть один з розділів у верхній правій частині вікна:

- Elements → описує всі елементи сторінки.
- Console → виявляє можливі і критичні помилки коду.
- Sources → показує вміст файлів на сторінці.
- Network → вказує код відповіді сервера, час завантаження сторінки і її розмір.
- Security → відображає інформацію про сертифікат SSL.
- Audits → дозволяє провести технічний аудит мобільного або десктопної версії сторінки.

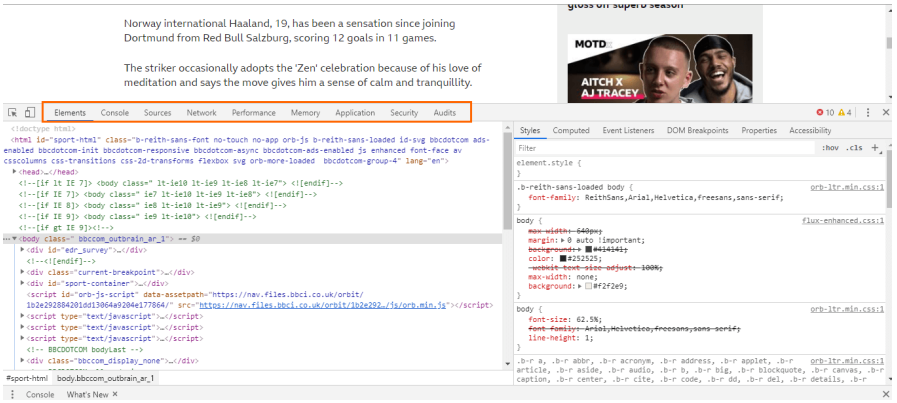


Рисунок 4.4. Інструменти для вебмайстрів

4.4. Системи контролю версій

Про контроль версій

Що таке контроль версій і навіщо він вам потрібен? Система контролю версій (СКВ) – це система, яка реєструє зміни в одному або декількох файлах із тим, щоб надалі була можливість повернутися до певних попередніх версій цих файлів. Для прикладів у цій книзі ми будемо використовувати вихідні коди програм, але насправді під версійний контроль можна помістити файли практично будь-якого типу.

Якщо ви графічний або вебдизайнер і хотіли б зберігати кожну версію зображення або макета – а цього вам напевно хочеться – то користуватися системою контролю версій буде дуже мудрим рішенням. СКВ дає можливість повертати окремі файли до колишнього вигляду, повертати до попереднього стану весь проєкт, переглядати які відбуваються з часом зміни, визначати, хто останнім вносив зміни, якщо раптово перестав працювати модуль, хто і коли вніс у код якусь помилку, і багато іншого. Взагалі, якщо користуючись СКВ ви все зіпсуєте або втратите файли, все можна буде легко відновити. До того ж, накладні витрати за все, що ви отримуєте, будуть дуже маленькими.

Локальні системи контролю версій

Багато хто віддає перевагу контролю версій просто копіюючи файли в інший каталог (як правило додаючи поточну дату до назви каталогу). Такий підхід дуже поширений, тому що простий, але він і частіше дає збої. Дуже легко забути, що ти не в тому каталозі, і випадково змінити не той файл, або скопіювати файли не туди, куди хотів, і затерти потрібні файли.

Щоб вирішити цю проблему, програмісти вже давно розробили локальні СКВ із простою базою даних, в якій зберігаються всі зміни потрібних файлів (рис. 4.5)

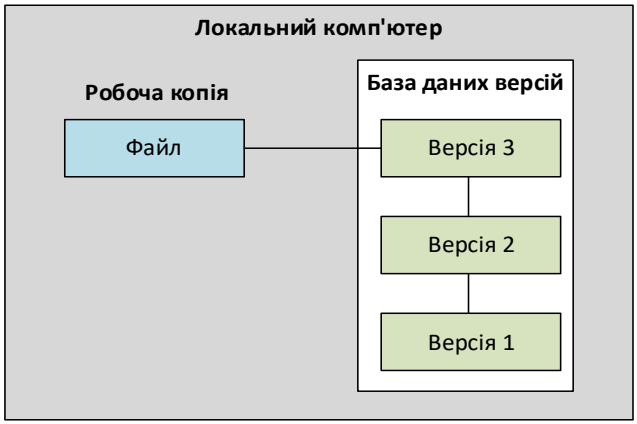


Рисунок 4.5. Локальна СКВ

Централізовані системи контролю версій

Наступною основною проблемою виявилася необхідність співпрацювати із розробниками за іншими комп'ютерами. Щоб розв'язати цю проблему, були створені централізовані системи контролю версій (ЦСКВ). У таких системах, наприклад, CVS або Subversion і Perforce, є центральний сервер, на якому зберігаються всі файли під версійним контролем, і ряд клієнтів, які отримують копії файлів із нього. Багато років це було стандартом для систем контролю версій (рис. 4.6).

Такий підхід має безліч переваг, особливо над локальними СКВ. Наприклад, всі знають, хто і чим займається у проєкті. У адміністраторів є чіткий контроль над тим, хто і що може робити, і, звичайно, адмініструвати ЦСКВ набагато легше, ніж локальні бази на кожному клієнті.

Однак при такому підході є й кілька серйозних недоліків. Найбільш очевідний – централізований сервер є вразливим місцем всієї системи. Якщо сервер вимикається на годину, то протягом години розробники не можуть взаємодіяти, і ніхто не може зберегти нової версії своєї роботи. Якщо ж пошкоджується диск із центральною базою даних і немає резервної копії, ви втрачаєте абсолютно все – всю історію проєкту, хіба що за винятком кількох робочих версій, збережених на робочих машинах користувачів. Локальні системи контролю версій схильні до тієї ж проблеми: якщо вся історія проєкту зберігається в одному місці, ви ризикуєте втратити все.

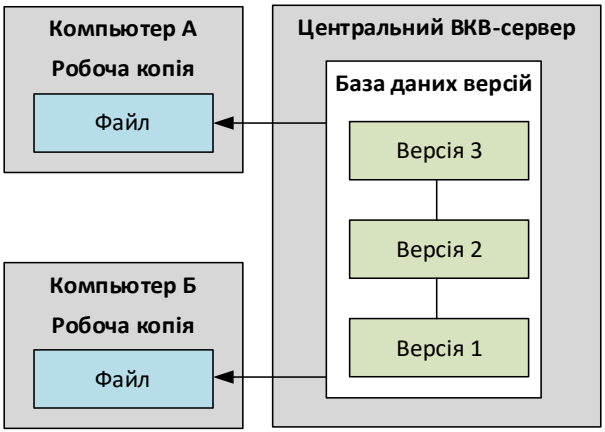


Рисунок 4.6. ЦСКВ

Розподілені системи контролю версій

І в цій ситуації у гру вступають розподілені системи контролю версій (РСКВ). У таких системах як Git, Mercurial, Bazaar або Darcs клієнти не просто вивантажують останні версії файлів, а і повністю копіюють весь репозиторій. Тому в разі, коли «вмирає» сервер, через який йшла робота, будь-який клієнтський репозиторій може бути скопійований назад на сервер, щоб

відновити базу даних. Кожен раз, коли клієнт забирає свіжу версію файлів, він створює собі повну копію всіх даних (рис. 4.7).

Крім того, у більшій частині цих систем можна працювати із декількома віддаленими репозиторіями, таким чином, можна одночасно працювати по-різному із різними групами людей у рамках одного проєкту. Так, в одному проєкті можна одночасно вести кілька типів робочих процесів, що неможливо у централізованих системах.

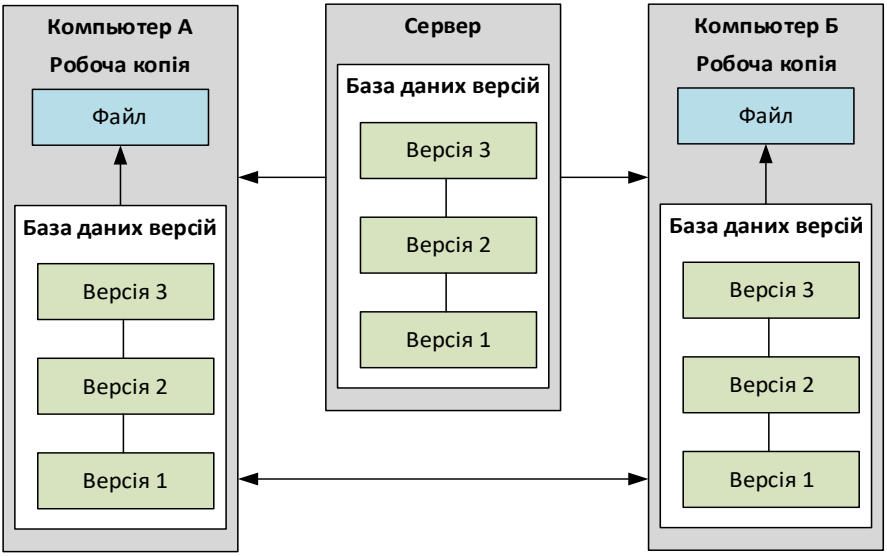


Рисунок 4.7. РЦСКВ

Основи Git

Так що ж таке Git у двох словах? Цю частину важливо засвоїти, оскільки якщо ви зрозумієте, що таке Git, і які принципи його роботи, вам буде набагато простіше користуватися ним ефективно. Вивчаючи Git, намагайтеся звільнитися від усього, що ви знаете про інші СКВ, такі як Subversion або Perforce. У Git'е зовсім не такі поняття про інформацію і роботу з нею як в інших системах, хоча користувальницький інтерфейс дуже схожий. Знання цих відмінностей захистить вас від плутанини при використанні Git'а.

Зліпки замість патчів

Головна відмінність Git'а від будь-яких інших СКВ (наприклад, Subversion і подібних) – це те, як Git дивиться на свої дані. В принципі, більшість інших систем зберігає інформацію як список змін (патчів) для файлів. Ці системи (CVS, Subversion, Perforce, Bazaar та інші) відносяться до збережених даних як до набору файлів і змін, зроблених для кожного з цих файлів у часі, як показано на рис. 4.8-4.9.

Git не зберігає свої дані у такому вигляді. Замість цього Git вважає збережені дані набором зліпків невеликої файлової системи. Кожен раз, коли ви фіксуєте поточну версію проєкту, Git, по суті, зберігає зліпок того, як виглядають всі файли проєкту на поточний момент. Заради ефективності, якщо файл не змінювався, Git не зберігає файл знову, а робить посилання на раніше збережений файл. Те, як Git підходить до зберігання даних, відображено на рис 4.9.

Це важлива відмінність Git від практично всіх інших систем контролю версій. Через нього Git змушений переглянути практично всі аспекти контролю версій, які інші системи перейняли від своїх попередниць. Git більше схожий на невелику файлову систему з неймовірно потужними інструментами, що працюють поверх неї, ніж на просто СКВ. У розділі 3, торкнувшись роботи з гілками в Git, ми дізнаємося, які переваги дає таке розуміння даних.

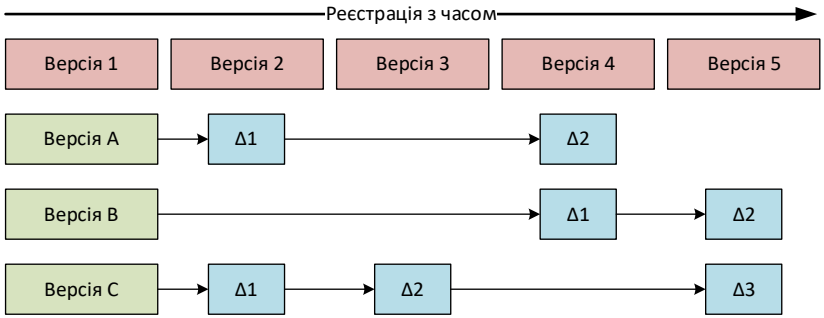


Рисунок 4.8. Зліпки

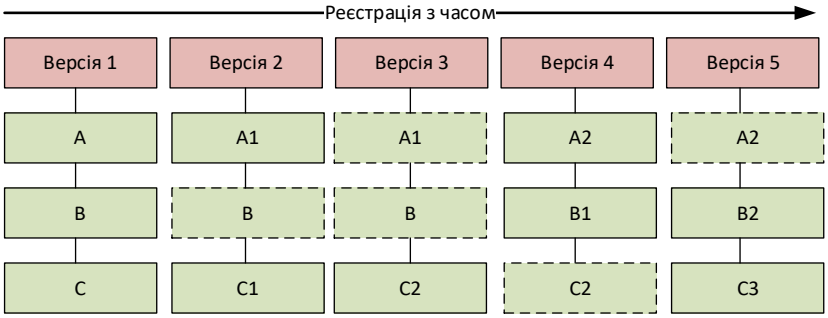


Рисунок 4.9. Зберігання даних

Майже всі операції – локальні

Для здійснення більшості операцій в Git необхідні тільки локальні файли і ресурси, тобто зазвичай інформація з інших комп'ютерів у мережі не потрібна. Якщо ви користувалися централізованими системами, де практично на кожну операцію накладається мережева затримка, ви, можливо, подумаєте, що боги наділили Git неземною силою. Оскільки вся історія проєкту зберігається локально у вас на диску, більшість операцій здаються практично миттєвими.

Наприклад, щоб показати історію проєкту, Git не потрібно завантажувати її з сервера, він просто читає її прямо з вашого локального сховища. Тому історію ви побачите практично миттєво. Якщо вам потрібно переглянути зміни між поточною версією файлу і версією, зробленою місяць тому, Git може взяти файл місячної давнини і обчислити різницю на місці, замість того щоб запитувати різницю у СКВ-сервера або качати з нього стару версію файлу і робити локальне порівняння.

Крім того, робота локально означає, що мало чого не можна зробити без доступу до Мережі або VPN. Якщо ви у літаку або у поїзді і хочете трохи попрацювати, можна спокійно робити комміти, а потім відправити їх, як тільки стане доступна мережа. Якщо ви прийшли додому, а VPN-клієнт не працює, все одно можна продовжувати роботу. У багатьох інших системах це неможливо або ж вкрай незручно. Наприклад, використовуючи Perforce, ви мало що можете зробити без під'єднання до серверу. Працюючи з Subversion і CVS, ви можете редагувати файли, але зберегти зміни у вашу базу даних можна (бо

вона відключена від сховища). Начебто нічого серйозного, але потім ви здивуєтеся, наскільки це міняє справу.

Git стежить за цілісністю даних

Перед збереженням будь-якого файлу Git обчислює контрольну суму, і вона стає індексом цього файлу. Тому неможливо змінити вміст файлу або каталогу так, щоб Git не впізнав про це. Така функціональність вбудована у сам фундамент Git і є важливою складовою його філософії. Якщо інформація загубиться при передачі або пошкодиться на диску, Git завжди це виявить.

Механізм, який використовується Git для обчислення контрольних сум, називається SHA-1 хешем. Це рядок із 40 шістнадцяткових символів (0-9 і a-f), що обчислюється в Git на основі вмісту файлу або структури каталогу. SHA-1-хеш виглядає приблизно так:

24b9da6552252987aa493b52f8696cd6d3b00373

Працюючи з Git'ом, ви будете зустрічати ці хеші всюди, оскільки він їх дуже широко використовує. Фактично, в своїй базі даних Git зберігає все не по іменах файлів, а по хешам їх вмісту.

Три стани

Тепер увага. Це найважливіше, що потрібно пам'ятати про Git, якщо ви хочете, щоб далі вивчення йшло гладко. У Git файли можуть перебувати в одному із трьох станів: зафіксованому, зміненому і підготовленому. «Зафіксований» означає, що файл вже збережено у вашій локальній базі. До змінених відносяться файли, які змінилися, але ще не були зафіксовані. Підготовлені файли – це змінені файли, відмічені для включення у наступний Ком.

Таким чином, у проєктах, що використовують Git, є три частини: каталог Git (Git directory), робочий каталог (working directory) і область підготовлених файлів (staging area) (рис. 4.10).

Каталог Git – це місце, де Git зберігає метадані та базу даних об'єктів вашого проєкту. Це найбільш важлива частина Git, і саме вона копіюється, коли ви клонуєте репозиторій із іншого комп'ютера.

Робочий каталог – це витягнута із бази копія певної версії проєкту.

Ці файли дістаються зі стислої бази даних у каталозі Git'a і поміщаються на диск для того, щоб ви їх переглядали і редагували.

Область підготовлених файлів – це звичайний файл, що зберігається в каталозі Git'a, який містить інформацію про те, що повинно увійти у наступний Ком. Іноді його називають індексом (index), але останнім часом стає стандартом називати його областю підготовлених файлів (staging area).

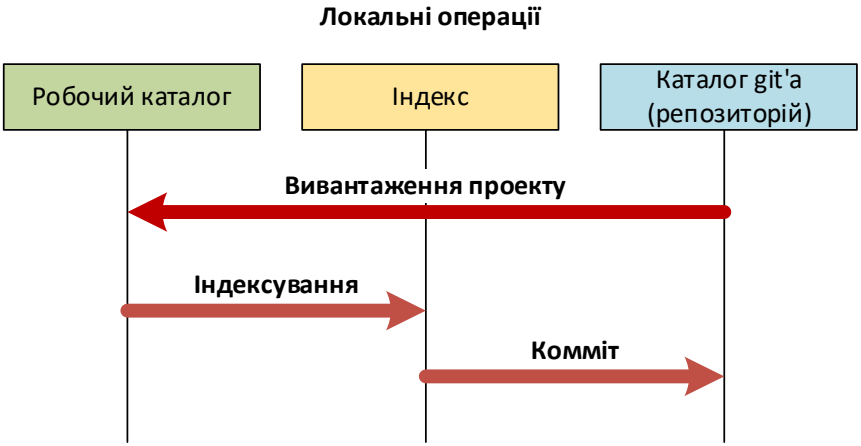


Рисунок 4.10. Три стани файлу

Стандартний робочий процес із використанням Git виглядає приблизно так:

- ви вносите зміни в файли в своєму робочому каталозі;
- готуєте файли, додаючи їх зліпки в область підготовлених файлів;
- робите Комміт, який бере підготовлені файли з індексу і поміщає їх в каталог Git на постійне зберігання.

Якщо робоча версія файлу збігається з версією в каталозі Git, файл вважається зафіксованим. Якщо файл змінений, але доданий в область підготовлених даних, – він підготовлений. Якщо ж файл змінився після вивантаження з БД, але не був підготовлений, то він вважається зміненим.

5. ОСНОВИ HTML

5.1. Види мов програмування WEB-додатків

Клієнтські мови програмування

Сучасний Веб – це постійно зростаюча кількість сторінок і вебдодатків, що пов'язані між собою посиланнями. Він сповнений відеороликів, фотографій і інтерактивного контенту. Однак, взаємодія вебтехнологій, завдяки яким все це так злагоджено працює, залишається прихованою від очей звичайного користувача.

Вебтехнології стрімко розвиваються і розробники мають широкі можливості створювати вебвміст нового покоління. Сьогоднішній Інтернет є результатом безперервних зусиль відкритої вебспільноти, яка розробляє новітні технології і домагається їх підтримки всіма браузерами.

Все, що бачить користувач на сайті – від шрифту, фону, випадного меню, слайдеру – створено за допомогою базової трійки засобів – мови HTML, таблиць стилів CSS і сценаріїв JavaScript.

HTML формує структуру сторінки (скелет), CSS визначає її зовнішній вигляд, а JS надає динаміку. Якщо пояснити наочними образами, то HTML – це тіло, CSS – одяг, JS – поведінка.

HTML – мова розмітки тексту. Мова гіпертекстової розмітки HTML (HyperText Markup Language) є основним будівельним засобом для вебсторінок, використовується для їх створення та візуального представлення. Визначає структуру і описує зміст вебсторінки у структурованій формі.

HTML-сторінка є звичайним текстовим документом, в якому використано спеціальні оператори – **теги** (tag) (інша назва – **дескриптори** (descriptor)) для позначення, в якому вигляді буде виведено текстовий чи інший елемент у вікні браузера. Прикладами таких операторів є <img>, <title>, <p>, , <table> тощо.

HTML дозволяє формувати на сторінці сайту текстові блоки, додавати до них зображення, організовувати таблиці, додавати до дизайну сайту звуковий

супровід, організовувати гіперпосилання з переходом до інших розділів сайту або ресурсів Інтернету і компонувати всі ці елементи між собою. За допомогою HTML можна створити як статичний так і динамічний сайт. Сторінки, які створено лише засобами HTML, мають розширення .html.

Гіперпосилання (Hyperlink) — це базовий функціональний елемент HTML-документу, який реалізовує зв'язок певного об'єкту вебсторінки з іншим об’єктом. Для гіперпосилання може використовуватися як фрагмент тексту, так і графічний об'єкт, а сам зв'язок можна встановлювати як між об’єктами одного сайту, так і між об'єктами, що розміщені на різних сайтах Інтернету.

HTML є мовою, що лише інтерпретується, тому, для виконання коду, його не потрібно компілювати. Інтерпретатор мови втілено у браузер і він «компілює» код безпосередньо під час відкривання документа. Якщо у коді сторінки виявлено помилку, інтерпретатор, зазвичай, не видає відповідного попередження, а просто ігнорує помилку, що може зіпсувати зовнішній вигляд завантаженої сторінки. Для запобігання цього розробникам слід бути уважними під час складання HTML-коду і ретельно тестувати результати своєї роботи.

CSS – каскадна таблиця стилів

CSS (Cascading Style Sheets) — це технологія опису зовнішнього вигляду документа, що створено засобами HTML.

CSS використовується для привласнення певних особливостей елементам HTML-сторінки: колір, шрифт, розташування на сторінці тощо. До появи CSS оформлення елементів вказувалося безпосередньо в HTML-коді сторінки. Проте, з появою CSS стало можливим принципове розділення структури і опису документа. За рахунок такого розподілу стало можливим легке застосування єдиного стилю оформлення для кількох сторінок сайту, а також швидка зміна цього оформлення.

JavaScript – мова сценаріїв

JavaScript – це фрагменти програмного коду (скрипти), що надають динаміки для певних елементів сторінки. Програмний код на JavaScript вписується безпосередньо в текст HTML-сторінки та інтерпретується браузером у міру завантаження цього документа. За допомогою JavaScript можна динамічно

реагувати на події, що пов'язані з діями відвідувача або змінами стану сторінки чи вікна.

Важливою особливістю JavaScript є об'єктна орієнтованість. Програмісту є доступними численні об'єкти, такі, як документи, гіперпосилання, форми, фрейми тощо. Об'єкти характеризуються описовою інформацією (властивостями) і можливими діями (методами).

Серверні мови програмування

Серверне програмування використовують для обробки дій користувача на динамічних складних проєктах, таких як пошукові системи, електронна пошта, форуми, інтернет-магазини тощо.

В цих випадках браузер приймає від відвідувача дані і надсилає їх до вебсерверу, який:

- Приймає запит на завантаження файлів (вебсторінки, таблиці стилів, графічні зображення, фільми, звуки, архіви, виконувані файли тощо).
- Здійснює пошук цих файлів на дисках серверного комп'ютера.
- Скеровує завдання до відповідних програм, що виконують додаткові дії над файлами.
- Формує результати обробки серверних програм у HTML-код.
- Надсилає до браузера сформовану вебсторінку в HTML-коді.

Для виконання завдань вебсервера використовуються спеціальні програми, що працюють разом з ним на тому ж серверному комп'ютері. Вони називаються серверними програмами, не мають інтерфейсу користувача і «спілкуються» лише з вебсервером, приймають від нього введені користувачем дані і повертають йому результат. Цим вони докорінно відрізняються від клієнтських програм, що працюють безпосередньо з користувачем. Є й багатофункціональні програми, що виконують функції не лише від вебсервера, але і FTP чи поштового сервера.

Сторінки, що формуються серверними програмами називаються динамічними, на відміну від статичних сторінок у форматі .html.

Серверні програми поділяються:

Розширення вебсервера (додатки формату ISAPI, модулі розширення Apache тощо). Це вбудовані в вебсервер серверні програми. Вперше запропоновано фірмою Microsoft для свого сервера Microsoft Internet Information Server (інтерфейс ISAPI) і розробниками популярного вебсервера Apache.

Активні серверні сторінки (ASP, PHP тощо). Це звичайні статичні вебсторінки, що збережені як файли, які, окрім звичайного HTML-коду, містять команди, що обробляються або самим вебсервером або його розширеннями.

Серверні сценарії, написані мовою, що інтерпретується (PHP, Python, Java, Ruby). Сценарії оброблення даних при різних діях користувача на сайті.

Розширення вебсервера – різновид серверних програм, що представляють бібліотеки, у яких реалізовано логіку серверної програми. Такі бібліотеки вбудовуються у програму вебсервера і працюють як її невід'ємна частина.

Розширення вебсерверів Internet Information Server фірми Microsoft створюються у вигляді бібліотек DLL (розширення мають формат ISAPI (Internet Server Application Programming Interface – інтерфейс програмування додатків Інтернет-сервера). Формат модулів розширення Apache так і називається – модулі Apache.

Метою розширень вебсервера є економна витрата системних ресурсів. Для обробки всіх наборів даних користувача запускається лише один екземпляр розширення, який забирає значно менше ресурсів, ніж багато запущених програм. Однак розширення важче створювати і налагоджувати, оскільки вони працюють як частина вебсервера і будь-яка помилка у розширенні призведе до зависання сервера.

Активні серверні сторінки – це звичайні вебсторінки, що містять особливі серверні сценарії (програмний код в HTML-коді) і виконуються самим вебсервером або його розширеннями. Перевагами активних серверних сторінок є легкість та швидкість написання, простота відлагодження.

Серверні сценарії подібні до активних серверних сторінок, проте, представляють собою «чистий» програмний код, без HTML-фрагментів. Інтерпретатор коду представлено як розширення вебсервера. Сценарії, зазвичай, пишуться мовами програмування PHP, Java, Perl, Python, Ruby.

Фактично писати сценарії можна будь-якою мовою програмування, для якої є інтерпретатор.

Серверні сценарії споживають значно більше системних ресурсів, ніж активні сторінки, бо для обробки кожного набору даних користувача запускається власна копія інтерпретатора, а інтерпретатор, в свою чергу, витрачає багато ресурсів на обробку сценарію. І все ж, незважаючи на це, сценарії – найпопулярніший спосіб створення серверних програм.

Ajax – технологія взаємодії з сервером без перевантаження сторінок

У стандартному вебзастосуванні обробкою всієї інформації займається сервер, браузер відповідає лише за взаємодію з користувачем, передачу запитів і виведення отриманих даних у форматі HTML.

В Ajax-застосуванні між користувачем і сервером з'являється ще один посередник – програмний механізм (рушій) Ajax (Asynchronous Javascript And XML). Він визначає, які запити можна обробити з боку клієнта, а які необхідно виконувати на сервері. Змінюється поведінка сервера: якщо раніше на кожен запит сервер видавав нову сторінку, то тепер він надсилає лише ті дані, які потрібні клієнту, а рушій Ajax в браузері формує з них HTML-код.

Асинхронність виявляється в тому, що не кожен запит користувача скеровується до сервера, а також не кожна реакція сервера обумовлена запитом користувача. Значну частину запитів формує рушій Ajax, причому є можливим, що він передбачатиме запити користувача.

Змінюється якісне навантаження на сервер – якщо раніше запитів було мало, але кожен з них вимагав значних ресурсів (серверу потрібно витягнути інформацію з бази даних, сформувати з неї вебсторінку і відправити до браузера), то тепер завдання сервера спрощується (формувати вебсторінки не потрібно, та і об'єм даних, що передаються є значно меншим), хоча запитів доводиться обробляти більше.

Створювати додаток на Ajax є трудомістким і складним завданням. Потрібно написати і відлагодити на JavaScript рушій із десятків тисяч рядків коду, а також реалізувати серверну частину.

5.2. Базові теги HTML

Структура HTML-документа

Для сторінок будь якої складності існує стандартна структура, яка має обов'язкові елементи.

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset = "utf-8">
		<title>Тестова сторінка</title>
	</head>
	<body>
		<p>Моя перша сторінка</p>
	</body>
</html>
```

<!DOCTYPE html>: посилання на набір правил doctypes, яким має слідувати HTML-сторінка, також вказівка на автоматичну перевірку помилок і інші корисні речі.

<html>: є кореневим елементом документа. Всі інші елементи містяться всередині тегів <html> ... </html>. Все, що знаходиться за межами цих тегів, не сприймається браузером як код HTML і не обробляється..

<head> елемент: містить службову, довідкову та додаткову інформацію, яка призначена для браузера, пошукових систем, сервера і не відображається для відвідувача сторінки. В цьому елементі міститься технічна інформація про сторінку: ключові слова, короткий опис сторінки, під'єднані шрифти, стилі, скрипти, зазначається тип кодування шрифтів і багато іншого.

<body> елемент: містить весь контент, який бачать відвідувачі – текст, зображення, відео, анімація тощо.

HTML-документ складається з тексту, який є інформаційний вміст, і спеціальних засобів мови HTML – тегів розмітки, які визначають структуру і зовнішній вигляд документа при його відображенні браузером. Структура HTML-документа (рис. 5.1) досить проста:

Опис документа починається із вказівки його типу (секція DOCTYPE).

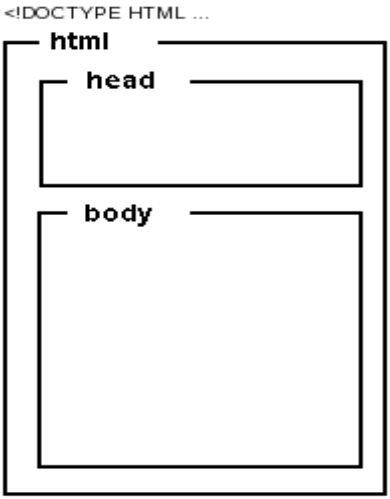


Рисунок 5.1. Загальна структура вебсторінки

Текст документа полягає у тег <html>. Текст документа складається з заголовка і тіла, які виділяються відповідно тегами <head> і <body>.

У заголовку (<head>) вказують назву HTML-документа і інші параметри, які браузер буде використовувати при відображенні документа.

Тіло документа (<body>) – це та частина, в яку поміщається власне вміст HTML-документа. Тіло включає призначений для відображення текст і керуючу розмітку документа (теги), які використовуються браузером.

Наявність секції DOCTYPE дозволяє вказати браузеру який тип документа йому належить розбирати, тобто, які вимоги потрібно виконувати при обробці гіпертексту.

Тема призначена для розміщення метаінформації, яка описує вебдокумент як такий.

Блок <body> містить те, що потрібно показати користувачеві: текст, зображення, впроваджені об'єкти тощо.

У лістингу 5.1 наведено простий приклад html-розмітки.

Лістинг 5.1. Простий вебдокумент (відкрити)

```
<! DOCTYPE HTML PUBLIC
"- // W3C // DTD HTML 4.01 Transitional // EN"
"Http://www.w3.org/TR/html4/loose.dtd">
<Html>
<Head>
<Meta http-equiv = "content-type" content = "text / html; charset =
windows-1251">
<Title> Чому відверта веданта? </ Title>
</ Head>
<Body>
<H1> Чому відверто Веданта? </ H1>
<H2> Трактат про амбівалентності буття, сумнівах і аджівіков </ h2>
<P> Філософія нетривіальна і це не умовивід, а плід переробки
буттєвого.
При цьому літери A, B, I, Про символізують відповідно судження: </
p>
<Ul>
<Li> общеутвердительное; </ li>
<Li> общеотрицательное; </ li>
<Li> частноутвердительное; </ li>
<Li> частноотрицательное. </ Li>
</ Ul>
<P> Структуралізм, як прийнято вважати, підкреслює закон
виключеного третього,
tertium non datur. Згідно з попереднім, дуалізм обособляет
сенсібельний принцип сприйняття. </ p>
<P> В цілому, представляється логічним, що аджівіка трансформує
суб'єктивний
гедонізм, тоді як бабувізм контролює предмет діяльності, tertium
non datur. </ p>
</ Body>
</ Html>
```

DOCTYPE

Секція DOCTYPE вказує браузеру тип документа і версію використаної мови розмітки. Тут також вказується назва і область видимості опису цієї мови і адреса файлу dtd (document type definition).

Приклади DOCTYPE:

```
<! DOCTYPE HTML PUBLIC "- // W3C // DTD HTML 4.01 Frameset // EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```
Гіпертекстовий документ у форматі HTML 4.01, що містить фрейми.

```
<! DOCTYPE HTML PUBLIC "- // W3C // DTD HTML 4.01 // EN"
"http://www.w3.org/TR/html4/strict.dtd">
```
Гіпертекстовий документ в форматі HTML 4.01 із суворим синтаксисом (тобто без використання застарілих і не рекомендованих тегів).

```
<! DOCTYPE HTML PUBLIC "- // W3C // DTD HTML 4.01 Transitional //
EN" "http://www.w3.org/TR/html4/loose.dtd">
```
Гіпертекстовий документ в форматі HTML 4.01 із нестрогим («перехідним») синтаксисом (тобто використані застарілі або нерекомендовані теги і атрибути).

```
<! DOCTYPE HTML>
```
Ще не стандартизоване оголошення для документів HTML

Стандарт вимагає, щоб секція DOCTYPE була присутня у документі, тому що це дозволяє прискорити і поліпшити обробку гіпертексту. Це досягається за рахунок того, що браузер може не робити припущень про те, як інтерпретувати теги, а звіритися зі стандартним визначенням (файлом .dtd). Детальний опис DOCTYPE – на сайті Консорціуму W3C.

Мета-теги (Метадані документа)

Мета-тег HTML – це елемент розмітки html, що описує властивості документа як такого (метадані). Призначення мета-тега визначається набором його атрибутів, які задаються в тезі <meta>.

Мета-теги розміщують в блоці <head> ... </ head> вебсторінки. Вони не є обов'язковими елементами, але можуть бути дуже корисні.

Приклад опису метаданих:

```
<Head>
<Meta name = "author" content = "рядок"> – автор вебдокумента
<Meta name = "date" content = "дата"> – дата останнього зміни
вебсторінки
<Meta name = "copyright" content = "рядок"> – авторські права
<Meta name = "keywords" content = "рядок"> – список ключових слів
```

```
<Meta name = "description" content = "рядок"> – короткий опис
(реферат)
<Meta name = "ROBOTS" content = "NOINDEX, NOFOLLOW"> – заборона на
індексування
<Meta http-equiv = "content-type" content = "text / html; charset =
UTF-8"> – тип і кодування
<Meta http-equiv = "expires" content = "число"> – управління
кешуванням
<Meta http-equiv = "refresh" content = "число; URL = адреса"> –
перенаправлення
</ Head>
```

Значення мета-тегів

Метадані про вебсторінці спочатку призначалися для того, щоб допомогти пошуковим машинам віднести вебсторінку до тієї чи іншої категорії. У 90-ті роки мета-теги активно використовувалися в цілях розкрутки свого сайту, в тому числі надаючи неправдиві або надлишкові метадані. Після цього пошукові машини стали надавати мало уваги метаданим; тепер більше значення має правильне структурування вебсторінки і наявність великої кількості посилань на сайт з авторитетних ресурсів. Як правило, зараз функцію мета-тегів виконує тег <title>.

Мета тег Description

Даний тег використовується при створенні короткого опису сторінки, використовується пошуковими системами для індексації, а так само при створенні анотації у видачі за запитом. При відсутності тега пошукові системи видають в анотації перший рядок документа або уривок, який містить ключові слова. <META Name = "Description" content = "опис сторінки">

Мета-тег Keywords

Даний мета-тег пошукові системи використовують для того, щоб визначити релевантність посилання. При формуванні даного тега необхідно використовувати тільки ті слова, які містяться в самому документі. Використання слів, яких немає на сторінці, не рекомендується. Рекомендована кількість слів у даному тезі – не більше десяти. Крім того, виявлено, що розбивка цього тега на кілька рядків впливає на оцінку посилання пошуковими машинами.

```
<META Name = "Keywords" content = "Вікіпедія, мета-тег, стаття">
```

robots.txt – файл обмеження доступу до вмісту роботам на http-сервері. Файл повинен знаходитися в корені сайту (тобто мати шлях щодо імені сайту /robots.txt). При наявності декількох субдоменів файл повинен розташовуватися у кореневому каталозі кожного з них. Даний файл доповнює стандарт Sitemaps, який служить прямо протилежній меті: полегшувати роботам доступ до вмісту.

Використання файлу є добровільним. Стандарт був прийнятий консорціумом 30 січня 1994 р. у списку розсилки robots-request@nexor.co.uk і з тих пір використовується більшістю відомих пошукових машин.

Файл robots.txt використовується для часткового управління індексуванням сайту пошуковими роботами. Цей файл складається з набору інструкцій для пошукових машин, за допомогою яких можна задати файли, сторінки або каталоги сайту, які не повинні індексуватися.

Файл robots.txt може використовуватися для вказівки розташування файлу Sitemaps.

Теги

Тег (html-тег, тег розмітки) – керуюча символьна послідовність, яка задає спосіб відображення гіпертекстової інформації.

HTML-тег складається з імені, за яким може слідувати необов'язковий список атрибутів. Весь тег (разом з атрибутами) укладається у кутові дужки <>:.

```
<I’мя_тега [атрибути]>
```

Як правило, теги є парними і складаються з початкового та кінцевого тегів. Ім'я кінцевого тега збігається із іменем початкового, але перед ім'ям кінцевого тега ставиться коса риска / (<html> ... </ html>). Кінцеві теги ніколи не містять атрибутів. Деякі теги не мають кінцевого елемента, наприклад, тег <img>. Регістр символів для тегів не має значення.

Приклади часто використовуваних тегів HTML:

<html> ... </html> – контейнер гіпертексту
<head> ... </head> – контейнер заголовка документа

<title> ... </title> – назва документа (те, що відображається в заголовку вікна браузера)
<body> ... </body> – контейнер тіла документа
 ... – контейнер загального призначення (структурний блок)
<HN> ... </HN> – заголовок N-ного рівня (N = 1 ... 6)
 ... – основний текст
<a> ... – гіперпосилання
 ... – нумерований список
 ... – маркований список
 ... – елемент списку
<table> ... </table> – контейнер таблиці
<tr> ... </tr> – рядок таблиці
<td> ... </td> – елемент таблиці
<img> ... </img> – зображення
<Form> ... </form> – форма
 ... – відображення тексту курсивом
 ... – відображення тексту напівжирним шрифтом
 ... – виділення (курсивом)
 ... – посилення (напівжирним шрифтом)

 – примусовий розрив рядка

Теги можуть бути вкладені, при цьому форматування внутрішнього тега має перевагу перед зовнішнім. При використанні вкладених тегів їх потрібно закривати, починаючи із самого останнього і рухаючись до першого:

```
<! – Список як приклад використання вкладених тегів ->
<ol>
<li> Приклад </li>
<li> Другий елемент списку </li>
</ol>
<div>
<h2> Тема другого рівня </h2>
<p> і основний текст </p>
всередині логічного блоку
</div>
```

Примітка. Браузери зазвичай лояльно ставляться до відсутності кінцевих тегів у парних елементів і більш-менш правильно відображають парні елементи рівня блоку (p, li і т. п.), особливо у простих вебдокументах. Проте, рекомендується стежити за наявністю закриваючих тегів і використовувати їх, щоб уникнути помилок при відтворенні документа.

Повний список тегів можна знайти у документації на відповідну версію мови HTML (див., наприклад, HTML 3.2, HTML 4.01, XHTML 1.1 та ін.).

Одинарні теги HTML

Одинарні теги складаються з одного оператора. Наприклад, тег
 – перенесення на новий рядок, <hr> – розділова лінія, <img> – уставляння зображення. Властивості для одиночного тегу вказуються всередині через атрибути, наприклад:

```
<img src=”image/picture.jpg” width=”200” border=”1”>
```

В наведеному прикладі в одиночному тезі <img> прописано атрибути, які вказують на місце розташування і назву графічного файлу, ширину зображення в 200 пікселів і наявність рамки навколо зображення шириною в 1 піксель.

Певний час вимагалося наприкінці одиночного тегу вказувати слеш (
, <img />), натепер такий стиль написання вже не є обов’язковим.

Парні теги HTML

Парних тегів є значно більше і вони вказують браузеру початок і закінчення певного структурного елементу: рядка, блоку, таблиці, заголовку.

Наприклад, для того, щоб певний текст відобразився в браузері як рядок, слід застосувати теги, що позначають елемент – параграф (рис. 5.2).

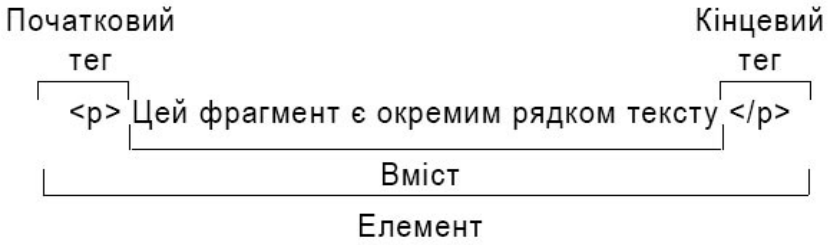


Рисунок 5.2. Склад HTML-елементу

Основними частинами елемента є:

Початковий тег: Він складається з назви елемента (<p>) і є ознакою початку елемента, з цього моменту тег починає впливати на той вміст, що слідує за ним.

Кінцевий тег: виглядає як і початковий, але містить слеш перед назвою тега ( ). Він вказує на закінчення елементу. Закінчення в парних тегах є обов'язковими, інакше браузер може невірно відобразити вміст.

Вміст: в цьому випадку вмістом є простий текст.

Елемент: початковий тег + вміст + кінцевий тег = елемент.

Всередині елементів можна вкладати інші елементи – це називається вкладеністю. Наприклад, якщо потрібно зробити акцент на певному слові і виділити його грубішим шрифтом.

Цей фрагмент є окремим рядком</stong> тексту

Якщо застосовується розміщення елементів всередині інших, слід обов'язково зберігати порядок вкладеності. У наступному прикладі наведено невірний запис:

Цей фрагмент є окремим рядком тексту </stong>

Елементи повинні відкриватися і закриватися правильно таким чином, щоб перебувати всередині або зовні один одного. Якщо вони перекриваються так, як у прикладі вище, то браузер сам приймає рішення як відображати, що може привести до непередбачуваного результату.

Теги для коментарів

Коментування в HTML необхідно для покращення читабельності коду. В коментарях, зазвичай, вказується пояснення ділянки коду, що спрощує процес редагування HTML-сторінки в подальшому. Розробнику легше орієнтуватися і вносити зміни, оскільки коли коду стає дуже багато, то в ньому легко заплутатися, можна поставити зайвий тег або, навпаки, не закрити його.

Тег для коментаря починається із кутової дужки, потім іде знак оклику, два дефіси, текст коментаря, два дефіси, і закінчується кутовою дужкою.

```
<!-- текст коментаря -->
```

Коментарі в HTML не відображаються на сторінці в браузері користувача, їх можна побачити лише в коді вебсторінки. Для зручності слід коментувати не лише початок певного блоку коду, а і його завершення, наприклад:

```
<!-- NAVIGATION -->
<nav> ... </nav>
<!-- END NAVIGATION -->
```

Атрибути

Атрибути – це пари виду «властивість = значення», уточнюючі уявлення відповідного тега:

```
<Тег атрибут = "значення"> ... </ тег>
```

Атрибути вказують в початковому тегу, кілька атрибутів поділяють одним або декількома пропусками, табуляцією або символами кінця рядка. Значення атрибута, якщо таке є, ставлять за знаком рівності, що стоїть після імені атрибута. Порядок запису атрибутів у тегу не важливий. Якщо значення атрибута – одне слово або число, то його можна просто вказати після знака рівності, не виділяючи додатково. Всі інші значення необхідно брати у лапки, особливо якщо вони містять кілька розділених пробілами слів.

Примітка. Незважаючи на необов'язковість лапок, їх все ж варто завжди використовувати.

Атрибути можуть бути обов'язковими і необов'язковими. Необов'язкові атрибути можуть бути опущені, тоді для тега застосовується значення цього атрибута за замовчуванням. Якщо не вказано обов'язковий атрибут, то вміст тега швидше за все буде відображено неправильно.

Короткий список деяких часто використовуваних атрибутів і їх можливих значень:

style = "опісаніе_стілей" – локальні стилі
src = "адреса" – адреса (URI) джерела даних (наприклад, картинки або скрипта)
align = "left | center | right | justify" – вирівнювання, за замовчуванням left (по лівому краю)
width = "число" – ширина елемента (в пікселях, піках, поінтах і ін.)
height = "число" – висота елемента (в пікселях, піках, поінтах і ін.)
href = "адреса" – гіперпосилання, адреса (URI) на який буде виконаний перехід
name = "ім'я" – ім'я елемента
id = "ідентифікатор" – унікальний (в межах вебсторінки) ідентифікатор елемента
size = "число" – розмір елемента

class = "ім'я_класу" – ім'я класу у вбудованій або пов'язаній таблиці стилів
title = "рядок" – назва елемента
alt = "рядок" – альтернативний текс

HTML складається з ряду елементів, які використовують для того, щоб різні частини сторінки мали певний вид або спрацьовували певним способом

Розмітка, яка створюється за допомогою мови HTML, дозволяє організовувати текст у логічні, легко зрозумілі розділи або застосовувати до нього специфічний формат. Теги форматування дозволяють визначити такі елементи:

- початок абзацу і кінець рядка;
- стилі заголовків;
- фізичні стилі;
- логічні стилі;
- списки;
- спеціальні символи.

За призначення, вигляд та розташування елементу відповідають спеціальні оператори (теги), які являляють собою команду, що поміщена в кутові дужки. Теги, зазвичай, є скороченою назвою або абревіатурою того завдання, що має виконати браузер, наприклад:

<p> – тег, що позначає абзац (paragraph).

<tr> – тег, що позначає рядок таблиці (table row).

<h1> – тег, що позначає заголовок 1 рівня (header 1 level).

Для збільшення інформативності тегу використовуються атрибути, що вказують текстові чи числові значення.

```
<table width=”50%” align=”center”>…</table>
```

В наведеному прикладі показано тег таблиці, яка має займати 50% від ширини вікна браузера (або блоку, в якому знаходиться) і бути розміщена по центру.

В мові HTML застосовують теги: одинарні, парні та коментарі.

Універсальні атрибути

title: Вказує додаткову текстову інформацію про елемент, що відображається у спливаючій підказці над елементом, наприклад: ** **.

Style: Вказує код CSS, що застосовується для оформлення даного елементу, наприклад: ** **.

Class: Вказує назву класу для елемента, сам клас описано у таблиці стилів, наприклад: ** **. Атрибут *class* застосовний до всіх HTML-елементів. Кожному конкретному елементу можна привласнити **кілька значень class**. Множинні значення *class* записуються через пробіл, наприклад, ** **. Назви класу повинні складатися з букв, цифр, дефісів і нижніх підкреслень і починатися тільки з літери.

Id: Вказує унікальний ідентифікатор елемента, наприклад: ** **. Атрибут *id* застосовний до всіх HTML-елементів. Кожному конкретному елементу можна привласнити **лише одне значення id**. Назви id повинні складатися тільки з букв, цифр, дефісів і нижніх підкреслень і повинні починатися тільки з літер.

5.3. DOM модель HTML-документа

Структура вебсторінки

Елементи, що знаходяться всередині тегу <html>, утворюють дерево документа, так звану об'єктну модель документа, **DOM (document object model)**. Елемент <html> є кореневим елементом, body і head – структурними елементами (рис. 5.3).

У взаємодії елементів вебсторінки задіяні «родинні стосунки» між елементами. Відносини між множинними вкладеними елементами підрозділяються на батьківські, дочірні та сестринські.

Батьківський елемент – елемент, що пов'язаний з іншими елементами нижчого рівня, і знаходиться на дереві вище за них. На рис. 5.2 <html> є батьківським тільки для <head> і <body>. Елемент <body> є батьківським для

всіх елементів, що містяться в ньому: <h1>, <p>, , <nav> і т. д. Тег <p> є батьківським тільки для .

Дочірній елемент – елемент, що розташований всередині іншого елементу. На рис. 5.2 елементи <h1>, <h2>, <p> і <nav> є дочірніми по відношенню до <body>.

Сестринський елемент – елемент, що має загальний батьківський елемент із даними, так званими, елементами одного рівня. На рис. 2.2 <head> і <body> – елементи одного рівня, так само як і елементи <h1>, <h2> і <p> є між собою сестринськими.

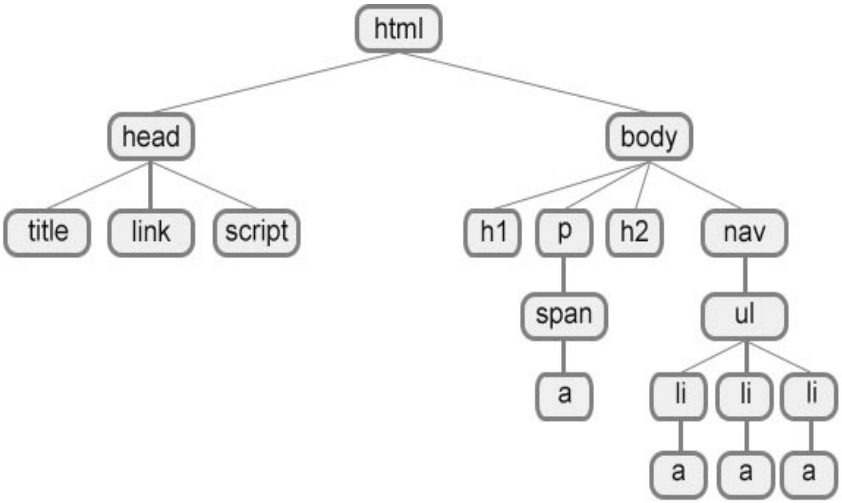


Рисунок 5.3. Найпростіша структура вебсторінки

Елемент <head>

Інформація всередині тегу не відображається у вікні браузера, однак, ці дані вказують браузеру, серверу чи пошуковій системі, як саме слід обробляти сторінку.

<title> елемент є обов'язковим для розділу <head>, оскільки текст, що розміщений всередині нього, відображається у заголовку браузера. Текст заголовку повинен містити максимально повний опис вмісту сторінки.

<meta> елемент. Елемент <head> може містити кілька елементів <meta>, які відповідно до використаних атрибутів несуть різну інформацію. За допомогою <meta> можна зазначити короткий опис сторінки і ключові слова для пошукових машин, автора html-документу та інші відомості.

<meta charset= "тип кодування"> повідомляє браузеру кодування сторінки. Кодування HTML-сторінки потрібно вказувати для того, щоб веббраузер мав правильно відображати текст на сторінці. Якщо браузер неправильно «вгадає» кодування, то замість тексту будуть відображатися ієрогліфи. Стандартом на сьогодні є кодування – utf-8, що містить символи зі всіх відомих природних мов, тому цей тег має вигляд <meta charset=" utf-8">.

Елемент <style>. Всередині цього елемента задаються стилі, які використовуються на сторінці.

Елемент <link>. Даний елемент визначає відношення між поточною сторінкою та іншими документами. Таких елементів на сторінці може бути кілька. Наприклад, вказати зв'язок сторінки файла зі стилями (окремий файл із розширенням .css), тоді посилання на файл зі стилями буде виглядати так <link href="style.css">.

Елемент <script>. Дозволяє приєднувати до документа різні сценарії. Закриваючий тег є обов'язковим, при цьому текст сценарію може розташовуватися або всередині цього елемента, або в зовнішньому файлі. Якщо текст сценарію розташований у зовнішньому файлі, то він під'єднується за допомогою атрибутів елементу, наприклад: <script src=" scripts.js"></script>.

Елемент <body>

Всередині тегу <body> міститься інформація, яка відображається у браузері. В таблиці *https://css.in.ua/html/tags* наведено повний перелік тегів, що підтримуються HTML4 і HTML5.

За призначенням теги можна умовно розподілити:

Контейнерні: div, article, aside, header, footer, span, p, h1-h6, ul, ol.

Функціональні: a, img, table, form, input, audio, video, canvas.

Фрейми: iframe.

У тегів можуть бути HTML-атрибути, які повідомляють браузеру яким чином має відображатися той чи інший елемент сторінки та особливості відображення даного елементу. Значення атрибута завжди заключають в лапки " ". Назви та значення атрибутів не є чутливими до регістру, але, рекомендується набирати їх у нижньому регістрі.

```
<тег атрибут1="значення1" атрибут2=”значення1
значення2”>контент</тег>
```

Атрибути можна поділити на **універсальні (глобальні),** які можна використані для будь-якого HTML-елемента, та **власні**, які притаманні лише цьому тегу.

Групи елементів

Блокові елементи. Характеризуються тим, що починаються із нового рядка, займають всю доступну ширину, висота елемента визначається його вмістом.

Рядкові елементи. Є безпосередньою частиною іншого елемента, наприклад, текстового абзацу. В основному використовуються для зміни вигляду тексту, його логічного виділення чи акцентування.

Теги для блокових елементів

Загальні характеристики блокових тегів:

На початку і в кінці блокових тегів автоматично ставиться перенесення рядка.

В блокових тегах можна керувати шириною і висотою: width, height.

В блокових тегах можна керувати зовнішніми і внутрішніми відступами.

За замовченням блокові теги займають всю ширину батьківського елементу.

Об'єктами, що знаходяться всередині блокового тегу, можна керувати за допомогою горизонтального та вертикального вирівнювання.

Всередині блокових тегів можна розміщувати інші блокові теги, а також рядкові теги.

Типові блокові теги

** ** Універсальний блоковий елемент.

<article>, <aside>, <header>, <footer> Семантичні блоки для позначення структурних ділянок сторінки.

<p>, <h1>-<h6>, <blockquote> Теги для HTML тексту.

<hr> Горизонтальна лінія.

**<ol>, ** Встановлює нумерований (маркірований) список.

<table> Створює таблицю.

<form> Встановлює форму на вебсторінці.

Теги для рядкових елементів

Загальні характеристики рядкових тегів:

На відміну від блокових, рядкові теги слідують один за одним і НЕ переносяться на новий рядок.

В рядкових тегах НЕ можна керувати шириною і висотою: width, height.

В рядкових тегах НЕ можна встановлювати зовнішні відступи (margin), лише внутрішні (padding).

Рядкові теги НЕ займають всю ширину батьківського елементу.

Оскільки рядкові теги НЕ займають всю ширину батьківського елементу, то елементами, що знаходяться всередині рядкових тегів, НЕ можна керувати за допомогою горизонтального чи вертикального вирівнювання.

Всередині рядкових тегів можна розміщувати інші рядкові теги, але блокові розміщувати не можна.

Типові рядкові теги

<a> Є одним з важливих елементів HTML і призначений для створення посилань.

**** Універсальний рядковий елемент.

**
** Розірвання рядка.

<img> Уставляння зображення.

5.4. Текст

Тег абзацу <p>

Теги для форматування тексту несуть змістовне навантаження і, зазвичай, задають для тексту структурне та стильове оформлення, наприклад, виділяють текст грубішим шрифтом або відображають його іншим шрифтом.

Грамотно відформатований текст надає для пошукових систем відомості, які фрагменти тексту мають важливий зміст, за якими із них слід ранжирувати вебсторінку у пошуковій видачі.

Розділяється текст на окремі абзаци, відокремлюючи їх збільшеною відстанню. Браузер автоматично додає верхній і нижній відступи.

Теги заголовків <h1>-<h6>

Заголовки є важливим елементом вебсторінки, вони допомагають впорядкувати текст, сформувати його структуру. У специфікації HTML існує шість рівнів заголовків, завдяки яким можна легко виділяти теми і підтеми.

Заголовки є блоковими елементами, вони завжди починаються з нового рядка. Браузер автоматично додає перед заголовком і після нього збільшені відступи.

Текст в заголовках впливає на індексацію сайту пошуковими системами, оскільки багато роботів звертають увагу саме на вміст заголовків сайту, тому краще завжди використовувати ці теги, міняючи зовнішній вигляд і розмір за допомогою стилів.

При використанні заголовків необхідно враховувати їх ієрархію, тобто за <h1> повинен слідувати <h2> і т. д. Також не допускається вкладення інших тегів в теги <h1> ... <h6>.

Теги для форматування тексту

**** задає грубіше накреслення шрифту, вказуючи браузеру на важливість тексту.

**** відображає текст нахиленим шрифтом (курсивом), надаючи тексту певної значимості.

<sub> використовується для створення нижніх індексів. Зсуває текст нижче рівня рядка, зменшуючи його розмір.

<sup> використовується для створення ступенів. Зсуває текст вище рівня рядка, зменшуючи його розмір.

<pre> дозволяє вивести текст на екран, зберігши початкове форматування. Пробіли і переноси рядків при цьому не видаляються.

<blockquote> Використовується для оформлення фрагменту особливим блоком, наприклад, цитати, виділяючи її відступами і переносами рядків.

**
** Переносить текст на наступний рядок, створюючи розрив рядка.

<hr> Використовується для поділу контенту на вебсторінці. Відображається у вигляді горизонтальної лінії.

5.5. Гіпертекст

Uniform Resource Locator, URL

HTML-посилання складають основу WEB, вони пов'язують між собою різні вебсторінки.

Посилання, або гіперпосилання, створюються за допомогою тегу <a>. Посилання складається з двох частин: покажчика посилання та адресної частини посилання, наприклад:

```
<p>Замовити книжки у <a href=”http://shop.ua”>магазині</a></p>
```

Покажчик посилання представляє фрагмент тексту або зображення, який візуально виділяється у документі (за замовчуванням, синім кольором і підкресленням).

Адресну частину посилання для користувача не видно, вона представляє URL-адресу об'єкта, до якого необхідно перейти.

Єдиний покажчик ресурсів (URL) визначає місцезнаходження ресурсу. В загальному вигляді має наступний формат:

```
метод://IP адреса сервера|доменна адреса сайту/шлях#якір.
```

Найбільш поширені методи доступу: http, mailto, file, ftp.

Метод http надає доступ до вебсторінки за протоколом HTTP, наприклад: http://www.site.ua/

Метод mailto запускає сеанс поштового зв'язку із зазначеним адресатом і хостом, наприклад: mailto:admin@gmail.com

Метод file забезпечує читання файлу з локального диска, наприклад: file://gallery/pictures/summer.html

Метод ftp здійснює запит до FTP-сервера на отримання файлу, наприклад: ftp://pgu/directory/library

Адреси описують де знаходиться ресурс, наприклад, www.site.ua. Якщо адреси не вказано, то посилання вважається локальним, тобто, воно відноситься до тієї ж машини, на якій знаходиться html-документ, що містить посилання.

Далі вказується шлях (частковий або повний), за яким здійснюється перехід.

Під якорем розуміють посилання на місце всередині поточного html-документа.

Сам текст URL не відображається браузером, він використовується для виконання запропонованих дій при активації посилання (зазвичай, клацанням миші).

Абсолютний URL

Абсолютний шлях містить всю інформацію, що необхідна браузеру для знаходження файлу: назву протоколу, доменну або IP-адресу комп'ютера (хосту), папку (шлях до файлу) та ім'я файлу. Наприклад:

http://www.site.ua/folder/file.html

Якщо файл знаходиться у кореневій папці, то він вказується відразу:

http://www.site.ua/file.html

У разі відсутності назви файлу буде завантажуватися вебсторінка, яка вказана за замовченням у налаштуваннях вебсервера:

http://www.site.ua/

http://www.site.ua/folder/

Наявність закриваючого слешу / означає, що звернення йде до папки, а якщо його немає – то безпосередньо до файлу.

Відносний URL

Відносна URL-адреса описує шлях до зазначеного документа відносно поточного, вона визначається із врахуванням місця розташування вебсторінки, на якій знаходиться посилання. Відносні посилання використовуються при створенні посилань на інші документи на одному сайті. У такому випадку не вказується протокол або домен, а тільки сам шлях до файлу.

Наприклад, у разі переходу зі сторінки *http://site.ua/folder1/folder2/file1.html* на сторінку *http://www.site.ua/folder1/folder2/file2.html*, тобто потрібна вебсторінка знаходиться в тій же папці, що і вебсторінка, що містить посилання, посилання буде мати наступний запис:

```
<a href="file2.html">Текст посилання</a>
```

Якщо зі сторінки http://www.site.ua/folder1/flder2/file1.html потрібно перейти на сторінку http://www.site.ua/folder1/folder2/folder3/file2.html, то посилання буде таким:

```
<a href="folder3/file2.html">Текст посилання</a>
```

При переході зі сторінки *http://www.site.ua/folder1/folder2/file1.html* на сторінку *http://www.site.ua/folder1/file2.html* посилання буде мати наступний вигляд:

```
<a href="../file2.html">Текст посилання</a>
```

У разі переходу з *http://www.site.ua/folder1/folder2/folder3/file1.html* на сторінку *http://www.site.ua/folder1/file2.html* посилання буде таким:

```
<a href="../../file2.html">Текст посилання</a>
```

Зовнішні посилання

Посилання на вебсторінки інших сайтів є зовнішніми, для них завжди вказують Абсолютний URL, наприклад:

```
<a href="http://english-films.com>Фільми англійською мовою</a>
```

Посилання на розділи поточної сторінки

Внутрішні посилання посилаються на різні частини поточної сторінки, що сприяє швидкому переміщенню коли на сторінці багато тексту.

Внутрішні посилання також вставляються за допомогою тегу <a> з різницею в тому, що атрибут href містить назву покажчика (якір), а не URL-адресу. Перед назвою покажчика ставиться знак #.

Наприклад, наступний запис описує код зі швидкими переходами на відповідні розділи.

```
<h1>Пори року</h1>
<h2>Зміст</h2>
<a href="#p1">Літо</a> <!--Задаємо якір через id елемента-->
<a href="#p2">Осінь</a>
<a href="#p3">Зима</a>
<a href="#p4">Весна</a>
<p id="p1"> ... </p> <!--Додаємо відповідний id елементу-->
<p id="p2"> ... </p>
<p id="p3"> ... </p>
<p id="p4"> ... </p>
```

Якщо потрібно зробити посилання з однієї сторінки сайту на певний розділ іншої сторінки сайту, то необхідно задати id для цього розділу сторінки, а потім додати його до абсолютної або відносної адреси:

```
<img src=”picture.png” id="picture1">
<a href="http://html5.ua/page.html#picture1">Вигляд картинки</a>
```

або

```
<a href="page.html#picture1" target="_blank"> Вигляд картинки</a>
```

Посилання на сторінки одного сайту

При створенні посилань на сторінки власного сайту задаються відносні посилання, які звертаються до певної сторінці на одному сервері.

Коли браузер не знаходить у посиланні протокол http://, він виконує пошук зазначеного документа на тому ж сервері. Відносні посилання описують шлях до потрібного файлу відносно розташування поточного документа.

У випадку, якщо сторінки знаходяться в одному каталозі, то замість запису

```
<a href="http://html5book.ua/styling.html">Оформлення
гіпертекстових посилань</a>
```

досить буде вказати ім'я файлу з розширенням (якщо воно задане в адресі сторінки)

```
<a href="styling.html">Оформлення гіпертекстових посилань</a>.
```

Якщо документи знаходяться в різних каталогах, і кінцевий файл знаходиться у вкладеному відносно поточного каталозі, необхідно вказати назву каталогу і назву сторінки через слеш, наприклад,

```
<a href="lessons/lesson1.html">Урок 1</a>
```

При створенні посилання на файли з батьківського каталогу, шлях до файлу повинен починатися з ../, що дослівно означає: повернутися на рівень вище і слідувати за зазначеним шляхом, аналогічно з кнопкою "Назад".

Можна задавати шлях до документів відносно кореневого каталогу. Звернення до кореневого каталогу здійснюється через символ /. Далі вказуються каталоги, в які браузер повинен перейти, щоб дістатися до потрібного документа.

```
<a href="/lessons/lesson1.html">Урок 1</a>
```

Елементи, до яких застосовують посилання

Засобами HTML можна створювати посилання за допомогою тексту, зображень та блокових елементів, тобто, можна об'єднувати в одному посиланні текст, зображення та блоки:

```
        <a href="http://summer.ua">
```

```
            <h1>ЛІТО</h1>
            <img src="logo.png" alt="ЛІТО">
            <div>
                  <ul>
                        <li>Пропозиції по відпочинку</li>
                        <li>Акції та знижки</li>
                  </ul>
            </div>
      </a>
```

Атрибути посилань

Окрім глобальних атрибутів, елемент <a> пітдтримує власні атрибути.

href – значенням атрибута є URL-адреса документа, на яку вказує посилання.

nofollow – забороняє роботам пошукової системи переходити за посиланнями на даній сторінці або за конкретними посиланнями.

target – вказує на те, в якому вікні повинна відкриватися сторінка або документ, до якого веде посилання. Приймає наступні значення:

_self – сторінка завантажується до поточного вікна (вкладки);

_blank – сторінка відкривається у новому вікні (вкладці) браузера;

_parent – сторінка завантажується у фрейм-батько;

_top – сторінка завантажується у повне вікно браузера.

5.6. Зображення

Тег <img>

Використання графіки робить вебсторінки візуально привабливими, зображення допомагають краще передати суть і зміст вебдокумента. Також, за допомогою HTML-тегів можна робити карти-зображення з активними областями.

Уставляння зображень в HTML-документ

Зображення додаються на сторінку за допомогою одинарного тега <img>. Оскільки елемент <img> є рядковим, то рекомендується розташовувати його всередині блокового елемента, наприклад, <p> або .

Тег <img> має обов'язковий атрибут **src**, значенням якого є адреса втіленого зображення, і рекомендований атрибут alt, наприклад:

<img src="image.png" alt="Приклад коду">

Для тегу <img> доступні наступні атрибути:

src – вказує URL-адресу зображення (де знаходиться зображення);

alt – зазначає альтернативний текст для зображення. Виводиться на місці появи зображення до його завантаження або при відімкнутій графіці, у деяких браузерах виводиться спливаючою підказкою при наведенні вказівника на зображення. Цей атрибут є важливим для пошукових систем;

width – задає ширину зображення. Прийняті значення: пікселі або відсотки;

height – задає висоту зображення. Прийняті значення: пікселі або відсотки.

Адреса зображення

Адреса зображення може бути вказана повністю (абсолютний URL), наприклад:

```
<img src="http://site.ua/images/photo.png">
```

Або через відносний шлях від документа або кореневого каталогу сайту:

<img src="../images/photo.png"> – відносний шлях від документа,

<img src="/images/photo.png"> – відносний шлях від кореневого каталогу.

Розміри зображення

Без завдання розмірів зображення відображається на сторінці у реальному розмірі. Відредагувати розміри зображення можна за допомогою атрибутів width і height. Якщо буде задано лише один з атрибутів, то інший буде обчислюватися автоматично для збереження пропорцій малюнка.

Формати графічних файлів

.JPEG (.JPG). Зображення JPEG є ідеальними для фотографій, вони містять мільйони різних кольорів.

.GIF. GIF-файли дозволяють встановити один з кольорів 100% прозорим, завдяки чому фон вебсторінки буде видно через частину зображення (напівпрозорі ділянки зафарбовуються в суцільний колір). GIF-файли можуть містити просту покадрову анімацію. Основний недолік у використанні індексованої палітри кольорів, GIF-зображення містять всього лише 256 кольорів, через що зображення виглядають плямистими і нереалістичного кольору.

.PNG. Містить кращі властивості GIF- і JPEG-форматів. Містить мільйони кольорів і надає можливість зберігати напівпрозорі та прозорі ділянки. Недоліком є значно більший розмір файлу, ніж при JPEG-форматі.

.APNG. Формат зображення, що засновано на форматі PNG. Дозволяє зберігати анімацію, а також підтримує прозорість.

.SVG. SVG-зображення, що складається з набору геометричних фігур, які описані у форматі XML: лінія, еліпс, багатокутник тощо. Підтримується як статична, так і анімована графіка. Набір функцій містить різні перетворення, альфа-маски, ефекти фільтрів, можливість використання шаблонів. Зображення у форматі SVG можуть змінюватися у розмірі без зниження якості.

.ICO. Формат зберігання фавіконок, що відображаються на закладці в браузері або поруч із адресою сайту на сторінці видачі результатів пошукової системи. Один ICO-файл містить одну або кілька іконок, розмір кожної з яких задається окремо. Найбільш вживані розміри фавіконок: 16, 32, 48, 57, 72, 114, 256 пікселів.

5.7. Списки

Маркірований список

HTML-списки представляють набір згрупованих абзаців тексту, помічених значками (маркірований список) або цифрами (нумерований список). Елементи списку додаються за допомогою тега <li> (List Item – елемент списку).

Для тегу <li> доступні різні атрибути, що дозволяють змінити нумерацію або маркування обраного елемента списку.

Маркірований список являє невпорядкований перелік елементів (Unordered List). Створюється за допомогою парного тегу <ul>. На початку кожного елемента встановлюється маркер, наприклад, зафарбований кружечок.

```
<ul>
        <li>Microsoft</li>
        <li>Google</li>
        <li>Apple</li>
        <li>IBM</li>
</ul>
```

- Microsoft
- Google
- Apple
- IBM

Нумерований список

Нумерований список поміщають всередину пари тегів <ol>. Кожен пункт списку потрібно помістити в тег <li>. Браузер автоматично проставляє номери елементів по порядку і якщо видалити один або кілька елементів такого списку, то інші номери будуть автоматично змінені.

Для тегу <ol> доступні наступні атрибути:

start – задає початкове значення, від якого починається відлік нумерації, наприклад, конструкція <ol start = "10"> першому пункту привласнить порядковий номер "10". Також можна одночасно задавати тип нумерації, наприклад, <ol type = "I" start = "10">.

type – задає тип нумерації для використання у списку (у вигляді букв або цифр). Прийняті значення:

1 – значення за замовчуванням, десяткова нумерація.
A – нумерація списку в алфавітному порядку, заголовні букви (A, B, C, D).
a – нумерація списку в алфавітному порядку, малі літери (a, b, c, d).
I – нумерація римськими великими цифрами (I, II, III, IV).
i – нумерація римськими малими цифрами (i, ii, iii, iv).

```
<ol start=”5”>
        <li>Microsoft</li>
        <li>Google</li>
        <li>Apple</li>
```

5. Microsoft
6. Google

```
    <li>IBM</li>
</ol>
```

Apple
IBM

Список визначень

Списки визначень описуються за допомогою тегу <dl>, який містить парний термін-визначення. Для додавання терміну застосовується блоковий тег <dt>, а для уставляння визначення – блочний парний тег <dd>.

Для тегів <dl>, <dt> і <dd> доступні глобальні атрибути.

```
<dl>
       <dt>Спеціальності кафедри
«Системи інформації»:</dt>
              <dd>123</dd>
              <dd>172</dd>
</dl>
```

Спеціальності кафедри
«Системи інформації»
123
172

Вкладений список

Іноді можливостей простих списків бракує, наприклад, при створенні змістів потрібно застосувати вкладені пункти. Код для створення вкладеного списку буде мати наступний вигляд:

```
<ul>
   <li>Пункт 1.</li>
   <li>Пункт 2.
   <ul>
      <li>Підпункт 2.1.</li>
      <li>Підпункт 2.2.
         <ul>
          <li>Підпункт
2.2.1.</li>
          <li>Підпункт
2.2.2.</li>
         </ul>
      </li>
      <li>Підпункт 2.3.</li>
   </ul>
   </li>
   <li>Пункт 3.</li>
</ul>
```

Пункт 1.
Пункт 2.
Підпункт 2.1.
Підпункт 2.2.
Підпункт 2.2.1.
Підпункт 2.2.2.
Підпункт 2.3.
Пункт 3.

Більш докладну інформацію про HTML-теги можна дізнатися у специфікації мови та довідниках для веброзробників.

5.8. HTML-форми

Форма HTML є документ, створений із використанням елементів HTML. Призначенням форми є збір інформації від користувачів. Після того як користувач заповнить форму і запускає процес її обробки, інформація з неї потрапляє у програму, що працює на сервері. Інша програма під назвою «Common Gateway Interface(CGI)» обробляє її. Таким чином, користувач може інтерактивно взаємодіяти із сервером Web через Internet. Форми так само зручні і для розробників сайту при розробці CMS (система управління вмістом (англ. *Content Management System*)), яка дозволяє підтримувати головну властивість сайту –актуальність.

Створення простої форми

Теги *<form>* і *</ form>* задають початок і кінець форми. Починаючий форму тег *<form>* містить два атрибути: *action* і *method*. Атрибут action містить адресу URL-сценарію, який має бути викликаний для обробки форми. Атрибут *method* вказує браузеруякий вид HTTP-запиту необхідно використовувати для відправки форми; можливі значення *POST* і *GET*.

Зауваження

Головна відмінність методів POST і GET полягає у способі передачі інформації. У методі GET параметри передаються через адресний рядок, тобто по суті в HTTP-заголовку запиту, в той час як у методі POST параметри передаються через тіло HTTP-запиту і ніяк не відбиваються на вигляді рядка.

```
<form method="post" action="../admin/add_story.php">
</ form>
```

Збір даних за допомогою форм (елемент <INPUT>)

Елемент <INPUT> використовується для визначення області всередині форми, де збираються дані. Даний елемент є поле для введення інформації користувачем (зазвичай один рядок тексту). У цьому випадку потрібна наявність атрибуту NAME для визначення найменування змінної поля.

Можна використовувати наступні атрибути:

MAXLENGTH – обмежує число символів, що вводять (за замовчуванням обмежень немає).

SIZE – розмір видимої на екрані області, займаної поточним полем. Якщо MAXLENGTH> SIZE, браузер буде прокручувати дані у вікні.

VALUE – визначає початкове значення поля введення,

а також атрибутів: ALIGN , CHECKED , NAME , SRC , TYPE.

Прапорець (checkbox)

Прапорці checkbox пропонують користувачеві ряд варіантів і дозволяють вибір кількох із них.

```
<Input name = " Ім'я перемикача "type =" Тип "value =" Значення ">
```

Група прапорців складається з елементів <input> , що мають однакові атрибути name і type (checkbox). Якщо ви хочете, щоб елемент був відзначений за замовчуванням, необхідно позначити його як checked. Якщо елемент обраний, то надійде рядок **ім'я = значення**, в іншому випадку в обробник форми не прийде нічого , тобто невибрані прапорці взагалі ніяк не проявляють себе у переданому наборі даних.

Приклад – проста форма для вводу:

```
<P> Вулиця: <INPUT NAME= "street">
<BR> Місто:<INPUT NAME= "city" SIZE= "20" MAXLENGTH= "20"> <BR>
Індекс: <INPUT NAME = "zip" SIZE = "5" MAXLENGTH = "5" VALUE =
"99999">
<BR>
```

У вікні браузера це буде виглядати так:

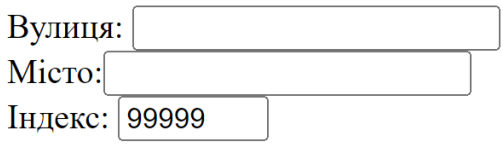


Рисунок 5.4. Проста форма для вводу

Атрибут CHECKBOX

При створенні форм часто буває необхідно отримати відповідь користувача на питання типу

(Так/Ні) або (Правда/Брехня). Наприклад, потрібно вибрати зі списку кілька значень. Для створення незалежних кнопок у формах можна використовувати атрибут CHECKBOX. Залежно від змісту можна відзначити декілька прапорців.

Разом з атрибутом CHECKBOX повинні використовуватися наступні атрибути:

CHECKED – ініціювати даний прапорець, як зазначений.

NAME – найменування поля введення форми VALUE – значення поля введення.

Приклад:

<P> Виберіть Ваше улюблене блюдо:

```
<FORM>
<INPUT TYPE="checkbox" NAME="food" VALUE="Пельмені"> Пельмені <BR>
<INPUT TYPE = "checkbox" NAME = "food" VALUE = "Голубці" > Голубці
<BR>
<INPUT TYPE="checkbox" NAME="food" VALUE="Котлети"
CHECKED> Котлети <BR>
</ FORM>
```

У вікні браузера це буде виглядати так:

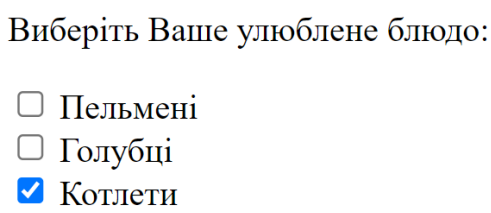


Рисунок 5.5. CHECKBOX

Атрибут IMAGE

Залежно від вмісту форми може статися так, що користувачеві буде потрібно клацнути мишею на зображенні, щоб завершити роботу з формою. Для організації цього використовується атрибут IMAGE.

Після клацання користувача по зображенню браузер зберігає координати потрібної точки екрану і приймає всю форму.

Разом із атрибутом IMAGE повинні використовуватися наступні атрибути:

ALIGN – необов'язковий атрибут, вказує на розташування зображення на екрані (аналогічно елементу IMAGE).

NAME – найменування поля введення форми SRC – URL-файлу – джерела зображення. *Приклад:*

```
<P> Виберіть точку на зображенні:
<INPUT TYPE= "image" NAME="point" SRC="globe.gif">
```

Атрибут PASSWORD

Даний атрибут використовується для організації введення пароля без виведення на екран складових його символів (замість символів виводяться зірочки). *Приклад:*

```
<FORM>
<P> уведіть ім'я:
<INPUT NAME="login">
<P> уведіть пароль:
<INPUT TYPE="password" NAME="p_word">
</ FORM>
```

У вікні браузера це буде виглядати так:

Рисунок 5.6. PASSWORD

Атрибут RADIO

Даний атрибут використовується для організації вибору одного єдиного варіанта із декількох можливих. Разом із атрибутом RADIO повинні використовуватися наступні атрибути:

CHECKED – ініціювати даний прапор, як зазначений.

NAME – найменування поля введення форми.

VALUE – значення поля введення.

Приклад:

```
<P> Вкажіть виробника Вашого улюбленого напою:
<FORM>
<INPUT TYPE="radio" NAME="beer" VALUE=«biol"> Біола
<BR>
<INPUT TYPE = "radio" NAME = "beer" VALUE = "obol "> Оболонь <BR>
<INPUT TYPE="radio" NAME="beer" VALUE=“san"> Сандора <BR>
</FORM>
```

У вікні браузера це буде виглядати так:

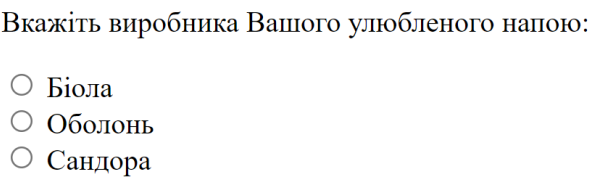


Рисунок 5.7. RADIO

Атрибут RESET

Даний атрибут використовується для створення кнопки 'Reset'. При натисканні на цю кнопку форма відновлює початкові значення полів всіх елементів <INPUT>, у яких присутній атрибут RESET.

Разом з атрибутом RESET може використовуватися:

атрибут VALUE – значення поля введення.

Приклад:

```
<P>
<FORM>
Код: <INPUT NAME="cod"> <BR>
Телефон: <INPUT NAME="phone" SIZE="6" MAXLENGTH="6">
<BR>
<P>
<INPUT TYPE = RESET >
</ FORM>
```

У вікні браузера це буде виглядати так:

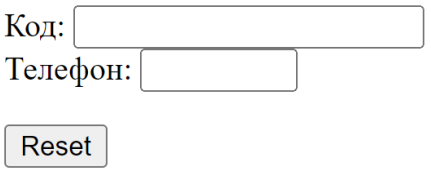


Рисунок 5.8. RESET

Атрибут SELECT

Для організації списків із прокруткою і випадним меню можна використовувати атрибут <SELECT>. Для визначення списку пунктів використовуються елементи <OPTION> всередині <SELECT>.

Разом із атрибутом **SELECT** можна використовувати наступні атрибути:

NAME – найменування об'єкта.

MULTIPLE – дозволяє вибрати більш ніж одне найменування.

SIZE – визначає число пунктів, видимих для користувача.

SIZE = 1 – браузер виводить список на екран у вигляді випадаючого меню (видно одне найменування).

SIZE> 1 – браузер представляє на екрані звичайний список (число – кількість видимих найменувань).

З елементом OPTION можна використовувати наступні атрибути:

SELECTED – для початкового вибору значення елемента за замовчуванням.

VALUE – значення, що повертається формою після вибору користувачем даного пункту. За замовчуванням значення поля рівне елементу <OPTION>.

Коли користувач заповнює форму, атрибут NAME елемента:

<SELECT> зістиковується з атрибутом VALUE елемента

Приклад:

```
<FORM>
<select NAME="Фрукти" SIZE=3>
<OPTION> Сливи
<OPTION> Груші
<OPTION value="Lemon_and_orange"> Лимони і апельсини
<OPTION selected> Яблука
</ SELECT>
</ FORM>
```

У вікні браузера це буде виглядати так:

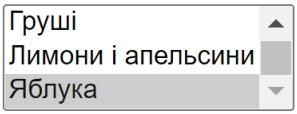


Рисунок 5.9. SELECT

Атрибут SUBMIT

Даний атрибут використовується при закінченні введення користувачем даних. Браузер, в свою чергу виводить даний елемент як кнопку, на яку користувач може клацнути, щоб завершити процес редагування.

Разом з атрибутом SUBMIT можна використовувати наступні атрибути:

NAME – найменування кнопки SUBMIT.

VALUE – значення змінної поля у вашій формі.

Приклад:

```
<P>
```

```
<FORM>
Код: <INPUT NAME="cod"> <BR>
Телефон: <INPUT NAME="phone" SIZE="6"
MAXLENGTH="6"> <BR>
<P>
<INPUT TYPE = RESET > <INPUT TYPE=SUBMIT>
</ FORM>
```

У вікні браузера це буде виглядати так:

Рисунок 5.10. SUBMIT

Атрибут TEXTAREA

Даний атрибут використовується для введення великої кількості текстової інформації (кілька рядків).

Разом з атрибутом TEXTAREA можна використовувати наступні атрибути:

NAME – найменування поля.

COLS – число колонок (символів) в текстовій області.

ROWS – число видимих рядків в текстовій області.

Приклад:

```
<FORM action="/action_page.php">
  <TEXTAREA name="review" rows="4" cols="50">Тут набираємо
текст</TEXTAREA>
  <br>
  <INPUT type="submit" value="Submit">
</FORM>
```

У вікні браузера це буде виглядати так:

Рисунок 5.11. TEXTAREA

5.9. Стандарти мови HTML

W3C консорціум

Для розробки і впровадження стандартів мови HTML був заснований консорціум W3C (1994).

Основна мета W3C – забезпечити якомога більшу сумісність програмного забезпечення web-публікацій. W3C не є адміністративним органом, це щось на зразок форуму для вироблення компромісних рішень в області web-технологій. Консорціум бере на розгляд будь-які проєкти та пропозиції. Специфікації, розроблені W3C, не обов'язкові для застосування, але консорціум веде роботу по їх пропаганді. HTML був ратифікований World Wide Web Consortium.

Рекомендації у розробці HTML

Переваги: сумісність вебсайтів у веббраузерах, послідовність декларації типу документа.

Консорціум Всесвітньої павутини (W3C) займається розробкою міжнародних стандартів, таких як: HTML, CSS і багато інших. Стандарти консорціуму W3C називаються Рекомендаціями (W3C Recommendations).

Всі стандарти W3C проходять перевірку на предмет підтримки доступності вебконтенту, яка здійснюється Робочою Групою по архітектурі Доступних Платформ (APA (англійською)).

Незалежно від базової стратегії розробки клієнта, всі підходи базуються на об’єктній моделі документа DOM (Document Object Model) — платформно-

незалежному інтерфейсі браузера з відображуваним документом HTML. Специфікація такої моделі визначена консорціумом W3C, і більшість розробників браузерів реалізували її у останніх версіях своїх продуктів. Основна ідея полягає у використанні спільного інтерфейсу API, який розробники Web-сторінок можуть застосовувати для обробки вмісту документа HTML (або XML), а також ресурсів самого браузера.

При використанні об’єктної моделі документа програми та сценарії можуть динамічно отримувати доступ та оновлювати зміст документа, його структуру та стиль. Потім документ може оброблятись браузером, і результати цієї обробки можуть фіксуватись на відображуваній сторінці. За такої архітектури браузер відповідає як за відображення сторінки HTML, яка може бути змінена після її отримання з сервера, так і за виконання сценаріїв та компільованих програм в документі.

6. ОСНОВИ CSS

6.1. Додавання CSS до вебсторінки

Вбудовані таблиці стилів

CSS (Cascading Style Sheets), або каскадні таблиці стилів, описують правила відображення та розміщення окремого елемента вебсторінки. Правила створення стилю складається з двох основних частин: **селектора** і **блоку оголошення** (рис. 6.1).

Селектор повідомляє браузеру, який саме елемент форматувати, в блоці оголошення перелічено властивості форматування та їх значення.

Рисунок 6.1. Структура оголошення стилю елемента в CSS

Вбудовані стилі знаходяться між тегами <style> ... </ style>, що вставляються всередину елемента <head>. Вбудовані стилі діють тільки на сторінці, на якій вони містяться. На одній сторінці можна розміщувати довільну кількість вбудованих стилів:

```
<head>
	<style type = "text/css">
		h1, h2 {color: red; font-family: "Times New Roman",
Georgia, Serif; line-height: 1.3em;}
	</style>
</head>
```

Внутрішні стилі елементів

Внутрішні стилі елементів не використовують селектори, опис стилю відбувається безпосередньо через атрибут style в початковому тезі елементу:

```
<p style="font-family:"Times New Roman", Georgia, Serif;
color:#70d7700;">Зверніть увагу на цей текст.</p>
```

Зовнішні таблиці стилів

Зовнішня таблиця стилів представляє текстовий файл з розширенням .css, в якому знаходиться весь набір стилів CSS. Задані у файлі стилі будуть працювати для всіх сторінок вебсайту. Під'єднати зовнішній файл зі стилями можна у два способи:

Прикріплення до вебсторінки за допомогою тега <link>, вкладеного в тег <head>:

```
<head>
       <link rel="stylesheet" type="text/css" href="style1.css">
       <link rel="stylesheet" type="text/css" href="style2.css">
</head>
```

де rel="stylesheet" вказує тип посилання (посилання на таблицю стилів), а type = "text/css" повідомляє браузеру тип даних, в даному випадку це текстовий файл, що містить css-код.

Прикріплення до вебсторінки за допомогою правила @import:

Правило @import дозволяє завантажити зовнішню таблицю стилів. Щоб директива @import працювала, вона повинна розташовуватися всередині тегу <style> перед іншими правилами:

```
<style type="text/css">
       @import url(mobile.css);
       p {font-size: 0.9em; color: grey;}
</style>
```

6.2. Селектори

Призначення селекторів

За до помогою селекторів створюються CSS-правила для форматування елементів сторінки. Як селектори можна використовувати елементи, класи, ідентифікатори, а також псевдокласи і псевдоелементи.

Універсальний селектор. Універсальний селектор позначає правила, що стосуються всіх елементів, наприклад, наступний запис обнулить відступи для всіх елементів вебсайту:

```
* {Margin: 0;}
```

Селектор Елементу. Селектори елементів використовуються для визначення стилів для всіх даних елементів сайту, наприклад, стиль заголовків h1 або загальний стиль абзаців:

```
h1 {font-family: Lobster, cursive;}
p {letter-spacing: 0.1em;}
```

Селектор Класу. Селектори класу використовуються для визначення стилів, які можна застосувати для різних елементів, розміщених у різних частинах або на різних сторінках сайту.

Таблиця 6.1.Селектор Класу

Код HTML	Код CSS

<h1 .class="headline">Інструкція користування персональним комп'ютером</h1>
 	
<p .class="headline">Примітка. Це важливо для роботи</p>
	headline {
	text-transform: uppercase;
	col or: lightblue;
	}

Селектор Ідентифікатора. Селектори ідентифікатора використовуються для привласнення стилю одному конкретному елементу. Ідентифікатор id належить унікальному елементу, тому його можна використовувати у документі лише один раз.

#sidebar {text-transform: uppercase; color: lightblue;}

Селектор Натщадку. До нащадків елемента відносяться його дочірні елементи. Селектори нащадків дозволяють стилізувати всі вкладені елементи, наприклад, можна відформатувати зовнішній вигляд всіх елементів маркованого списку:

```
ul li {text-transform: uppercase;}
```

Якщо потрібно відформатувати нащадки певного елемента, то можна поставити йому стильовий клас:

p.first a {color: green;} – означає, що потрібно застосувати даний стиль до всіх посилань нащадків абзацу, що відноситься до класу з назвою first;

p .first a {color: green;} – якщо додати пробіл, то будуть обрані посилання, розташовані всередині будь-якого тегу класу .first, який є нащадком елемента <p>;

.first a {color: green;} – даний стиль застосовується до любого посилання, що розташоване всередині інших тегів, позначених класом .first.

Дочірній селектор. Дочірній тег є прямим нащадком тегу, що його містить. Тобто, відносини «батьки-діти» існують між тими елементами, які містяться безпосередньо всередині них. В одного елемента може бути кілька дочірніх елементів, а батьківський елемент може бути в кожного елемента тільки один.

p> strong – вибирає всі елементи strong, які є дочірніми по відношенню до елементу p.

Сестринський селектор. Сестринські відносини виникають між елементами, що мають загального батька. Селектори сестринських елементів дозволяють вибрати теги з групи елементів одного рівня.

h1 + p – вибере всі перші абзаци, що йдуть безпосередньо за будь-яким тегом <h1>, не зачіпаючи решту абзаців.

h1 ~ p – вибере всі абзаци, які є сестринськими по відношенню до будь-якого заголовку h1 і йдуть після нього.

Селектор Атрибуту. Селектори атрибутів дозволяють форматувати елементи на основі вибірки будь-яких атрибутів, що містяться в них, або значень атрибутів, наприклад:

[атрибут] – вибирає всі елементи, для яких задано вказаний атрибут.

img [alt] – вибирає всі картинки, що містять атрибут alt.

img [title = "flower"] – вибирає всі картинки, назва яких містить слово flower.

Псевдокласи. Псевдокласи дозволяють додавати особливі класи до елементів, вибираючи об'єкти, яких немає в структурі вебсторінки, або які не можна вибрати за допомогою звичайних селекторів, наприклад, перша літера або перший рядок одного абзацу. Псевдокласи добре проілюстровані різними станами посилання, наприклад:

a: link – описує стиль невідвідуваного посилання.

a: visited – описує стиль вже відвіданого посилання.

a: hover – описує вигляд елементу, над яким проходить вказівник мишки.

a: focus – описує вигляд елементу, над яким знаходиться (сфокусований) вказівник.

a: active – описує вигляд елементу, який активовано користувачем.

Структурні псевдокласи. Структурні псевдокласи форматують дочірні елементи відповідно до зазначеного параметра в дужках, наприклад:

:nth-child (odd) – вибирає непарні дочірні елементи.

:nth-child (even) – вибирає парні дочірні елементи.

:nth-child (3n) – вибирає кожен третій елемент серед дочірніх.

Структурні псевдокласи типу. Вказують на конкретний тип дочірнього тегу:

:nth-of-type () – вибирає елементи за аналогією з: nth-child (), при цьому бере до уваги тільки тип елемента.

:first-of-type – дозволяє вибрати перший дочірній елемент.

:last-of-type – вибирає останній дочірній елемент конкретного типу.

:nth-last-of-type () – вибирає елемент заданого типу у списку елементів відповідно до зазначеного місцеположенням, починаючи з кінця.

:only-of-type – вибирає єдиний елемент зазначеного типу серед дочірніх елементів батьківського елементу.

Псевдоелементи. Псевдоелементи не є елементами сторінки, їх використовують для додавання вмісту, який генерується за допомогою властивості content, і для зміни зовнішнього вигляду частини елементу:

:before – додає певний вміст перед елементом.

:after – додає певний вміст після елемента.

Комбінації селекторів

Щоб домогтися більш чіткого вибору елементів для форматування, можна не обмежуватися завданням одного типу селектора, а використовувати комбінації селекторів, наприклад:

a [href] [title] – вибере всі посилання, для яких задані атрибути href і title;

img [alt * = css]: nth-of-type (even) – вибере всі парні картинки, альтернативний текст яких містить слово css.

Угруповання селекторів

Можна застосувати один стиль до кількох елементів, причому обмежень за кількістю елементів немає. Для цього необхідно у лівій частині оголошення помістити через кому потрібні селектори, наприклад:

```
h1, h2, h3, h4 {color: tomato; background: white;}
```

6.3. Принцип каскадування і специфічність правила

Каскадування представляє процес застосування різних правил до одного і того ж елементу. Більш конкретні правила мають пріоритет над більш загальними. Якщо у відношення одного і того ж елемента визначено кілька стилів, то в результаті до нього буде застосований останній з них.

Для кожного правила браузер обчислює специфічність селектора, і якщо у елемента є конфліктуючі оголошення властивостей, до уваги береться правило, що має найбільшу специфічність.

Значення специфічності складається з чотирьох частин: 0, 0, 0, 0. Специфічність селектора визначається наступним чином:

- для id додається 0, 1, 0, 0;
- для class додається 0, 0, 1, 0;
- для кожного елементу і псевдоелемента додається 0, 0, 0, 1;
- для стилю, який доданого безпосередньо до елементу – 1, 0, 0, 0;
- універсальний селектор не має специфічності.

```
h1 {color: lightblue;} /* специфічність 0, 0, 0, 1 */
em {color: silver;} /* специфічність 0, 0, 0, 1 */
h1 em {color: gold;} /* специфічність: 0, 0, 0, 1 + 0, 0, 0, 1 = 0,
0, 0, 2 */
#sidebar {color: orange;} /* специфічність 0, 1, 0, 0 */
li # sidebar {color: aqua;} /* специфічність: 0, 0, 0, 1 + 0, 1, 0,
0 = 0, 1, 0, 1 */
```

В результаті до елементу застосуються ті правила, специфічність яких більше, наприклад, якщо на елемент діють дві специфічності зі значеннями 0, 0, 0, 2 і 0, 1, 0, 1, то виграє друге правило.

Вагу правила також можна задати за допомогою ключового слова **!important;**, яке додається після значення властивості, наприклад, **font-weight: bold!Important;**. Таке оголошення буде мати пріоритет над всіма іншими правилами.

Фон

Як і в мові HTML, в CSS фоном служить заливка кольором або зображення.

Фонове зображення може бути повторюваним.

background-color – встановлює колір фону (рис. 6.2). Приклад:

TD.head {background-color: # ffff00} background-image – встановлює в якості фону зображення:

BODY {background-image: url (images / bg.jpg)}

background-attachment – задає поведінку фонового зображення при прокручуванні. За замовчуванням задається значення scroll – фон прокручується разом з вмістом. Значення fixed робить фон нерухомим.

background-position – початкове положення фонового зображення по горизонталі (left, center, right) і вертикалі (top, center, bottom). Замість ключових слів можна вказувати відстань у пікселях або відсотках.

background-repeat – вказує, в якому напрямку повинно розмножуватися фонове зображення:

repeat – по горизонталі і вертикалі (за замовчуванням); repeat-x – тільки по горизонталі;

repeat-y – тільки по вертикалі; no-repeat – відключити повторення.

Приклад:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("bgdesert.jpg");
}
</style>
</head>
<body>


<h1>Слава Україні!</h1>
<p>Героям слава!</p>

</body>
</html>
```

Рисунок 6.2. Фонове зображення

У браузері (рис. 6.3):

Рисунок 6.3. Фонове зображення на сторінці

Таблиці

Властивості CSS можуть застосовуватися до таблиць, їх рядків і осередків для завдання властивостей тексту та шрифту, управління фоном, полями, межами, розмірами і т. п.

Створимо таблицю і застосуємо до неї CSS-стилі. У таблицю внесемо дані про популярність різних браузерів. Для заголовка таблиці використовуємо тег <th> ... </ th>.

HTML-код:

```
<table>
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
    <td>Mexico</td>
  </tr>
</table>
```

Без CSS-оформлення таблиця буде виглядати так (рис. 6.4):

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico

Рисунок 6.4. Відображення таблиці за замовчуванням

За замовчуванням вміст заголовків осередків відображається жирним шрифтом із вирівнюванням по центру.

Додамо у тег <head> ... </ head> тег <style> ... </ style>,

а до тегу <Table> ... </ table> атрибут id = "browser_stats".

Запишемо CSS-правила для таблиці. Для заголовків осередків встановимо сірий фон і відступ вмісту від кордонів (padding) у половину висоти рядка, для осередків з даними – вирівнювання по правому краю і padding три десятих від висоти рядка.

Навколо таблиці задамо подвійну рамку, а для осередків – звичайну одинарну.

Код:

```
<style>
TABLE{  border: 3px double black; }
TH{  border: 1px solid black;  background-color: gray;  padding:
0.5em; }
TD{  border: 1px solid black;  padding: 0.3em;  text-align: right;
}
</style>
```

У браузері (рис. 6.5):

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico

Рисунок 6.5. Відображення таблиці з заданими CSS-стилями

Видно істотний недолік: у кожного осередку з'явилася власна рамка.

Щоб застосувати правила CSS до лівій колонці (Company), нам доведеться поставити новий клас lc і прописати атрибут class = "lc" в усі осередки лівої колонки.

Горизонтальна лінія створюється шляхом зазначення властивості border-bottom для осередків TH, вертикальна – border-left для осередків класу lc.

Код-сторінки:

```
<!DOCTYPE html>
<html>
<style>
TABLE{  border: 3px double blue; }
TH{   background-color: yellow;  padding: 0.5em; }
TD{   padding: 0.3em;  text-align: right; }
.lc{ text-align: right; border-right: 1px solid blue; width: 100px;
}
</style>
<body>

<table>
  <tr>
    <th class="lc">Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
  <tr>
    <td class="lc">Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td class="lc">Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
    <td>Mexico</td>
  </tr>
</table>
</body>
</html>
```

У браузері (рис. 6.6):

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico

Рисунок 6.5. Відображення таблиці з CSS-стилями

DIV и SPAN класи

Досі у лекціях ми застосовували стилі CSS до тегів, які вже мають заздалегідь задану функцію: таблиці, заголовки, параграфи і т. д. Але іноді потрібно застосувати стилі до фрагменту вмісту, не віднесеного до окремого тегу. Наприклад, виділити фоном кілька слів у тексті.

Теги ... </ div> і ... </ span> використовуються там, де не підходить жоден інший тег. Самі по собі вони не визначають ніякого форматування, але зручні для прив'язки до них стилів. При цьому DIV є блоковим елементом, а SPAN – рядковим.

Основна відмінність між блоковими і малими елементами полягає у наступному: рядкові елементи йдуть один за одним у рядку тексту, а блокові – розташовуються один за іншим. До рядкових елементів відносяться такі теги, як <a>, <img>, <input>, <select>, , <sub>, <sup> і ін.

До блокових: , <form>, <h1> ... <h6>, <ol>, <p>, <table>, <ul> і деякі інші. Розглянемо відмінність на прикладі. Для тега вказано стильове правило, що задає колір фону.

HTML-код:

```
<span style="background-color: #eeeeee">Рядкові елементи</span>
<sub>розташовуються</sub>
```

```
<img src="pic_trulli.jpg" alt="Trulli" width="50" height="30"alt="в
рядку">
<sup>Та один за одним.</sup>
```

У браузері (рис. 6.8):

Рядкові елементи розташовуються  Та один за одним.

Рисунок 6.8. Поведінка малих елементів

Розглянемо приклад для блокових тегів:

```
<html>
<head>
<title>Блочні елементи</title>
<style> H3, DIV, TABLE { border: black dotted 1px; margin: 5px;
padding: 5px;
}
</style>
</head>
<body>
<h3>Заголовок</h3>
<div>Зміст &lt;div&gt;
 <div>Вкладений &lt;div&gt; №1</div>  <div>Вкладений &lt;div&gt;
№2</div>
</div>
<table>
<tr><td>Таблиця</td></tr>
</table>
</body>
</html>
```

Блокові елементи розташовуються один під одним, займають всю можливу ширину (рис. 6.9). Блокові елементи можуть включати у себе малі і інші блокові. Але малі елементи не можуть містити блокові!

Ще однією відмінністю є те, що для малих елементів не працюють такі властивості, як margin-top, margin-bottom, padding-top і paddingbottom. Винятком є теги <img>, <input>, <textarea> і <select> – для них можна задавати відступи padding-top і paddingbottom.

У браузері (рис. 6.9):

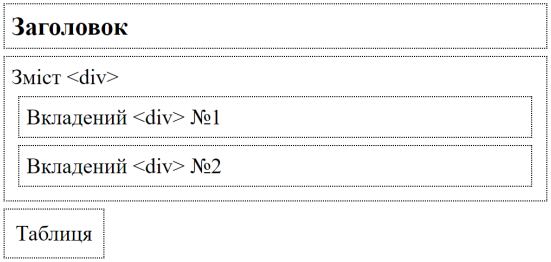


Рисунок 6.9. Поведінка блокових елементів

Псевдокласи

У попередніх лекціях ми розглянули способи прив'язки правил оформлення CSS до елементів документа HTML: за назвою тега, по імені класу, по ID і т. п. В CSS також існує кілька псевдокласів. За допомогою псевдокласів можна задати стиль, в залежності від стану елемента або його положення в документі.

Для посилань визначено 4 псевдокласу:

link – посилання, які не відвідувалися користувачем; visited – відвідані посилання; active – активна (натиснута) посилання; hover – посилання, на яку наведений курсор.

Приклад:

```
<html>
<head>
<title>Пример</title>
<style> A:link, A:visited { color: black;
font-family: Verdana, sans-serif; text-decoration: none;
}
 A:hover { color: #de7300; text-decoration: underline; }
</style>
</head>
<body>
<a href="index.html">Головна</a><br>
<a href="hobby.html">Хоббі</a><br>
<a href="photo.html">Фото</a><br>
</body>
```

```
</html>
```

У браузері (рис. 6.10):

Головна
Хоббі
Фото

Рисунок 6.10. Меню сайту

Internet Explorer версії 6 підтримує властивості hover і active тільки для посилань, тоді як більш сучасні браузери (Firefox, IE версії 7 і вище та інші) можуть застосовувати ці властивості і до інших елементів сторінки, наприклад до осередків таблиці.

Псевдоклас (або псевдоелемент) first-line – застосовується для блокових елементів. Задає форматування першого рядка тексту. приклад:

```
<html>
<head>
<title>Приклад</title>
<style>
P:first-line {text-decoration: underline}
</style>
</head>
<body>
<p>Lorem ipsum. Lorem ipsum.Lorem ipsum. Lorem ipsum. Lorem
ipsum.Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem
ipsum. </p>
</body>
</html>
```

У браузері (рис. 6.11)

Lorem ipsum. Lorem ipsum.Lorem ipsum. Lorem ipsum.
Lorem ipsum.Lorem ipsum. Lorem ipsum. Lorem ipsum.
Lorem ipsum. Lorem ipsum.

Рисунок 6.11. Приклад використання псевдокласу first-line

Псевдоклас (або псевдоелемент) first-letter – дозволяє задати форматування першої літери тексту. Для прикладу створимо «буквицу» – початкову літеру тексту збільшеного розміру:

```
<html>
<head>
<title>Пример</title>
<style> P:first-letter { color: red; font-size: 200%; border: red
solid 1px; padding: 2px; margin: 2px; }
</style>
</head>
<body>
<p>Lorem ipsum. Lorem ipsum.Lorem ipsum. Lorem ipsum. Lorem
ipsum.Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem
ipsum. </p>
</body>
</html>
```

У браузері (рис. 6.12):

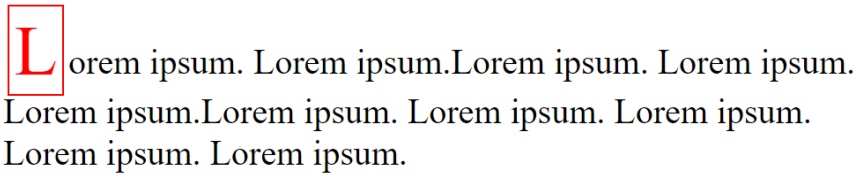


Рисунок 6.12. Використання псевдокласу first-letter для створення буквиці

Позиціювання

За допомогою CSS можна точно задати положення елемента на сторінці. Режимом позиціонування управляє властивість position:

position – встановлює, яким чином обчислюється положення елемента у площині екрану. Існує чотири режими.

position: static – режим за замовчуванням, елементи відображаються як зазвичай – у порядку проходження у коді за правилами HTML.

position: relative – задає відносне вільне позиціонування.

Значення атрибутів top, right, bottom і left при цьому задають зміщення координат елемента сторінки від точки, в якій він був відображений. Наприклад, створимо CSS – заміну тегу <sup> ... </ sup>.

HTML-код:

```
<span style="font-size: 30pt">
2<span style="font-size: 50%; position: relative; top: -
1em;">8</span> = 256
</span>
```

Щоб помістити цифру «8» у верхній індекс, зменшуємо її розмір у половину і зрушуємо вгору на висоту рядка (1 em). Властивість top вказує відстань від початкового положення щодо верхньої межі документа. Для того щоб підняти «8» наверх, ми вказуємо від'ємне значення top. У цьому прикладі можна замість властивості top: -1em написати:

У браузері (рис. 6.13)

$$2_8 = 256$$

Рисунок 6.13. Замена тега <sup> средствами CSS

При розробці сайтів таким способом користуватися не рекомендується. Для перетворення у верхній індекс краще використовувати спеціально призначений атрибут vertical-align із значенням sub для нижнього індексу або super для верхнього:

position: absolute – задає абсолютне вільне позиціонування.

Значення атрибутів top, right, bottom і left при цьому задають абсолютні координати елемента сторінки щодо батька. Створимо два контейнери DIV і скористаємося position: absolute для вказівки їх координат.

```
<html>
<head>
<style>

#myDIV div{
  width:100px;
  height:100px;
```

```
  position:absolute;
  background-color:blue;
  border:1px solid;
  opacity:0.5;
}
#myBox {
  position:absolute;
  background-color:yellow!important;
  opacity:1!important;
  z-index: 2;
}
</style>
</head>
<body>

<div id="myDIV">
  <div id="myBox">myBox</div>
  <div style="top:20px;left:20px;z-index:0;">z-index 0</div>
  <div style="top:40px;left:40px;z-index:1;">z-index 1</div>
 </div>
</body>
</html>
```

Для блоків задається відступ від верхнього і лівого краю властивостями top і left. Так як другий блок оголошений у HTML-коді пізніше, він перекриває перший блок на сторінці.

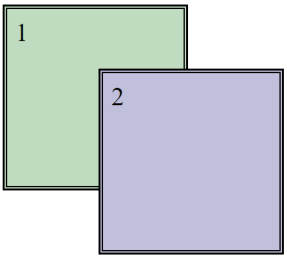


Рисунок 6.14. Використання абсолютного позиціонування

Для управління порядком накладення елементів один на одного необхідно використовувати властивість z-index. Значним z-index є позитивне або негативне число, що задає «висоту», на якій розташовано елемент. Елементи з великим z-index накладаються зверху елементів із меншим z-index. Щоб в

попередньому прикладі перший блок виявився вищим другого, необхідно для першого блоку задати z-index, наприклад, рівним двом, а для другого – одиниці.

```
<html>
<head>
<style>

#myDIV div{
  width:100px;
  height:100px;
  position:absolute;
  background-color:yellow;
  border:1px solid;
  opacity:0.5;
}
#myBox {
  position:absolute;
  background-color:blue!important;
  opacity:1;
  z-index: 2;
}
</style>
</head>
<body>

<div id="myDIV">
  <div id="myBox">1</div>
  <div style="top:20px;left:20px;z-index:0;">2</div>
 </div>

</body>
</html>
```

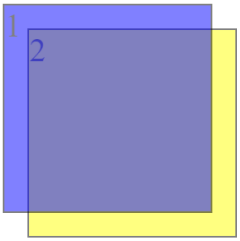


Рисунок 6.15. Використання z-index для зміни порядку накладання елементів

position: fixed – фіксує елемент щодо вікна. Елемент залишається на місці навіть при прокручуванні сторінки. На жаль, режим fixed не працює у браузері Internet Explorer версії 6 і нижче, тому застосовувати його не рекомендується.

Плаваючі елементи

У минулій лекції ми дізналися, що за замовчуванням блокові елементи йдуть строго один під одним. Змінити цей порядок можна зробивши елементи «плаваючими». Для цього служить CSS атрибут float. Він задає відносно якої сторони буде вирівнюватися елемент: лівої (left) або правої (right). Плаваючий елемент буде прагнути до лівої чи правої стороні батьківського елемента, а з інших сторін він може обтікати текстом або іншими елементами.

При цьому слід пам'ятати, що властивість float не працює одночасно із завданням позиціонування, розглянутим у першій частині лекції.

Наочно роботу float видно на прикладі:

```
<html>
<head>
<title>Плаваючі елементи</title>
<style> DIV#floating { float: left; }
</style>
</head>
<body>
Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem
ipsum.Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem
ipsum. <br>

<div id="floating"><img src="pic_trulli.jpg" width="120"
height="168"></div>
Lorem ipsum. Lorem ipsum.Lorem ipsum. Lorem ipsum. Lorem
ipsum.Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem ipsum. Lorem
ipsum. <br>
</body>
</html>
```

Контейнер DIV із зображенням прагне лівої сторони документа, а всі інші три сторони обтікає текст (рис. 6.16)

Lorem ipsum. Lorem ipsum.Lorem ipsum. Lorem ipsum.
Lorem ipsum.Lorem ipsum. Lorem ipsum. Lorem ipsum.
Lorem ipsum. Lorem ipsum.

Lorem ipsum. Lorem ipsum.Lorem ipsum.
Lorem ipsum. Lorem ipsum.Lorem ipsum.
Lorem ipsum. Lorem ipsum. Lorem ipsum.
Lorem ipsum.

Рисунок 6.16. Обтікання текстом блочного елемента

Створимо приклад з декількома плаваючими блоками. Задамо основний контейнер із фіксованою шириною, а в нього помістимо п'ять плаваючих блоків із вирівнюванням по лівому краю.

```
<html>
<head>
<title>Плавающие элементы</title>
<style>
#mmain {
border: double black 3px;
width: 150px;
height: 100px;
padding: 5px;
}

.lefty {
border: dashed black 1px;
width: 30px;
height: 30px;
float: left; margin: 5px;
text-align: center;
}
</style>
</head>
<body>
<div id="mmain">
<div class="lefty">1</div>
<div class="lefty">2</div>
<div class="lefty">3</div>
```

```
<div class="lefty">4</div>
<div class="lefty">5</div>
</div>
</body>
```

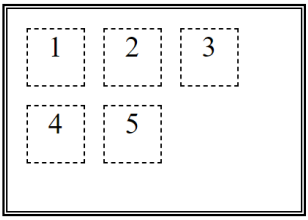


Рисунок 6.17. Кілька плаваючих блоків

Перший блок вирівнюється по лівому краю батьківського контейнера. Другий блок теж прагне до лівого краю, але так як місце вже зайнято першим блоком, другий блок стає (обтікає) праворуч від першого. Аналогічно поводиться і третій блок. Четвертий блок вже не може встати праворуч від третього, тому він поміщається нижче інших і вирівнюється по лівому краю. І нарешті, п'ятий блок обтікає четвертий праворуч.

Можна одночасно використовувати блоки з вирівнюванням по лівому і правому краю.

```
<div style="float: left">&larr; ліворуч</div>
<div style="float: right">праворуч &rarr;</div>
```

← ліворуч праворуч →

Рисунок 6.18. Блоки з вирівнюванням по різних краях

Ще однією властивістю, пов'язаною з плаваючими елементами, є clear. Clear забороняє обтікання елемента з лівої (left), правої (right) або з обох сторін (both). За замовчуванням значення – none – обтікання дозволено. Розглянемо приклад:

```
<style> DIV {
border: solid black 1px; width: 75px;
}
```

```
DIV.floating { float: left;
}
</style>
</head>
<body>
<div class="floating">Блок 1</div>
<div class="floating">Блок 2</div>
<div style="clear: both">Блок з забороною обтікання</div>
<div class="floating">Блок 3</div> <div class="floating">Блок
4</div> <div class="floating">Блок 5</div>
</body>
</html>
```

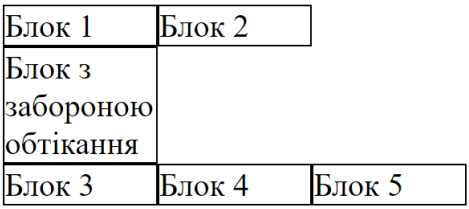


Рисунок 6.19 .Використання правила clear

При створенні сайтів плаваючі елементи, властивості float і clear часто використовуються для створення «каркаса» сторінок сайту.

7. ДИЗАЙН

7.1. Стандарт HTML5

Переваги HTML5

Інтернет зараз сильно відрізняється від того, що був у 1999 році, коли вийшло останнє велике оновлення HTML4. Смартфони, планшети та інші мобільні пристрої – це нові виклики для інженерів і розробників програм. Глобалізація зробила стандартизацію інтернет-технологій пріоритетним завданням, оскільки світове споживання Інтернету невпинно зростає, а популярні технології розвиваються швидше.

HTML5 покликаний надати зручний і послідовний перехід до нових стандартів. HTML5 сильно спростив процес створення вебдодатків та прибрав багато проблем щодо кросбраузерності та мультиплатформості. Завдяки HTML5 вебсторінки навчилися зберігати дані локально у браузерах користувачів, що дозволяє отримати контент швидше і безпечніше. Раніше, браузери використовували різні плагіни для програвання мультимедіа-файлів, на тепер вбудована підтримка аудіо і відео усуває проблеми із сумісністю.

Здебільшого сумісний із існуючими стандартами HTML4. Нові засоби розмітки працюють згідно існуючих, нові API ґрунтуються на існуючих JavaScript / DOM, які розробники використовуються протягом багатьох років.

Додає нові потужні засоби в HTML, які були раніше доступні у Веб тільки за допомогою технології плагінів, таких як Flash, або за допомогою складного коду JavaScript чи спеціальних прийомів.

Краще підходить для написання динамічних додатків, ніж попередні версії HTML.

Має чіткий алгоритм синтаксичного аналізу, тому всі браузери, які реалізують HTML5, створюють однакове дерево DOM із однією розміткою, що є величезним виграшем для сумісності.

Властивості HTML5

Мова HTML5 містить багато нових властивостей, що робить HTML значно більш потужним і зручним для створення вебдодатків.

Зміни щодо елементів мови

Застарілі елементи, які не вживаються: isindex, noframes, acronym, applet, basefont, dir, font, frame, frameset, big, center, strike, tt.

Нововведені елементи: summary, time, aside, audio, command, data, datalist, details, embed, wbr, figcaption, figure, footer, header, article, hgroup, bdi, canvas, keygen, mark, meter, nav, output, progress, rp, rt, ruby, section, source, track, video.

Семантичні елементи: <nav>, <header>, <footer>, <aside>, <article>.

Нове призначення старих елементів: <b> vs , <i> vs , <small>, <s>, <del>.

Нові види елементів управління: dates and times, email, number, range, tel, url, search и т. д.

Заміна елементів:

<applet>, <object> замінено тегом <embed> (використовується для втілення об'єктів);
<acronym> замінено тегом <abbr> (використовується для аббревіатур);
<bgsound> замінено тегом <audio> (для втілення програвача на сторінку);
<frame> замінено тегом <iframe>;
<listing>, <xmp> замінено тегами <pre> и <code> (для уставляння лістингів програм та кодів);
<plaintext> — замінено тегом <pre>.

Нові властивості форм

HTML5 надає стандартизований, простий спосіб реалізації таких властивостей, як вибір дати, повзунки і клієнтська перевірка – засіб перевірки заповнення полів форми з боку користувача.

Якщо користувач у полі, що призначене для введення електронної адреси зазначить номер телефону, то йому висвітиться попередження про помилку.

Аналогічно, у полі для номера телефону можна вводити лише цифри, інакше висвітиться помилка. Для такої клієнтської перевірки HTML5 використовує нові атрибути min, max, step, pattern, і required для введених даних.

Логічний атрибут required вказує браузеру, що форму можна відправити лише при заповненні даного поля, воно не може бути порожнім, а також містити допустимі типи значень.

Якщо обов'язкове поле є пустим, форма не відправиться. Браузер видасть користувачеві повідомлення із помилкою. Наприклад, «будь ласка, заповніть це поле» або «необхідно вказати значення».

Власна підтримка відео та аудіо

Протягом багатьох років відео і аудіо в Веб втілювалося за допомогою Flash-технологій. Технологія Flash стала популярною на початку 21 століття, оскільки відкриті стандарти не надавали для різних браузерів сумісний механізм реалізації таких речей. Як наслідок, різні браузери реалізували різні способи виконання одних функцій (наприклад, < object> і <embed>), роблячи тим самим весь процес дійсно складним. Flash надавав високоякісний, легкий спосіб відтворення відео в різних браузерах, яке останніми роками трасформувалось.

HTML5 містить елементи <video> і <audio> для простої реалізації власних відео і аудіо плеєрів за допомогою відкритих стандартів, і також містить API, що дозволяє легко реалізувати індивідуальні елементи управління плеєром.

API малювання на полотні

Елемент <canvas> і відповідний API дозволяють визначити на сторінці область для малювання, і використовувати команди JavaScript для малювання ліній, фігур і тексту, імпорту та маніпуляцій із зображеннями і відео, експорту в різні формати зображень, і багатьох інших речей. Canvas задає прямокутну область, що контролюється розробником і призначена для використання сценаріїв візуалізації двовимірних 2D-форм і растрових зображень.

Елемент <canvas> ідеально підходить для створення візуальних матеріалів, які покращують користувацькі інтерфейси, діаграми, фотоальбоми, схеми, графіки, анімовані матеріали і вбудовані графічні додатки.

Вебсховище Web Storage

До HTML5 єдиним способом локального зберігання даних було використання механізму файлів cookies. Файли cookies призначені для зберігання невеликих обсягів даних, але модель JavaScript для роботи з ними незручна.

В HTML5 запроваджено кращу альтернативу для файлів cookies, яка дозволяє легко і просто зберігати інформацію на комп'ютері відвідувача. Ця інформація може зберігатися на клієнтському комп'ютері необмежений час, не надсилається до вебсерверу (якщо тільки розробник не зробить це сам), може бути великого обсягу і для роботи з нею потрібно лише пара простих об'єктів JavaScript.

Ця можливість називається вебсховищем (Web Storage) і добре підходить для застосування автономного режиму роботи сайтів, оскільки дозволяє створювати автономні програми, які можуть зберігати всю необхідну їм інформацію, навіть за відсутності під’єднання до Інтернету.

Функціональність вебсховища HTML5 дозволяє вебсторінці зберігати дані на комп'ютері відвідувача. Ця інформація може бути короткочасною, яка видаляється після закривання браузера, або довготривалою, яка залишається доступною при наступному відвідуванні вебсторінки.

Автономні вебдодатки

HTML5 надає можливості для вебдодатків виконуватися в автономному режимі. Кеші додатків дозволяють зберігати копії всіх ресурсів, необхідних для локального виконання вебдодатків, а бази даних Web SQL – локальну копію даних вебпрограми. Спільно вони спроможні використовувати додаток, за відсутності з'єднання з Інтернетом. Згодом, коли мережа стане доступною, зміни синхронізуються з основною версією на сервері.

Web workers

Загальною проблемою вебдодатків є зменшення продуктивності, якщо потрібно обробити багато даних. В HTML4 всі JavaScript скрипти на сторінці могли виконуватися тільки по черзі. У деяких випадках користувачі повинні були чекати завершення виконання скрипта, щоб продовжити взаємодію з вебсторінками.

Завдяки технології Web Worker у HTML5 стало можливим створення та запуск скриптів, які будуть виконуватися у фоновому режимі і не впливати на швидкість завантаження основної сторінки. Потік Worker може виконувати завдання без втручання в користувацький інтерфейс.

Вебсокети WebSocket

Протокол http, за яким передаються дані, є синхронним, тобто побудованим за моделлю «запит-відповідь». WebSocket – це серйозне розширення HTTP5, яке дозволяє додаткам підтримувати багатокористувацьку взаємодію у режимі реального часу. Завдяки цьому клієнт і сервер стають рівноправними учасниками обміну даними. Кожен при цьому працює сам по собі, і за потребою надсилає дані до іншого. Інша сторона відповідає в міру необхідності, і тільки тоді, коли ця необхідність виникне.

WebSocket – протокол повнодуплексного двохбічного зв'язку поверх TCP-з'єднання, що призначений для обміну повідомленнями між браузером і вебсервером у режимі реального часу. WebSockets встановлює одне TCP-з'єднання і підтверджує, що сервер може спілкуватися з WebSocket, роблячи спеціальні перевірки. Після цього сервер і клієнт можуть надсилати текстові повідомлення через встановлене з'єднання за необхідності, в результаті чого зв'язок стає швидше. З'єднання постійно тримається відкритим, поки порт не буде закрито. При цьому у вебсокетах немає обмежень на кількість з'єднань.

Така спроможність істотно покращує ефективність вебдодатків, оскільки дані можуть безперервно передаватися між клієнтом та сервером без опитування сервера чи немає доступних оновлень, зайвих HTTP-заголовків та постійного перезавантаження сторінки.

Геолокація

Специфікація геолокації (*http://dev.w3.org/geo/api/spec-source.html*) визначає API, який дозволяє вебдодатку легко отримувати доступ до даних у будь-якому місці розташування, яке стало доступним, наприклад, за допомогою засобів GPS пристроїв. Це дозволяє надавати додатку різних корисних властивостей, що пов'язані з місцем розташування, наприклад, розмістити контент, який більше підходить для місця розташування.

Максимальна підтримка мобільних пристроїв

Швидке поширення мобільних пристроїв змусило скорегувати HTML-стандарти. HTML5 спрощує мобільну підтримку та враховує особливості смартфонів і планшетів.

При створенні документа HTML5 ми можемо використовувати два різні стилі: HTML і XML.

Стиль HTML

Стиль HTML передбачає наступні моменти:

- Початкові відкриваючі теги можуть бути відсутніми у елементів.
- Кінцеві закриваючі теги можуть бути відсутніми у елементів.
- Тільки порожні елементи (void elements) (наприклад, br, img, link) можуть закриватися за допомогою слеша />.
- Регістр назв тегів і атрибутів не має значення.
- Можна не укладати значення атрибутів у лапки.
- Деякі атрибути можуть не мати значень (checked і disabled).
- Спеціальні символи НЕ екрануються.

Документ повинен мати елемент DOCTYPE.

Це так званий «дозвільний» стиль, заснований на послаблення при створенні документа.

Стиль XHTML (XML)

Документ HTML5 також може бути описаний за допомогою синтаксису XML. Такий стиль ще називають "XHTML". Він використовується, якщо заголовок content-type має значення application / xml + xhtml. Для цього стилю характерні наступні правила:

- Кожен елемент повинен мати початковий відкриваючий тег.
- Непорожні елементи (non-void elements) із початковим відкриваючим тегом також повинні мати кінцевий закриваючий тег.
- Будь-який елемент може закриватися за допомогою слеша />.
- Назви тегів і атрибутів чутливі до регістру, як правило, використовуються у нижньому регістрі.

- Значення атрибутів повинні бути укладені в лапки.
- Атрибути без значень не допускаються (checked = "checked" замість просто checked).
- Спеціальні символи повинні бути екрановані.

7.2. Особливості стандартів дизайну

Типи дизайну сайтів

Швидкий розвиток пристроїв потребує сучасного підходу до розробки вебсайтів. Інтернет-користувачі переглядають сайти на різних пристроях із екранами різних розмірів. Розміри екранів постійно змінюються, тому, важливо, щоб сайт адаптувався до будь-якого з них.

Першим підходом до вирішення цієї проблеми було створення окремої версії сайту, яка розміщалася на окремому домені виключно для користувачів із різніми пристроями. На сьогодні розробники відмовляються від окремих версій вебсайтів і створюють єдину версію, яка працює і адаптується безпосередньо під всі пристрої: стаціонарні комп'ютери, ноутбуки, планшети або смартфони.

Існує два основні підходи створення сайтів, що легко адаптуються для різних типів пристроїв:

Responsive Design (Responsive Web Design, RWD) – чуйний дизайн – проєктування сайту з певними значеннями властивостей, наприклад, гнучка сітка макета, які дозволяють одному макету працювати на різних пристроях;

Adaptive Design (Adaptive Web Design, AWD) – адаптивний дизайн – проєктування сайту, що базується на кількох макетах фіксованої ширини.

Чуйний і адаптивний вебдизайн тісно пов'язані і вирішують одну задачу, але в різний спосіб. Основна відмінність між цими прийомами – чуйний дизайн – один макет для всіх пристроїв, адаптивний дизайн – різні макети для різних пристроїв. Їх поєднання забезпечує ідеальну формулу для функціональних сайтів.

Чуйний дизайн

Чуйний дизайн об'єднує три методики – гнучкий макет на основі сітки(fluid grid), гнучкі зображення (fluid images) і медіа запити. Пропорції та розміри елементів задаються у відсотках. При зменшенні ширини сторінки весь вміст плавно стискається, структурні елементи зменшуються відносно один одного. Так, наприклад, якщо вебсайт мав 3-х колонкову структуру, то на вузькому екрані він буде мати дві або одну колонки контенту.

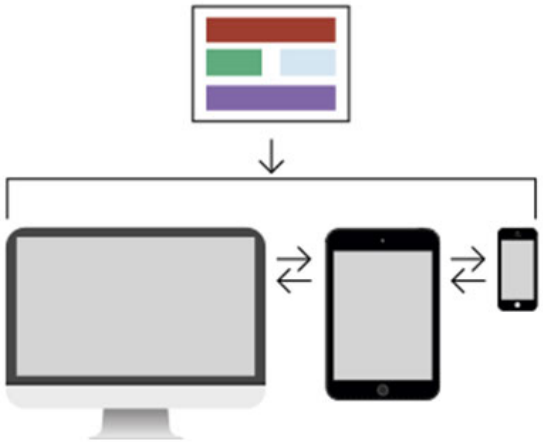


Рисунок 7.1. Макет для чуйного дизайну

Для створення чуйної версії вебсайту використовуються медіа запити (media queries) – блоки коду із вказуванням параметрів екрану. Медиа запити дозволяють застосовувати певні правила (стилі) для виведення контент-блоків у різний порядок та пропорції залежно від ширини екрана та можливостей пристрою, на якому відбувається перегляд сайту.

Адаптивний дизайн

На відміну від чуйного дизайну, адаптивний дизайн орієнтується на розміри пристроїв. Він використовує кілька статичних макетів для різних типів пристроїв, базуючись на контрольних (переломних) точках. Макети завантажуються при певних розмірах вікна браузера пристрою, а переходи між макетами відбуваються стрибкоподібно, а не плавно.

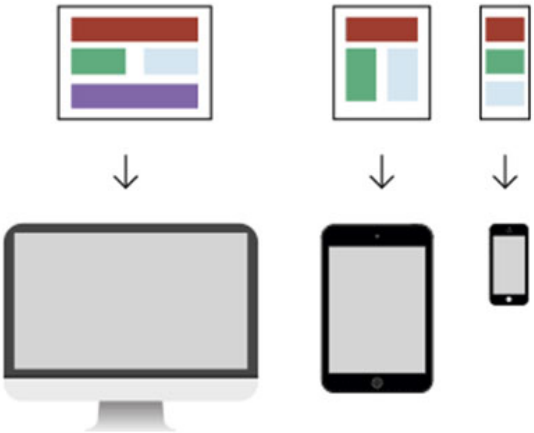


Рисунок 7.2. Макети для адаптивного дизайну

В адаптивному дизайні серверні скрипти спочатку визначають тип пристрою, за допомогою якого користувач намагається отримати доступ до сайту (настільний ПК, телефон або планшет), потім завантажує саме ту версію сторінки, яка найбільш оптимізована для нього. Для елементів сітки задаються фіксовані розміри.

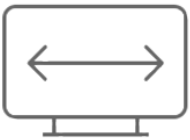
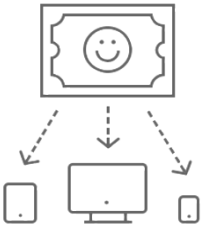
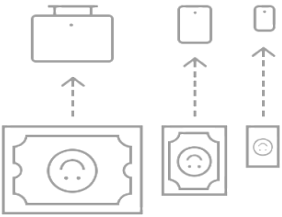
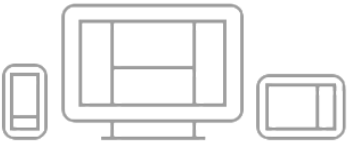
Порівняння чуйного та адаптивного дизайну

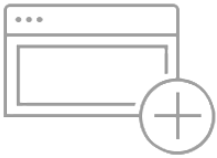
Чуйний дизайн більше підходять тоді, коли немає різниці між переглядом і використанням вебсайту на мобільному і десктоп пристрої, а також коли бюджет і термін розробки обмежені.

Адаптивні сайти найбільш затребувані тоді, коли важливою є швидкість завантаження сторінок і взаємодія користувача з мобільною версією сильно відрізняється від використання десктоп версії.

Кожен випадок створення мобільної версії сайту потрібно розглядати окремо. Для простоти прийняття рішення нижче наведено порівняльну таблицю використання чуйного і адаптивного дизайну.

Таблиця 7.1 Використання чуйного і адаптивного дизайну

Характеристики	Responsive Design	Adaptive Design
Визначення пристрою	 Використовуються медіа запити, що формують структуру сторінки щодо розмірів екрану пристрою, для всіх пристроїв використовується один і той же шаблон.	 Використовується серверне або браузерні визначення пристрою, після чого застосовується конкретний шаблон саме для цього типу пристрою.
Оптимізація контенту	 Завантажується весь контент, для всіх типів пристроїв, незалежно від того, використовується він чи ні.	 Завантажується підготовлений, і оптимізований контент під конкретний пристрій.
Адаптація під пристрій і його особливості	 Використовується один шаблон для всіх пристроїв. Використовуються однакові блоки, модулі, медіа файли, як для мобільного, так і для десктопного перегляду.	 Створюються різні унікальні шаблони, що призначені для різних пристроїв. Поведінка сайту змінюється залежно від типу пристрою.

Характеристики	Responsive Design	Adaptive Design
Швидкість завантаження	 Швидкість завантаження нижче, оскільки користувачу потрібно чекати, поки завантажитися весь контент і всі структурні елементи для всіх розмірів екрану.	 Швидкість завантаження вище, оскільки завантажується оптимізований контент, створений безпосередньо для певного пристрою і розміру його екрану.
Розробка і запуск	 Потрібна переробка основного сайту. При проєктуванні і розробці сайту, потрібно враховувати і оптимізувати взаємодію користувача відразу під всі можливі пристрої.	 Більш затратне, але без змін існуючого сайту. Адаптивні шаблони можуть розроблятися на базі існуючого сайту, без його зміни. Потрібно більш висока кваліфікація розробників, етап проєктування і дизайну триває довше, відповідно, створення такого сайту буде більш витратним.

7.3. Анімація у ВЕБдодатку

Вебанімація

Дизайн – це більше ніж візуальне представлення даних, це управління взаємодією користувача і вебсторінки. Як результат, мультимедійні об’єкти відіграють надзвичайно важливу роль у справі передачі інформації.

Проєктуючи сторінки, необхідно з самого початку пам'ятати про інтерактивну природу вебпростору і сприймати її прояви як природну частину дизайну.

HTML5 є платформою для реалізації не просто сторінок, а вебдодатків, що містять анімацію, графіку, відео і аудіо. Все це робить HTML однією з найпопулярніших мов, яка динамічно розвивається і вдосконалюється.

Взаємодія користувачів із сучасними вебсайтами серйозно зав'язано на анімації. Якісна і доречна анімація пожвавлює сторінку і сприяє кращому сприйняттю інформації. Вона здатна повідомляти про певні стани сторінки та привертати увагу. Анімація допомагає користувачеві побачити результат його дій і може впливати на його поведінку.

Анімація при завантаженні даних

Такий прийом створює відчуття того, що дія виконується швидше, ніж насправді. Анімована картинка свідчить, що прогрес завантаження відбувається і відвідувач легше сприймає цей час. Чим простіше анімація завантаження – тим краще. При цьому слід дуже обережно ставитися до будь-яких додаткових ефектів, на зразок звуків. Зазвичай вони просто не потрібні.

Рисунок 7.3. Приклад анімації при завантаженні даних

Анімація процесів

Анімація може показати виконання певної лінійної послідовності дій, що не залежить від користувача. Типовим прикладом є прогрес завантаження.

Рисунок 7.4. Приклад анімації процесів

Анімація покрокових операцій

Анімацію можна використовувати для процесів, які передбачають виконання користувачем покрокових операцій, щоб відображати послідовність виконаних дій.

Рисунок 7.5. Приклад анімації покрокових операцій

Анімація і каркасне відображення вебсторінок

Варіант каркасного відображення вебсторінок передбачає наявність простору, яке поступово заповнюється завантаженою інформацією. Поступове, не розірване періодами «порожнечі», наповнення сторінки, дає відчуття швидкості виконання дій. Цю техніку можна використовувати на практично будь-якому сайті разом із ненав'язливою анімацією процесу завантаження, що забезпечує утримання уваги користувача.

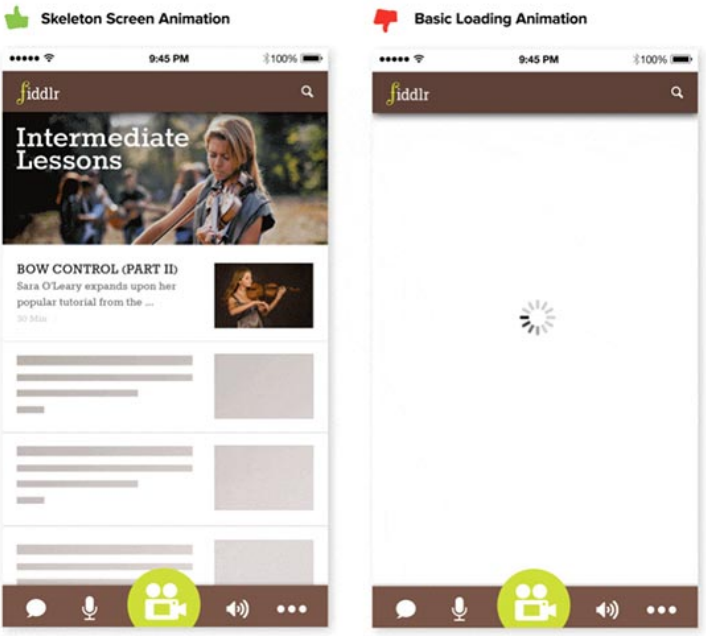


Рисунок 7.6. Приклад каркасной анімації

Каркасні екрани завантажують елементи користувацького інтерфейсу ще до того, як буде повністю відображено вміст сторінки.

Візуальний зворотний зв'язок

Анімована реакція на дії користувача

Хороший дизайн взаємодій із користувачем передбачає зворотний зв'язок, що повідомляє про результати взаємодії, робить його видимим і зрозумілим. Якщо відвідувач сайту не знає, з якими саме елементами сторінок можна працювати, чого від них можна очікувати, він буде відчувати себе розгубленим. Потрібно продумано спроєктувати варіанти взаємодії, зробивши їх чіткими і зрозумілими.

Анімація елементів управління для настільних і мобільних сайтів

Один з найпоширеніших прикладів візуального зворотного зв'язку – анімація при наведенні курсору на інтерактивний елемент.

Якщо користувач не впевнений у призначенні елемента управління, він намагається навести на нього курсор миші. Анімація елемента при наведенні курсору, на інтуїтивному рівні, повідомить користувача про те, що з об'єктом можна взаємодіяти.

На мобільних сайтах кнопки та інші інтерактивні елементи повинні бути оснащені візуальними підказками. Підказки повинні вказувати на те, як взаємодіяти до того, як користувач торкнеться елемента. Після торкання потрібно надати візуальний зворотний зв'язок, щоб засвідчити реакцію системи.

Рисунок 7.7. Приклад анімації при наведенні курсору на інтерактивний елемент

Привертання уваги за допомогою анімації

Зміна стану елементів на сторінці скеровує увагу користувача і підкріплює виконувані ним дії. Наприклад, при введенні даних в форму, поле підсвічується і користувач розуміє, що потрібно вводити текст.

Плавна зміна станів

Анімація переходів надає можливість плавно змінити колір фону, прозорість блоку, затемнення. Без анімації подібні зміни будуть із різким переходом, який відчувається як неприродний «стрибок». За допомогою анімації можна забезпечити плавні переходи між екранами. Добре спроєктовані переходи дають користувачам чітке розуміння того, на чому слід зосередитися.

Довгий скрол

Раніше дизайнери прагнули заповнити верхню область сторінки цінною інформацією. Тепер же «правило верхньої половини» втратило абсолютну істинність. Дизайнери застосовують красиві плавні ефекти для скролінгу сторінки як вниз, так і вгору. Це робить прокрутку простою і зручною.

Анімації оживляють процес скролінгу і приносять задоволення користувачеві для створення анімації.

7.4. Типи анімацій

Зображення, що створюються програмним шляхом

Є чотири способи створення зображення на вебсторінці програмним кодом: CSS, SVG, Canvas, JS-анімація.

CSS

Створення зображення на чистому CSS.

Замість програмного забезпечення для створення векторних ілюстрацій (такого, як Illustrator, Affinity Designer або Sketch) використовується безпосередньо CSS-код. Створення зображень на CSS може вплинути на креативність мислення і надихнути на підкорення нових вершин цієї мови.

Приклад CSS-зображень:

https://a.singlediv.com/

Анімація за домомогою CSS

Властивість CSS-animation дозволяє анімувати переходи від однієї конфігурації стилю CSS до іншої. Анімація складається з двох компонентів: стилю, що описує анімацію і набору ключових кадрів, які вказують початковий і кінцевий стани елементу, а також можливі проміжні точки.

Існують три ключові переваги використання CSS-анімації перед традиційними методами анімації, керованими скриптами:

Вони прості у використанні для простих анімацій; їх можна створювати не знаючи JavaScript.

Анімація працює добре, навіть при помірному завантаженні системи. Механізм рендеринга може використовувати пропуски кадрів і інші методи, щоб підтримувати оптимальну продуктивність.

Надання браузеру управління послідовністю анімації дозволяє йому оптимізувати продуктивність, наприклад, зменшуючи частоту оновлення анімацій, запущених на вкладках, які в даний час не відображаються.

Фонова картинка в CSS

Уставляння основною картинкою до елементу, псевдоелементу або контенту. В такому SVG-зображенні також не можна змінювати оригінальний вміст.

```
.picture {background-image:url (picture.svg);}
```

Цей спосіб підходить для оздоблювальної графіки, для якої не потрібна взаємодія: фони, іконки та інші нескладні зображення.

Уставляння через тег <object>

Третій спосіб – уставляння через тег object. У цьому випадку працюють скрипти, взаємодія, анімація – якщо вони описані всередині SVG. Між тегами object можна вставити код, що повідомить користувача про використання застарілої версії браузера.

```
<object type = "image/svg + xml"  data="picture.svg">
       <img src = "picture.png">
```

```
</object>
```

Замість тегу object можна використовувати тег iframe, що буде сприймати SVG-код як іншу сторінку. Але object працює краще і підлаштовується під розміри картинки.

За гнучкість доводиться платити: через те, що це вже не просто графіка і там можна додавати скрипти, до такого способу висуваються інші вимоги безпеки. Наприклад, картинку з іншого домену просто так вже не вставити.

Цей спосіб підходить, коли потрібно додати інтерактивну графіку: ігри, графіки, діаграми.

Зображення SVG

SVG – це формат масштабованої векторної графіки. У векторних форматах зберігається не саме зображення, а інструкція з його побудови по точках і кривих.

У растрових форматах інформація про конкретне число точок зображення щільно упакована в бінарний код. У нього марно заглядати і міняти його можна лише в редакторах.

Формат SVG також можна створювати і змінювати в редакторах, наприклад, Illustrator, Sketch або Inkscape. Але ще він текстовий, а значить його можна відкрити у любому текстовому редакторі.

```
<svg width = "20">
       <rect fill="#fc0" width="20" height="20" />
       <line stroke="black" x1="0" y1="0" x2="20" y2="20" />
</svg>
```

При втіленні SVG-файлу, насправді, уставляється не просто картинка, а ціла сторінка. Зі своєю системою координат, стилями, скриптами і особливостями. Існує чотири основних способи втілення SVG-зображень зі своїми особливостями.

Уставляння в елементі img HTML-коду.

Це самий ефективний спосіб завантажити будь-яку картинку.

```
<img src="picture.svg" alt="За склом">
```

В такому SVG-зображенні не працюватимуть скрипти і будь-які спроби взаємодії з елементами всередині не є успішними. Файл буде як за склом: дивитися можна, а чіпати ні. Хоча всередині все інше прекрасно працює, включаючи CSS-анімації.

Такий спосіб найкраще підходить для контентних зображень, яким не потрібна взаємодія: логотипи, графіки, схеми.

Пряме уставляння вмісту SVG-файлів на сторінку

Цей спосіб став можливим при впровадженні стандартів HTML5 і вміст SVG-файлів тепер можна вставляти прямо на сторінку, як будь-які інші теги.

```
<h1>Квадрат</h1>
<svg>
       <rect fill="#fc0" width="120" height="120" />
</svg>
```

З таким SVG-елементом можна маніпулювати як зі звичайними HTML-елементами: стилі, скрипти. Наприклад, можна змінювати колір заповнення при наведенні і описувати все в загальних стилях.

```
<style>
rect:hover {fill:#090;}
</style>
```

Нажаль, такі зображення не можна зберігати.

Canvas

HTML-елемент <canvas> створює область на вебсторінці, яку можна використати для відтворення графіки в режимі реального часу за допомогою сценаріїв (наприклад, за допомогою мови JavaScript). Він може бути використаний для створення онлайн-ігор, годинників, анімації тощо.

Додавання на сторінку

Елемент <canvas> є звичайним HTML-елементом:

```
<canvas id="test" width="600" height="200"> </canvas>
```

Цей елемент має три атрибути: id, width і height. Атрибут id (ідентифікатор) надає елементу унікальне ім'я – test, це потрібно для того, щоб мова сценаріїв

могла отримати до нього доступ. Атрибут width визначає кількість пікселів по горизонталі, яке <canvas> буде займати на вебсторінці. Аналогічним чином height визначає вертикальну займану область. Таким чином, браузер зможе виділити простір потрібного розміру на сторінці, відповідно до зазначених значень.

За замовчуванням полотно є прозорим і не має рамки, тобто є порожнім простором у вікні браузера, в якому можна малювати.

Анімація є важливою складовою дизайну вебсайту і у вебдизайнера є широкі можливості вибору багатьох технологій, які допоможуть у досягненні конкретних цілей, що ставляться перед сайтом. Тип анімації і програмного забезпечення, необхідного для її створення визначається в залежності від конкретного призначення.

GIF-анімація

Анімація з використанням формату GIF (Graphic Interchange Format) є простою в реалізації і відтворенні на сторінках. Анімовані GIF-зображення працюють як традиційна покадрова анімація, яку застосовують у мультфільмах. Анімоване зображення містить певне число кадрів, де кожен кадр є частиною анімації, також задаються параметри, які дозволяють регулювати швидкість і тривалість анімації. Програма, за допомогою якої створюється GIF-анімація (наприклад, PhotoShop) зберігає кадри та відмічені опції в GIF-файлі, а браузер інтерпретує задані параметри і відтворює анімацію, створюючи зоровий образ руху.

APNG. Анімований PNG

aPNG (Animated Portable Network Graphics) запропоновано в 2004 році компанією Mozilla. Новий стандарт базується на форматі PNG, додано можливість анімації та покращено алгоритми стиснення. Втім компанія, що підтримує формат PNG не визнала новий стандарт, і aPNG не пішов в реліз. Основна причина відмови звучала так «PNG – це формат для нерухомих зображень».

До 2008 року не було жодних спроб широкого впровадження aPNG. В 2008 році компанія Mozilla вносить його у свій браузер Firefox, пізніше таку підтримку додано у браузер Opera. З приходом HTML5 ситуація почала змінюватися. Google, Twitter, Facebook і інші популярні сервіси, стали один за одним

відмовлятися від застарілих технологій, і переходити на нові, паралельно, задаючи стандарти розробки.

WebP

Сучасний формат зображень з відкритим вихідним кодом, розроблений компанією Google спеціально для Інтернету, є відгалуженням відеоформату WebM.

Подібно до APNG, він також підтримує 24-бітні зображення, 8-розрядну прозорість. У ньому є стиснення з втратами та без втрат, що у деяких випадках дозволяє досягти невеликого розміру файлів із забезпеченням високої якості, що робить його універсальним. Формат поєднує переваги JPG і PNG без збільшення розміру файлу. Сьогодні YouTube використовує перетворення мініатюр для відео у формат WebP.

Поки формат підтримується лише браузером Chrome і має певні недоліки щодо відображення дрібних деталей. Для підтримки в інших браузерах існують способи обходу обмежень (додаткові плагини або конфігурація), але вони не сприяють використанню формату повсюдно.

FLIF Free Lossless формат зображення

FLIF – це новий та інноваційний формат зображення, що підтримує анімацію із альфа-прозорістю, прогресивне завантаження (що дозволяє відображати зображення в меншій якості, поки він довантажується), а також інші можливості у порівнянні з різними форматами. Формат поки не підтримується всіма браузерами, але розробляються JavaScript-плагіни, що дозволяють використовувати FLIF у будь-якому браузері.

JavaScript

Створювати анімацію за допомогою JavaScript складніше, ніж писати переходи або анімацію засобами CSS, проте, JavaScript, як правило, надає розробнику ширші можливості. Зазвичай, використовується функція requestAnimationFrame, а потім на кожному кадрі анімації вручну визначається значення кожної властивості анімованого елемента.

Оскільки анімація засобами JavaScript дозволяє повністю контролювати стилі елементів на кожному кроці, анімацію можна уповільнювати, зупиняти, відтворювати у зворотному напрямку і виконувати інші маніпуляції.

7.5. Формати графічних файлів для Веб

У підготовці зображень для вебсторінок варто бути обізнаним у популярних графічних форматах та їх властивостях.

На щастя, сьогодні немає такого різноманіття форматів, як на початку 90-х, коли кожна компанія-виробник графічних редакторів створювала свої файлові формати, проте кожен із залишившихся на сьогодні форматів пройшов природний відбір, довів свою життєздатність і користь. Вони мають характерні особливості і можливості, що роблять їх незамінними в роботі.

Комп’ютерні графічні файли можна поділити на дві великі гілки: растрову і векторну.

Векторні файли представляють математичний опис об'єктів відносно точки відліку координат. Наприклад, для того, щоб відобразити пряму потрібно вказати координати двох точок, які об'єднаються за коротшим шляхом, для дуги задається радіус тощо. Таким чином, векторне зображення є набором геометричних примітивів. Більшість векторних форматів можуть містити втілені у файл растрові об'єкти. Складність при переведенні чи перенесенні даних із одного векторного формату до іншого полягає у використанні в програмах різних алгоритмів, різних математичних формул для побудови векторних примітивів та опису растрових об'єктів.

Растровий файл влаштовано простіше. Він представляє прямокутну матрицю (*bitmap*), що поділена на пікселі. Растрові формати різняться між собою здатністю містити додаткову інформацію: різні колірні моделі, вектори, альфа-канали, прошарки різних типів, інтерліньяж (черезрядкове підвантаження), анімацію, можливості стиснення тощо.

Растрові вебформати

GIF (CompuServe Graphics Interchange Format)

Апаратно незалежний формат GIF розроблено в 1987 році (GIF87a) фірмою CompuServe для передачі растрових зображень по мережах. У 1989-му формат модифіковано (GIF89a), до нього додано підтримку прозорості та анімації.

GIF використовує LZW-компресію, що дозволяє добре стискати файли, в яких багато ділянок із однорідним заповненням (логотипи, написи, схеми).

Метод стиснення LZW (Lempel-Ziv-Welch) розроблено в 1978 році ізраїльтянами Лемпелом і Зівом і допрацьовано пізніше в США. Принцип стискання полягає у пошуку однакових послідовностей – фраз у всьому файлі. Виявлені послідовності зберігаються в таблиці, їм привласнюються короткі маркери – ключі. Так, якщо в зображенні є набори з рожевого, оранжевого і зеленого пікселів, що повторюються 50 разів, LZW виявляє цей набір, привласнює йому окреме число (наприклад, 7) і зберігає ці дані 50 разів у вигляді числа 7.

Метод LZW ефективно діє на ділянках однорідних, вільних від шуму кольорів, добре стискає довільні графічні дані, але процес кодування і розпаковування відбувається відносно повільно.

GIF дозволяє записувати зображення «через рядок» (Interlaced), завдяки чому, маючи лише частину файлу, можна побачити зображення цілком, але з меншою роздільністю. Це досягається за рахунок завантаження, спочатку 1, 5, 10 і далі рядків, за другим проходом підвантажуються 2, 6, 11 рядки, і згодом зображення набуває початкового вигляду. Черезрядковий запис дещо збільшує розмір файлу, але надає більшої зручності для користувачів.

У GIF можна застосовувати прозорі ділянки, вони лишаються прозорими в браузерах та інших програмах і через них просвічується фоновий колір. Прозорість забезпечується за рахунок додаткового alpha-каналу, що зберігається разом із файлом.

Файл GIF спроможні містити кілька растрових картинок, які браузери підвантажують одну за іншою із вказаною у файлі частотою. Так досягається ілюзія руху (GIF-анімація).

Особливості

Кількість кольорів у зображенні може бути від 2 до 256, але це можуть бути будь-які кольори з 24-бітової палітри.

Файл у форматі GIF може містити лише 100% прозорі ділянки. Якщо використовується відмінний від білого кольору фон, він буде просвічуватися крізь прозорі ділянки в зображенні.

GIF підтримує покадрову зміну зображень, що робить формат популярним для створення банерів і простої анімації.

Використовує вільний від втрат метод стиснення.

Область застосування

Текст, логотипи, ілюстрації з чіткими краями, анімовані малюнки, зображення з прозорими ділянками, банери.

JPEG (Joint Photographic Experts Group)

JPEG – це є назва формату та алгоритму стиснення, який засновано не на пошуку однакових елементів, як у LZW, а на різниці між пікселями.

Кодування даних відбувається у кілька етапів. Спочатку графічні дані конвертуються в колірний простір типу LAB, відкидається половина або три чверті інформації про колір (залежно від реалізації алгоритму).

Далі аналізуються блоки 8x8 пікселів. Для кожного блоку формується набір чисел. Перші кілька чисел представляють колір блоку в цілому, в той час, як подальші числа відображають тонкі деталі.

На наступному етапі, залежно від обраного рівня якості, відкидається певна частина чисел, що представляють тонкі деталі. На останньому етапі використовується кодування за методом Хафмана для ефективного стиснення кінцевих даних. Відновлення даних відбувається у зворотному порядку.

Метод стиснення Хафмана (Huffman) розроблено в 1952 році і використовується як складова частина в ряді інших схем стиснення, в тому числі і у LZW. У методі Хафмана аналізується набір символів для визначення частоти кожного символу. Для символів, що найчастіше зустрічаються,

використовується позначення у вигляді мінімальної можливої кількості бітів. Наприклад, найчастіше в англійських текстах зустрічається буква «е». Використовуючи кодування Хафмана можна представити літеру «е» лише двома бітами (1 і 0), замість вісьмох бітів, необхідних для представлення букви «е» в кодуванні ASCII.

Таким чином, чим вище рівень компресії, тим більше даних відкидається, тим нижчою є якість. Використовуючи JPEG можна отримати файл в 2-500 разів менше, ніж BMP.

Формат JPEG є апаратно незалежним, повністю підтримується на PC і Macintosh. JPEG відтворює спектр кольорів TrueColor (2^{24}).

JPEG краще стискає растрові картинки фотографічної якості, ніж логотипи або схеми – в них більше півтонових переходів, у той час серед однотонних заливок з'являються небажані переходи.

Краще, і з меншими втратами стискаються великі зображення для Веб із високою роздільністю (200-300 dpi і більше), ніж із низькою (72-150 dpi), оскільки в кожному квадраті 8x8 пікселів переходи виходять м'якшими, за рахунок того, що таких квадратів є більше.

Даний формат називають стисненням із втратами, оскільки алгоритм JPEG вибірково відкидає дані. Не бажано зберігати у JPEG-форматі будь-які зображення, де важливими є тонкі нюанси кольорів, оскільки під час стиснення відбувається відкидання колірної інформації. У JPEG слід зберігати лише кінцевий варіант роботи, оскільки кожне повторне збереження призводить до нових втрат (відкидання) даних і початкове зображення може бути вкрай зіпсованим.

Формат JPEG не підтримує прозорість і при збереженні зображення з прозорими ділянками, вони зафарбовуються у певний колір.

Особливості

У зображенні може бути понад 16 мільйонів кольорів, що цілком достатньо для збереження фотографічних зображень.

Основною характеристикою формату є якість, яка визначає кінцевий розмір файлу. Слід пам’ятати, що формат застосовує стиснення з втратами. Чим вище стиснення, тим менше якість і навпаки.

Підтримка технології прогресивного JPEG. Спочатку у вікні перегляду з'являється версія зображення з низькою роздільністю, яке при повному завантаженні поступово набуває початкового вигляду.

Область застосування

Використовується переважно для фотографій. Не є доцільним для зображень з прозорими ділянками, великими одноколірними ділянками.

PNG (Portable Network Graphics)

PNG – Інтернет формат, який долає обмеження GIF. Використовує стиснення без втрат Deflate, подібне до LZW. Стиснуті індексовані файли PNG, зазвичай, є меншими за аналогічні GIF.

Глибина кольору може бути любою, до 48 біт. Використовується двохвимірний interlacing (не лише рядків, але і стовпців), що, подібно до GIF, дещо збільшує розмір файлу. На відміну від GIF, де застосовується 100% прозорість, PNG підтримує також напівпрозорі пікселі (в діапазоні прозорості від 0 до 99%) за рахунок альфа-каналу з 256 градаціями сірого.

У файл формату PNG записується інформація про гамму. Гамма є певним числом, що характеризує залежність яскравості світіння екрану монітора від напруги на електродах кінескопа. Це число обчислюється з файлу і дозволяє вводити поправку яскравості при відображенні. Воно потрібне для однакового відображення інформації незалежно від апаратної платформи комп’ютера. PNG підтримується у всіх сучасних браузерах.

PNG-8

PNG-8 — формат подібний до GIF і має покращений формат стиснення даних.

Особливості

Використовує 8-бітову палітру (256 кольорів) у зображенні, за що і отримав в своїй назві цифру вісім. При цьому можна вибирати, скільки кольорів буде задіяно у файлі — від 2 до 256.

На відміну від GIF, не відображає анімацію.

Область застосування

Текст, логотипи, ілюстрації з чіткими краями, зображення з градієнтною прозорістю.

PNG-24

PNG-24 — формат, аналогічний до PNG-8, але використовує 24-бітову палітру кольору. Подібно до формату JPEG, зберігає яскравість і відтінки кольорів у фотографіях. Подібно до форматів GIF і PNG-8, зберігає деталі зображення та прозорість.

Особливості

Використовує понад 16 млн кольорів, тому застосовується для повнокольорових зображень.

Підтримує багаторівневу прозорість, це дозволяє створювати плавний перехід від прозорої області зображення до колірної, так званий градієнт.

Алгоритм стиснення зберігає всі кольори і пікселі в зображенні незмінними. Якщо порівнювати з іншими форматами, то в PNG-24 кінцевий об'єм графічного файлу виходить найбільшим.

Область застосування

Фотографії, малюнки, що містять прозорі ділянки, малюнки з великою кількістю кольорів і чіткими краями зображень.

Векторні вебформати

SVG. Масштабована векторна графіка

SVG (Scalable Vector Graphics) – це тип векторних файлів, що описують зображення у форматі XML. Формат з'явився у 2001, однак популярність серед веброзробників він отримав нещодавно, після впровадження підтримки у сучасні браузери. Формат є відкритим стандартом, на відміну від більшості інших форматів, SVG не є чиєюсь власністю.

Файл із зображенням у цьому форматі є звичайним текстовим файлом, який можна відкрити в блокноті і відредагувати. У цьому форматі можна описати не тільки статичну, а й динамічну картинку (анімація), змішати створені вектори з растровою картинкою. Завдяки тому, що кожна фігура для браузера є елементом DOM, за допомогою JavaScript можна описувати досить складні сценарії, взаємодіяти з користувачем.

Розмір об'єктів SVG є меншим за розмір растрових зображень, а самі зображення не втрачають якості при масштабуванні. На відміну від растрових форматів можна взаємодіяти із зображеннями у форматі SVG – за допомогою CSS можна змінювати параметри графіки: колір, прозорість або межі, а за допомогою JavaScript – анімувати зображення.

У SVG є маса функцій, які роблять цей формат рекомендованим для Вебу, особливо якщо SVG використовується для простих зображень типу логотипів, карт, іконок, маркерів.

Переваги формату SVG

SVG часто важать менше за растрові зображення.

Формат масштабується, що забезпечує чіткість за любої роздільності екрану.

SVG-код можна помістити в HTML і налаштовувати через CSS.

SVG-зображення можна анімувати, в тому числі окремі частини, як за допомогою CSS, так і JS.

Втім занадто складні SVG-зображення збільшують розмір файлу. SVG не застосовний до фотографій, тут краще підійдуть формати JPG і webP.

Інші формати графічних файлів

PSD (Adobe Photoshop Document)

Внутрішній формат популярного растрового редактора Photoshop останнім часом підтримується великою кількістю програм. Він дозволяє записувати зображення з багатьма прошарками, їх масками, додатковими альфа-каналами і каналами базових кольорів, контурами та іншою інформацією.

Для стиснення застосовують метод RLE (Run Length Encoding), кодування із змінною довжиною рядка. Дія методу полягає у пошуку однакових пікселів у одному рядку. Якщо в рядку, наприклад, є 3 пікселя білого кольору, 21 – чорного, 14 – білого, то застосування RLE надає можливість не запам'ятовувати кожен з них (38 пікселів), а записати як 3 білих, 21 чорний і 14 білих у першому рядку.

Подібно до методу LZW, алгоритм RLE добре працює зі штучними й ілюстрованими картинками і гірше з фотографіями. У випадку, якщо фотографія має багато дрібних деталей, RLE може навіть збільшити розмір файлу.

TIFF (Tagged Image File Format)

Апаратно незалежний формат TIFF на сьогоднішній день є одним із найпоширеніших і надійніших, його підтримують практично всі програми на PC і Macintosh, що пов’язані з графікою. TIFF є кращим вибором при імпорті растрової графіки у векторні програми та видавничі системи. Йому доступно весь діапазон колірних моделей від монохромної до RGB, CMYK і додаткових кольорів Pantone. TIFF може зберігати контури, альфа-канали та інші додаткові дані.

TIFF має два різновиди: для Macintosh і PC. Це пов'язано з тим, що процесори Motorola читають і записують числа зліва направо, а процесори Intel – навпаки. Сучасні програми можуть без проблем використовувати обидва варіанти формату. Зазвичай, дані у форматі TIFF не стискаються, але може бути використано LZW-стиснення.

BMP (Windows Device Independent Bitmap)

Рідний формат Windows, який підтримується всіма графічними редакторами, що працюють під управлінням цієї операційної системи. Застосовується, в основному, для збереження растрових зображень, що призначені для використання у Windows. Може зберігати як індексовані (до 256 кольорів), так і RGB-кольори (понад 16 млн відтінків). Можливе застосування стиснення за алгоритмом RLE.

PDF (Portable Document Format)

PDF запропоновано фірмою Adobe, як платформо-незалежний формат для створення електронної документації, презентацій, передачі верстки та графіки через мережі.

Односторінкові файли PDF відмінної якості може створювати Photoshop. Багатосторінкові PDF можуть створювати програми Adobe Acrobat, PageMaker і програми пакету MS Office.

PDF спочатку проєктувався як компактний формат для електронної документації. Тому, всі дані в ньому можуть стискатися, причому, до різного типу інформації застосовуються різні, найбільш прийнятні для них типи стиснення: JPEG, RLE, CCITT, ZIP. Програма Acrobat дозволяє розставляти гіперпосилання, заповнені поля, додавати у файл PDF відео чи звук, інші дії.

Файл PDF може бути оптимізованим. Із нього видаляються елементи, що повторюються, встановлюється посторінковий порядок завантаження сторінок через Веб, із пріоритетом спочатку для тексту, потім для графіки. Якщо елементів, що повторюються, немає, файл, після оптимізації, як правило, дещо збільшується.

PDF найчастіше використовується для передачі по мережах графіки і зверстаного тексту в компактному виді. Він може зберігати всю інформацію для пристрою виведення, яка була у початковому файлі.

WMF (Windows Metafile)

Векторний формат WMF є рідним форматом Windows і використовує його графічну мову. Призначений для передачі векторних даних через буфер обміну (Clipboard). Розпізнається практично всіма програмами Windows, так чи інакше пов'язаними з векторною графікою. Користуватися форматом WMF варто лише у випадках передачі «чистих» векторів. WMF спотворює колір, не зберігає певні параметри, які привласнюються об'єктам у різних векторних редакторах, не містить растрові об'єкти, не розпізнаєтьс я багатьма програмами на Macintosh.

AI (Adobe Illustrator Document)

Adobe Illustrator – популярний графічний редактор від Adobe. Векторний формат Illustrator можна безпосередньо відкрити у Photoshop, його підтримують майже всі програми Macintosh і Windows так чи інакше пов'язані з векторною графікою і графікою взагалі.

Формат Illustrator є найкращим посередником при передачі векторів із однієї програми в іншу, з PC на Macintosh і навпаки. Втілені або пов'язані з документом растрові файли при обміні через формат Illustrator втрачаються.

CRD (CorelDraw Document)

Векторний формат, що має незаперечне лідерство на платформі PC. Багато програм на PC (FreeHand, Illustrator, PageMaker тощо) можуть імпортувати файли CorelDraw.

У файлах застосовується окрема компресія для векторів і растрів, можуть втілюватися шрифти, файли CorelDraw мають величезне робоче поле 45x45 метрів (цей параметр є важливим для зовнішньої реклами), підтримується багатосторінковість.

7.6. Стиль дизайну

Для вибору стилю дизайну, варто враховувати особливості проєкту, для якого розробляється сайт:

Тематика майбутнього проєкту.

Цільова аудиторія – вікова група, основна соціальна та статева приналежність, освіта, коло інтересів.

Очікувані емоції від дизайну сайту.

Кожен проєкт є особливим, і до його розробки потрібно підходити індивідуально. Для створення успішного проєкту потрібен нестандартний підхід, із помірною часткою креативу. Сучасні основні стилі вебдизайну надають можливість реалізовувати творчі ідеї, створювати і використовувати нові свіжі стилі. Тренди вебмоди швидкоплинні і тому перед розробкою сайту

варто ознайомитися з останніми новинками, стильними і колірними тенденціями, поданням інформації, сучасними засобами зв'язку тощо.

Колірні схеми

Від вибору колірної гами безпосередньо залежить успішність сайту і позитивне сприйняття відвідувачами. Вибір колірної схеми сайту не можна робити навмання, слід звернути пильну увагу на вдале поєднання кольорів, що відповідають бізнесу, брендингу або особі. Знайти ідеальне поєднання кольорів може стати складним завданням для певного дизайну, що відповідатиме вимогам клієнта і в той же час бути в тренді.

Колірна схема має вибиратися на підставі:

Особливостей фірмового стилю. Оздоблення сайту повинно відповідати його тематиці та фірмовому стилю компанії. Наприклад, для солідного корпоративного сайту дизайн повинен бути строгим, лаконічним. І навпаки, якщо сайт присвячено організації свят, то яскраві та веселі кольори швидше сподобаються потенційному клієнту.

Фізіологічних і психологічних особливостей цільової аудиторії. Комфорт для відвідувача створює умови для позитивного сприйняття інформації. Вдало підібрані кольори задають бажаний настрій і сприяють запам'ятовуванню вебресурсу. Кольори сайту не мають бути дратівливими і не повинні напружувати очі. Потрібно забезпечити достатній контраст між фоном та звичайним текстом і не використовувати надмірну кількість кольорів.

Розуміння колірної психології буде відігравати важливу роль в успішному дизайні сайту. Колірна психологія – це теорія, що пояснює, як певні кольори впливають на свідомість та переконують людей у прийнятті певних рішень. Згідно з дослідженнями, люди приймають підсвідомі рішення щодо продуктів протягом перших 90 секунд перегляду. І на 90% ці рішення засновані на кольорі. Наприклад, червоний колір є одним з небагатьох кольорів, здатних миттєво привернути увагу людини, тому великі написи «Розпродаж» у магазинах завжди пофарбовані в червоний колір.

Для дизайнерів існує мільйони відтінків на вибір, які можна комбінувати. Але для вибору правильних колірних комбінацій для дизайну сайту існує багато чинників. Врахування цього сприяє вибору колірної схеми, яка протримається

протягом тривалого часу, і буде пасувати як до тематики так до очікувань аудиторії.

Поетапний вибір кольорів для сайту (основний, вторинний та фоновий кольори):

Основний колір, як правило, використовується в більшій мірі, ним можуть виділятися основні заголовки чи важлива інформація.

Вторинний колір, яким буде виділятися другорядна за важливістю інформація.

Акцентований колір – важливий інструмент для привертання уваги відвідувача. Повинен бути контрастним, виділятися і на основному і на фоновому кольорі.

Фоновий колір, який переважає на сайті, заповнюючи вільний простір. Повинен взаємодіяти зі всіма вибраними відтінками, не відволікати на себе увагу.

Існують класичні колірні схеми, які допоможуть у виборі гармонійно поєднаних кольорів. Якщо розглядати колірні схеми, то їх є кілька: монохромні (відтінки одного основного кольору), аналогічні (три кольори, що знаходяться поруч на колірному колі), взаємодоповнюючі (з двох кольорів, що знаходяться на протилежних сторонах колірного кола), додаткові розщеплені, тріада та тетраедр.

Лідером для підбору колірних схем є сервіс **Adobe Color CC.** Даний ресурс дозволяє створювати палітри на основі обраного кольору чи завантаженого зображення. Для використання представлено великий архів палітр інших користувачів.

Гарячі кольори у вебдизайні викликають почуття тепла, пристрасті. Вони агресивні і сміливі, привертають увагу, тому їх використовують для критично важливих повідомлень, іконок та іншого.

Холодні кольори привносять прохолоду, підходять для витриманих тематик, нічних, сумних або водних. Вони неагресивні і заспокійливі, несуть строгість і холод.

Монохромні кольори
Використовується один основний колір і найближчі до нього відтінки.

Аналогічні кольори
Використовується один основний колір і суміжні з ним кольори.

Рисунок 7.1. Схеми монохромні та аналогічні кольори

Взаємодоповнюючі кольори
Використовуються діаметрально протилежні кольори згідно спектрального кола

Додаткові розщеплені
Використовуються діаметрально протилежні кольори і колір, суміжний з одним із них

Тріада
Використовуються три діаметрально протилежних кольори згідно спектрального кола (120 градусів)

Тетраедр
Використовуються чотири діаметрально протилежні кольори згідно спектрального кола (90 градусів)

Рисунок 7.2. Складні кольорні схеми

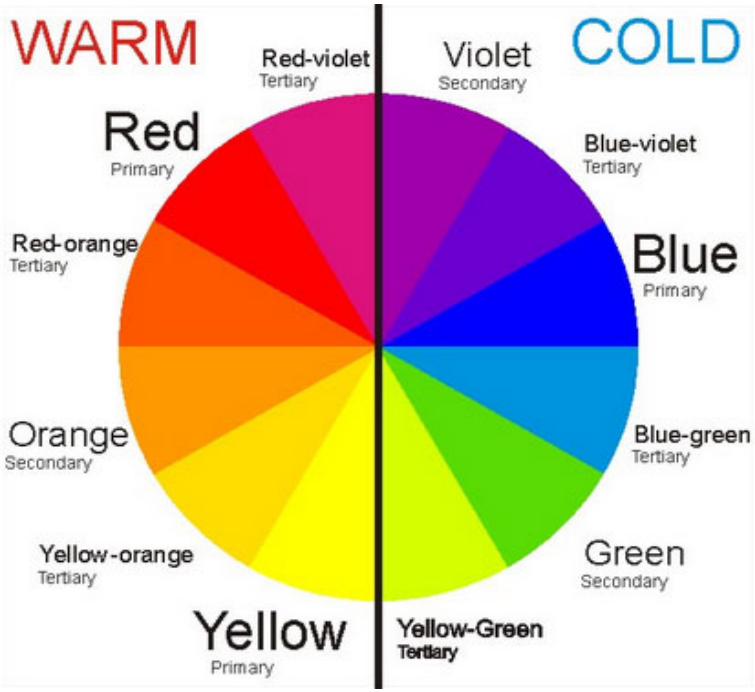


Рисунок 7.3. Шкала поділу кольорів на теплі та холодні

Температура. Всі існуючі кольори можна розділити на три групи з точки зору температури: теплі (червоний, оранжевий, жовтий), холодні (зелений, синій) і нейтральні (чорний, білий). Використання кількох тонів, що є різними за температурою (наприклад, червоний і білий) дозволить створити достатньо динамічний, але логічно комбінований контраст.

Відтінки і тіні. Відтінки утворюються додаванням білого до кольорового, а тінь навпаки, додаванням чорного. Відтінки і тіні дозволяють створити монохромні кольори і палітри, шляхом додавання різної кількості білого і чорного.

Насиченість, тон, світло. Кольори також розподіляються за насиченістю, тоном і освітленості (рис. 7.4).

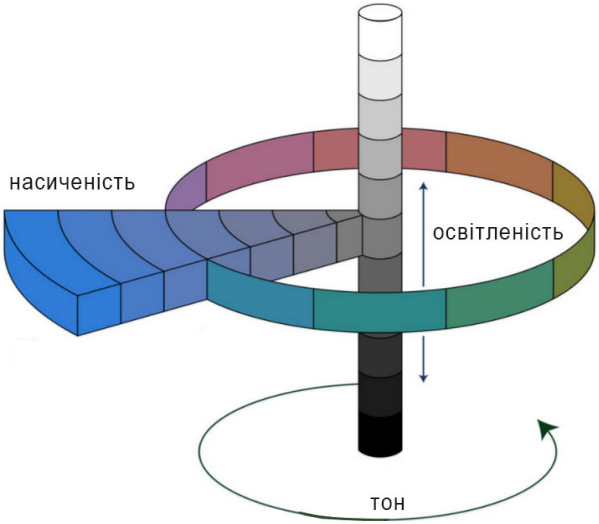


Рисунок 7.4. Насиченість, тон та освітленість кольорів

Насичення дозволяє зробити колір багатшим і темнішим. Поняття «світло-блакитний» або «темно-зелений» – має на увазі насиченість.

Тон визначає відмінність від основного кольору (їх сім – за кольорами райдуги). Тобто, наприклад, «зеленувато-блакитний».

Освітленість визначає колір по відношенню до чисто білого, наскільки він яскравий або тьмяний.

Контрастність. При підборі кольорів для палітри завжди варто вибирати як мінімум 2 контрастних кольори. Досягнення потрібного контрасту між кольорами є необхідною умовою для того, щоб вийшов хороший дизайн.

Поради щодо вибору палітри кольорів для сайту:

До вибору колірної палітри для сайту слід підходити дуже серйозно. Не варто спиратися на власні бажання. Слід враховувати смаки цільової аудиторії, за допомогою кольорів поєднувати їх потреби з цілями сайту.

Для створення читабельного тексту використовуються контрастні кольори, що приверне увагу потенційного клієнта.

Не варто зловживати надмірною кількістю кольорів, краще підбирати поєднані між собою відтінки.

Для вибору гармонійної палітри допоможуть спеціальні сервіси та приклади оздоблення дизайнів популярних сайтів.

Колірна схема сайту створює загальний настрій подачі ресурсу, і, як наслідок, впливає на популярність ресурсу, прибуток та конверсію ресурсу.

Генерація ідей дизайну

Як і мода, вебдизайн слідує за модними трендами, тому, дизайнер має бути обізнаним щодо останніх тенденцій у вебдизайні. Як і при знайомстві з новою людиною, тут важливо справити хороше перше враження.

Простота і акуратність. Реклама, банери, іконки, стрічки, різні знаки, спливаючі вікна, кнопки і багато іншого – часом цього стає занадто багато. Використовуючи елегантний дизайн і вільний простір, можна значно вплинути на враження відвідувачів.

Дотримання нових трендів у вебдизайні. Графічні та колірні рішення, цікаві деталі, оригінальне меню – все це можна обіграти в конкретному дизайні і красиво поєднати.

Легке сприйняття тексту. Текстова інформація надає необхідну інформацію і відповідає на питання ще до того, як вони були задані. Тому, важливо, щоб користувачам було легко читати тексти на сайті. Існує кілька простих правил, яких варто слідувати:

Достатня контрастність. Наприклад, шрифт світло-сірого кольору на білому тлі може позбавити бажання читати або дратувати. При застосуванні колірної схеми варто протестувати наскільки легко читається текст.

Достатній розмір шрифтів. Дрібний шрифт є абсолютно не практичним. Користувачі не повинні напружувати свій зір, щоб прочитати текст.

Обмежена кількість шрифтів. Використовувати максимум 2-3 шрифти. Художні чи рукописні шрифти використовувати лише для заголовків чи особливих фрагментів.

Схема перегляду сторінки

Це шлях, за яким погляд рухається по сторінці; шлях, який допоможе привернути увагу до важливого контенту на сайті (рис 7.5). Згідно цьому, погляд рухається зліва направо і зверху вниз. Відповідно, значна кількість відвідувачів зупинить свій погляд на кнопці, що розташована в лівому верхньому кутку сайту, і це може привести до зростання числа кліків. В цьому місці варто розміщувати самий важливий контент.

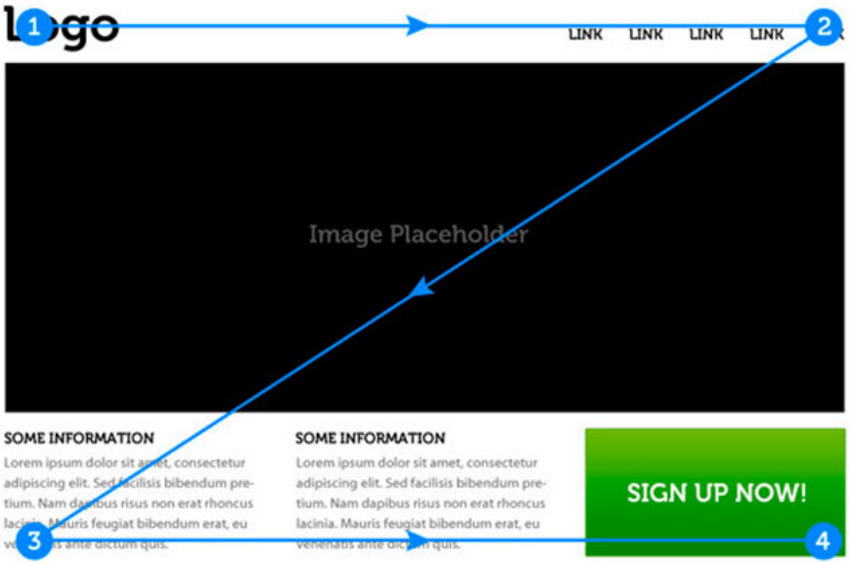


Рисунок 7.5. Схема перегляду сторінки

Зазвичай, елементи сторінки розташовують наступним чином: логотип зліва вгорі, меню справа вгорі, інформаційні блоки, картинки зліва внизу, кнопка із закликом до дії справа внизу.

Візуальні напрямки

Візуальними напрямками називають декоративні елементи сторінки, які скеровують погляд користувача на ті чи інші елементи дизайну, форми, кнопки та ін. Як візуальні напрямки можуть бути стрілки, напрям погляду людини на зображення, напрям вказівного пальця, в загальному все, що може вказувати у певний бік (рис. 7.6).

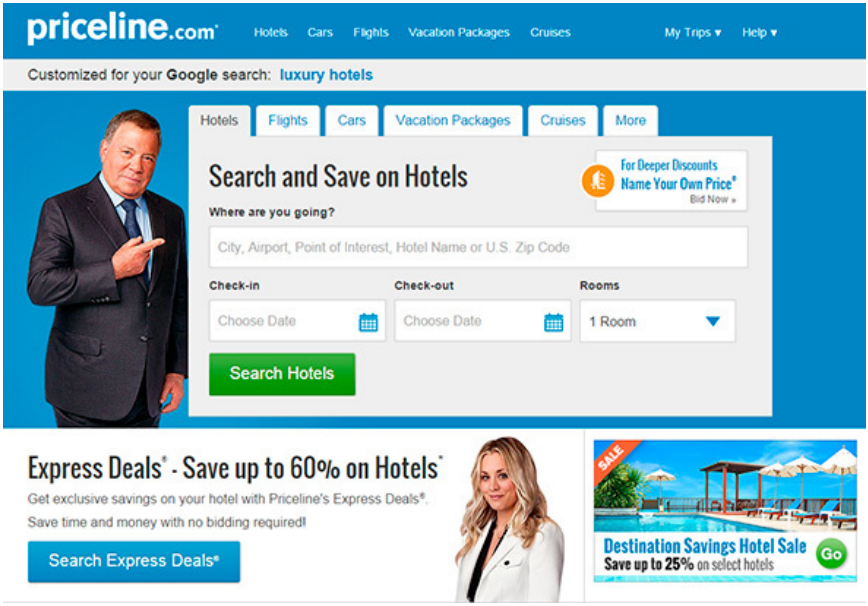


Рисунок.7.6. Візуальні напрямки на сторінці

Елементи для спонукання здійснити дію

Спонукання до дії (Call to Action) відноситься до інтерактивних елементів сайту: кнопки, банери та інше. Дані елементи оформлюються таким чином, щоб користувач захотів неодмінно на них натиснути. Наприклад, це може бути кнопка із закликом до дії (Натисніть, Дізнайтеся тощо), яскравий банер із привабливою пропозицією чи яскравою картинкою.

Дане поняття добре вписується у принцип AIDA (Attraction Interest Desire Action), який часто застосовується при дизайні головних сторінок, сторінок акцій, де необхідно підштовхнути користувача до тієї або іншої дії: підписка, покупка тощо (рис. 7.7).

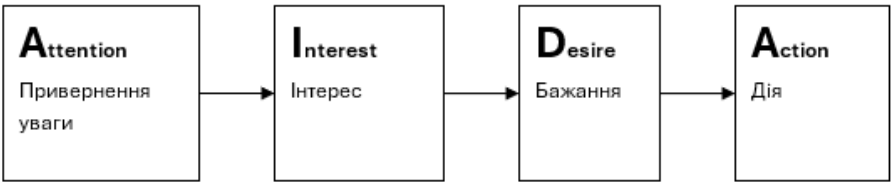


Рисунок 7.7. Ланцюжок принципу AIDA

Таким чином стає зрозумілим принцип побудови дизайну, що спирається на дане поняття: наприклад, яскрава картинка чи банер повинні привернути увагу користувача, супутній підпис у тексті повинен викликати в нього інтерес і бажання, а завершальним акордом повинна стати, наприклад, кнопка із закликом до дії.

Але цей принцип не працює сам без деяких інших, наприклад, схеми перегляду сторінки чи візуальних напрямків.

Вигляд мобільної версії

Важливим є адаптація задумів до втілення на дизайні для мобільних пристроїв. Це стосується розміщення елементів, вигляд слайдерів, таблиць, великих зображень. Якісно виконаний дизайн для телефону заохочує клієнтів частіше користуватися сайтом у вільні хвилини під час робочого дня чи за потребою.

Структура

На підставі наведених правил складають схематичний ескіз проєкту або на папері або в графічному редакторі (рис. 7.6). Це виконується для того, щоб узгодити розташування основних інформаційних блоків, графіки та інших елементів дизайну. Змовник узгоджує основний напрямок дизайну з певними зауваженнями по деталях.

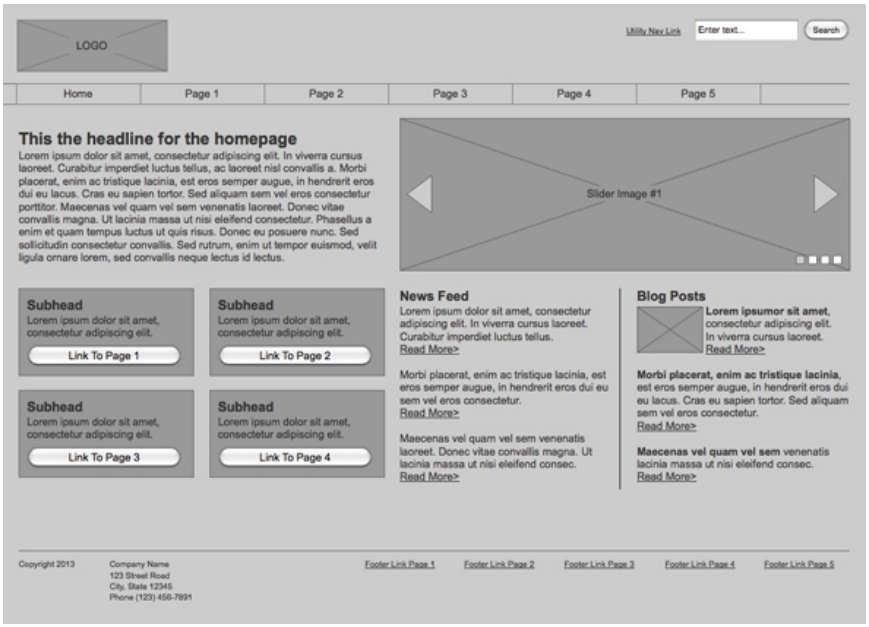


Рисунок.7.8. Схематичний ескіз сторінки

Розроблення макету головної сторінки сайту

Для створення головної сторінки, що здатна привернути увагу користувачів з першого погляду, варто дотримуватися певних правил.

Структура. Структура головної сторінки повинна бути простою і не містити зайвих елементів. Не слід зловживати занадто великою кількістю інформації і картинок. Контент сайту подається в акуратному і добре продуманому вигляді. У верхній частині сторінки розміщується сама важлива інформація: логотип, назва сайту та його напрямок або вид послуг, що надаються.

Зображення. Доречно застосовані зображення можуть краще представити проєкт ніж багато рядків тексту. Слід використовувати зображення високої якості, які викличуть у користувачів бажання продовжити вивчення сайту.

Кольори і фон. Для підкреслення бренду компанії або тематики сайту слід використовувати колірну схему і фон, які добре працюють разом. Варто уникати використання надмірної кількості різних кольорів і не

використовувати складний фон, що відвертає увагу від основного тексту та зображень.

Кнопки. Кнопки із закликом до дії ведуть на інші сторінки, сайти, промо-акції, каталоги і т. д. Їх слід зробити привабливими, щоб переконати користувачів натиснути на кнопку і зробити перехід. Текст на кнопці повинен бути коротким і чітким – одне-два слова.

Текст. Складно створити ідеальну головну сторінку, не використовуючи текст. Текст головної сторінки має коротко пояснити суть сайту. Не варто надавати великі секції з текстом, на головній сторінці обмежуються 1-2 абзацами, більш розлога інформація подається на відповідних сторінках сайту.

Основні елементи сторінки

Зазвичай, основними елементами сторінки є основний блок сторінки (wrapper, container), логотип, навігація, контент, хедер (верхній колонтитул), футер (нижній колонтитул), вільний простір (рис. 7.9).

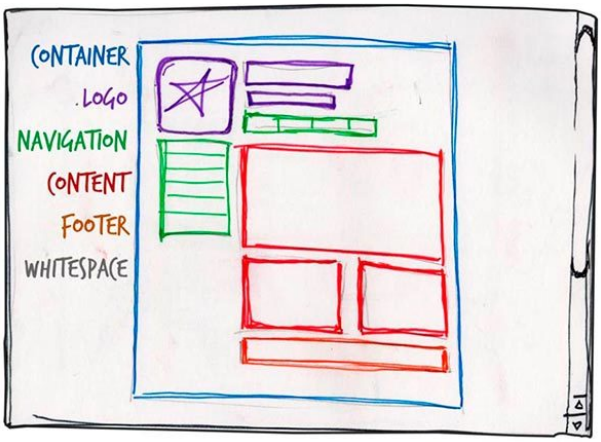


Рисунок.7.9. Основні елементи сторінки

Основний блок (контейнер). Роль контейнера на сторінці може виконувати безпосередньо елемент body або div. Ширина основного блоку може бути еластичною (fluid), а може бути фіксованою (fixed).

Логотип. Текстовий або графічний логотип найчастіше розташовується у верхньому лівому кутку сторінки або ж посередині (в залежності від ідеї, макета).

Навігація. Основна навігаційна панель містить посилання на основні розділи сайту. Навігаційна панель часто розташовується у верхній частині сторінки (незалежно від того вертикально або горизонтально розташовуються елементи навігації).

Контент. Основна складова вебсторінки. Він займає чільну роль в дизайні сторінки, тому займає більший простір, підкріплений, окрім тексту, графікою.

Нижній колонтитул (footer). Даний елемент розташовується внизу сторінки і, зазвичай, містить інформацію про власників, контактні дані, посилання на основні розділи сайту (може дублювати основну навігацію), посилання на соціальні мережі, форму зворотного зв'язку та інше.

Фіксований та еластичний макет

Фіксований макет. За фіксованим макетом незалежно від роздільності екрану сторінки сайту будуть мати однакову ширину (рис. 7.10).

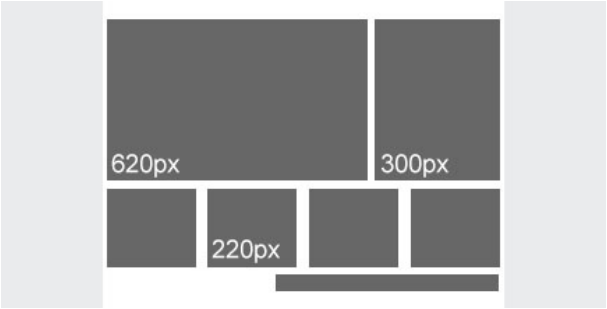


Рисунок 7.10. Фіксована ширина елементів

Еластичний макет. За еластичним макетом сторінка сайту буде займати весь доступний простір на екрані, підлаштовуючись під його розмір (рис. 7.11).

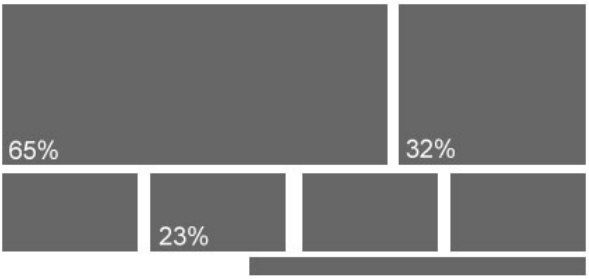


Рисунок 7.11. Еластична ширина елементів

На сьогоднішній день введено такі поняття як **чуйний вебдизайн** (Responsive Web Design) і **адаптивний вебдизайн** (Adaptive Web Desing). Перше поняття вкладається у концепцію «еластичного» і означає, що при зміні розміру екрану сайт підлаштовується під нього, друге поняття передбачає, що при розробці зазначаються основні розміри екрану, під які буде підлаштовуватися (адаптуватися) контент. В обох випадках слід розробляти не один, а кілька макетів, які будуть відповідати різним розмірам екрану. Часто створюється 3 макети під розміри Phone, Tablet і Desktop.

При розробці макету мобільної версії сайту намагаються на перший план винести основний контент, тому, часто навігаційне меню перетворюється в «гамбургер-меню», великі банери та декоративні елементи приховуються, блоки контенту, зазвичай, розташовують один під одним. На заздалегідь складеному макеті можна визначити, які елементи залишаються видимими на мобільному, а які будуть приховані.

Модульна сітка

Модульна сітка представляє набір невидимих напрямних, вздовж яких розташовуються елементи вебсторінки. Це полегшує розміщення даних у макеті, забезпечує візуальний зв'язок між окремими блоками і зберігає спадкоємність дизайну при переході від однієї сторінки до іншої.

Переваги застосування модульної сітки

Сітка задає стандарт розташування елементів: це полегшує вирівнювання елементів, додавання нових блоків і підтримку сторінки в подальшому.

Напрямні інтуїтивно підказують, де краще розташувати елементи. Можна додавати елементи не перевантажуючи дизайн макету.

Зменшується ймовірність помилок при перенесенні елементів з однієї сторінки на іншу. На основі сітки простіше робити адаптивний дизайн.

Сітка дозволяє створювати макети та шаблони швидше, спрощує роботу для дизайнера і верстальника. Допомагає швидше розібратися в макеті новим учасникам, оскільки у сітці завжди є логіка.

Сітка створює візуальний порядок і допомагає користувачеві швидше орієнтуватися та зчитувати інформацію. Допомагає сторінці виглядати більш естетично за рахунок того, що елементи є пропорційними і структурованими.

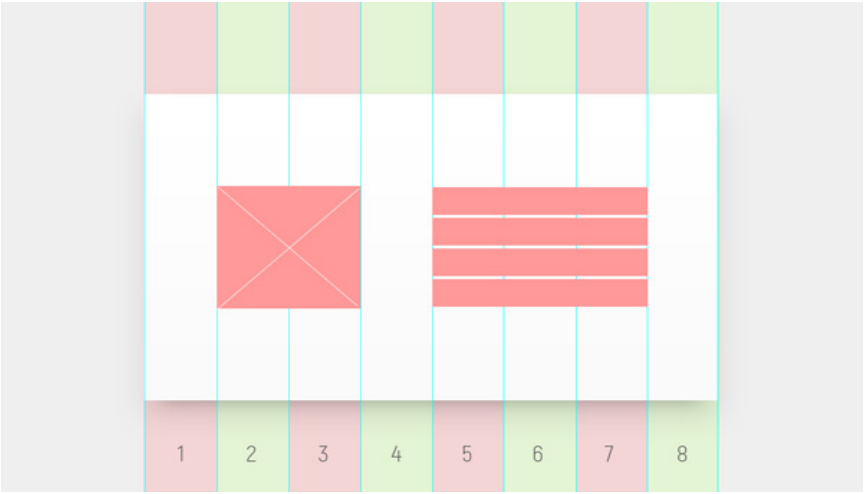


Рисунок7.12. Модульна сітка макету

Найбільш популярною є модульна сітка від фреймворку Bootstrap, яка поділяє сторінку на 12 колонок. Сітка Bootstrap надає можливість створювати адаптивні сторінки для різних пристроїв. Для цього використовується система з 12 колонок та п'ять контрольних точок розмірів екрану (брейкпоїнтів). Контрольні точки розташовуються на певній ширині «viewport» (видима частина вікна браузера). При досягненні контрольної точки відбувається перебудування елементів сторінки.

Завдяки модульній сітці блоки контенту і елементи будуть розташовуватися на певній відстані один від одного, мати прийнятну ширину, що в подальшому візуально буде приємно користувачеві і не викликати в нього незручностей у сприйнятті сайту.

Навігаційне меню

Навігація є важливим елементом дизайну, який допомагає користувачеві швидко отримати доступ до потрібних частин сайту. Планування навігації заслуговує на особливу увагу, оскільки сайт стає логічним і впорядкованим, підвищується його юзабіліті і відвідувачі швидко орієнтуються в географії сторінок.

Навігаційні меню є кількох типів, які можуть комбінуватися залежно від розгалуженості структури сайту.

Горизонтальне меню

Горизонтальне меню підходить для сайтів із невеликою кількістю розділів, а також для односторінкових сайтів. Зазвичай, розташовується під або над шапкою сторінки.

Опції меню виконують у вигляді тексту, іконок, вкладок. Назви опції мають бути в 1-2 слова. Хорошим тоном є виділення поточної опції кольором, підкресленням або в інший спосіб. Якщо сайт достатньо довгий, горизонтальне меню можна зафіксувати, щоб воно не прогорталося.

Слабке місце горизонтальних меню полягає в обмеженні кількості опцій. Тому, сайти зіскладною структурою використовують випадні меню, які допомагають приховати підрозділи і не захаращувати простір сайту. Випадні меню підходять для сайтів із розгалуженою структурою: магазинів із широким асортиментом і великою кількістю груп товарів.

Часто на потужних ресурсах одночасно використовують горизонтальне і вертикальне меню.

Вертикальне меню

Вертикальне меню може містити значно більшу кількість опцій, що складаються з більшої кількості слів. Таке меню є зручним для магазинів і

каталогів, де кількість товарів може збільшуватися. Широко вживаються групування опцій, розгортання підменю, іконки.

Меню-гамбургер

Популярний тип меню, що на сьогодні використовують не лише на мобільних, а й на десктопних версіях сайтів. У ньому може ховатися кілька пунктів або повний набір розділів і підрозділів. Недолік такого типу меню – непомітність, тому варто подбати про те, щоб користувачеві було зрозуміло, що перед ним саме меню.

Рисунок 7.13. Гамбургер-меню

Рекомендації при розробці навігаційного меню:

Логотип компанії завжди повинен вести на головну сторінку сайту. Згідно досліджень 36% людей схильні використовувати логотип як засіб переходу до головної сторінки.

Горизонтальне меню має містити якомога менше його опцій (від 3 до 7). Більша кількість заплутує відвідувачів. Назви опцій повинні бути лаконічними, але разом з тим і ємними.

Меню повинні бути оптимальними за розміром для екранів різних пристроїв: текст буде простіше читати, а на кнопки – натискати.

Ефективним є виділення опцій кольором або hover-ефектом, щоб мати можливість вказати відвідувачам, в якій області сайту вони знаходяться в даний момент.

Фіксовані меню, що залишаються на екрані при скролінгу, значно допомагають відвідувачам. Це відноситься до будь-якого стилю навігації: мобільного, десктопного, горизонтального і вертикального.

Для користувачів мобільних і десктопних пристроїв пропонуються різні варіанти меню, що пов’язано з розмірами екрану.

Меню для мобільної версії сайту варто робити більш виразним і краще оточити рамкою, щоб надати вигляд кнопки.

Застосовані іконки мають бути легко впізнаваними за змістом.

Глибокі, багаторівневі меню часто приводять до розгублення відвідувачів. Як альтернативу можна використовувати мега-меню і навігаційні ланцюжки (breadcrumbs), щоб полегшити подальшу навігацію.

Креативний варіант меню краще розміщати в очікуваному для відвідувачів місці. Багато дизайнерів експериментують з popup-формами, оскільки несподіваний рух на екрані має добре захоплювати увагу відвідувачів. Але спливаючі вікна не повинні бути зведені до вікон, що оповіщають про маркетингові пропозиції та інше, вони повинні використовуватися суто для навігації.

Інформаційне наповнення сайту

Контент (content, вміст) — це інформаційне наповнення сайту, інформація, яку розробник складає самостійно або копіює з дотриманням відповідних законодавств. Весь контент охороняється законом про авторське право, оскільки він є продуктом інтелектуальної праці і має своїх авторів і власників.

Якісний контент – це матеріал, що поєднує ряд властивостей: унікальність, корисність, актуальність, оздоблення, інформативність. Якісний контент дозволяє сайту краще ранжуватися у пошукових системах і привертає на сайт більше відвідувачів.

Користувач, заходячи в Інтернет, має на меті певні потреби і задовольнити їх є першочерговим завданням будь-якого контенту.

Потреба в інформації. Користувачеві потрібна інформація, для нього Інтернет – це величезна бібліотека, де відповіді можна знайти на будь-яке питання.

Потреба в розвагах. Людина хоче відпочити і подивитися відео-ролики, пограти в онлайн-ігри, розглядати фотографії.

Потреба в комунікаціях. Користувачеві хочеться з кимось поговорити, зв'язатися.

Алгоритми роботи пошукових систем стрімко розвиваються і тепер потрібно наповнювати сайт хорошим контентом. Пошукові системи враховують унікальність контенту через середній час, які користувачі проводять на сайті, читаючи статті. Враховується дата додавання матеріалу та дата останнього оновлення. Чим вона більш свіжа – тим вище актуальність вмісту, тим швидше сайт потрапить в топ видачі.

Хороші матеріали швидше зацікавлять користувачів. Якщо написано цікаву статтю, нею стануть ділитися, збільшиться потік відвідувачів, про сайт дізнаються, складеться хороша репутація. Все це допоможе досягти основної мети – сайт стане популярним.

Типи контенту

Текст. Це статті, новини, замітки.

Копірайтинг. Авторський унікальний матеріал, що ґрунтується на особистому досвіді автора, його знаннях.

Рерайтінг. Дані беруться з різних джерел, обробляються, переписуються. Текст виходить унікальним, але дані, що використані для написання, запозичено ззовні.

Оптимізовані тексти (SEO-тексти). Текст, що перероблений спеціально під пошукові системи і містить ключові слова. Від грамотності автора залежить наскільки ключі будуть вдало вписані в текст, чи не будуть вони різати око.

Копіпаст, плагіат. Скопійований текст із джерела без змін або з мінімальною корекцією.

Відео-контент. Це зміст сайту, що представлений у відеоформаті. Ролики, мальоване відео (дудл-відео), огляди, інструкції – кожному виду знайдеться своє застосування. У кожного відео повинна бути мета: залучення користувачів, підвищення довіри, навчання, демонстрація продукту і так далі. І в залежності від мети вибирається вид контенту.

За призначенням відеоконтент буває:

Іміджевий. Формує необхідне відношення споживача до продукції або компанії, породжує певні емоції. Це можуть бути кейси (успішні приклади послуг), відгуки споживачів.

Презентаційний. Завданням роликів є візуальна демонстрація послуги, продукту, явища, користі або вигоди для споживача. Прикладом такого ролика є звичайна телереклама.

Вірусний. Повинен привертати увагу, заражати ідеєю і спонукати ділитися цим роликом з іншими користувачами.

Навчальний. Це відео вчить споживача конкретним діям, показує наслідки дій, розповідає як поводити себе в різних ситуаціях, як товар або послуга допоможуть йому. Якщо товар складний в експлуатації, то відеоінструкції допоможуть з цим розібратися.

Соціальний. У такому відео показується проблема та шляхи її вирішення. Подібна реклама повинна бути тонкою, ненав'язливою, максимально природною.

Самим популярним відеоконтентом є **відеоролики**. Вони поділяються на кілька видів, наприклад, можуть бути простими і постановочними, із закадровим озвученням чи музикою.

Різновид таких роликів – **дудл-відео**. Це анімований відеоролик, в якому певна особа малює у режимі реального часу. До нього можна додавати корпоративні персонажі, текст, озвучення. Динамічні ролики мимоволі утримують увагу глядача. А значить, він активно сприймає всю інформацію з дудл-відео.

На сайті можна розмістити **просту презентацію**, що створена зі слайдів. Якщо до неї додати оригінальні спецефекти, музику або голосове озвучення – за належної частки креативу вийде гідний ролик.

Ще один варіант відеоконтенту – **скрінкасти**. Це запис того, що відбувається на екрані, з додаванням пояснюючих коментарів. Так записуються навчальні відео.

Популярністю серед користувачів користуються і **GIF-анімації**. Вони короткі, але здатні передати більше інформації, ніж невеликий текст.

Залежно від цілей компанії та її аудиторії можна використовувати **мультиплікаційні ролики** або **ролики з 3D-анімацією**. Цей формат допомагає змоделювати найважчі робочі процеси і наочно показати їх споживачеві.

Відео в реальному часі – це можливість для покупця задати питання та отримати відповіді, побачити в ролику знаменитість або подивитися демонстрацію продукту, наприклад, на виставці. Деякі компанії навіть проводять навчальні вебінари.

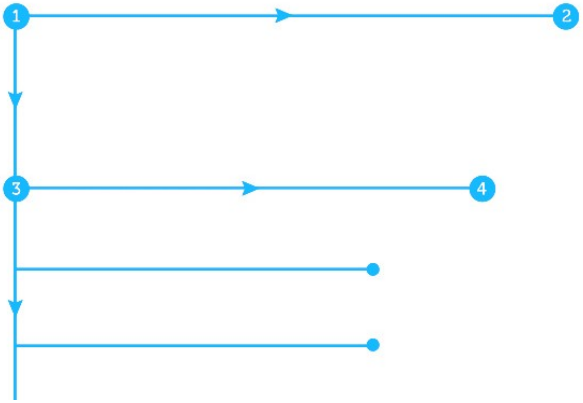


Рисунок 7.14. Траєкторія перегляду тексту за F-патерном

Очевидно, що траєкторія сканування не завжди складається з трьох чітких відрізків. Коли користувач знаходить щось цікаве, він починає читати, формуючи горизонтальні лінії.

F-макет – це хороше рішення для сайтів, де багато тексту: наприклад, блогів і новинних сайтів. Якщо на сторінці багато контенту – особливо тексту – користувачам буде простіше вивчати його за природною, звичною траєкторією сканування. F-макет дозволяє дизайнеру контролювати, на що користувач зверне увагу.

Розставити контент за пріоритетністю. Перш ніж компонувати елементи на сторінці, їх слід розмістити за ступенем пріоритетності. Елементи, на які потрібно звернути увагу користувачів у першу чергу – розміщають в "гарячих точках" F-патерну, щоб домогтися потрібного ефекту.

Надання можливості швидкого ознайомлення. Перші два параграфи – найважливіші. Найважливіший контент розміщається якомога вище, щоб відразу повідомити читачеві головну ідею і призначення сайту (або сторінки).

Верхній рядок користувачі, зазвичай, читають цілком по горизонталі, а значить – це відмінне місце для панелі навігації.

Проєктування вмісту під сканування, а не під читання. У першу чергу варто орієнтуватися на користувачів-сканерів. Додавати контент, який може їх зацікавити, на лініях F-патерну:

Починати новий параграф із ключових слів, які привернуть увагу «сканерів».

Люди насамперед звертають увагу на елементи або області з великою візуальною вагою. Так можна збільшувати візуальну вагу важливих речей: якщо це текст, то використовувати типографіку (наприклад, виділити в тексті ключові слова), а якщо це кнопка – оформити її в яскравому кольорі.

Один параграф – одна ідея. Частіше використовувати списки.

Розміщення важливого контенту (наприклад, заклик до дії) зліва чи справа – там, де користувач починає і закінчує сканування. У цих точках погляд користувача на мить зупиняється – а значить, він приділить для важливої інформації дещо більше уваги.

Використання бічної панелі. Бічна панель потрібна, щоб залучати користувачів у взаємодію на більш глибокому рівні:

Додати до бічної панелі все, що не має першорядної важливості, але заслуговує на увагу користувача. Це може бути реклама, список статей по темі, кнопки соціальних мереж і т. д.

Перетворити бічну панель в джерело спеціалізованого контенту. Наприклад, там можна розмістити список категорій, хмару тегів, віджет із популярними постами і т. д.

Уникати довгих монотонних макетів. Основний недолік F-макету в тому, що він прагне до монотонності. Користувачам швидко набридають повторювані рядки, тому варто пожвавити область сканування якимось елементом (картинка, художня лінія, оформлена цитата тощо).

F-патерн повторює природну траєкторію руху погляду, тому можна оптимізувати під неї макет сторінки. Але не потрібно строго дотримуватися цього патерну – зрештою, це просто рекомендація, а не стандарт.

8. ТИПОГРАФІКА

8.1. Комп’ютерні шрифти

Типи шрифтів

Шрифт є одним із важливих елементів дизайну документа, який може підсилити чи навпаки зменшити ефективність донесення інформації до користувача. Важко переоцінити значення шрифту в оформленні тексту, будь то вебсторінка чи журнальна стаття. Вірно підібраний шрифт полегшує сприйняття і додає сторінці неповторного стилю.

Комп'ютерний шрифт – це файл, що містить кодований набір буквених, цифрових, службових і псевдографічних символів, які відтворюються програмою або операційною системою та відображаються на пристрої виведення (екрані або принтері), а також ряд спеціальних символів, що не відображаються (кінець рядка, пробіл тощо).

Екранні шрифти. Використовуються для відображення тексту на екрані. Кожен символ є двохколірною матрицею екранних пікселів. Як правило, в кожному шрифті представлено кілька фіксованих за розміром наборів символів. Це дозволяє більш гнучко змінювати розмір символів залежно від роздільності. Даний тип шрифтів є оптимальним для застосування в службових елементах інтерфейсу – діалогових вікнах, панелях, палітрах, меню тощо. При виведенні на друк екранні символи виглядають непрезентабельно.

Принтерні шрифти. Завантажуються безпосередньо в пам'ять друкованого пристрою, а іноді відразу записані в пам'яті принтера. Це дозволяє максимально пришвидшити процес друкування, але створює значні незручності через неможливість побачити текст на екрані. Теоретично ідеальною є зв'язка «екранний шрифт – принтерний шрифт тої ж назви». На практиці часто виявляється, що екранна і принтерна версії реалізовані по-різному, через що текст «пливе», деякі символи зникають і т. п.

Універсальні шрифти. Призначені як для виведення на екран, так і для друкування. На відміну від попередніх типів шрифту, завжди представляють собою набір векторних об'єктів, які растеризуються лише в момент відображення на вивідному пристрої. Це дозволяє домагатися максимальної

якості відображення символів, а також гарантує ідентичність екранного та надрукованого тексту. Головний недолік – залучення додаткових ресурсів комп'ютера для растеризації при виведенні на екран.

Існує кілька різних форматів векторних шрифтів, що різняться за способом збереження і представлення інформації про шрифт: це PostScript, Type1,TrueType, OpenType.

PostScript (Type 1) є першим форматом файлів для шрифтів. Створено Adobe у першу чергу для принтерів. PostScript відрізнявся від інших основних форматів принтерів, використанням мови програмування для опису друкування зображення. Шрифти PostScript мають гладкі, докладні, високоякісні літери, але файли шрифту є доволі великими. В основному використання обмежене для друкування книг, плакатів і журналів професійної якості. Формат файлу має розширення .ps.

TrueType (TTF) є найбільш поширеним форматом і стандартним шрифтом, що розроблений Apple Computer. Оригінальний формат засновано на мові PostScript. Цей стандарт призначений для використання на комп'ютерах із ОС Windows. Використовують для випадків, які вимагають виняткової якості друку. Формат файлу має розширення .ttf.

OpenType (OTF) створено Microsoft і Adobe у 1994 році, як розвиток TrueType і PostScript і метою було створення універсального формату відкритого стандарту для всіх комп'ютерів. Формат файлу має розширення .otf.

За способом створення

За способом створення шрифти бувають растровими і векторними (контурними):

В растрових шрифтах кожен символ описано у вигляді набору точок (пікселів), що розташовано у вузлах сітки растру. Тут, шрифт є звичайним точковим рисунком. Растрові шрифти є непридатними для високоякісного друку і використовуються, в основному, в програмах із текстовим інтерфейсом. Вони широко використовувалися в епоху матричних принтерів і моніторів низької роздільності.

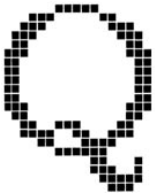


Рисунок 8.1. Растровий шрифт

У векторних (або контурних) шрифтах символи є криволінійними контурами, які описано математичними формулами. Кожен символ містить координати опорних точок, які сполучені прямими або кривими лініями і утворюють контур символу без прив'язки до абсолютного розміру чи роздільності. Такий опис дозволяє легко змінювати масштаб зображення без втрати якості, що є неможливим у випадку з растровими шрифтами.

Векторні шрифти добре відтворюються як на екрані, так і в друці. Для виведення векторного шрифту на растрові пристрої (монітори і принтери) його необхідно растеризувати – перетворити в набір точок. Для растеризації шрифтів у сучасні операційні системи Windows і Mac OS втілено растеризатор шрифтів.

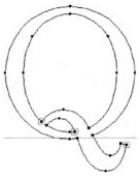


Рисунок 8.2. Векторний шрифт

Слід зазначити, що векторний комп'ютерний шрифт – це не просто таблиця символів, а невелика і досить складна програма. Створенням шрифтів часто займаються непрофесіонали, тому, буває, що шрифт, який виглядає добре на екрані, викликає проблеми при друкуванні.

За шириною символів

За шириною символів шрифти поділяють на моноширинні і пропорційні.

У моноширинному шрифті символи мають однакову ширину. В ранніх комп'ютерах це дозволяло спростити виведення тексту на екран: кожен символ розташовувався в межах відведеного місця, зображення символу було фіксованого розміру і процедура розміщення символу на екрані була надзвичайно простою.

В пропорційному шрифті символи можуть мати різну ширину. Наприклад, літера «І» займатиме значно менше місця, ніж літера «Ш». Це дозволяє в значній мірі зменшити середній розмір символу і зберегти при цьому легкість для читання. Текст, що набрано пропорційним шрифтом виглядає акуратніше і краще читається при великих об'ємах. Для друкування і відображення текстів майже завжди застосовується пропорційний шрифт.

За сімейством

За сімейством шрифти бувають:
Serif – шрифти із засічками на закінченнях літер.
Sans-serif – шрифти без засічок.
Cursive – шрифти вільного стилю (рукописні, курсивні, художні).

Таблиця 8.1. Види шрифтів

Приклад	Опис
Текст	Шрифт із засічками (serif) **Засічки або серіфи** (фр. *serif*) – це горизонтальні елементи закінчення основних чи сполучених штрихів, які можуть мати різні форми: прямокутну, зігнуту, заокруглену тощо. Засічки полегшують читання, ніби сполучаючи літери між собою, і одночасно розділяють окремі букви, щоб вони не зливалися. Шрифти із засічками також називають *антиквеними*, тобто античними. Шрифти із засічками виглядають традиційно, типово і полегшують сприйняттю великих об'ємів тексту. Яскравим прикладом серіфних шрифтів є *Times, Bodoni, Garamond*.

Текст

Шрифти без засічок (sans-serif)

Це так звані **гротески** (нім. *Grotesk*, анг. і фр. *Grotesque*, амер. *Gothic*). У шрифтах без засічок відсутні завершальні елементи на кінцях штрихів. Шрифти без засічок виглядають сучасно і стильно, є широко вживаними для електронних документів, що переглядаються на екрані монітору.

В операційних системах шрифти без засічок користуються більшою популярністю, ніж шрифти із засічками, тому перелік універсальних шрифтів без засічок, що доступні на платформах Windows, Linux і MACOS є більш широким. Це *Arial, Impact, Helvetica, Lucida, Tahoma, Trebuchet, Verdana.*

Текст

Шрифти курсивного зображення (cursive)

Це каліграфічні шрифти, що подібні до рукописних. Вони можуть бути зв'язаними (кожна літера з'єднується з сусідніми) і незв'язаними (де кожна літера візуально є окремою). Можливі також проміжні напівзв'язані форми, де з'єднуються певні частини літер або літери з'єднуються лише з одного боку.

Рукописні шрифти застосовують, зазвичай, не для інформативного тексту, а, скоріше, для привабливості заголовків.

Текст

Моноширинний шрифт (monospace)

Відрізняється від інших шрифтів однаковою шириною і висотою символів, ця особливість виділяє його серед пропорційних шрифтів, у якому символи відрізняються один від одного. Подібні види гарнітур використовують для таблиць, програм та інших випадків, коли необхідно забезпечити збіг знаків або колонок по вертикалі.

Текст

Альтернативні (художні) шрифти

Це шрифти, які створені у власному неповторному стилі. В основному – це новий вигляд шрифтових форм, що не базуються на традиційних символах шрифтів.

Іконкові шрифти

Іконкові шрифти – це шрифти, де кожен символ представлений картинкою. До появи іконкових шрифтів на вебсторінках використовували растрові зображення іконок, що було незручно. На сьогодні широко вживають векторні іконки, що виконані у вигляді шрифтів.

Принцип роботи таких шрифтів простий, до вебсторінки, за допомогою CSS стилів, підключається додатковий файл шрифту. В стилях сайту вказуються властивості іконок (розмір, колір, тінь та інші), і в розмітці сторінки пишуться спеціальні теги, на місці яких і з'являються іконки.

За роллю на сторінці

За роллю на сторінці шрифти можна поділити:

Текстові (text). Текстові шрифти використовуються для довгих фрагментів тексту в книгах і журналах, для набору основного тексту. Особлива увага в дизайні таких шрифтів приділяється зручності при читанні.

Акцидентні (display). Акцидентні шрифти проєктуються для великих розмірів і використовуються для набору заголовків.

Декоративні (decorative). Декоративні шрифти широко використовуються в рекламі, також можуть використовуватися для заголовків.

За зображенням

За зображенням шрифти поділяються на три групи.

До першої групи відносять шрифти за ознакою нахилу: прямий, нахилений, курсивний.

До другої групи відносять за ознакою контрастності: жирний, напівжирний, світлий.

До третьої групи шрифти відносять, виходячи зі співвідношення ширини символу до висоти: вузький, широкий.

Гарнітура шрифту

Гарнітура шрифту (*Type family*) – це сукупність зображень, які об'єднані загальними стильовими ознаками, характером графічної побудови символів і рішенням їх елементів.

В гарнітурі присутній комплект шрифтів, що мають схожий малюнок, але різняться за накресленням (звичайний, курсивний, жирний). Наприклад, шрифти «Arial», «**Arial Bold**», «*Arial Italic*» складають одну гарнітуру «Arial».

Деякі гарнітури містять більшу кількість зображень, ніж інші. Завдяки цьому можна побудувати весь документ на одній гарнітурі, використовуючи там, де потрібно різні варіанти зображень.

Розмір шрифту

Розмір шрифту є важливим інструментом представлення тексту на сайті, він визначає ієрархію розмірів заголовків, основного тексту, а також впливає на сприйняття тексту загалом. Розмір шрифту знаходиться в дуже тісному контакті з іншими властивостями тексту, такими як ширина колонки, висота відступів тощо (рис. 8.3).

Рисунок 8.3. Розміри символів шрифту

Одиниці виміру

Абсолютні одиниці є фіксованими і відносяться до будь-яких фізичних одиниць вимірювання. Після того, як вони були задані, розмір не змінюється.

Відносні одиниці не мають фактичного значення. Їх розмір визначається відносно батьківського елементу. Це означає, що розмір шрифту можна змінити шляхом зміни розмірів пов'язаного елемента.

Піксель

Піксель px – це базова, абсолютна і остаточна одиниця виміру. Пікселі поєднують властивості і абсолютних і відносних одиниць вимірювання.

Відносність визначається тим, що видимий розмір, вказаний в пікселях, залежить від роздільної здатності монітора. На моніторі з невеликою роздільною здатністю видимий розмір буде більшим, ніж той же розмір у пікселях на моніторі з високою роздільною здатністю. Це пов'язано з тим, що розміри пікселів є незмінними, але на великому моніторі для відображення вмісту екрану використовується більша кількість пікселів. Таким чином, при збільшенні роздільності монітору, розміри, що задані в пікселях, зменшуються.

Абсолютність пікселів обумовлено незалежністю даної одиниці вимірювання від налаштувань браузера. Піксель є мінімальною одиницею вимірювання, тобто, немає величини 0.5 пікселя. Шрифт, що заданий у розмірі 12px, завжди відображатиметься в цьому розмірі і його буде неможливо змінити через налаштування браузера. Таким чином, пікселі можна назвати відносними одиницями вимірювання, що є незалежними від налаштувань браузера.

Абсолютні одиниці

Завдання абсолютних одиниць вимірювання не дозволяє відвідувачеві змінити розмір шрифтів через меню браузера і адаптувати сторінку під свої налаштування.

cm Сантиметр
mm Міліметр
in Дюйм (1 дюйм = 2,54 см)
pt Пункт (1 пункт = 1/72 дюйми = 0.35 мм)
pc Піка (1 піка = 12 пунктам)

У більшості випадків неможливо заздалегідь передбачити на якому моніторі, якого розміру і роздільності буде відображено сайт. Текст, що нормально сприймається на екрані з невеликою чи середньою роздільною здатністю, може стати абсолютно нечитабельним при високій роздільній здатності.

Абсолютні одиниці вимірювання варто застосовувати лише тоді, коли відомо про точні фізичні розміри пристрою відображення (наприклад, розміри екрану або сторінки, що буде надрукувана).

Якщо сайт передбачає наявність версії для друку, то там доречні будуть pt, mm, cm, але потрібно точно розуміти, для яких елементів задаються значення. У більшості випадків для інтернет-видань використовуються відносні одиниці.

Відносні одиниці

Відносні одиниці вимірювань застосовуються у верстці набагато частіше. Відносні одиниці визначають розмір елементу через значення розміру іншого елементу. Таблиці стилів, які використовують відносні одиниці, легше масштабуються з одного середовища виведення в інше.

Використання відносних одиниць вимірювання надає значні переваги щодо вибору розміру шрифту. Відвідувач може самостійно регулювати розмір шрифту через меню браузера і адаптувати сторінку під власні вподобання. У разі, коли користувач змінює розмір шрифту основного тексту в браузері, пропорційно збільшуються чи зменшуються решта текстових елементів сторінки.

У більшості випадків для визначення розмірів шрифту, а також ширини, висоти і відступів елементів слід використовувати такі одиниці вимірювань, як px, em, rem і %. Для мобільних пристроїв добре підійдуть vh, vw або vmin, vmax.

Одиниці em, ex, ch є величинами, відносними до шрифту, що встановлено за замовченням на сайті. Одиниця 1 em дорівнює висоті найбільшої літери цього шрифту. 1 ex дорівнює висоті рядкових літер шрифту. 1 ch означає ширину символу «0» (нуль).

Найчастіше вживають одиницю **em**, що відповідає 100% розміру шрифту батьківського елементу. Якщо цей розмір змінити, задавши його більше за одиницю, то шрифт збільшиться. Якщо його поставити менше за одиницю, то шрифт зменшиться. Тобто, число перед em є множником для розміру шрифту.

Використання відсотків % або em – це вибір верстальника, тому що по суті обидві одиниці відштовхуються від розмірів шрифту батьківського елементу.

Одиниця rem (root em) відповідає розміру «кореневого» елемента, а саме – тегу <html>. Для нього не так часто задаються стилі, тому розмір береться з налаштувань розміру шрифту в браузері, найчастіше, це 16px. Саме він береться за «кореневий» – «root» em – rem. Від цього розміру і розраховуються розміри, що зазначені в rem.

Таблиця 8.2. Відносні одиниці

Одиниця	Відносно чого вимірюється
%	Відсоток від висоти шрифту батьківського елементу
em	Висота найбільшого символу шрифту «M» поточного елементу (m-height)
ex	Висота символу «x» шрифту поточного елемента в нижньому регістрі (x-height)
ch	Ширина символу «0» (zero) у шрифті поточного елементу
rem	Висота найбільшого символу шрифту «M» кореневого елементу <html>
vw	1% ширини вікна перегляду (viewport width)
vh	1% висоти вікна перегляду (viewport height)
vmin	Мінімально допустимий відсоток від висоти
vmax	Максимально допустимий відсоток від висоти

CSS одиниці для мобільних пристроїв

На даний момент зміна розмірів шрифту в залежності від розміру екрану є важливою частиною вебдизайну, оскільки під різні роздільності екранів шрифт потрібно «перебудовувати» – тобто збільшувати або зменшувати його для зручності користувача.

Для цього застосовують одиниці, які беруть відсоток від viewport, тобто вікна перегляду, яке в телефонах буде від 320px до 480px, у планшетах 768-1024px, а на моніторах – від 1024px. Тому, використовувати тут можна такі одиниці, як:

vw – 1% ширини вікна;
vh – 1% висоти вікна;
vmin – найменше з (vw, vh), в IE9 позначається vm;
vmax – найбільше з (vw, vh).

Дані одиниці також відмінно зарекомендували себе в тих випадках, коли, наприклад, потрібно встановити висоту елемента, що дорівнює висоті екрану.

Поради

Оптимальні розміри абзаців

Оптимальні розміри абзаців дозволяють оформити текст на сайті таким чином, щоб відображена в них інформація засвоювалася найкращим чином і створювалося враження легкості читання. Найкраще прописувати не більше 5-7 рядків.

Довжина рядка не повинна перевищувати 600 px (16-17 см). Це оптимальний розмір для комфортного переміщення погляду з одного рядка на інший. Дуже широку контентну частину важко читати – часто втрачається зміст після прочитання довгого попереднього рядка.

Якщо необхідно розтягнути текстовий блок на 1000 px і більше за шириною, можна спробувати розподілити текст на дві або більше колонок. Інший варіант – зробити межрядкову відстань більше за звичайну, щоб візуально сильніше відокремити рядки один від одного. Варто розділяти текст абзацами, це також допоможе зробити його легким для читання.

Вертикальні відступи (Margin) між абзацами

Відступи мають вплив на загальне сприйняття сторінки. Вони допомагають розміщувати текст на різних відстанях від суміжних елементів, а також від меж браузера.

На вебсторінках існують правила оформлення абзаців та інших елементів тексту. Наприклад, на паперових носіях прийнято робити горизонтальні відступи першого рядка, а на вебсторінках абзаци мають збільшені відступи між собою. Це пов’язано з тим, що сприйняття тексту з екрану є складніше ніж з паперу. Тому, великі об’єми тексту бажано розділяти на дрібніші фрагменти і розділяти збільшеними вертикальними відступами.

Відстань між рядками (Line Spacing)

Вертикальна дистанція між рядками має значний вплив на чіткість і стиль відображення тексту. Оптимальною відстанню між рядками вважається відстань, не менша за висоту символу.

Рисунок 8.4. Велика відстань між рядками

При встановленні відстані між рядками необхідно не забувати, що занадто велика відстань також може негативно позначитися на загальному сприйнятті тексту.

Вирівнювання (Alignment)

Для вебтексту можна використати кілька варіантів вирівнювання:

За лівим краєм. Традиційний підхід до вирівнювання основного тексту.

По центру. Здебільшого застосовується для заголовків чи важливих фрагментів.

За правим краєм. Використовується дуже рідко для специфічних цілей, наприклад, епіграфу.

За шириною. Є найбільш популярним у друкованих виданнях, але рідше використовується у вебтипографіці. При вирівнюванні за шириною колонки браузер збільшує або зменшує відстані між словами в кожному рядку, що істотно відбивається на естетичному вигляді загального тексту.

Відстань між літерами (Letter Spacing)

Зазвичай, стандартні шрифти у браузерах відображаються з оптимальною відстанню між літерами та словами. Зазвичай, зміну відстані між літерами застосовують для коротких заголовків або важливої інформації.

Вибір шрифту для сайту

Веброзробник може використовувати різні шрифти для тексту, заголовків, логотипу та інших елементів. Але, потрібно знати, що браузер може відобразити на сторінці лише ті шрифти, які встановлено на комп'ютері у користувача.

З цієї точки зору шрифти можна умовно розподілити на дві категорії:

Стандартні шрифти, які містяться в інсталяційних пакетах операційних систем, офісних програмах та інтернет-застосуваннях. Вони без проблем відображаються в переважної більшості користувачів.

Довільні шрифти, які користувач доставляє за власними потребами та уподобаннями. Вони можуть бути відсутніми у значної групи користувачів.

Стандартні шрифти

Стандартні шрифти – це набір шрифтів, що встановлюється разом із операційною системою. Оскільки операційні системи бувають різними, то і набір шрифтів у них є різним.

Стандартні шрифти Windows:
http://www.microsoft.com/typography/default.mspx

Стандартні шрифти Mac OS:
http://www.microsoft.com/typography/fonts/mac.htm

В **Unix/Linux** операційних системах єдиного набору шрифтів немає. За статистикою більшість Unix/Linux користувачів мають на своєму комп'ютері шрифти набору: *Core fonts for the Web. http://www.microsoft.com/typography/fonts/web.aspx*

Веббезпечні шрифти

У веб історично склалося таке поняття як «безпечні» вебшрифти. Безпечним шрифтом можна назвати шрифт, який є стандартним для всіх операційних систем. Основою для визначення «безпечних» шрифтів послужили шрифти Windows, які використовуються в інших операційних системах, наприклад, пакет шрифтів *Core fonts for the Web*.

Цей пакет містить наступні шрифти: Andale Mono, Arial Black, Arial, Comic Sans MS, Courier New, Georgia, Impact, Times New Roman, Trebuchet MS, Verdana, Webdings. Всі вони підтримують кирилицю, що важливо для української аудиторії.

Таким чином, на основі шрифтів Windows, що використовуються в інших операційних системах сформувався наступний список так званих «безпечних» вебшрифтів:

1. Сімейство sans-serif – шрифти без засічок, із прямими чітко прописаними контурами:

- Arial — Шрифти для сайту.
- Arial Black — **Шрифти для сайту**.
- Tahoma — Шрифти для сайту.
- Verdana — Шрифти для сайту.
- Lucida Sans Unicode — Шрифти для сайту.
- Trebuchet MS — Шрифти для сайту.
- MS Sans Serif — Шрифти для сайту.
- Impact — **Шрифти для сайту**.
- Century Gothic — Шрифти для сайту.

2. Сімейство serif – шрифти з засічками:

- Times New Roman — Шрифти для сайту.
- Georgia — Шрифти для сайту.
- Palatino Linotype — Шрифти для сайту.
- Sylfaen — Шрифти для сайту.
- Garamond — Шрифти для сайту.
- Century — Шрифти для сайту.

3. Сімейство monospase – моноширинні шрифти:

- Courier New — Шрифти для сайту.
- Lucida Console — Шрифти для сайту.
- Consolas — Шрифти для сайту.

4. Сімейство cursive:

- Monotype Corsiva — Шрифти для сайту.
- Mistral — *Шрифти для сайту.*

Ці шрифти є в кожного користувача Windows, Mac OS і в переважній більшості користувачів Unix/Linux.

Лінійки шрифтів

Для того, щоб текст сторінки міг відображатися однаково за задумом дизайнера в будь-якій операційній системі, існує можливість задавати лінійки шрифтів: комбінації безпечних і довільних шрифтів.

font-family: Arial, "Helvetica", sans-serif.

font-family: "Times New Roman", "Times CY", "Nimbus Roman No9 L", serif.

font-family: "Lucida Console", Monaco, monospace.

Властивість font-family вказує пріоритетний список шрифтів, що використовуються для відображення даного елемента. Якщо перший шрифт зі списку не встановлено на комп'ютері, з якого виконується доступ до сайту, шукається наступний шрифт списку, поки не буде знайдено відповідний.

Для категоризації шрифтів використовуються два типи імен:

Назва шрифту (Family-name). Приклад family-name «Arial», «Times New Roman» або «Tahoma».

Родовий шрифт (Generic family). Його можна простіше описати як групу family-names, що мають характерні загальні риси. Наприклад, sans-serif – набір шрифтів без засічок.

При вказуванні шрифтів для сайту першим вказується самий відповідний шрифт, далі перелічуються альтернативні. Рекомендується в кінці списку вказувати родовий шрифт. Тоді сторінка, як мінімум, буде відображена шрифтом того ж сімейства, якщо відсутні всі специфіковані конкретні шрифти.

Список шрифтів може виглядати так:

```
h1 {font-family: arial, verdana, sans-serif;}
h2 {font-family: "Times New Roman", serif;}
```

Заголовки <h1> будуть відображатися шрифтом «Arial». Якщо він не встановлений на клієнтській машині, буде використовуватися «Verdana». Якщо недоступні обидва шрифту, для показу заголовків буде використаний шрифт сімейства sans-serif.

Зверніть увагу, що назву шрифту «Times New Roman» укладено в лапки. Якщо в назві гарнітури присутні пробіли, тоді цю назву обов'язково потрібно укладати в лапки.

Формати шрифтів

Десктопні формати

Формат OpenType прийшов на зміну 8-бітовим форматам. Він працює у Windows, починаючи з версій 2000 і XP. На платформі Macintosh цей формат реалізовано без додаткової програмної підтримки з появою Mac OS X. Існують два різновиди даного формату: OpenType на базі мови TrueType і OpenType на базі мови PostScript з ідентичною загальною структурою, але різняться за іншими ознаками:

Кросплатформність – один і той же шрифтової файл можна встановлювати як у Windows, так і в Macintosh.

Підтримка стандарту Юнікод – можливість побудови шрифтів із розширеним знаковим комплектом.

Підтримка спеціальних типографських функцій – лігатури, капітель, мінускульним цифри, альтернативні гліфи, контекстуальні підстановки і т. п.

Більш економний «стислий» формат – шрифтові файли займають менше місця на диску і швидше пересилаються по мережі.

Перелічені особливості є відмінними при порівнянні OpenType із застарілим 8-бітовим форматом PostScript. Шрифти формату TrueType, що випускалися для Windows «до епохи» OpenType, фактично теж є OpenType. Вони підтримують Юнікод, можуть безпосередньо встановлюватися в Mac OS X і підтримувати друкарські функції.

Шрифти формату OpenType використовуються для роботи в десктопних додатках.

Вебформати

Для використання в Інтернеті потрібні шрифти спеціальних вебформатів. **Різні браузери історично підтримують різні вебформати, тому виробники** шрифтів змушені підтримувати різні формати, щоб їх шрифти працювали всюди.

EOT – Embedded OpenType (вбудовований Open Type) – формат розроблено компанією Microsoft і підтримується виключно браузером Internet Explorer, починаючи з версії 4.0. Представляє стислу копію шрифту TTF.

WOFF – Web Open Font Format – розроблено Mozilla Foundation для браузера Firefox. На даний час формат woff підтримується всіма основними браузерами.

WOFF2 – друга версія Web Open Font Format у середньому на 30% «легше» за першу, що пришвидшує завантаження шрифтів.

SVG – Scalable Vector Graphics. Окремий шрифтовий файл у форматі .svg , що представляє векторне накреслення шрифту. Як правило, має менші розміри файлів, тим самим дозволяючи покращити продуктивність на мобільних пристроях. Працює в Opera Mobile і iOS Safari.

OTF / TTF (OpenType Font і TrueType Font) працюють в більшості браузерів. Обидва формати поширюються вільно. Особливістю використовуваного у вебкомплекті TTF є захищеність формату, що перешкоджає його використанню локально на комп'ютері.

Вебкомплекти формуються на основі формату OpenType/TT і мають відповідати основним необхідним вимогам до вебшрифтів:

Якість відображення для різних режимів візуалізації.

Узгодженість вертикальних метрик при використанні на різних платформах і браузерах.

Підтримка всіх властивостей вихідного OpenType / TT.

Використання вебшрифтів

Якщо обраний для дизайну шрифт не міститься у переліку стандартних шрифтів, тоді, слід застосувати сучасні вебтехнології втілення вебшрифтів.

Використання властивості @ font-face в файлі стилів.

Використання API Google Font.

Правило @ font-face

Правило @ font-face дозволяє під’єднувати користувацькі вебшрифти. Браузер завантажує шрифти в кеш і використовує їх для оформлення тексту на сторінці. Такий підхід називається втіленням шрифтів.

Правило @ font-face потрібно розміщувати перед всіма іншими правилами css, оскільки цей прийом підвищує продуктивність сторінки. Файли зі шрифтом у відповідних вебформатах знаходяться у папці fonts, що розміщається на сервері. Властивість @ font-face має хорошу підтримку в браузерах, але різні браузери використовують різні формати шрифтів, тому це потрібно враховувати.

Складнощі використання втілених шрифтів

Файли шрифтів можуть бути великих розмірів, у деяких випадках додавання @ font-face уповільнює завантаження сторінок.

З деякими шрифтами пов'язані ліцензійні обмеження, що не допускають безкоштовного використання.

Деякі шрифти погано виглядають на вебсторінках.

Корисні ресурси

Font Squirrel (*http://www.fontsquirrel.com/tools/webfont-generator*) – сервіс, що дозволяє конвертувати вебшрифти різних форматів. Невеликий мінус – висота деяких символів після конвертації може розрізнятися. Шрифт завантажується на сервіс і конвертується.

drukarnia.com.ua (*https://drukarnia.com.ua/articles/pidbirka-bezkoshtovnikh-ukrayinskikh-shriftiv--jgxv*) – ресурс, що містить колекцію безкоштовних шрифтів для сайтів. На сайті також є форум, де можна задавати питання, що стосуються використання того чи іншого шрифту. Можна побачити як виглядатиме текст різного розміру. Використання API Google Font.

Google Font API (https://fonts.google.com/) – це безкоштовний сервіс від Google, який надає власникам сайтів можливість використовувати різні шрифти простим, зручним і ефективним способом.

Google Font API – надає високоякісні відкриті (open source) шрифти, які легко можуть бути використані у дизайні. Google Font API надає посилання на властивість CSS3 @ font-face. Після уставляння коду Google Font API до сторінки, він повертає таблицю стилів, що містить правило @ font-face для обраного шрифту, що може виглядати приблизно так:

```
<link href="https://fonts.googleapis.com/css?family=Open+Sans"
rel="stylesheet">
```

або

```
<style>
@import url('https://fonts.googleapis.com/css?family=Open+Sans');
</style>
```

Google виконує всю складну роботу по організації відображення шрифту в браузерах, що не підтримують CSS3 (наприклад, Internet Explorer).

Переваги використання Google Font:

Google Font API є одним із простих рішень для використання довільних шрифтів на веб ресурсах. За допомогою кількох рядків можна використовувати будь-який із представлених шрифтів Google Font.

Google Font API працює у всіх сучасних браузерах.

Всі шрифти Google Font API можна використовувати і в комерційних і персональних додатках.

Google Font API не пов'язаний з Javascript, таким чином шрифти виводяться навіть якщо користувач відключив Javascript.

Оскільки шрифти виводяться за допомогою CSS, то будь-які нововведення CSS3, наприклад, властивість text-shadow, можна додати до тексту.

Вибір типу шрифту – досить відповідальний етап. Шрифт – це частина вебпроєкту. Якщо вебвузол розробляється для банку або серйозної компанії, нереальним буде використання шрифту зразку Comic Sans, шрифт повинен бути підібраний строгим, що відповідає тематиці сайту.

Марно стверджувати, що існує самий кращий шрифт для верстання сайту. Якщо такий є, то для кожного розробника він свій.

Хороший екранний шрифт

До хороших екранних шрифтів можна віднести шрифти з такими характеристиками:

Низький контраст, прості блоки з відповідною вагою і товщиною.

Відповідна висота букв.

Оптимальна ширина символів і відстань між ними.

Наявність вільного простору в кожному символі.

Рисунок.8.4. Характеристики читабельного шрифту

Правила:

Шрифт за стилем повинен відповідати змістовному навантаженню сайту.

Зручність сприйняття і читабельності текстів відповідними шрифтами: довжина рядків, міжрядковий інтервал (висота рядків), розмір шрифту, співвідношення тексту і вільного простору на сторінці. Підвищення читабельності тексту забезпечує важливі чинники: увагу, швидкість прочитання та осмислення інформації.

Вибирати шрифти, що є доступними для користувачів більшості операційних систем, або забезпечити надійне використання довільних шрифтів.

Вебтипографіка домоглася значного прогресу за останні кілька років. Вона починає знаходити свої характерні риси, засновані на специфіці верстки статей, розміщення їх в Інтернеті, і відображення за допомогою браузерів.

Сьогодні шрифти і виглядають відмінно, і читаються виразно, окрім того, типографіка підказала дизайнерам цілий ряд прекрасних рішень в плані побудови сайтів.

8.2. Кодування тексту

Стандарт кодування ASCII

Розвиток кодування текстів відбувався одночасно з формуванням галузі IT і за цей час зазнав багато змін. Історично першим було кодування EBCDIC, яке дозволяло кодувати букви латинського алфавіту, арабські цифри і знаки пунктуації з керуючими символами.

Базовою точкою для розвитку сучасних кодувань текстів вважають кодування **ASCII** (*American Standard Code for Information Interchange*). Це кодування, що використовує 1 байт для опису одного символу. В ASCII описано 128 поширених символів – латинські літери, арабські цифри, розділові і деякі службові знаки: дужки, решітка, зірочка тощо (рис. 8.5):

;	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Рисунок 8.5. Базова таблиця кодування ASCII

Саме ці 128 символів із ASCII стали стандартом і в такому порядку присутні в інших кодуваннях. Але, за допомогою одного байта інформації можна закодувати не 128 (2^7), а 256 (2^8) різних значень, тому слідом за базовою версією ASCII з'явився цілий ряд розширених кодувань ASCII, в яких можна було крім 128 основних символів закодувати ще й символи національного кодування (наприклад, кирилицю).

Але в таблиці з символами кодування ASCII ці символи представлено в шістнадцятковому кодуванні. Наприклад, символ «зірочка» відповідає в ASCII шістнадцятковому числу 2A. Тут окрім арабських цифр використовуються латинські букви від A (десять) до F (п'ятнадцять).

Розширені ASCII кодування CP866, KOI8-R із псевдографікою

Символи на екрані комп'ютера формуються на основі двох речей – наборів векторних форм (представлень) різних символів (вони знаходяться у файлах із шрифтами, які встановлено на комп'ютері) і коду, який дозволяє витягнути з цього набору векторних форм (файлу шрифту) саме той символ, який потрібно вставити.

За векторні форми символів відповідають файли шрифтів, а за кодування відповідає операційна система і програми, що в ній використовуються. Тобто, любий текст на комп'ютері представлено набором байтів, у кожному з яких закодовано один символ тексту.

Програма, що відображає текст на екрані (текстовий редактор, браузер тощо), при розборі коду зчитує кодування чергового символу і шукає відповідну для нього векторну форму в потрібному файлі шрифту, який під’єднано для відображення даного текстового документа.

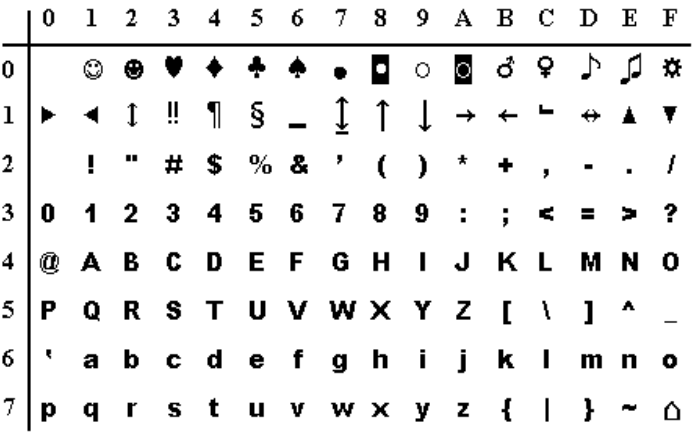
Щоб закодувати потрібний символ (наприклад, національного алфавіту) має бути виконано дві умови – векторна форма цього символу повинна бути у вживаному шрифті і цей символ можна було б закодувати в розширених кодуваннях ASCII в один байт. Тому, розширених кодувань ASCII існує багато. Лише для кодування символів кирилиці існує кілька варіантів розширеного кодування ASCII.

Кириличне кодування CP866

Першим з'явилося кодування тексту CP866, яке поширила компанія IBM. У ньому була можливість використовувати символи російського алфавіту і це кодування було розширеною версією кодування ASCII. Його верхня частина повністю збігалася з базовою версією ASCII (128 символів латиниці, цифр і знаків), яка представлена на наведеному вище скріншоті, а нижня частина таблиці з кодуванням CP866 мала зазначений на скріншоті нижче вид і дозволяла закодувати ще 128 символів (російські літери і псевдографіка) (рис. 8.8.).

У правому стовпчику цифри кодування починаються з 8, тому цифри з 0 до 7 відносяться до базового кодування ASCII. Наприклад, російська буква «М» у кодуванні CP866 матиме код 9C (вона знаходиться на перетині відповідних

рядка з 9 та стовпця з цифрою C у шістнадцятковій системі числення), який можна записати в одному байті інформації і за наявності відповідного шрифту з російськими символами ця буква без проблем відобразиться в тексті.



Риунок 8.6. Символи розширеної ASCII (перша половина)

Рисунок 8.7. Розширене кодування IBM

Кодування CP866 має велику кількість символів псевдографіки, оскільки розроблялося в роки, коли графічних операційних систем не було. А в DOS і

подібних текстових операційних системах, псевдографіка дозволяла дещо урізноманітнити оформлення текстів. Тому, в кодуванні CP866 та інших розширених кодуваннях ASCII тих років так багато символів псевдографіки.

Рисунок 8.8. Таблиця кодування CP866

Кириличне кодування KOI8-R

Кодування KOI8-R подібне до CP866 – кожен символ тексту кодується одним байтом. На скріншоті показано другу половину таблиці KOI8-R, оскільки перша половина повністю відповідає базовій версії ASCII (рис. 8.9).

Серед особливостей кодування KOI8-R можна відзначити те, що кириличні літери в її таблиці йдуть не в алфавітному порядку, як у кодуванні CP866. Якщо порівняти з базовою версією кодування ASCII (яка входить у всі розширені кодування), то можна помітити, що в KOI8-R кириличні літери розташовані в тих же елементах таблиці, що і співзвучні їм літери латинського алфавіту з першої частини таблиці. Це було зроблено для зручності переходу з кирилиці на латинку шляхом відкиданням всього одного біта (2^7 або 128).

Кодування KOI8-R використовувалася як транспортне кодування в Internet і як основне кодування в більшості безкоштовних операційних систем.

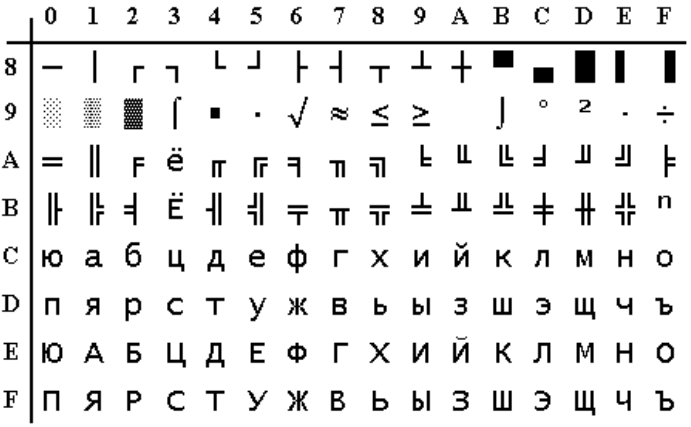


Рисунок 8.9. Таблиця кодування KOI8-R

Кириличне кодування ISO **8859-5**

Кодування ISO 8859-5 застосовується в більшості комерційних UNIX-сумісних операційних систем (рис. 8.10).

Рисунок.8.10. Таблиця кодування ISO 8859-5

Windows 1251 (розширена кодування ASCII)

Подальший розвиток кодувань тексту був пов'язаний із поширенням графічних операційних систем. Необхідність використання псевдографіки в текстах із часом зникла, тому її було забрано зі складу кодувань. У результаті виникла ціла група кодувань, які за своєю суттю і раніше були розширеними версіями ASCII (один символ тексту кодується всього одним байтом інформації), але вже без використання символів псевдографіки.

Такі кодування було розроблено американським інститутом стандартизації і відносилися до ANSI кодування. Для кирилиці популярним стало кодування **Windows 1251**.

Кодування Windows 1251 вигідно відрізнялося від ранішніх CP866 і KOI8-R. Тут місце символів псевдографіки зайняли додаткові символи російської типографіки (окрім знака наголосу), а також символи, що використовуються в близьких до російської слов'янських мовах (української, білоруської тощо) (рис. 8.11):

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Ђ	Ѓ	‚	ѓ	„	…	†	‡	€	‰	Љ	‹	Њ	Ќ	Ћ	Џ
9	ђ	‘	’	“	”	•	–	—	™	™	љ	›	њ	ќ	ћ	џ
A		Ў	ў	Ј	¤	Ґ	¦	§	Ё	©	Є	«	¬	—	®	Ї
B	°	±	І	і	ґ	µ	¶	·	ё	№	є	»	ј	Ѕ	ѕ	ї
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Э	Ю	Я
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я

Рисунок 8.11. Таблиця кодування Windows 1251

Заміна тексту на псевдосимволи

Через надмірну кількість кодувань текстів у виробників шрифтів і виробників програмного забезпечення постійно виникали проблеми. У разі невірно

використаного кодування в тексті у відвідувачів замість тексту міг відображатися безглуздий набір псевдосимволів.

Така проблема часто виникала при відправленні чи отриманні повідомлень електронною поштою, у результаті некоректного розкодування кирилиці, яке не відповідала початковому кодуванню текстового повідомлення.

Якщо символи, які закодовано за допомогою CP866, спробувати відобразити, використовуючи кодову таблицю Windows 1251, то з'являється набір псевдосимволів і повністю заміняє собою текст повідомлення (рис. 8.9).

Рисунок 8.12. Відображення псевдосимволів у разі невідповідного кодування

Проблему вирішували в різний, не завжди ефективний спосіб: користувачі для листування часто використовували транслітерацію латинських букв; поштові служби створювали складні перекодувальні таблиці.

Аналогічна ситуація часто виникає при створенні та налаштуванні сайтів, форумів або блогів, коли текст із кирилицею помилково зберігається не в тому кодуванні, яке використовується на сайті за замовчуванням або не в тому текстовому редакторі, який додає в код кодування від себе.

Зрештою така ситуація з множиною кодувань і постійними проблемами з відображенням символів національного алфавіту стала нестерпною, і з'явилися передумови до створення нового універсального кодування, яке б замінило собою всі існуючі і вирішило б, нарешті, на корені, проблему з появою псевдосимволів, що не читаються.

Юнікод – універсальне кодування тексту (UTF 32, UTF 16 і UTF 8)

У світі окрім кирилиці існують інші алфавіти, із значно більшою кількістю символів ніж в кирилиці. Наприклад, символів мовної групи південно-східної Азії є тисячі і їх неможливо описати в одному байті інформації, який виділяється для кодування символів в ASCII.

Для вирішення цієї проблеми було створено консорціум Юнікод (Unicode – Unicode Consortium) при співпраці багатьох лідерів IT-індустрії (виробники програмного та апаратного забезпечення, розробники шрифтів), які були зацікавлені у появі універсального кодування тексту.

Першим кодуванням тексту, що вийшов під егідою консорціуму Юнікод, було кодування UTF 32. Цифра в назві кодування UTF 32 означає кількість біт, що використовується для кодування одного символу. Для кодування одного символу в новому універсальному кодуванні UTF 32 використано 4 байти інформації.

У результаті чого файл із текстом, що закодований в UTF 32 матиме розмір в чотири рази більше, ніж у розширеному кодуванні ASCII, зате з'явилася можливість закодувати 232 символів.

Але багатьом країнам із мовами європейської групи такої величезної кількості символів використовувати в кодуванні зовсім необхідності не було, однак при використанні UTF 32 вони отримували чотириразове збільшення ваги текстових документів, а в результаті і збільшення обсягу Інтернет-трафіку і обсягу збережених даних. Це багато і таке марнотратство собі ніхто не міг дозволити.

Подальшим розвитком універсального кодуванням стало UTF 16, яке вийшло настільки вдалим, що було прийнято за замовчуванням як базовий простір для всіх символів, які використовуються. Для кодування одного символу UTF 16

використовує два байти. У UTF 16 можна закодувати 65536 символів (216), що було прийнято за базовий простір в Юнікод.

Наприклад, в операційній системі Windows в «Таблиці символів» можна переглянути векторні форми всіх встановлених у системі шрифтів (рис. 8.12).

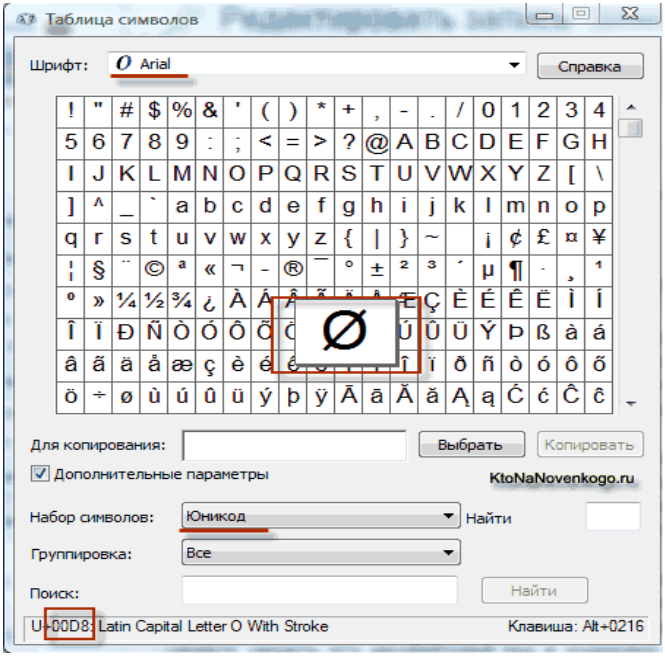


Рисунок 8.13. Перегляд шрифтів у Таблиці символів

Якщо вибрати в Додаткових параметрах набір символів Юнікод, то можна побачити для кожного шрифту окремо весь асортимент символів, що у ньому містяться. Якщо клацнути по будь-якому з цих символів– можна побачити його багатобайтовий код у кодуванні UTF 16, що складається з чотирьох шістнадцяткових цифр.

Вдала версія UTF 16 знов не принесла переваг для європейських користувачів, бо у них після переходу від розширеної версії кодування ASCII до UTF 16 вага документів збільшувалася в два рази (один байт на один символ в ASCII і два байти на той же символ у кодуванні UTF 16).

Саме для вирішення цього в консорціумі Юнікод запропоновано кодування тексту змінної довжини UTF 8. Він є повноцінним кодуванням змінної довжини, тобто кожен символ тексту може бути закодований у послідовність довжиною від одного до шести байт. На практиці ж у UTF 8 використовується тільки діапазон від одного до чотирьох байт, оскільки більше за чотири байти коду нічого вже навіть теоретично не можливо уявити.

В UTF 8 всі латинські символи кодуються в один байт, подібно до ASCII. У разі кодування тільки латиниці, навіть ті програми, які не розуміють Юнікод, все одно прочитають те, що закодовано в UTF 8. Тобто, базова частина кодування ASCII перейшла в UTF 8. Кириличні символи в UTF 8 кодуються в два байти, а, наприклад, грузинські – в три байти.

Після створення кодувань UTF 16 і UTF 8 консорціум Юнікод вирішив основну проблему – тепер у шрифтах існує єдиний кодовий простір. Виробникам шрифтів залишається тільки заповнювати цей кодовий простір розробленими векторними формами символів тексту.

У наведеній вище «Таблиці символів» видно, що різні шрифти підтримують різну кількість символів. Деякі насичені символами Юнікоду шрифти можуть мати велику вагу. Тепер шрифти відрізняються не тим, що вони створені для різних кодувань, а тим, що виробник шрифту заповнив або не заповнив єдиний кодовий простір векторними формами символів.

Кодування UTF 8 з BOM

Коли розробляли кодування UTF 16, було вирішено додати до неї можливість записувати код символу, як у прямій послідовності (наприклад, 0A15), так і в зворотній (150A). Для того, щоб програми розуміли в якій саме послідовності читати код символу в UTF 16 застосовують **BOM** (*Byte Order Mark*) – сигнатуру, яка містить три байти і додається на початок документів.

Консорціум Юнікод у кодуванні UTF 8 жодних BOM не передбачав і додавання сигнатури (додаткових трьох байтів у початок документа) деякими програмами заважає читати кодування Юнікод. Тому, завжди при збереженні файлів у кодуванні UTF 8 варто вибирати варіант **без BOM** (без сигнатури). Це допоможе запобігти від можливого відтворення псевдосимволів.

Деякі програми у Windows не вміють цього робити (зберігати текст у UTF 8 без BOM), наприклад, Блокнот Windows. Він зберігає документ у UTF 8, але додає на його початок сигнатуру (три додаткових байти). Ці байти сигнатури UTF 8 будуть завжди примушувати читати код у прямій послідовності. На серверах через цю дрібницю може виникнути проблема – з'являються псевдосимволи.

Інформацію про застосоване кодування слід прописати в HTML коді, щоб на сервері або локальному хості не виникло плутанини.

Зазначення кодування тексту

Для зазначення кодування тексту в HTML використовується елемент META, який прописується між тегами HEAD:

```
<head>
	<meta charset="utf-8">
</head>
```

Елемент META із зазначенням застосованого кодування краще ставити якомога вище в документі, щоб на момент появи в тексті першого символу не з базового кодування ANSI браузер вже повинен мати інформацію про те, як інтерпретувати коди цих символів.

Поради

Значна частка контенту, який потрібен користувачу – це текст: опис товарів, способи замовлення послуги, порівняння марок тощо. Багато користувачів не просто шукають, а порівнюють, читають аналітичні матеріали та статті на сайтах, тому, для них потрібно забезпечити максимальний комфорт і читабельність.

Нечитабельний текст на сайті розчаровує користувача, який скоріше проігнорує і текст і сайт. Дрібний шрифт, надвеликі текстові блоки та відсутність підзаголовків відлякують значний сегмент цільової аудиторії.

Основні параметри розміщення тексту та типові помилки:

Шрифт. Існує два типи шрифтів: із засічками (serif) і без засічок (sans serif).

SERIF SANS SERIF

Для перегляду на екрані використовується шрифт без засічок, шрифти класу serif підходять краще для друкованої продукції. Це пояснюється тим, що з екрану читати складніше, а невелика роздільна здатність дисплеїв не надає високої якості для дрібних засічок. Стандартними і популярними для основного тексту є шрифти Arial, Verdana, Helvetica. Розміщення на сайті тексту в стандартному шрифті Times New Roman виглядає убого і нагадує саморобні безкоштовні сайти.

На сайті краще використовувати один-два шрифти в різних накресленнях (грубіший, курсив). Більша кількість шрифтів виглядає неприємно, змушуючи користувача підсвідомо нервувати.

Колір шрифту. Кольори фону і основного тексту повинні бути контрастними. Чим контрастніше текст, тим він краще сприймається. Для з'ясування достатньої контрастності кольорів, потрібно створити дублікат макету сторінки і перетворити зображення у відтінки сірого, якщо текст читабельний, тоді контрастність обраних кольорів буде достатньою.

На сайті не слід використовувати більше чотирьох різних кольорів тексту. В ідеалі на сайті повинен бути темний текст на світлому фоні, або навпаки. Не варто використовувати синій колір для шрифту, оскільки він традиційно використовується для позначення посилань. Найгіршими поєднання вважають червоний чи синій на чорному тлі та світло-сірий на білому.

Розмір шрифту. Розмір шрифту для основного тексту повинен бути в межах від 16 до 20 пікселів, великий шрифт дратує, дрібний є не читабельним. Для заголовків таких обмежень немає, і якщо дизайн допускає – шрифт може бути доволі великого розміру.

Інтерліньяж. Це відстань між рядками. У тексті інтерліньяж повинен бути в півтори рази більше, ніж у друкованих виданнях, це полегшує перегляд і читання тексту.

Абзаци. Тут немає якихось жорстких обмежень за розміром і кількістю, але завжди хочеться зробити текст не просто інформативним, але і красивим.

Відносно довжини абзацу, яка буде оптимальною для швидкості читання, існує припущення, що людське око в певний момент часу може сфокусуватися

приблизно на 5-7 рядках тексту довжиною приблизно в 10 слів. Вважається, за таких умов збільшується швидкість читання і сприйняття матеріалу. Занадто великі абзаци погано сприймаються, занадто короткі виглядають «брудно», тому варто самотні рядки варто об'єднувати до одного абзацу.

Вирівнювання тексту за лівим краєм збільшує швидкість читання, оскільки рівний лівий край допомагає знайти початок нового рядка. Вирівнювання тексту за шириною може надати вкрай небажані великі відстані між словами, тому використовується рідко.

У друкованому виданні для виділення наступного абзацу використовується відступ першого рядка. Це дозволяє читачеві легко відшукувати поглядом новий абзац підвищуючи, таким чином, читабельність тексту. Для текстів на вебсторінках цей прийом не використовується, а для розмежування абзаців застосовується збільшені відстані між абзацами.

Підзаголовки. Тексти довше за 1,5-2 тисяч знаків потрібно ділити на підрозділи з заголовками. Зазвичай, користувач спочатку переглядає текст у пошуках потрібної інформації. Підзаголовки економлять час і дозволяють швидше знайти те, що потрібно.

Шрифтові виділення. В арсеналі дизайнера є три види виділення основних думок тексту: грубіший шрифт, курсив і великі літери (капслок). Виділення грубішим шрифтом використовується для позначення головного слова абзацу, його тематики, курсив традиційно позначає головну думку автора, а великі літери використовуються для резюмування і привертання уваги. Не рекомендується використовувати великі літери занадто часто – це на мові шрифту позначає крик і дратує відвідувача.

Виділення посилань. Доцільним вибором будуть традиційні кольори: синій для посилання, фіолетовим для відвіданого пункту. Якщо існуючий дизайн сайту не передбачає цього, тоді слід пам'ятати правило: посилання повинні бути виділені в інший ніж основний колір, текст посилання слід підкреслювати, використані посилання повинні відрізнятися за кольором від активних.

Не варто підкреслювати текст, який не є посиланням – це вводить користувача в оману. Людина заходить на сайт, бачить щось схоже на посилання (те, що повинно бути посиланням, судячи з його досвіду), натискає і не отримує нічого.

У мережі занадто велика конкуренція, щоб розчаровувати і ображати користувача такими дрібницями.

Порядок у вихідному макеті дизайну

Всі роботи по створенню макету сайту проводяться в графічному редакторі. Беззаперечним лідером серед редакторів, що використовують вебдизайнери, є Adobe PhotoShop.

Правильно складений макет повинен бути збережений у форматі *.psd із чіткою ієрархією прошарків та груп прошарків.

Вихідний макет очищають від елементів, які не використовуються, всі прошарки йменуються, для текстових прошарків роблять растрові копії. Прошарки, що відносяться до одного елементу чи блоку групуються по папках. Зайві прошарки макету видаляються, щоб не заплутувати верстальника та зменшити розмір файлу.

Наявність напрямних по модульній сітці макету. Такий порядок дозволяє швидко орієнтуватися в макеті, легко виділити, замінити чи змінити потрібний елемент.

Несистемні шрифти додаються до папки з макетами, оскільки PSD-файл не зберігає їхні дані. Для верстання за допомогою @font-face даний шрифт втілюється у сторінки, і верстальник не витрачає час для пошуку потрібного шрифту.

Розроблений макет також зберігають в популярному растровому форматі (.jpg або .png) для перегляду та остаточного затвердження клієнтом.

Вихідні файли з текстовими поясненнями передаються до верстальника, який буде робити шаблони сторінок. Через різне кодування в операційних системах важливо використовувати латиницю в назвах.

Для дизайнера варто мати базові знання щодо HTML та CSS кодування, принципи та основи верстки і тих фреймворків, що використовуються. Це дозволить створювати елементи та ефекти, які можна легко реалізувати кодом, мінімізуючи використання графічних зображень.

8.3. Верстання сторінки

По закінченні роботи зі створення графічного макету дизайну і схвалення його іншими учасниками проєкту чи замовником, приступають до створення HTML-шаблону сторінки.

Верстка – це процес перетворення дизайну сайту (картинки) у вебсторінку. Макет дизайну – це зображення у форматі PSD (Photoshop), а шаблон вебсторінки – це інтерактивне відображення сторінки, де можна виділити текст, перейти за посиланням, заповнити форму, додати товар в корзину і так далі.

Верстання сторінок – це процес написання HTML-коду сторінки, за яким сторінка набуває вигляд, подібний до дизайну макету. Безумовно, існують різні технології розмітки тексту, які підтримуються браузерами, проте, найпоширенішим варіантом є верстання сторінок за допомогою мови HTML та стилів CSS.

Верстка сторінки робиться поетапно: спочатку створюється HTML-структура (HTML-код), додаються стилі (CSS-код), а далі за потребою пишуться скрипти (JS), додаються необхідні плагіни і бібліотеки.

Для того, щоб навчитися верстанню, слід спочатку зрозуміти значення операторів HTML та CSS і приступати до вивчення верстки. Для початківців починати слід із верстання простих структур, але з часом ускладнювати завдання. Питання як зверстати це, як зверстати те відпадуть з накопиченням досвіду.

Наприклад, почати з верхньої частини сторінки: створити контейнер, додати до нього фон, встановити логотип картинкою або текстом. Далі – створення меню по частинах: вивести пункти, вирівняти їх, додати поля і відступи, зробити hover-ефект. І так поступово слід верстати всю сторінку.

Поради для початківців:

Вивчати HTML та CSS по свіжих матеріалах і від фахівців, що мають авторитет серед спільноти розробників.

Намагатися верстати, відтворювати сторінки відео-уроків, вивчати код готових шаблонів, яких є багато у відкритому доступі.

Накопичений досвід сприятиме доведення навичок верстальника до автоматизму, і в подальшому верстання сторінок відбуватиметься швидше і якісніше.

Редактори коду

З найбільш популярних редакторів коду на сьогодні можна виділити три, які схожі за принципом роботи, до них додатково можна під'єднати необхідні модулі та плагіни:

Sublime Text (http://www.sublimetext.com/3)
Atom (https://atom.io/)
Brackets (http://brackets.io/)
I

снують більш просунуті комплекси IDE (**Integrated Development Environment – Інтегроване середовище розробки)** такі, як Adobe DreaWeaver, Web Storm, PHP Storm тощо, але для верстки нескладного проєкту цілком підійде редактор коду, а не цілий комплекс.

Блокова верстка

Сучасний підхід до HTML-верстки передбачає розподілення елементів сегментів у різних місцях, і це називається «блоковою версткою». Блокова верстка базується на принципах розташування і взаємодії за допомогою блокових елементів.

В HTML4 як базовий елемент розмітки використовувався тег div, що не має певного семантичного значення та особливих вимог на його вміст. Тег ** ** є контейнером для тексту, зображень та інших елементів сторінки і вони позиціонуються в різних місцях. Теги **div** можна помістити в будь-яке місце вебсторінки, позиціонувати абсолютно (вказати координати **x** і **y**) або відносно (вказати відстань до інших елементів сторінки). На рис. 8.13 показано просту структуру сторінки в кодуванні HTML4.

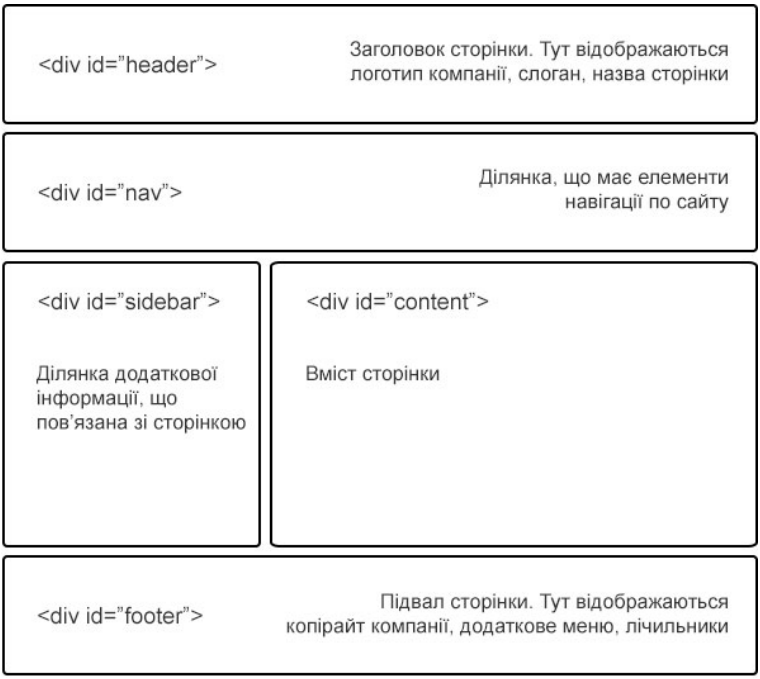


Рисунок 8.14. Типова структура сторінки для HTML4

У HTML5 з'явилися нові елементи, які служать заміною для звичайних блочних елементів з атрибутами class і id, наприклад, або . Натепер можна використати семантично змістовні блочні теги, що за призначенням аналогічні до тегу div. Це теги section, article, aside, header, footer та інші (рис. 8.14).

Для спрощення роботи можна скористатися генераторами структури сторінки, які створюють код загальної структури сторінки з CSS-файлом, в якому прописані загальні стилі. Як приклад таких генераторів – шаблонізатор Dreamweaver, онлайн-генератори CSStemplater.com і CSScreator.com.

Особливості сторінки з блочною верстkою:

Легкий і зрозумілий код сторінки – компактній, коментований, не захаращений зайвими стилями.

Блокова верстка дуже «дружньо» відноситься до позиціонування. Блоки можуть перекривати один одного, розробник вказує, який з блоків буде зверху.

Блокова верстка дозволяє розмістити блок, що відображає будь-яку частину сторінки на початку html-коду. Це буде доречним при застосуванні певних технологій просування сайтів.

Максимально ідентично відображає сторінку в різних браузерах при різних розмірах екрану.

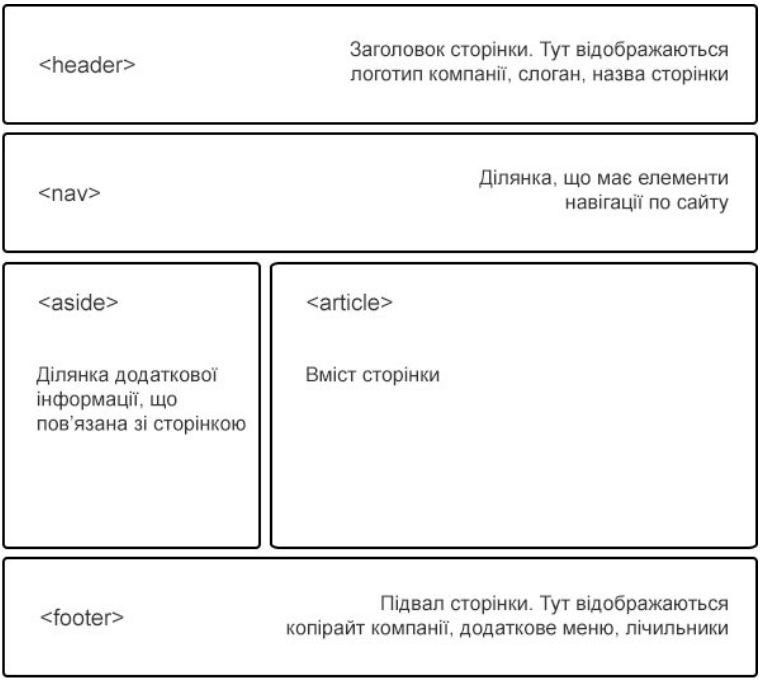


Рисунок 8.15.Типова структура сторінки для HTML5

Перевірка HTML і CSS-коду

Перевіряти код сторінки сайту необхідно, коли здійснюються кардинальні зміни в структурі сайту. Акуратний верстальник прагне це робити якнайчастіше. Перевірку HTML і CSS-коду краще робити з використанням

програм-валідаторів чи інтернет-сервісів, які відображають знайдені помилки відповідно до різних стандартів. Кращим валідатором за допомогою якого можна перевірити будь-яку сторінку, що розташована в Інтернеті або на локальному комп'ютері, є валідатор міжнародного Консорціуму W3C **(WorldWideWebConsortium)** *http://validator.w3.org/.*

Сучасні браузери підтримують стандарти W3C значно краще, ніж їх попередні версії. Якщо правильний код відповідає певним формальним правилам, його легше інтерпретувати й обробляти, він швидше аналізується і відображається в браузері, з ним легше працювати пошуковим системам.

Відповідність коду сторінки до стандартів W3C не є гарантією якості та ефективності. Грамотний код – це важливий, але далеко не єдиний показник для оцінювання сторінки.

Поради до HTML та CSS-коду

1. Кросбраузерність

Сайт повинен нормально працювати як в останніх, так і старших версіях популярних браузерів, зокрема Internet Explorer, Mozilla FireFox, Opera, Chrome, Edge.

2. Застосування коментарів

Основні HTML-блоки коментарями коментуються в такий спосіб:

```
<!-- BEGIN FOOTER -->
<!-- END FOOTER -->
```

CSS блоки в такий:

```
/* FOOTER */
```

Якщо використовуються CSS-префікси чи хакі, також потрібні коментарі, що це є і для якого саме браузера. Коментарі допомагають орієнтуватися в коді не лише верстальнику, але і решті учасників проєкту: програмісту, контент-менеджеру, оптимізатору.

3. Впорядкування в «Таблиці стілів»

CSS-файл повинен бути розділений за допомогою коментарів на блоки за функціональним або структурним призначенням, наприклад:

```
/* 1. Скидання CSS              */
/* 2. Типові елементи    */
/* 2.1. Заголовки               */
/* 2.2. Посилання               */
/* 2.3. Елементи форм    */
/* 3. HEADER (Шапка сайту) */
/* 4. FOOTER (підвал)    */
/* 5. SIDEBAR (Праворуч) */
```

4. Змістовні назви

Назви класів і id повинні за змістом відповідати застосуванню, наприклад, header, menu, footer, news.

5. Вживання Javascript

Все що можна зробити без скрипту, робити без нього. Якщо Javascript-коду багато – потрібно його виносити в окремий файл. Обробники подій теж краще відокремити і оголошувати в окремому файлі.

Заздалегідь узгодити JavaScript-бібліотеки, наприклад, jQuery чи PrototypeJS.

Якщо в макеті присутній JavaScript, що змінює DOM, – потрібно прослідкувати його поведінку в різних браузерах.

6. Ширина сторінки

Для еластичних макетів обов'язково повинна бути задана мінімальна і максимальна ширина. Для фіксованих дизайнів ширина має бути 960-980 пікселів. У іншому випадку може з’явитися горизонтальний скролінг, що свідчить про низький професійний рівень розробника.

7. Особливості відображення

Обов’язково вказувати колір для фону сторінки (body), навіть якщо він білий. Певна частка відвідувачів встановлює за замовченням фоновий колір браузера, відмінний від білого, що призводить до відображення сторінки не в тому вигляді як задумав дизайнер.

8. Порядок у файлах

Файлова структура повинна бути стрункою і не містити файлів, які не використовуються в сайті (HTML-файли, зображення, стилі, скрипти).

9. Розташування футера у нижньому боці браузера

У макетах, де висота сторінки залежить від контенту (а таких, як правило, більшість), футер має триматися низу браузера при відсутності або малій кількості контенту.

10. Кодування тексту

На сьогоднішній день для тексту стандартним є кодування UTF-8. Це кодування застосовують для HTML, CSS та JS-коду, інакше ймовірність довгого виправлення багів критично зростає.

11. Використання нестандартних шрифтів

Якщо в макеті використовуються нестандартні шрифти, слід з’ясувати за якою технологією їх буде втілено: @ font-face, Google Font API чи інше.

12. Використання невалідних фрагментів

Якщо макет не проходить 100% HTML-валідацію, варто вживати лише виправданий невалідний код.

13. Звітність про виправлення

Якщо здача верстки проводиться більш ніж одним етапом (наприклад, верстальник відправляє сторінки по одній, або якщо йому повертаються на доопрацювання вже зверстані сторінки), а система контролю версій для верстки не використовується, тоді виконавець повинен в обов'язковому порядку прикріпити файл з описом змін в макеті приблизно такого змісту:

Додано нові картинки в папку img.

Картинки btnHome.jpg і btnFeedback.jpg вже не потрібні, можна видаляти.

Змінено html-код в секції файлу anketa.html.

Додано в кінець файлу main.css нові стилі (починаючи з .form_row і нижче).

9. ВЕБАВТОРСТВО (ТОНКОЩІ ВЕРСТАННЯ)

9.1. Кодинг

По закінченні роботи зі створення графічного макету дизайну і схвалення його іншими учасниками проєкту чи замовником, приступають до створення HTML-шаблону сторінки.

Верстання сторінок – це процес написання HTML чи XML-коду сторінки, при якому сторінка набуває вигляд, подібний до дизайну макету. Безумовно, існують і інші технології розмітки тексту, які підтримуються браузерами, проте, найпоширенішим варіантом є верстання сторінок за допомогою мови HTML та стилів CSS.

До HTML-верстки можна застосувати два підходи щодо розподілення елементів сегментів у різних місцях.

Таблична верстка

Таблична верстка – умовна назва методу верстання HTML-документів, при якому за структурну основу для розташування текстових чи графічних елементів документа використовуються таблиці.

Метод широко застосовувався до появи стандартів CSS, оскільки на той момент не було іншої можливості точно розташувати елементи на сторінці. Таблиці є спроможними автоматично змінювати свої розміри відповідно до вмісту, і навпаки, тримати точні розміри певних комірок, дозволяють швидко і зручно розставити по комірках різнорідну текстову чи графічну інформацію.

Таблиці в HTML можуть бути вкладеними, що дозволяє створювати цілі ієрархії таблиць при верстанні складних сторінок, окремі елементи яких мають зберігати своє розташування і розмір на екрані незалежно від розміру вікна браузера, тоді як інші елементи, навпаки, мають змінюватися в розмірах або змінювати своє розташування щодо решти об'єктів документа.

Таблична верстка натепер є застарілою і практично не використовується.

Блокова верстка

На відміну від табличного способу розташування даних блокова верстка не потребує чіткої прив'язки кожного логічного блоку до певної комірки. Блокова верстка базується на абсолютно інших принципах розташування і взаємодії за допомогою блокових елементів і .

HTML-тег є контейнером для тексту, зображень та інших елементів сторінки. При створенні HTML-макета теги div розміщуються на сторінці, в них додається вміст і вони позиціонуються в різних місцях.

На відміну від комірок таблиці, які існують лише всередині рядків і стовпців таблиці, теги div можна помістити в будь-яке місце вебсторінки. Теги div можна позиціонувати абсолютно (вказати координати **x** і **y**) або відносно (вказати відстань до інших елементів сторінки).

Для спрощення роботи можна скористатися генераторами структури сторінки які створюють HTML-сторінку з кодом загальної структури і CSS-сторінку з загальними стилями. Як приклад таких генераторів – шаблонізатор Dreamweaver, онлайн-генератори

Перевірка HTML і CSS коду

Перевіряти код сторінки сайту необхідно, коли здійснюються кардинальні зміни в структурі сайту. Акуратний верстальник прагне це робити якнайчастіше. Перевірку HTML і CS-коду краще робити з використанням програм-валідаторів чи інтернет-сервісів, які відображають знайдені помилки відповідно до стандартів W3C.

Кращим валідатором за допомогою якого можна перевірити будь-яку сторінку, що розташована в Інтернеті або на локальному комп'ютері, є валідатор Консорціуму W3C *http://validator.w3.org/*.

Сучасні браузери підтримують стандарти W3C значно краще, ніж їх попередні версії. Якщо правильний код відповідає певним формальним правилам, його

легше інтерпретувати й обробляти, він швидше аналізується і відображається в браузері, з ним легше працювати пошуковим і індексуючим системам.

Піклуючись про відповідність коду до стандартів W3C, не треба впадати в крайнощі і думати, що якщо сторінка успішно перевірена валідатором, то вона автоматично є якісною і ефективною. Грамотний код – це важливий, але далеко не єдиний показник якості сторінки.

1. Перевірка відстежує синтаксичні помилки

Під час верстання розробник може здійснити дрібні помилки, наприклад, неправильно вкладені теги, пропущені дужки чи лапки, які важко відстежити в надвеликому масиві коду. Автоматизована система швидко виявляє ці недоречності і безперечно буде корисною.

2. Перевірка сторінок сайту визначає коректність відображення в різних браузерах

Сторінка може містити HTML і CSS-елементи, які не відображаються в якомусь з популярних браузерів. Тому, відвідувач з таким браузером побачить або некоректне відображення коду або отримає непрацюючий динамічний елемент, наприклад, випадне меню чи слайдер.

Правильний HTML-код, зазвичай, гарантує вірне функціонування сайту, втім браузери можуть мати певні особливості в дотриманні існуючих HTML і CSS-стандартів.

3. Пошукова видимість

Помилки на сторінках браузери намагаються компенсувати по-різному. Деякі браузери можуть ігнорувати помилкові елементи, в той час як інші – відображають в свій спосіб.

Різні пошукові роботи також повинні приймати рішення щодо виявлених помилок, в результаті чого в деяких випадках сторінка може бути не проіндексованою або проіндексованою з певними обмеженнями.

4. Професіоналізм у створенні сайту

Перевірка вебсайту в різних браузерах свідчить про професійний підхід. Непрацюючі або некоректно відображені елементи свідчать, що вебдизайнер

не знає тонкощів своєї роботи або ставиться до неї недбало, що в підсумку відбивається на престижі сайту.

Найпоширеніші вимоги до HTML та CSS-коду

1. Кросбраузерність

Сайт повинен нормально працювати як в останніх так і старших версіях популярних браузерів, зокрема IE7 +, FF3 +, Opera9 +, Safari4 +, Chrome.

2. Застосування коментарів

Основні HTML-блоки коментарями коментуються в такий спосіб:

<! --- BEGIN FOOTER ->
<! --- END FOOTER ->
CSS блоки в такий:

/* FOOTER */
Якщо використовуються CSS-префікси чи хакі, також потрібні коментарі, що це і для якого браузера. Коментарі допомагають орієнтуватися в коді не лише верстальнику, але і решті учасників проєкту: програмісту, контент-менеджеру, оптимізатору.

3. Впорядкування в «Таблиці стілів»

CSS-файл повинен бути розбитий за допомогою рядків з коментарями на блоки за функціональним або структурним призначенням, наприклад:

```
/ * _____ 1. Скидання CSS_____________ * /
/ * _____ 2. Типові елементи__________ * /
/ * _____ 2.1. Заголовки_____________ * /
/ * _____ 2.2. Посилання_____________ * /
/ * _____ 2.3. Елементи форм_________ * /
/ * _____3. HEADER (Шапка сайту) ____ * /
/ * _____4. FOOTER (підвал) _________ * /
/ *_____5. SIDEBAR (Праворуч) ______ * /
```

4. Змістовні назви

Назви класів і id повинні за змістом відповідати застосуванню, наприклад, *header, menu, footer, news.*

5. Вживання Javascript

Все що можна зробити без Javascript – робити без нього. Якщо Javascript-коду багато – потрібно його виносити в окремий файл. Обробники подій теж краще відокремити і оголошувати в окремому файлі.

Заздалегідь узгодити JavaScript-бібліотеки: PrototypeJS чи jQuery.

Якщо в макеті присутній JS, що змінює DOM, – потрібно прослідкувати його поведінку в Опері, оскільки там часто для відвідувачів надається документ з кешу.

Якщо при натисканні на посилання відбувається обробка подій, потрібно, щоб обробники подій повертали false, або ж використати href = 'javascript: void (0)' замість популярного href = '#', щоб сторінка не поверталася вгору.

6. Ширина сторінки

Для еластичних макетів обов'язково повинна бути задана мінімальна і максимальна ширина. Для фіксованих дизайнів ширина має бути 960-980 пікселів. В іншому випадку може з’явитися горизонтальний скролінг, що свідчить про низький професійний рівень розробника.

7. Особливості відображення

Обов’язково вказувати колір фону для body, навіть якщо він білий. Певна частка відвідувачів встановлює за замовченням фоновий колір браузера, відмінний від білого, що призводить до відображення не в тому вигляді як задумав дизайнер.

8. Порядок в файлах

Файлова структура повинна бути стрункою і не містити файлів, які не використовуються в сайті (HTML-файли, зображення, стилі, скрипти).

9. Розташування футера у нижньому боці браузера

В макетах, де висота сторінки залежить від контенту (а таких, як правило, більшість), футер має триматися низу браузера при відсутностості або малій кількості контенту.

10. Кодування тексту

Обумовити, в якому кодуванні (наприклад, UTF-8, windows1251) має бути HTML-макет. CSS і JS-файли повинні бути в тому ж кодуванні, що і макет, інакше ймовірність довгого та виснажливого виправлення багів критично зростає.

11. Використання нестандартних шрифтів

Якщо в макеті використовуються нестандартні шрифти, слід з'ясувати за якою технологією їх буде втілено: @ font-face, Cufon, Google Font API чи інше.

12. Використання невалідних фрагментів

Якщо макет не проходить 100%-у HTML-валідацію, варто вживати лише виправданий невалідний код.

13. Звітність про виправлення

Якщо здача верстки проводиться більш ніж одним етапом (наприклад, верстальник відправляє сторінки по одній, або якщо йому повертаються на доопрацювання вже зверстані сторінки), а система контролю версій для верстки не використовується, тоді виконавець повинен в обов'язковому порядку прикріпити файл з описом змін в макеті приблизно такого змісту:

- Додано нові картинки в папку img.
- Картинки btnHome.jpg і btnFeedback.jpg вже не потрібні, можна видаляти.
- Змінено html-код в секції файлу anketa.html.
- Додано в кінець файлу main.css нові стилі (починаючи з. Form_row і нижче).

9.2. Таблиці

<table> позначає таблицю.

<tr> розшифровується як «table row», позначає рядок таблиці.

<td> розшифровується як «table data», позначає комірку всередині рядка таблиці.

Теги <td> розташовуються усередині тегів <tr>, а ті, у свою чергу, всередині <table>. Дані комірок розміщуються всередині тегів <td>. У простій таблиці в кожному рядку має бути однакова кількість комірок, тобто всередині всіх <tr> має бути однакова кількість <td>.

Горизонтальні та вертикальні границі

Іноді потрібно, щоб границі клітинок у таблиці відображалися не повністю. Наприклад, щоб відображалася лише нижня границя клітинки, тоді таблиця виходить розкресленою по горизонталі (на стрічки). Аналогічно, якщо відображати тільки бокові границі клітинок, то таблиця виходить розбитою на стовпці. Ці ефекти легко робляться за допомогою CSS. Для цього необхідно використовувати не border, що задає рамки для всіх границь клітинок, а одну з властивостей:

- border-top,
- border-bottom,
- border-right,
- border-left.

Ці властивості відповідають за відображення кожної окремої границі клітини: верхньої, нижньої, правої або лівої відповідно.

Відступи всередині клітин таблиці

За замовчуванням текст в клітинках таблиці «прилипає» до границь. Якщо додати відступи всередині клітин, то інформація буде сприйматися набагато краще. Відступи всередині комірок можна додавати за допомогою CSS-властивості padding, яка задає «внутрішні відступи елементу» з усіх боків. Можна задавати відступи для кожної зі сторін окремо, використовуючи властивості:

- padding-top;
- padding-right;
- padding-bottom;
- padding-left.

Наприклад, щоб задати для клітинок таблиці всі відступи в 10 пікселів, а відступ зліва в 20 пікселів, потрібно написати такий CSS-код:

```
td {
  padding: 10px;
  padding-left: 20px;
}
```

Відступи між клітинами таблиці

Більшість завдань з оформлення таблиць вирішуються за допомогою роботи з границями, відступами всередині клітин, зміною кольору фону клітин. Окрім внутрішніх відступів можна задавати відступи між клітинами таблиці. Вони не працюють при border-collapse:collapse, що досить логічно, адже границі клітин у цьому режимі «склеєні» і їх не розірвати. В цьому випадку використовується border-collapse:separate – властивість, що «розклеює» клітини. Насправді це значення за умовчанням, а ми використовуємо його лише для наочності. Якщо видалити властивість border-collapse, то результат не зміниться, клітини будуть відображатися роздільно. Відступи між клітинами можна задати:

- за допомогою атрибуту cellspacing тега <table>;
- або с допомогою CSS-властивості border-spacing.

Відзначимо, що властивість border-spacing задається для таблиці, на відміну від padding, що задається для клітин.

Заголовки стовпців (стрічок)

Зазвичай у таблицях виділяють назви стовпців або рядків. У HTML для цього передбачений спеціальний тег <th>, що розшифровується як «table header» і позначає клітинку-заголовок. Тег <th> аналогічний <td>, він так само повинен розташовуватися всередині <tr>, і для нього стилями можна задавати такі ж самі властивості. За умовчанням текст всередині <th> виділяється жирним і вирівнюється по центру.

Заголовок таблиці

Для створення підпису до таблиці (або заголовка таблиці) використовується тег <caption>. Тег <caption> повинен розміщуватися всередині тега <table>, причому безпосередньо всередині нього і першим, відносно решти вкладених тегів. Ось так:

```
<table>
  <caption> Текст </ caption>
  ...
```

```
</ table>
```

Тег заголовка йде перший всередині таблиці, але за допомогою CSS можна перемістити заголовок таблиці в будь-яке місце: зверху чи знизу таблиці, по центру, праворуч або ліворуч. По вертикалі заголовок таблиці переміщається CSS-властивістю caption-side зі значеннями top, що розміщує заголовок перед таблицею, і bottom, що розміщує заголовок після таблиці відповідно.

По горизонталі заголовок таблиці вирівнюється CSS-властивістю text-align зі значеннями left, right і center.

Об’єднання клітинок у таблиці

Ми підійшли до одного з найскладніших питань у роботі з таблицями – це об'єднання клітинок. Коли ви об’єднуєте клітинки в текстовому редакторі, наприклад у Word, то програма багато чого робить за вас. У чистому HTML завдання об'єднання складніше, однак не варто лякатися, сам принцип об'єднання клітинок не такий вже складний, просто потрібно більше уважності.

Браузер	Посещения	
	Количество	В процентах
Mozilla Firefox	163	59%
Google Chrome	78	28%
Safari	35	13%

Рисунок 9.1. Об’єднання клітинок в таблиці

Вихідна таблиця в прикладі не дуже красива, але за допомогою об'єднання клітинок ми зробимо складну «шапку» в таблиці, що має ось такий вигляд:

Почнемо з об'єднання клітин по горизонталі. Щоб це зробити необхідно використовувати атрибут colspan у тегів <th> або <td>. Коли ви задаєте клітині атрибут colspan зі значенням 2, то вона ніби «розтягується» на клітинку праворуч, але та клітинка не зникає, а відсувається, і в таблиці з'являється

новий стовпець. Щоб видалити його, потрібно видалити клітину, що знаходиться праворуч від «розтягнутою».

Об'єднання клітин по вертикалі трохи складніше. Воно здійснюється за допомогою атрибуту rowspan тега <td> або <th>. Коли Ви задаєте клітині атрибут rowspan зі значенням 2, то вона ніби «розтягується» на наступний рядок. При цьому клітина, яка була під «розтягнутою», відсувається у своєму ж рядку вправо, що робить в таблиці зайвий стовпець. Видаливши клітинку, яка була під «розтягнутою» ми позбудемося цього стовпця. Фактично клітинки, місце яких займають об'єднанні клітинки, ніби видавлюються далі поздовж рядка.

<table>
<tr><td colspan="2">1.1</td><td>1.2 (з'їхала через 1.1)</td><td>1.3 (лишній стовпець)</td></tr>
<tr><td rowspan="2">2.1</td><td>2.2</td><td>2.3</td><td></td></tr>
<tr><td>3.1 (з'їхала через 2.1)</td><td>3.2</td><td>3.3 (лишній стовпець)</td></tr>
<tr><td>4.1</td><td>4.2</td><td>4.3</td><td></td></tr>
</table>

Рисунок 9.2. Приклад1 об'єднання стовпців

Чи можна об'єднувати більше двох клітинок по горизонталі? Можна! При цьому так само використовується атрибут colspan. Однак, оскільки клітинка «розтягується» вправо більш, ніж на одну сусідню клітинку, то і зайвих стовпців з'являється більше. Наприклад, якщо встановити colspan рівним 4, то клітинка розтягнеться на три сусідні клітини праворуч, а вони, в свою чергу, змістяться, додавши в таблицю три стовпці.

Об'єднання кількох комірок по вертикалі теж можливе. Як ви пам'ятаєте, при вертикальному об'єднанні витісняються клітинки, які знаходяться в рядках під «розтягнутою» клітинкою. І ці клітини «з'їжджають» праворуч у своїх рядках.

Можна розтягувати комірку одночасно по вертикалі та по горизонталі. Для цього потрібно задати комірці два атрибути: rowspan і colspan. Витіснення сусідніх комірок відбуватиметься так само, як і при звичайному об'єднанні комірок. Щоправда, витіснених комірок виявиться більше.

Наприклад, комірка з rowspan = "2" і colspan = "2" витіснить три сусідні комірки – одну в першому рядку і дві у другому.

<table>
<tr><td colspan="2" rowspan="2">1.1</td><td>1.2 (з`їхала через 1.1)</td><td>1.3(лишній стовпець)</td><td></td></tr>
<tr><td>2.1 (з`їхала через 1.1)</td><td>2.2(лишній стовпець)</td><td>2.3(лишній стовпець)</td></tr>
<tr><td>3.1</td><td>3.2</td><td>3.3</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>4.1</td><td>4.2</td><td>4.3</td></tr>
</table>

Рисунок 9.3. Приклад2 об’єднання стовпців

Вирівнювання тексту в комірках

Вміст комірок можна вирівнювати по горизонталі і по вертикалі за допомогою CSS. За вирівнювання по горизонталі відповідає CSS-властивість text-align. Найчастіше використовуються значення left, center і right. За вирівнювання по вертикалі відповідає CSS-властивість vertical-align. Найчастіше використовуються значення top, middle і bottom. Насправді, значень у обох властивостей більше, але у випадку з комірками нас цікавлять лише перераховані. Щоб задати вирівнювання контенту комірок, треба в стилях вказати:

```
td {
  vertical-align: значення;
  text-align: значення;
}
```

Ці стилі вплинуть на всі комірки таблиці. Щоб задати вирівнювання тільки в деяких, потрібно назначити їм класи і створити стилі для відповідних класів.

Зміна кольору фону в таблиці

Таблиці можна розфарбовувати, задаючи колір фону окремих комірок, колір тексту в комірках, а також колір рамок (границь). Кольором елементів можна керувати за допомогою властивостей:

- background-color – задає колір фону,
- color – колір тексту,

- border-color – колір рамки.

До цього ми використовували компактну форму для опису рамок: border: 1px solid lightgray. У цьому записі колір задає третій параметр – lightgray. Значення кольорів у CSS задаються різними способами, ми будемо використовувати ключові слова для опису кольору. Таким чином, щоб задати колір для комірки в CSS, потрібен наступний код:

```
td {
  color: колір;
  background-color: колір;
  border: 1px solid колір;
}
```

Звичайно, розфарбовувати можна і td, і th, і навіть table. Якщо задавати стилі для тега, наприклад, td, то вони застосуються до всіх тегів – td. Як бути, якщо стилі потрібно задати для якоїсь певної комірки, групи комірок, або рядка? Можна використовувати класи і застосовувати стилі для цих класів. Наприклад, ось так:

```
.my-class {
  стилі
}
```

Також для фону можна задавати картинки.

Задання розмірів таблиці

За замовчуванням ширина і висота таблиці залежить від вмісту і відступів всередині її комірок. Чим менший вміст, тим менший розмір таблиці. За допомогою CSS можна управляти розмірами таблиці, задавати бажану ширину і висоту. Також розмірами можна керувати за допомогою атрибутів таблиці, але ми розглянемо лише CSS. Варто відзначити, що у таблиці є мінімальні розміри, які залежать від змісту, менше яких вона не стиснеться, яке б значення ширини або висоти не задавалося.

Ширина таблиці задається за допомогою CSS-властивості width, а висота за допомогою властивості height, наприклад:

```
table {
  width: 100px;
  height: 100px;
```

```
}
```

Розміри таблиці можна задавати як в абсолютних одиницях, наприклад, в пікселях – 20 px, так і у відносних – відсотках – 20%.

При використанні відсотків розміри таблиці будуть обчислюватися з урахуванням розмірів батьківського елементу, наприклад, ширини вікна браузера. Особливе значення auto включає розрахунок розмірів за замовчуванням. Наприклад: width:auto; або height:auto;. Варто зазначити, що відсотки при заданні висоти зазвичай не працюють.

Розміри окремих комірок і стовпців теж можна задавати вручну, особливо якщо вам не подобається як браузер розподілив ширину колонок. Розміри комірок задаються точно так само, як і розміри таблиці: за допомогою CSS-властивостей width і height.

Є два варіанти використання стилів для окремих комірок:

- призначати коміркам унікальні імена класів, наприклад: class="cell-11", і застосовувати стилі для цих класів;
- використати атрибут style, всередині якого можна писати CSS-код.

Приклад другого варіанту:

```
<td style="width:100px;">   ... </ td>
```

На щастя, не часто є потреба задавати розміри кожної комірки. Зазвичай розміри комірок прописують, якщо треба вручну встановити ширину стовпців таблиці: для цього досить задати ширину для кожної комірки першої строки.

9.3. Верстка сайту

Блокова верстка сайту, що називається також версткою div'ами , останнім часом застосовується найбільш часто. Верстка div представляє собою різновид html-верстки сайтів і використовує таку синтаксичну конструкцію:

```
<div id="blok1"> вміст блоку 1</div>
<div id="blok2"> вміст блоку 2</div>
<div id="blok3"> вміст блоку 3</div>
...
```

Тег – це так званий контейнер (блок), який може містити форматований текст, зображення та ін. Важливою особливістю блоків є їх здатність накладатися один на одного при верстці. Ця особливість надає блокам набагато більше можливостей при верстці сайту, ніж, приміром, таблиці.

Верстка div активно використовує CSS для гнучкого й зручного форматування елементів вебсторінок. За допомогою CSS можна з точністю до пікселя задати розташування блоків на сторінці, необхідні відступи, колір, розмір шрифтів і багато іншого.

Блокова верстка сайту має безліч переваг. Ось деякі з них:

- блоки легко можна переміщати або «ховати» без перезавантаження всієї сторінки, що дозволяє отримувати динамічні ефекти (меню, що випадає, спливаючі підказки);
- зручно застосовувати для одержання «гумового» дизайну, оскільки блоки легко позиціонувати відносно меж вікна браузера, накладати один на одного і т. д.;
- верстка div дозволяє позбавитися від «розпорок» – прозорих зображень в 1 піксель, що використовуються для фіксації відстані між комірками таблиць при табличній верстці;
- програмний код при блоковій верстці сайту виходить більш компактним і зрозумілим; вважається, що завантаження сторінки з блоковою верстkою виконується швидше, ніж із табличною (проте не завжди це так).

Однак, поряд з перевагами, верстка div має також деякі недоліки . Наприклад, завдання кросбраузерності сайту вирішується складніше при використанні блочної верстки, оскільки різні браузери по-різному інтерпретують html-код сторінок із блоками.

Тому блокова верстка сайту нерідко застосовується в комплексі з табличною верстkою. У цьому випадку застосування кожного виду верстки сайтів для вирішення конкретних завдань виявляється більш привабливим. Наприклад, часто «каркас» сторінки задається за допомогою таблиці, а комірки цієї таблиці заповнюються необхідними блоками.

Таким чином, верстка div не є строго обов'язковою і повинна використовуватися в міру необхідності. Професійна верстка сайту, будь вона

блоковою або табличною, завжди правильно відображається в різних браузерах (кроссбраузерна), не містить зайвого коду, сприяє швидкому завантаженню сторінок.

Побудова моделі блоків

Властивість відображення display

Властивість display має багато різних значень. Зазвичай, використовуються тільки три з них: none, inline і block, тому що коли-то браузери інші не підтримували.

Значення властивості display:

- none (приховати);
- block (блоковий);
- inline (рядковий);
- inlineblock (строчноблочний);
- tablecell (осередок таблиці);
- flex (гнучкий).

Блоковий елемент (display : block;) створює розрив рядка перед тегом і після нього. Він утворює прямокутну область, по ширині займає всю ширину вебсторінки або батьківського блока, якщо для нього не задано значення width.

Блокові елементи можуть містити в собі елементи будь-якого типу. Не можна розміщувати блокові елементи всередині малих, за виключенням елемента < img >. Для блокових елементів можна задавати margin і padding .

Властивості width і height встановлюють ширину і висоту області вмісту елемента. Фактична ширина елемента складається з ширини полів (внутрішніх відступів), кордонів і зовнішніх відступів.

Рядкові елементи (display : inline;) не створюють блоки, вони відображаються на одному рядку з вмістом тегів поруч. Рядкові елементи є нащадками блокових елементів. Вони ігнорують верхні і нижні margin і padding , але якщо для елемента заданий фон, він буде поширюватися на верхній і нижній padding , заходячи на сусідні рядки тексту.

Ширина і висота статичного елемента залежить тільки від його змісту, задати розміри можна за допомогою CSS. Також можливе корегування відстані між сусіднімі елементами по горизонталі за допомогою горизонтальних полів і відступів.

Існує ще одна група елементів, які браузер обробляє як **строчноблочні** (display : inline block;). Такі елементи є вбудованим, але для них можна задавати поля, відступи, ширину і висоту.

Сучасні браузери (IE8+) дозволяють описувати таблицю будь-якими елементами, якщо поставити їм відповідні значення display. Це добре для семантичної верстки і дозволяє позбутися зайвих тегів.

flexbox – це цілий модуль, а не просто одинична властивість, він об'єднує в собі безліч властивостей. Деякі з них мають застосовуватися до контейнера (батьківського елемента, так званого, flex-контейнера), в той час як інші властивості застосовуються до дочірніх елементів або flex-елементів.

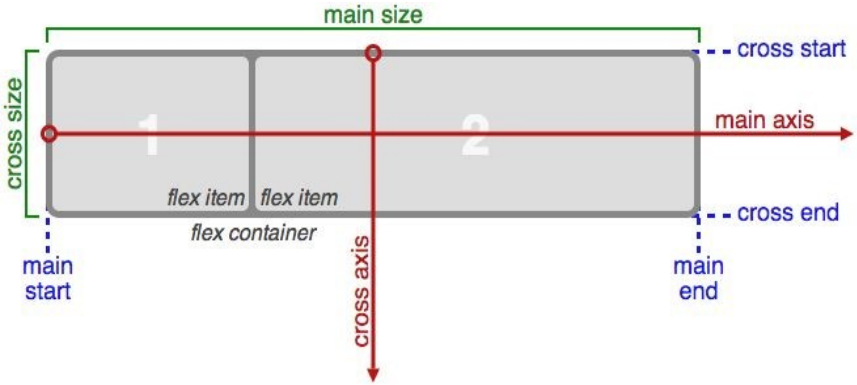


Рисунок 9.4. Значення елементів таблиці

Якщо звичайний layout ґрунтується на напрямках потоків блокових і inlin-елементів, то flex-layout ґрунтується на напрямках flex-потоку. Ознайомтеся зі схемою в специфікації, що роз'яснює основну ідею flex-layout.

В основному елементи будуть розподілятися або уздовж головної осі (від main start до main end), або вздовж поперечної осі (від cross start до cross end).

- main axis – головна вісь, уздовж якої розташовуються flex-елементи. Зверніть увагу, вона необов'язково повинна бути горизонтальною, все залежить від якості justify-content (див. нижче).
- main start / main end – flex елементи, що розміщуються в контейнері від позиції main start до позиції main end.

main size – ширина або висота flex-елемента в залежності від вибраної базової величини. Основна величина може бути або шириною , або висотою елемента.

- cross axis – поперечна вісь, перпендикулярна до головної. Її напрямок залежить від напрямку головної осі.
- cross start / cross end – flex рядки, що заповнюються елементами і розміщуються в контейнері від позиції cross start і до позиції cross end.
- cross size – ширина або висота flex-елемента, в залежності від обраної розмірності, дорівнює цій величині. Це властивість збігається з width або height елемента в залежності від обраної розмірності.

Елементи в контейнері піддаються вирівнюванню за допомогою властивості justify-content уздовж головної осі. Ця властивість приймає цілих п'ять різних варіантів значень:

- flex start (default): гнучкі елементи вирівнюються по початку головної осі;
- flex end: елементи вирівнюються по кінцю головної осі;
- center: елементи вирівнюються по центру головної осі;
- space betwee : елементи займають всю доступну ширину в контейнері, крайні елементи впритул притискаються до країв контейнера, а вільний простір рівномірно розподіляється між елементами;
- space around: гнучкі елементи вирівнюються таким чином, що вільний простір рівномірно розподіляється між елементами. Але варто відзначити, що простір між краєм контейнера і крайніми елементами буде в два рази менше, ніж простір між елементами всередині ряду.

Ми також маємо можливість вирівнювання елементів по cross-осі. Застосувавши властивість align items, яка приймає також п'ять різних значень, можна домогтися цікавого результату. Ця властивість дозволяє вирівнювати елементи в рядку відносно один одного.

- flex start: всі елементи притискаються до початку рядка;
- flex end: елементи притискаються до кінця рядка;
- center: елементи вирівнюються по центру рядка;
- baseline: елементи вирівнюються по базовій лінії тексту;
- stretch (default): елементи розтягуються, заповнюючи повністю рядок.

Ще одну або кілька схожих властивостей можемо спостерігати на елементі align-content. Тільки він відповідає за вирівнювання цілих рядків щодо гнучкого контейнера. Але ми не побачимо ефекту, якщо гнучкі елементи займають всього один рядок.

Властивість приймає шість різних значень.

- flex start: всі лінії притискаються до початку cross-осі;
- flex end: всі лінії притискаються до кінця cross-осі;
- center: flex елементи вирівнюються по центру flex-контейнера;
- space between: лінії розподіляються від верхнього краю до нижнього, залишаючи вільний простір між рядками, крайні ж рядки притискаються до країв контейнера;
- space around: лінії рівномірно розподіляються по контейнеру;
- stretch (default): лінії розтягуються, займаючи весь доступний простір.

Однією з основних властивостей є flex basis . За допомогою цієї властивості ми можемо вказувати базову ширину гнучкого елемента. За замовчуванням вонамає значення auto. Ця властивість тісно пов'язана з flex grow і flex shrink , про які буде розказано трохи пізніше. Приймає значення ширини в px ,%, em і інших одиницях. По суті, це не строго ширина гнучкого елемента, це свого роду відправна точка, відносно якої відбувається розтягування або усадка елемента. У режимі auto елемент отримує базову ширину щодо контенту всередині нього.

Схлопнути

Коли два або більше вертикальних margin стикаються, вони зливаються, при цьому ширина загального відступу дорівнює ширині більшого з вихідних відступів.

Злиття виконується тільки для блокових елементів у нормальному потоці документа. Зовнішні вертикальні відступи малих, плаваючих і елементів , що позиціонуються абсолютно, не зливаються.

Щоб отримати бажаний проміжок, можна задати, наприклад, для верхнього елементу padding bottom , а для нижнього елемента – margin top.

Існують винятки для схлопування:

- з блоками, яким присвоєно float;
- з основними елементами (html , body);
- для блоків, яким присвоєно властивість і значення position absolute;
- для малих елементів.

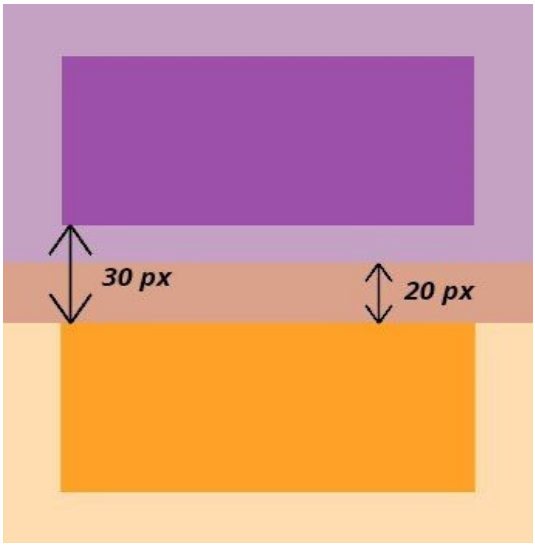


Рисунок 9.5. Злиття

9.4. Формування блокової моделі

Формування блоків

На перший погляд може здатися, що width – це остаточна ширина елемента, а height це остаточна висота елемента. Насправді це не так, width і height це не

остаточні розміри елемента. Для того, щоб обчислити розміри , необхідно враховувати наступні моменти .

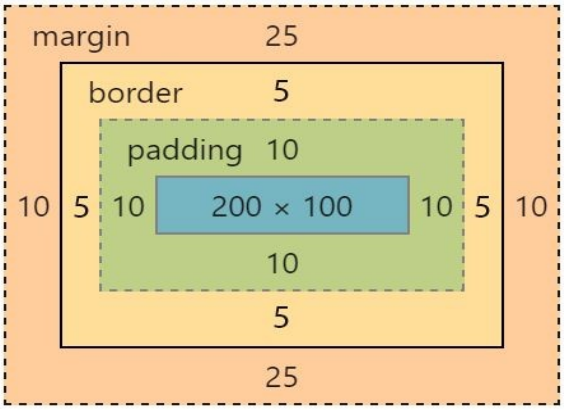


Рисунок 9.6. Відступи блоків

Якщо уважно придивитися до цієї схеми, то можна зробити висновок, що ширина блоку складається з наступних властивостей:

margin-left + border-left + padding-left + width + padding-right + border-right + margin-right

Відповідно, висота наступна:

margin-top + border-top + padding-top + height + padding-bottom + border-bottom + margin-bottom

Внутрішній відступ або поле елемента (padding) додає відступи всередині елемента, між його основним вмістом і його кордоном. Якщо для елемента задати фон, то він поширюється також і на поля елемента. Внутрішній відступ не може приймати негативних значень, на відміну від зовнішнього відступу.

Зовнішній відступ (margin) додає відступи за межами елемента, створюючи тим самим проміжки між елементами. Вони завжди залишаються прозорими, і через них видно фон батьківського елемента. Значення padding і margin задаються в наступному порядку: верхнє, праве, нижнє і ліве.

Кордон або рамка елемента задається за допомогою властивості border . Якщо колір рамки не заданий, вона приймає колір основного вмісту елемента, наприклад, тексту. Якщо рамка має розриви, то крізь них буде проступати фон елемента.

Зовнішні, внутрішні відступи і рамка елемента не є обов'язковими, за замовчуванням їх значення дорівнює нулю. Тим не менш, деякі браузери додають до цих властивостей позитивні значення за замовчуванням на основі своїх таблиць стилів.

Обтічні елементи

Обтічні елементи, або як їх ще називають «плаваючі», використовуються для реалізації обтікання текстом зображень, створення врізок, і навіть створення багатостовпчикових компоновок.

Перший прийом заснований на використанні обтічних блоків, так як із їх допомогою можна довільно розташовувати елементи по горизонталі. По суті – це використання властивості float не за призначенням, яке зрушує блок вліво або вправо і включає обтікання тексту.

Обтічні елементи мають ширину, залежно від змісту, і займають все вільне для цієї ширини місце. Якщо його не вистачає, вони зміщуються вниз до тих пір, поки є місце або не залишиться інших обтічних блоків.

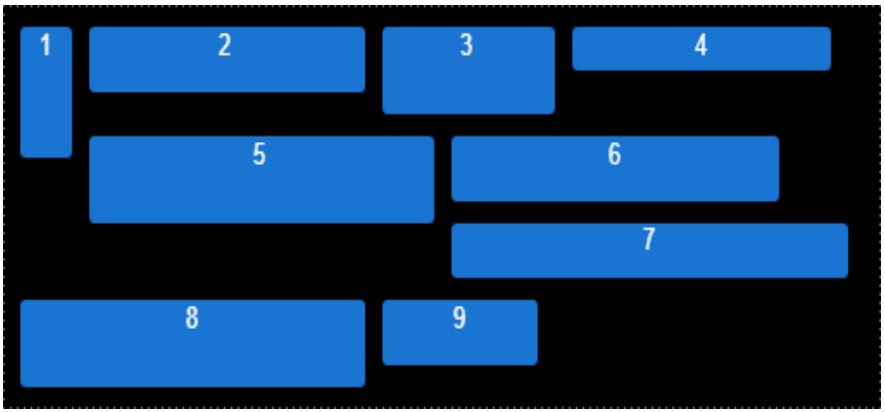


Рисунок 9.7. Обтічні блоки

Приклад можливого розташування довільних обтічних блоків.

Обтічні елементи частково вилучаються з потоку і звичайні блоки проходять крізь них. Змінюється лише розташування тексту через звуження рядків. Рядкові елементи розташовуються там, де їм вистачає місця в рядках.

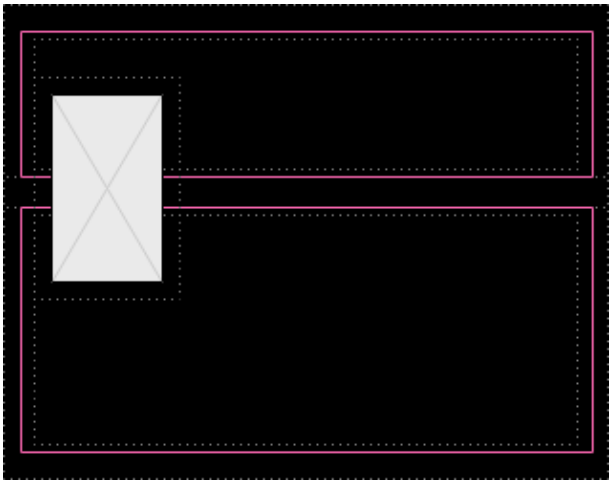


Рисунок 9.8. Плаваючі елементи

Також обтічні елементи активно використовуються при верстці вебсторінок, і за допомогою їх можна замінити табличну верстку на верстку шарами. Для того, щоб задати обтікання, у CSS існує тільки одна властивість float, яка може приймати всього два значення – це left і right.

В цьому випадку картинка займає положення зліва і дозволить будь-яким елементам, будь вони малі або блокові, обтікати себе справа.

Далі розглянемо приклад. Створимо два елементи < div > і один заголовок першого

У обох елементів < div > задано властивість float : left; , тобто вони повинні займати ліве положення і дозволяти обтікати себе справа.

```
<!DOCTYPE html>
<html>
<head>
```

```
<style>
div {width: 150px ;
height:150px ;
marginright : 15px;
border: 1px solid blue;
float: left}
h1 {
border: 1px solid red}

</style>
</head>
<body>

<div> Це перший блок </div>
<div> Це другий блок </div>
<H1> Це заголовок першого рівня </H1>

</body>
</html>
```

Подивимося на роботу цього прикладу в браузері.

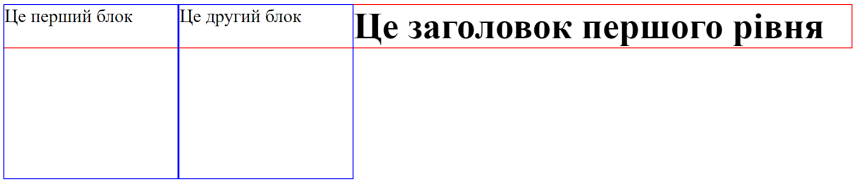


Рисунок 9.9. Приклад 1

Розберемося, що ж сталося. Елементи < div > знаходяться на одній лінії по горизонталі, що і очікувано, тому що у них задано властивість float : left; . Перший < div > посів становище зліва, дозволив обтікати себе справа. Другий < div >, відповідно, в свою чергу також дозволив обтікати себе справа. Тема першого рівня знаходиться праворуч другого елемента < div >, але його рамка обрамляє також обидва елементи < div >. Це відбувається тому, що у властивості float є особливість: елементи, яким задано цю властивість, починають притягувати до себе всі сусідні елементи і змушують їх теж брати участь в обтіканні. Але з цим можна боротися.

Для цього:

1. Заголовок не повинен брати участь в обтіканні і повинен перебувати під елементами < div >.

В цьому випадку необхідно застосувати заборону на обтікання. Для цього в css існує властивість clear. Вона може приймати три значення: це left , що скасовує обтікання з лівого краю, right – з правого краю, і значення both, яке скасовує обтікання з обох сторін. Додамо властивість clear зі значенням both для заголовка першого рівня.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
width: 150px ;
height: 150px ;
marginright : 15px;
border: 1px solid blue;
float: left ;}
h1 {   border: 1px solid red;
clear: both ; }

</style>
</head>
<body>

<div> Це перший блок </div>
<div> Це другий блок </div>
<H1> Це заголовок першого рівня </H1>
</body>
</html>
```

Якщо запустити даний код у браузері, то заголовок вже не братиме участі в обтіканні, а буде знаходитися під елементами < div >.

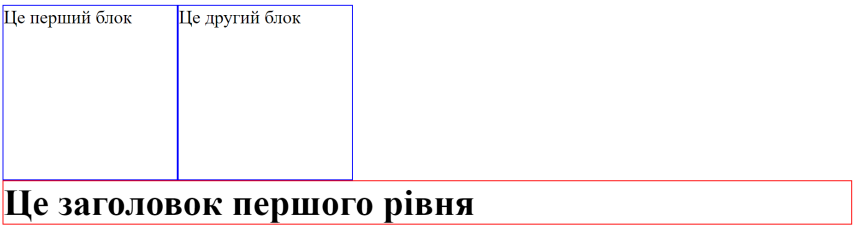


Рисунок 9.10. Приклад 2

Тема залишається на тому ж місці, де він зараз перебуває, але рамка повинна оточувати тільки сам заголовок.

Для вирішення цього завдання, допоможе css-властивість overflow . Вона визначає як буде вести себе блоковий елемент в разі його переповнення, і при значенні hidden відображає тільки вміст цього елементу.

```
<!DOCTYPE html>
  <html>
  <head>
  <style>
  div {width: 150px ;
  height: 150px ;
  marginright : 15px;
  border: 1px solid blue ;
  float: left}
  h1 { border: 1px solid red;
  overflow: hidden}
  </style>
  </head>
  <body>
  <div> Це перший блок </div>
  <div> Це другий блок </div>
  <H1> Це заголовок першого рівня </H1>
  </body>
  </html>
```

Якщо запустити цей приклад у браузері, то заголовок залишається на тому ж місці, і рамка тепер обрамляє тільки елемент < h1 >.

Ідея, що лежить в основі позиціонування, досить проста. Воно дозволяє точно визначити де з'являться блоки щодо іншого елемента, або щодо вікна

браузера. За замовчуванням всі елементи розташовуються послідовно один за іншим у тому порядку, в якому вони визначені в html-документі ({ position : static }). Властивість не успадковується.

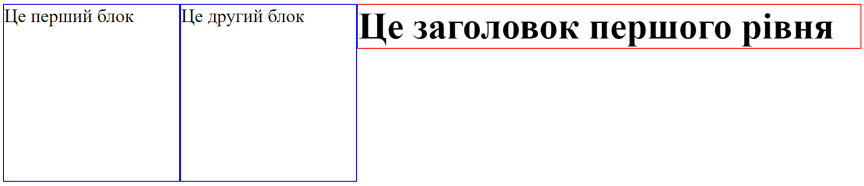


Рисунок 9.11. Приклад 3

Блоковий елемент (p , div , h 1 і ін.). Займає 100% ширини батьківського елементу (за замовчуванням – body). Тому блокові елементи відображаються один під другим відповідно до запрограмованих шаблонів.

Рядковий елемент (em , strong , span і ін.). Займає ширину, яка відповідає ширині вмісту всередині нього. Тому малі елементи відображаються поруч один з одним.

Властивість position разом зі значеннями top , right , bottom і left відображає елемент з порушенням звичайного порядку, зміщуючи його на задану відстань. При позиціонуванні елементів можна використовувати як позитивні, так і негативні значення. Таким чином, існують 4 види позиціонування.

Абсолютне позиціонування

При абсолютному позиціонуванні елемент не існує в потоці документа, і його положення задається щодо країв браузера. Задати цей шаблон можна через значення absolute властивості position. Координати вказуються відносно країв вікна браузера, так званої «видимої області».

Для режиму характерні наступні особливості:

Ширина шару, коли її не вказано явно, дорівнює ширині контенту плюс значення полів, меж і відступів.

- Шар не змінює своє початкове положення, якщо у нього немає властивостей right , left, top і bottom.

- Властивості left і top мають більш високий пріоритет у порівнянні з right і bottom. Якщо left і right суперечать один одному, то значення right ігнорується. Те ж саме стосується і bottom.
- Якщо left задати негативне значення, то шар піде за лівий край браузера, смуги прокрутки при цьому не виникне. Це один із способів заховати елемент від перегляду. Те саме можна сказати і до властивості top , тільки шар піде за верхній край.
- Якщо left задати значення більше ширини видимої області або вказати right із від'ємним значенням, з'явиться горизонтальна смуга прокрутки. Подібне правило працює і з top , тільки мова піде про вертикальну смугу прокрутки.
- Одночасно зазначені властивості left і right формують ширину шару, але тільки якщо width не вказано. Варто додати властивість width , і значення right буде проігноровано. Аналогічно відбудеться і з висотою шару, тільки вже беруть участь властивості top , bottom і height .
- Елемент з абсолютним позиціонуванням переміщується разом із документом при його прокручуванні.

Відносне позиціонування

Relative (відносне позиціонування) елемент буде зміщуватися щодо його певного в даний час положення, і при цьому його місце буде залишатися незаповненим. Додавання властивостей left , top , right і bottom змінює позицію елемента і зрушує його в ту чи іншу сторону від первісного розташування. Позитивне значення left визначає зрушення вправо від лівої межі елемента, негативне – зрушення вліво. Полож даткови значення top задає зсув елемента вниз, негативне – зрушення вгору.

Властивості bottom і right виробляють зворотний ефект. При позитивному значенні right зрушує елемент вліво від його правого краю, при негативному – зрушує вправо. При позитивному значенні bottom елемент піднімається вгору, при негативному – опускається вниз.

Для відносного позиціонування характерні наступні особливості:

- Цей тип позиціонування не застосовний до елементів таблиці на зразок осередків, рядків, колонок та ін.

- При зміщенні елемента щодо вихідного положення, місце, яке займав елемент, залишається порожнім і не заповнюється нижче або вище встановленими елементами.

Фіксоване позиціонування

Фіксоване позиціонування шару задається значенням fixed властивості position і за своєю дією схоже на абсолютне позиціонування. Але, на відміну від нього, прив'язується до зазначеної властивостями left , top , right і bottom точці на екрані і не змінює свого положення при прокручуванні вебсторінки. Ще одна різниця від absolute полягає в тому, що при виході фіксованого шару за межі видимої області праворуч або знизу від неї, не виникає смугу прокрутки.

Застосовується такий тип позіціонування для створення меню, заголовків, вкладок, – загалом, для будь-яких елементів, які повинні бути закріплені на сторінці і завжди видні відвідувачу.

Значення за замовчуванням

Якщо для елемента властивість position не задано або його значення static , елемент виводиться в поток документа як зазвичай. Іншими словами, елементи відображаються на сторінці в тому порядку, як вони йдуть у вихідному коді HTML.

Властивості left , top , right , bottom, якщо визначені, ігноруються.

Поєднане значення

Призначивши батьківському блоку відносне позиціонування (position : relative), ми зможемо позиціонувати будь-які дочірні елементи щодо його меж. Якщо в елемента є позиційований предок, то position : absolute працює щодо нього, а не відносно документа. Потрібно користуватися таким позиціонуванням із обережністю, тому що воно може перекрити текст. Цим воно відрізняється від float.

```
position: relative + position: absolute
```

Z-index

Будь-які позиційовані елементи на вебсторінці можуть накладатися один на одного в певному порядку, імітуючи тим самим третій вимір,

перпендикулярний екрану. Кожен елемент може перебувати як нижче, так і вище інших об'єктів вебсторінки, їх розміщенням по z-осі і управляє z-index . Ця властивість працює тільки для елементів, у яких значення position задано як absolute , fixed або relative.

Як значення використовуються цілі числа (позитивні, негативні і нуль). Чим більше значення, тим вище знаходиться елемент у порівнянні з тими елементами, у яких воно менше. При рівному значенні z-index , на передньому плані знаходиться той елемент, який у коді HTML описаний нижче. Хоча специфікація і дозволяє використовувати негативні значення z-index , але такі елементи не відображаються в браузері Firefox до версії 2.0 включно.

Крім числових значень застосовується auto – порядок елементів у цьому випадку будується автоматично, виходячи з їхнього економічного становища в коді HTML і приналежності до батька чи матері, оскільки дочірні елементи мають той же номер, що їх батьківський елемент. Значення inherit указує, що воно успадковується у батька.

10. ПІДГОТОВКА ЗАВАНТАЖЕННЯ ТА ОПУБЛІКУВАННЯ

10.1. Толерантний код

Спосіб, яким браузери читають HTML більш толерантний, ніж у мов програмування, що інтерпретують свій код суворіше. Це одночасно і погано, і добре.

Так що ж означає толерантний? У загальних рисах, коли ви напартачили в коді, є два типи помилок, з якими ви зіткнетеся:

- синтаксичні помилки (Syntax errors): це помилки в правильності написання, як це було вище, в прикладі з Rust. Такі зазвичай легко виправляти, в тій мірі, в якій ви знайомі з синтаксисом мови і знаєте, що означають повідомлення про помилки.
- логічні помилки (Logic errors): це помилки, що з'являються в тому випадку, якщо синтаксис коректний, але код не виконує свого призначення, тобто програма виконується невірно. Такі виправляти складніше, ніж синтаксичні, оскільки не виводиться повідомлень, що вказують місце, де ви помилилися.

HTML не страждає від синтаксичних помилок, тому що браузер читає код толерантно, в тому сенсі, що сторінки можуть відображатися, навіть якщо синтаксичні помилки присутні. Браузери мають вбудовані правила по інтерпретації невірно написаної розмітки, і ви можете запустити що-небудь, навіть якщо ви мали на увазі інше. Це може стати справжньою проблемою!

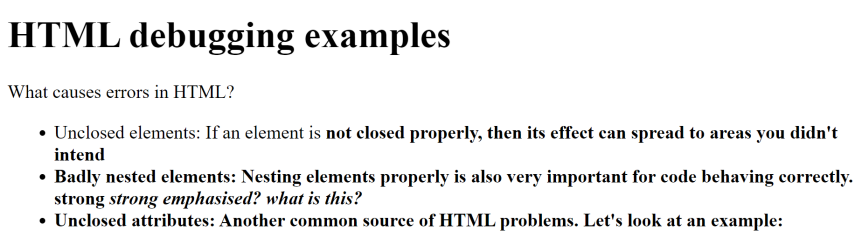
На замітку: HTML читається толерантно, тому що коли веб тільки з'явився, було вирішено дозволити людям публікувати контент навіть за умови неточностей у коді, так як це куди важливіше, ніж впевненість в абсолютно правильному синтаксисі. Веб не був би зараз такий популярний, якби ставився до новачків суворо.

```
Час вивчити природу толерантного коду в HTML.
  <!DOCT
  YPE
  html>
```

```
<html>
  <head>
    <meta charset="utf-8">

    <title>HTML debugging examples</title>

    <!--[if lt IE 9]>
      <script
src="https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3
.7.3/html5shiv.js"></script>
    <![endif]-->
  </head>

  <body>
    <h1>HTML debugging examples</h1>

    <p>What causes errors in HTML?
    <ul>
      <li>Unclosed elements: If an element is
<strong>not closed properly, then its effect can spread
to areas you didn't intend

      <li>Badly nested elements: Nesting elements
properly is also very important for code behaving
correctly. <strong>strong <em>strong
emphasised?</strong> what is this?</em>

      <li>Unclosed attributes: Another common source of
HTML problems. Let's look at an example: <a
href="https://www.mozilla.org/>link to Mozilla
homepage</a>
    </ul>
  </body>
</html>
```

Для початку, приклад налагодження збережіть локально. Ця демонстрація навмисно написана з помилками, які нам належить виявити.

Далі, відкрийте її в браузері. Ви побачите щось на зразок цього:



HTML debugging examples

What causes errors in HTML?

- Unclosed elements: If an element is **not closed properly, then its effect can spread to areas you didn't intend**
- **Badly nested elements: Nesting elements properly is also very important for code behaving correctly. strong *strong emphasised? what is this?***
- **Unclosed attributes: Another common source of HTML problems. Let's look at an example:**

Рисунок 10.1. Приклад 1

Зараз документ виглядає не особливо добре; давайте подивимося в код і з'ясуємо чому (показано лише тіло документа):

```
<h1>HTML debugging examples</h1>

<p>What causes errors in HTML?

<ul>
  <li>Unclosed elements: If an element is <strong>not closed properly,
      then its effect can spread to areas you didn't intend

  <li>Badly nested elements: Nesting elements properly is also very
important
      for code behaving correctly. <strong>strong <em>strong
emphasised?</strong>
      what is this?</em>

  <li>Unclosed attributes: Another common source of HTML problems.
Let's
      look at an example: <a href="https://www.mozilla.org/>link to
Mozilla
      homepage</a>
</ul>
```

Розглянемо проблеми:

У параграфа і елемента списку не закриті теги. На зображенні вище видно, що розмітка не постраждала, так як браузеру легко зробити висновок про те, де закінчується один елемент і починається інший.

Перший елемент також не має закритого тега. Це вже більш проблематично, так як складно сказати, де елемент повинен закінчуватися. На ділі, весь текст був виділений жирним.

Наступна частина порушує правила вкладеності: strong strong emphasised? </ Strong> what is this? </ Em>. В цьому випадку код теж складно проінтерпретувати через описане вище.

В атрибуті href відсутні закриваючі лапки. Це послужило причиною великої проблеми – посилання не відтворилося зовсім.

Зараз подивимося, як браузер згенерував власну розмітку, на противагу вихідній розмітці документа. Щоб зробити це, скористаємося інструментами розробника. Якщо ви не знайомі з інструментами розробника, витратьте декілька хвилин на огляд інструментів розробки в браузерах.

В DOM дереві ви можете побачити як згенерувати нову розмітку (рис. 10.2):

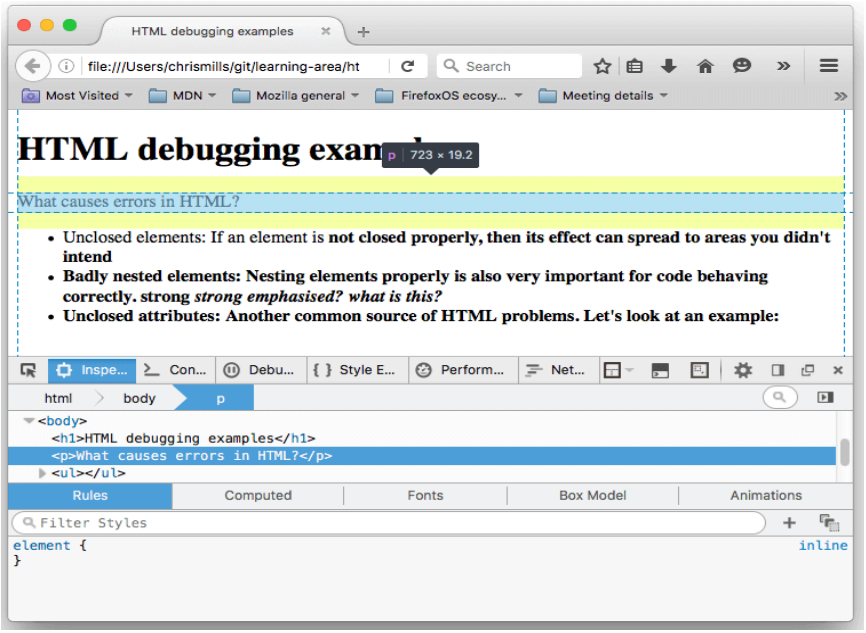


Рисунок 10.2. Приклад 2

Використовуючи DOM інспектор, давайте розглянемо деталі нашого коду, щоб побачити, як браузер намагається виправити наші помилки в HTML (ми оглядаємо в Firefox, другий сучасний браузер повинен видати ті ж результати):

Параграфи і елементи списку отримані з закриваючими тегами.

Було неочевидно, де елемент повинен був закритися, так що браузер обернув кожен окремий блок тексту своїми власними тегами strong , причому до самого низу документа!

Некоректна вкладеність була виправлена браузером наступним чином:

```
<Strong> strong
  <Em> strong emphasised? </ Em>
</ Strong>
<Em> what is this? </ Em>
```

Посилання з відсутніми подвійними лапками було видалено назовсім. Останній елемент списку буде виглядати так:

```
<Li>
   <Strong> Unclosed attributes: Another common source of HTML
problems.
   Let's look at an example: </ strong>
</ Li>
```

Валідація HTML

З прикладу вище ясно, що варто перевіряти валідність HTML. У простому прикладі зверху можна просто переглянути весь код і знайти помилки, але як бути з величезними, складними сторінками?

Краще за все перевірити сторінку в сервісі валідації розмітки. Його створив і підтримує W3C – організація, яка займається специфікаціями HTML, CSS та інших вебтехнологій. Сервіс перевірить ваш HTML і складе звіт по помилках у ньому.

10.2. Посилання на сайті

Як відомо, посилання на сайті мають дуже важливе значення: вони грають і навігаційну роль, допомагаючи відвідувачам переміщатися по сторінках сайту, і SEO-роль, впливаючи на індексацію сайту пошуковими системами. Щоб

оцінити, чи достатньо ефективно виконують свою роботу посилання на сайті, необхідно регулярно проводити їх аналіз.

Всі посилання сайту діляться на внутрішні, що ведуть на сторінки всередині сайту, і зовнішні (вихідні) – провідні на сторінки інших сайтів. Щоб провести комплексний аналіз посилань на сайті, необхідно приділити увагу як внутрішнім, так і вихідним гіперпосиланнями. Спочатку зазвичай перевіряються на відповідність рекомендаціям пошукових систем внутрішні посилання.

Аналіз внутрішніх посилань сайту зазвичай включає кілька етапів. Насамперед, наприклад, слід упевнитися, що всі наявні на сайті посилання є робочими. Якщо це не так – необхідно знайти причину помилок і усунути її. Прямі посилання на більш неіснуючі сторінки слід видалити.

Далі варто переконатися, що на кожну сторінку сайту є хоча б 1 текстове посилання. Якщо таке правило не виконується, то необхідно найближчим часом цю ситуацію виправити: сторінку сайту, на яку немає посилань, не побачать ні відвідувачі, ні пошукові роботи.

На наступному етапі необхідно підрахувати кількість внутрішніх посилань, наявних на кожній сторінці сайту – воно не повинно перевищувати 100 штук. У іншому випадку пошукові роботи можуть вважати сторінки «посилальним смітником» і тоді високих позицій у пошуковій видачі ці сторінки отримати не зможуть.

Далі слід переконатися, що у внутрішніх посиланнях на сайті використовуються в якості анкорів тематичні ключові слова. Використання в анкорах внутрішніх посилань ключових слів важливо як з точки зору просування сайту, так і з точки зору юзабіліті.

На завершення аналізу внутрішніх посилань сайту потрібно перевірити, чи доступні навігаційні посилання пошуковим роботам, що не використовують Javascript. Для цієї мети можна використовувати, наприклад, інструмент «Подивитися як Googlebot» в «Інструментах для вебмайстрів» від пошукової системи Google.

Наступна частина робіт по виконанню аналізу посилань на сайті включає аналіз зовнішніх посилань. Такий аналіз також ділиться на кілька основних етапів. На

першому етапі слід переконатися, що кількість зовнішніх вихідних посилань на кожній сторінці мінімальна і наявність кожного такого посилання обумовлено об'єктивною необхідністю.

На останньому етапі перевірки вихідних посилань кожної сторінки сайту варто переконатися, що наявні на сторінці зовнішні посилання ведуть на якісні сайти, причому бажано – щоб вони відповідали тематиці поточного сайту. В іншому випадку наявність зовнішніх посилань може негативно позначитися на позиціях сайту в пошуковій видачі.

Ефективний аналіз посилань сайту можна виконувати вручну, по черзі перевіряючи кожну сторінку. Однак для великих сайтів, що містять тисячі сторінок, подібна операція може бути дуже стомлюючою і до того ж займати дуже багато часу. Тому часто для аналізу посилань застосовуються різні програми і онлайн-сервіси, що дозволяють автоматизувати дану процедуру.

Перевірка

Текст

Інструмент, який допомагає нам щось підказати користувачу про його подальші дії. Величезна частина контенту в інтернеті – текст.

- не використовуйте занадто багато слів;
- використовуйте особисті займенники;
- пишіть короткі речення;
- задавайте питання і відповідайте на них;
- ретельно підбирайте заголовки.

Сучасні тенденції оформлення тексту:

1. Використовуйте піктограми, значки, емодзі.
2. Здійснюйте форматування (оптимальна довжина абзацу становить 3-5 рядків.)
3. Забудьте про КАПСЛОК! (великі літери сприймаються як крик і викликають у читача лише негатив).
4. Обмежте довжину рядка (не більше 60% ПК, 30-40% телефон)
5. Слідкуйте за міжрядковим інтервалом (розмір інтервалу повинен бути на 30% більше висоти символів).

Відображення

мати близько 10 слів в рядку

80 символов ~ 10 слов в строке

```
.text-wrapper {
    max-width: 80ch;
}
```

міжстрочна відстань

```
.text-wrapper {
    line-height: 1.3;
}
```

- ✓ розмір текста та переноси

```
.text-wrapper {
     max-width: 80ch;
     line-height: 1.3; I
     font-size: 16px;
     hyphens: auto;
}
```

Шрифти

Вебшрифти – це функція CSS дозволяє вам вказувати файли шрифтів, що завантажуються разом із вашим вебсайтом у міру доступу до нього, це означає, що будь-який браузер, що підтримує вебшрифти, може мати в своєму розпорядженні саме ті шрифти, які ви вкажете (рис. 10.3) .

CSS-властивості

- font-family
- font-size
- letter-spacing
- word-spacing
- line-height
- font-kerning
- font-stretch
- font-feature-settings
- font-variant
- font-synthesis
- font-style
- font-weight
- text-decoration
- word-wrap

Рисунок 10.3. Приклад шрифта

OpenType

OpenType – мабуть один із найпопулярніших зараз шрифтових форматів. OpenType використовує один файл, у якому зберігається вся інформація по контурах, метриках шрифту, цифровому підпису та службових даних.

Variable fonts

Варіативні шрифти – це розширення формату OpenType, яке дозволяє зберігати всі варіанти накреслень в одному файлі, а для перемикання між ними використовувати не тільки фіксовані кроки, але і проміжні значення.

Завантаження шрифтів

- Попереднє завантаження;
- CSS;
- JS (FontFace API);
- заміна шрифтів.

Фон

Призначення фону:

- легкість в сприйнятті;
- універсальність;
- простота;
- вся увага на елементи.

Типи фонів:

- білий;
- градієнт;
- патерни і текстури;
- велике зображення;
- відеобекграунд.

Підбір кольорів для дизайну сайту:

- базовий колір для виділення основного контенту;
- додатковий відтінок для другорядної інформації, котрий гармоніюватиме з базовим;
- фоновий колір, що має бути спокійним і не здатним затьмарити базовий та додатковий відтінки;
- акцентуючий, яким можна було б виділити шрифти, аби привернути увагу відвідувачів.

Коефіцієнт контрасності:

- маленький текст повинен мати коефіцієнт контрастності не менше 4,5 : 1 по відношенню до фону.
- великий текст (при 14 pt жирний – 18 pt звичайний) повинен мати коефіцієнт контрастності принаймні 3 : 1 по відношенню до фону.

Коефіцієнт контрастності (L1 + 0,05) / (L2 + 0,05), де: L1 – відносна яскравість найбільш яскравого з кольорів; L2 – відносна яскравість найчастіше темного з кольорів.

Схеми поєднання кольорів:

Окремі кольори самі по собі іноді бувають дуже кричущими або занадто блідими – однак вони можуть бути добре обіграними у поєднанні з іншими кольорами. Психологи припускають, що враження від кольору може збільшити чи зменшити вплив змісту сайту на користувача. Кольорове коло – це простий інструмент, який використовується для пошуку гармонійного поєднання кольорів і запропонований він ще Ньютоном (рис. 10.4) .

Кольорові схеми, які рекомендується використовувати:

- Монохроматична.
- Комплементарна.
- Спліт.
- Аналогова.
- Тріада.
- Прямокутник.
- Квадрат.

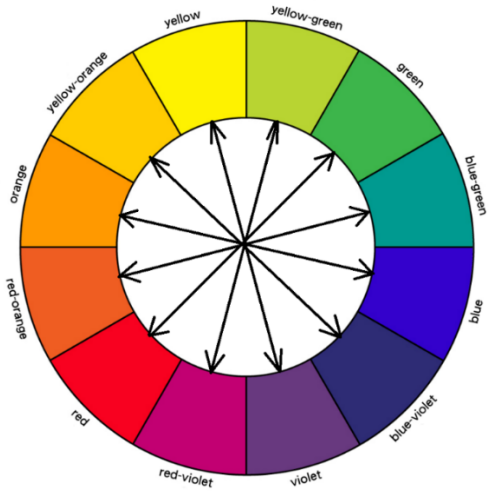


Рисунок 10.4. Кольоровий круг

Принципи при виборі фону сайту:

- специфіка ресурсу (легкість в сприйнятті);
- універсальність;
- відповідність фірмовому стилю;
- баланс між оригінальністю і юзабіліті (простота);
- важливість контентної частини (вся увага на елементи).

10.3. Зображення

Красиві вебсторінки обов'язково містять графічні зображення, часто приваблюють погляд веселими анімаційними картинками, а інколи пропонують прослухати музику чи переглянути кліп або фільм.

Ви ознайомитеся з форматами зображень, аудіо-та відеофайлів, які використовують в Інтернеті, дізнаєтеся про те, як вставляти на вебсторінки малюнки і працювати з картою гіперпосилань. На конкретних прикладах навчитеся створювати анімовані зображення та імпортувати їх у вебдокументи, а також розміщувати на своїх сайтах мультимедійну інформацію.

У практичній роботі, яку необхідно виконати, ви спробуєте оформити тло вебсторінки у вигляді малюнка, розмістити на ній графічні об'єкти і зробити їх посиланнями. У більшості професійно створених сайтів використовують графічне оформлення, що дає змогу яскраво та наочно подати інформацію. Для цього на вебсторінках розміщують відповідні графічні файли, які можуть мати різні формати.

Зображення мають бути розроблені у такий спосіб, щоб допомогти користувачу сприймати текстову інформацію та доповнювати її. Розглянемо особливості популярних графічних форматів, а також засоби розміщення та вирівнювання зображень на вебсторінках (рис. 10.5).

Рисунок 10.5. Формати зображень для WEB

Розподілений Git

Що знадобиться перед початком роботи?

акаунт в *Github*;

акаунт в *Cloudflare*;

доступ до користувацького домена

Якщо все готово, то давайте приступимо!

Покроковий план

Крок 1. Створити репозиторій на Github

Перейдіть на *головну сторінку GitHub Pages* і виберіть опцію “Project site”, щоб знайти інструкції зі створення базової сторінки з нуля (рис. 10.6):

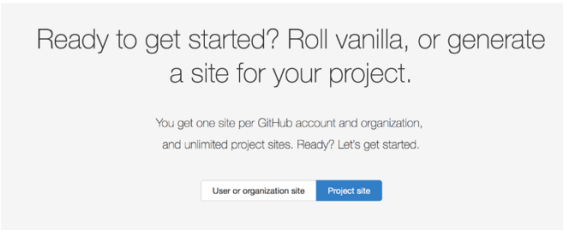


Рисунок 10.6. Створити репозиторій на Github

Крок 2. Налаштувати GitHub Pages для репозиторія

Перейдіть у налаштування (рис. 10.7.) для свого репозиторія (вкладка “Settings”):

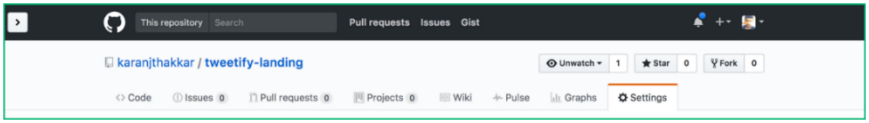


Рисунок 10.7. Налаштування в репозиторій на Github

У розділі “Github Pages” виберіть провідну гілку (рис. 10.8) для обслуговування Вашого вебсайту (master branch) :

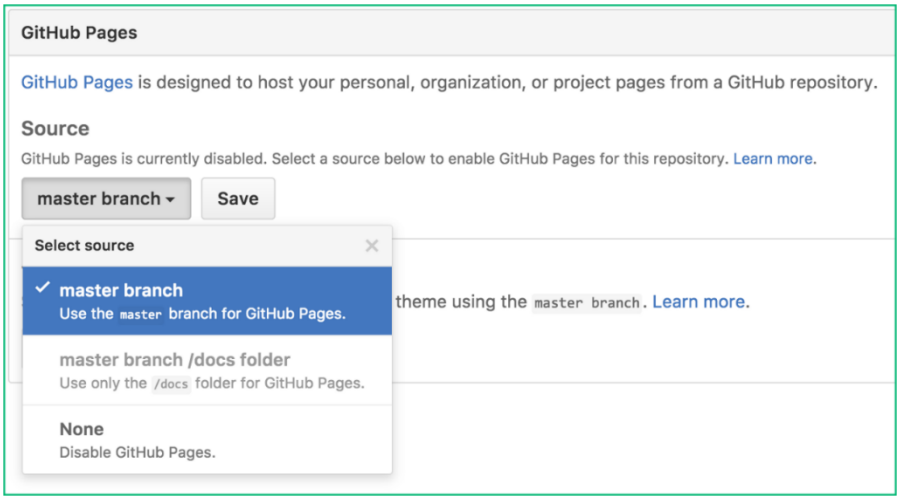


Рисунок 10.8. Гілка в репозиторії на Github

Після цього можете перейти на https://<назва Вашого сайту>.github.io/repository, щоб побачити сайт у дії, як показано на зображенні нижче (рис. 10.9):

Рисунок 10.9. Приклад сайту на Github

Крок 3. Додати власний домен

Додайте домен, який у вас є (рис 10.10.) і натисніть на кнопку “Save”:

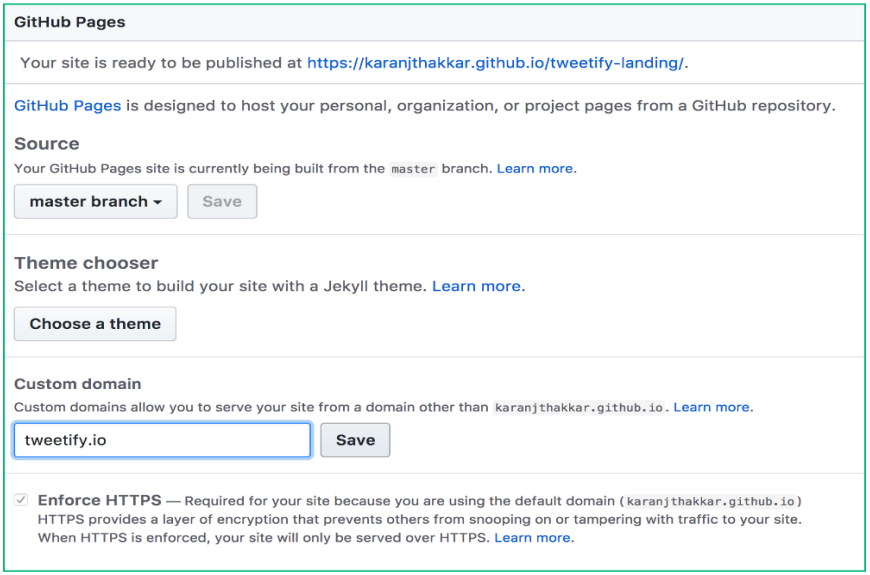


Рисунок 10.10. Задання домену на Github

Тепер треба налаштувати Cloudflare для використання всіх приємних фішок, про які згадувалося на початку статті.

Крок 4. Налаштувати домен у Cloudflare

Залогіньтесь у *Cloudflare*. Якщо Ви там уперше, то побачите таку ж картинку, як на зображенні нижче. Якщо Ви вже використали цей сервіс, то можете відразу натиснути на опцію “Add Site” в рядку навігації (вгорі праворуч), потім додати новий домен (рис .10.11.). Введіть ім’я домена, з яким працюватимете, і натисніть на “Begin Scan”:

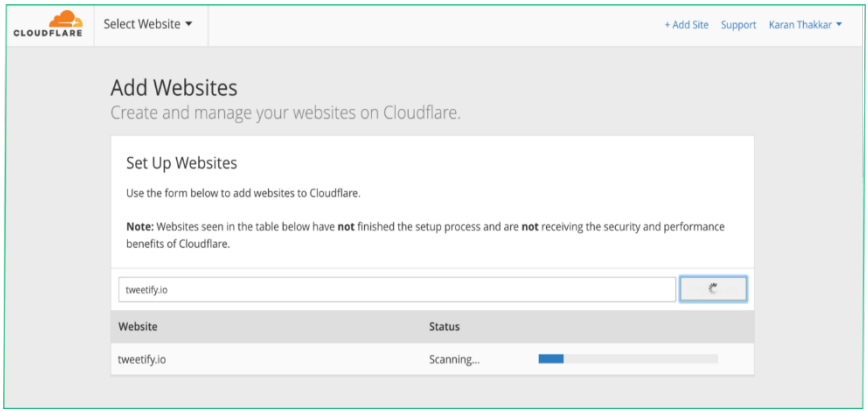


Рисунок 10.11. Налаштування домену на Github

Крок 5. Налаштувати записи DNS для Вашого домена

На цьому етапі Ви вказуєте Cloudflare на Ваш аліас на GitHub (рис. 10.12), використовуючи два записи A (адресні записи, відповідність між ім’ям і IP-адресою):

192.30.252.153

192.30.252.154

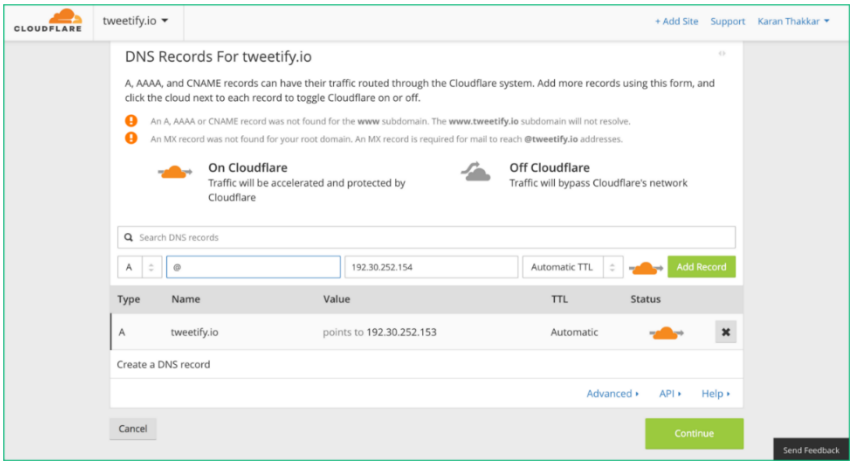


Рисунок 10.12. Налаштування запису DNS на Github

Після налаштування всі запити до Вашого домена (наприклад, Ваш власний домен.com) будуть перенаправлені на Ваш сайт на GitHub (рис. 10.13), налаштований на крок 3.

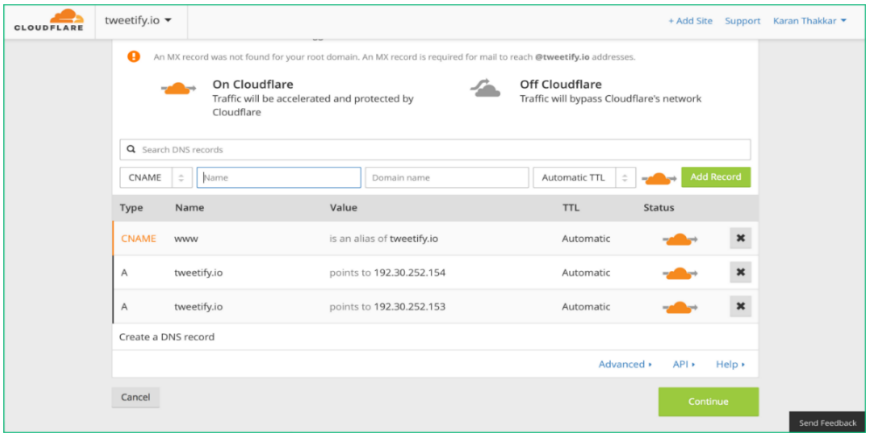


Рисунок 10.13. Властивості запису DNS на Github

І є ще дещо. Наприклад, Вам хочеться використати субдомен типу www для свого вебсайту, тобто www.Ваш власний домен.com. Для цього Вам треба буде

додати запис CNAME, який вкаже кореневому домену (@) на Ваш піддомен (www).

Примітка. Не намагайтеся відразу перейти до свого користувацького домена. Це не спрацює, адже Ви виконали тільки налаштування Cloudflare для GitHub.

Натисніть «Continue» для переходу до наступного кроку.

Крок 6. Вибрати безкоштовний план у Cloudflare

Безкоштовний план для Cloudflare надає багато опцій. Виберіть план (рис. 10.14):

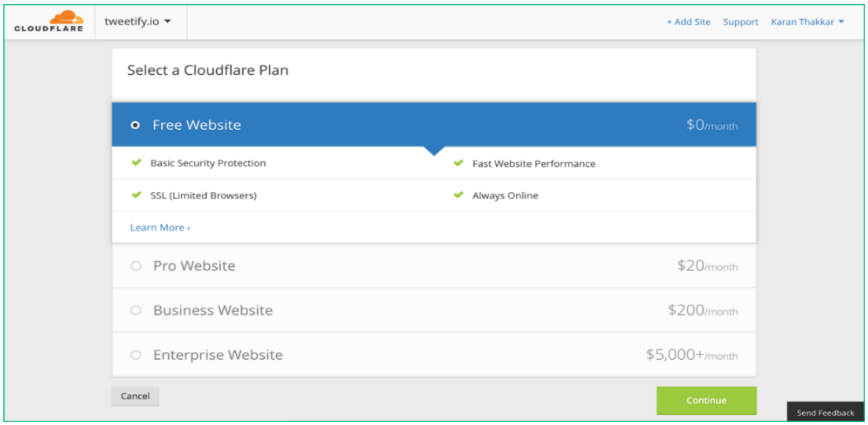


Рисунок 10.14. Вибір плану у Cloudflare

Натисніть «Continue», щоб перейти до наступного кроку.

Крок 7. Оновити сервери імен у реєстратора

Після того, як Ви потрапили на цю сторінку, відкрийте її в одній вкладці, а реєстратор доменних імен (місце, де Ви купили свій домен) – в іншій.

Тепер Вам треба замінити наявні сервери імен (nameservers) у налаштуваннях Вашого домена на сторінку реєстратора тими, що показує Cloudflare. Ось приклад того, як реєстратор поводиться після успішної заміни (рис. 10.15):

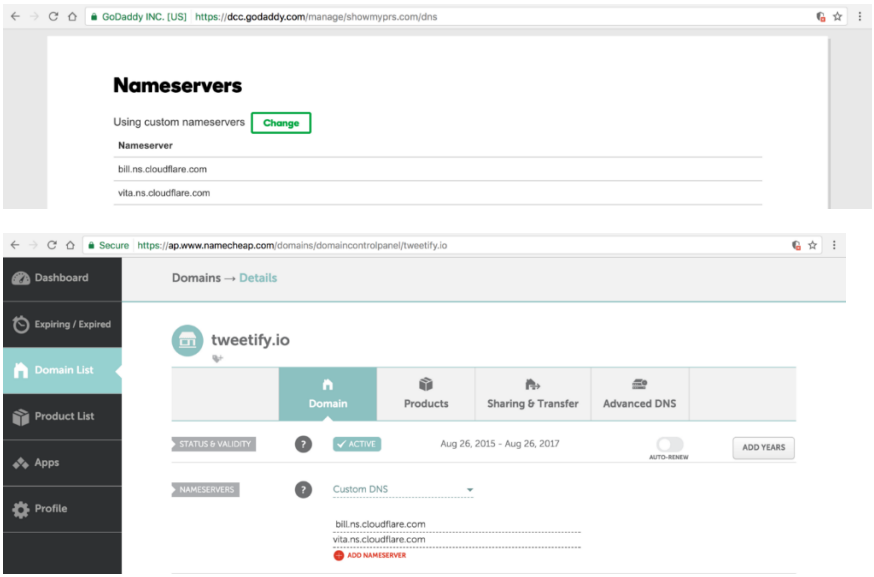


Рисунок 10.15. Приклад

Ви майже налаштували власний домен, щоб використати Cloudflare як DNS-провайдер. Тепер, якщо Ви перейдете до пункту «Overview» згори, то виявите, що сервіс все ще чекає зміни серверів імен.

Треба дочекатися, поки статус на вкладці «Overview» зміниться на «Active» (рис. 10.16):

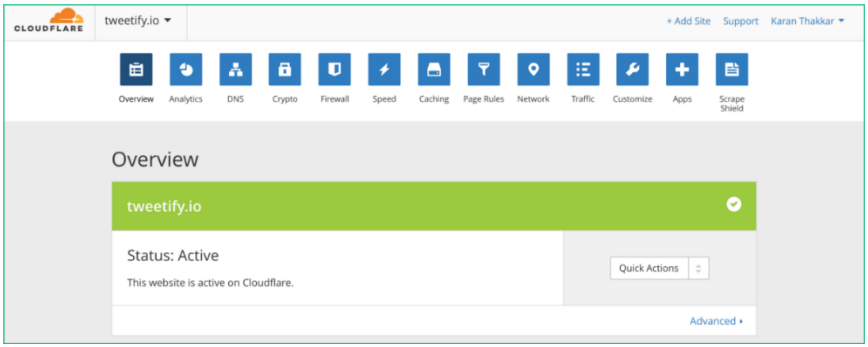


Рисунок 10.16. Зміна серверів імен

Тепер можете спробувати зайти на свій сайт, він повинен запрацювати.

Крок 8. Налаштувати мініфікацію

У налаштуваннях швидкості («Speed») у розділі «Auto Minify» поставте галочку для автоматичної мініфікації біля всіх пунктів: JavaScript, CSS, HTML.

Ці налаштування застосовуються Cloudflare «на льоту» один раз, а потім кешуются. Щоразу, коли будь-яке з Ваших особистих налаштувань змінюватиметься, Cloudflare все кешуватиме.

Перевага мініфікації полягає в тому, що розмір вихідного коду істотно зменшиться, оскільки Cloudflare видалить усі непотрібні пропуски і коментарі.

Крок 9. Налаштувати термін дії кеша браузера

Якщо Ви прокрутите вниз ту саму сторінку, де знаходиться «Auto Minify», то знайдете розділ «Browser Cache Expiration». В ідеалі термін дії має бути встановлений на 30 днів / 1 місяць, щоб *WebpageTest* не видавав ніяких попереджень.

Цей час показує, що після завантаження Вашого сайту в будь-якому браузері останній не проситиме дані вдруге, поки період кешування не сплине.

До того як виконати наступний крок, перевірте криптографічні налаштування («Crypto») на Cloudflare. У розділі SSL повинен з'явитися зелений напис «Active Certificate».

Далі треба налаштувати постійне використання HTTPS на Вашому сайті. Щоб все працювало без проблем, знадобиться активний сертифікат на Cloudflare.

Крок 10. Налаштувати правила для сторінок

На цьому кроці Ви зробите дві важливі речі:

переадресацію всіх запитів з www.Ваш власний домен.com на Ваш власний домен.com;

переадресацію всіх запитів з http://Ваш власний домен.com на https://Ваш власний домен.com.

Зайдіть у вкладку «Page Rules» і натисніть на «Create Page Rule» (рис. 10.17):

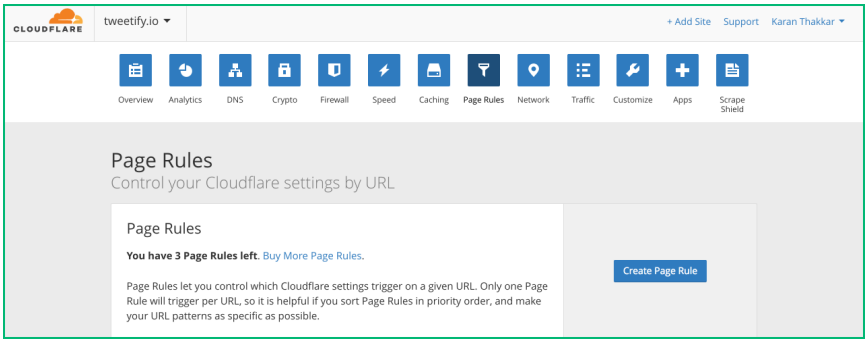


Рисунок 10.17. Вкладка «Page Rules»

Для обробки переадресації з www.Ваш власний домен.com на Ваш власний домен.com замініть tweetify.io на картинці нижче на Ваш власний домен.com. Натисніть «Save and Deploy» (рис. 10.18):

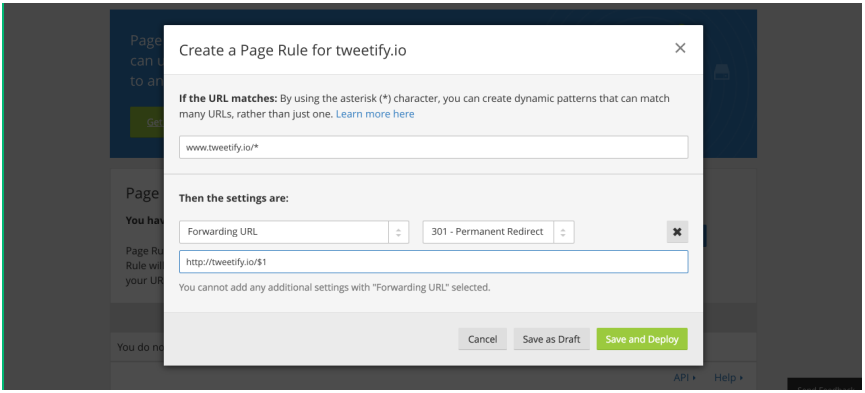


Рисунок 10.18. Переадресація

Для обробки переадресації з http:/www.Ваш власний домен.com на https:/Ваш власний домен.com, замініть tweetify.io на Ваш власний домен.com. Натисніть «Save and Deploy»:

Нагадаємо, що популярність протоколу HTTPS підвищилась: частка HTTPS-запитів перетнула позначку в 50 %, а число сайтів, що використовують цей

протокол, за останній рік подвоїлося. Використання HTTPS – це норма, а не виняток, тому радимо не нехтувати необхідними налаштуваннями.

Крок 11. Налаштувати HSTS

Тепер треба знову зайти в криптографічні налаштування («Crypto»), після чого прокрутити сторінку вниз до «HTTP Strict Transport Security (HSTS)». Кликніть «Enable HSTS». Вас запитають про те, чи точно Ви знаєте, що робите (рис. 10.19). До того як Ви натиснете на «I understand», прочитайте, навіщо нам взагалі вмикати HSTS.

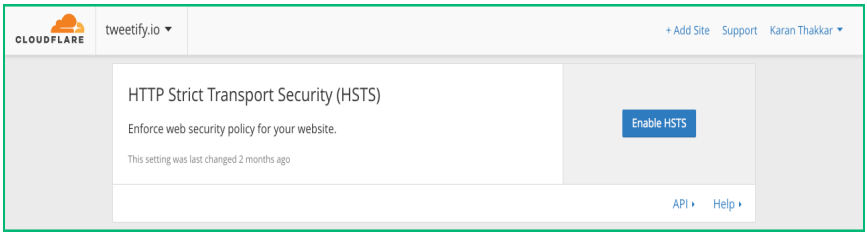


Рисунок 10.19. Налаштування HSTS

Якщо користувач раніше вже відкривав Ваш сайт, то тепер він завжди автоматично прямуватиме на HTTPS-версію сайту. Це прискорює завантаження сайту при подальших відвідуваннях, оскільки перенаправлення з HTTP на HTTPS відбувається на клієнтській стороні, а не за допомогою правила в Cloudflare, яке ми створили на попередньому кроці.

Залишилося активувати ті самі опції, що й на скриншоті (рис. 10.20):

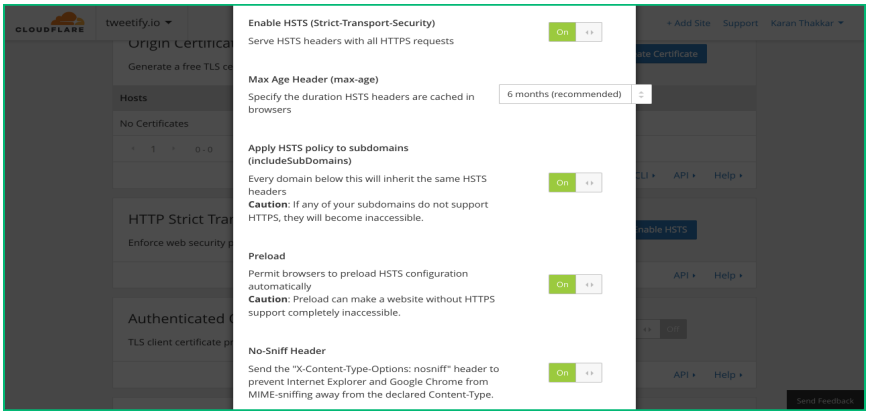


Рисунок 10.20. Активування опцій

Такі заголовки Cloudflare додаватиме до запитів до Вашого домена після того, як Ви налаштували HSTS:

```
strict-transport-security: max-age=15552000; includeSubDomains; preload
x-content-type-options: nosniff
```

Рисунок 10.21. Налаштування HSTS

Сайт готовий.

11. ХОСТІНГ

11.1. Сервіс хостингу

Дата-центр хостингової компанії

Поняття «хостинг» (hosting, host) означає сервіс із надання обчислювальних потужностей (ресурси процесора, ОЗП, дискового простору тощо) та фізичного розміщення інформаційних ресурсів (сайтів) на серверах, що знаходяться на території провайдера і постійно під'єднані до магістралей Інтернету.

Послуга хостингу, як правило, складається з надання місця, що використовується для зберігання сайтів, а також забезпечення працездатності необхідних сервісів: електронної пошти, баз даних, зберігання файлів та послуги служби DNS, які можуть надаватися як окремі послуги або міститися в комплексі.

Дата-центр (Data Center) хостингової компанії представляє окрему будівлю або велике приміщення, де розміщено сервери, телекомунікаційне обладнання та інші сучасні пристрої, що забезпечують збір, зберігання та обробку різної інформації.

Головною особливістю всіх дата-центрів є планування приміщень, розвинена інфраструктура і значні площі, що відповідають потребам стабільного функціонування центру і наданню послуг клієнтам. Розміщення дата-центру в окремих будівлях обґрунтовано економічно. Зосередження в одному місці всіх виробничих потужностей призводить до зменшення сукупної вартості наявного в розпорядженні організації обладнання. Для того, щоб дата-центр повністю окупав себе, необхідно щоб він був досить великим, тут економія будується на принципі масштабу.

Дата-центри поділяються на корпоративні, які обслуговують інтереси тільки однієї компанії, і комерційні, основною метою яких є обслуговування сторонніх організацій. Також розрізняють дата-центри залежні і незалежні від провайдера.

За розмірами дата-центри є:

Малі – розміщуються в непристосованих для серверів приміщеннях, можуть виконувати мінімум завдань і використовувати звичайне обладнання.

Середні – розташовуються на орендованих майданчиках необхідного розміру і працюють за наявними (не завжди достатньо широкосмуговими) каналами зв'язку.

Великі – розміщуються в окремих будівлях, які побудовані спеціально для дата-центру. Тут є власні (оптоволоконні, супутникові) канали зв'язку, через які сервери під'єднуються до високошвидкісних магістралей.

Дата-центри різняться за надійністю. Фахівці виділяють чотири типи дата-центрів: перший матиме найнижчий показник надійності, а четвертий – найвищий. Тип надійності залежить від інфраструктури та забезпечення вимог безпеки.

Тип надійності Tier 1 (стандарт TIA-942). Дата-центри не зможуть функціонувати, якщо відмовило обладнання або відбуваються планові ремонтні роботи.

Тип надійності Tier 2. Дата-центр продовжить працювати, якщо станеться невелика поламка або збій, але при проведенні ремонтних робіт функціонування дата-центрів припиняється.

Тип надійності Tier 3. Дата-центр працює в будь-яких ситуаціях і має хоча б один резервний сервер.

Тип надійності Tier 4. Дата-центр працює в будь-яких ситуаціях, зарезервовано все обладнання та інформація, що зберігається на ньому.

Хостинг

Виділений сервер (Dedicated hosting)

Оренда фізичного сервера цілком для одного користувача. Цей вид хостингу використовується для розміщення складних Інтернет-проєктів, а також для реалізації нестандартних сервісів, які не можуть поділяти місце з іншими проєктами на одному сервері і вимагають для успішного функціонування всі ресурси сервера.

Хостингова компанія підтримує і забезпечує резервне копіювання сервера, безпеку, електроживлення та інші аспекти підтримки. Всі проблеми з програмним забезпеченням вебсервера вирішує орендар серверу.

Власник виділеного сервера може встановити на комп'ютер будь-яку операційну систему будь-якої конфігурації, вести різні технічні роботи, не турбуючись, що його дії порушать роботу інших служб або серверів. Сервер баз даних, розташований на окремому комп'ютері, не дозволить кіберзлочинцю серйозно зіпсувати бази, у гіршому випадку буде завдано мінімальної шкоди.

Виділений сервер є кращим рішенням для компанії, що вимагає широкої смуги пропускання, з малим штатом і ресурсами для встановлення, підтримки та створення безпечного мережного оточення. Також, виділений вебсервер часто використовується як варіант для бізнесу – перепродажу хостингу.

Оренда виділеного сервера в Україні становить $50-500 на місяць.

Віртуальний хостинг

Віртуальний хостинг (Shared hosting)

Користувачу надається місце на диску, що використовується для розміщення сайтів. Ресурси одного сервера розподіляються між всіма користувачами. На одному сервері може розміщуватися від 50 до 1000 користувачів. Якщо провайдер хостингу знехтує засобами безпеки, користувач на сервері може мати безперешкодний доступ до сайтів інших користувачів, втім у професійних провайдерів такої проблеми не існує.

Віртуальний хостинг як послуга має ряд обмежень.

Кількісні обмеження

- Розмір дискового простору.
- Кількість місячного трафіку.
- Кількість сайтів, які можна розмістити в рамках хостингу, як однієї послуги.
- Кількість баз даних і кількість місця під бази даних.
- Кількість поштових скриньок і FTP-акаунтів.

Якісні обмеження

- Вільні ресурси процесора, оперативної пам'яті, які впливають на швидкодію сервера. У зв'язку з тим, що на сервері віртуального хостингу, зазвичай, знаходиться багато різних сайтів, навантаження є непропорційним і деякі хостери обмежують ресурси сервера (в основному процесора) для скриптів користувача.
- Критерієм вибору віртуального хостингу є операційна система сервера (UNIX/Linux-хостинг, Windows-хостинг), оскільки від цього залежить програмне забезпечення, яке буде підтримувати функціональність тих чи інших сервісів. Для організації послуги віртуального хостингу використовуються сервери, що працюють під управлінням ОС Unix-клонів, наприклад, FreeBSD, GNU/Linux, а також під управлінням ОС Windows.

Віртуальний хостинг є самим економним видом хостингу, і буде кращим вибором для невеликих проєктів. Вартість оренди складає $1-10 на місяць.

Віртуальний виділений сервер (Virtual Server)

Віртуальний виділений сервер виглядає для користувача як і звичайний виділений сервер, проте, на одному реальному сервері може розташовуватися кілька віртуальних виділених. Клієнту надається окремий віртуальний сервер із власною операційною системою. Подібно до справжнього фізичного сервера, в користувача є власні IP-адреси, таблиці маршрутизації, root-доступ, процесорний час і частка загальної пам’яті.

Всередині віртуального виділеного сервера користувач вільний як створювати свої версії системних бібліотек, так і змінювати вже існуючі, виконувати будь-які дії зі всіма файлами, навіть з тими, які знаходяться в кореневій або іншій службовій директорії.

Сервіс призначений для Інтернет-проєктів середньої складності. Утримання виділеного віртуального хостингу в України складає $10-100 на місяць

Провайдери розрізняють два види віртуального виділеного хостингу: VDS (Virtual Dedicated Server) або VPS (Virtual Private Server). Різниця між VPS і VDS проглядається не з боку користувача, а з боку провайдера і сервера.

VPS реалізується за допомогою роботи програмного забезпечення, що дозволяє для кожного виділеного сервера запустити свою копію операційної системи. Для цього на машині сервера встановлюється операційна система, в якій менеджер віртуальних серверів і виділяє сервер для клієнта. Віртуалізація забезпечується засобами операційної системи.

VDS представляє такий же виділений віртуальний сервер, що керується власною операційною системою, а не її клонами. У кожного віртуального сервера є власна ОС. Схема реалізації виходить приблизно така: на машині сервера встановлюється менеджер віртуальних серверів, потім на кожен з них встановлюється своя операційна система, і тільки потім створюються сервери для кінцевого користувача. Таким чином, віртуалізація виходить фактично апаратною.

Вважається, що VDS дещо надійніше і зручніше, оскільки не виникає проблем сумісності, наприклад, з розрядністю ядер операційних систем, оскільки ОС може бути запущена будь-яка. VDS може бути істотно дорожче в тарифних планах хостинг-провайдерів.

Переваги

Ціни на послуги. Віртуальні виділені сервери мають меншу ціну на виділені лінії, втім, вони мають достатню потужність, порівняну з виділеними серверами.

Налаштування. У порівнянні з виділеними серверами, віртуальні виділені налаштовуються простіше і швидше. Виділений сервер повинен бути зібраним, змонтованим і встановленим, а віртуальний виділений може бути готовий до використання вже за кілька годин після встановлення необхідного програмного забезпечення.

Доступ адміністратора. На відміну від інших хостингових схем, клієнт має повний доступ від імені адміністратора і може вносити зміни в налаштування в будь-який час.

Потужність. Віртуальні виділені сервери встановлюються на потужних машинах, що надає багато можливостей.

Безпека. Віртуальний виділений хостинг здатний витримати більшість відомих DDoS атак і вразливостей. Безпека сервера є важливою і повинна бути вирішальним фактором у виборі плану хостингу.

Подальша модернізація. Якщо є потреба в додаткових ресурсах, їх можна легко і швидко збільшити. З виділеним сервером необхідно чекати, поки буде встановлено інший апаратний компонент.

Недоліки

Менша потужність. VPS є менш потужними за виділені сервери. Якщо бізнес передбачає багато функцій, таких як форуми, значна кількість електронної пошти та інших ресурсоємних завдань, то для забезпечення безперебійної роботи необхідний виділений сервер.

Обмеження. Як і у випадку з віртуальною хостинговою схемою можуть з'явитися обмеження. Якщо один із VPS споживає занадто багато ресурсів, робота інших VPS може сповільнитися. Пропозиції «безлімітної смуги пропускання» в реальності замовчують ці важливі моменти.

Компетентність. На відміну від віртуальної хостингової схеми, яка є легкою в налаштуванні і запуску, віртуальні виділені сервери вимагають більш високого рівня технічних знань щодо налаштування, підтримки та управління.

Повільна підтримка. Якщо виникає проблема з віртуальним виділеним сервером, яку клієнт не може вирішити, може пройти деякий час, перш ніж технічна підтримка відреагує на запити. Користуючись віртуальним хостингом, можна позбутися від проблем значно швидше.

Cloud-hosting (Хмарний хостинг)

Хмарні технології – це одночасне використання ресурсів кількох серверів. Хмарний хостинг (Cloud-hosting, кластерний хостинг) є особливою моделлю послуг, яка за вимогою забезпечує зручний і повсюдний мережний доступ до серверів, сервісів, додатків, пристроїв зберігання даних. При цьому послуги надаються оперативно, з невеликими експлуатаційними витратами.

Головною особливістю є можливість придбання ресурсів за потребами і оплата за послуги відповідно до навантаження на сервер. При замовленні хмарного хостингу, потрібно лише вибрати, який розмір дискового простору потрібен

для сайту. Оплата хостингу залежить від того, скільки процесорного часу було витрачено на роботу з даними.

Чим більше буде зростати відвідуваність сайтів, тим більше буде використано ресурсів серверів. Інших параметрів, таких, які зазначаються при замовленні звичайного віртуального хостингу, наприклад, пропускна здатність на місяць (трафік), кількість можливих доменів, суб-доменів, сайтів, баз даних, у хмарному хостингу немає. Все це надається без обмежень.

Характеристики хмарного хостингу

On-demand self service. Користувачеві доступний будь-який обсяг послуг. Причому для «дозакупки» послуг користувачеві не потрібно докладати зусиль (він робить все з панелі управління) і все відбувається миттєво.

Broad network access. Принцип мережної доступності. Хмарне рішення є доступним з будь-якого комп'ютера, з якого є доступ в Інтернет.

Metered use. Принцип оплати за фактом: скільки спожито послуг, стільки й буде оплачено.

Elasticity. Принцип гнучкості закупівлі. Можливість споживання потужності невеликими квантами і довільний час використання будь-якого обсягу послуг протягом часу, який хоче користувач. Тобто, можна рік використовувати малий сервер, потім пару днів половину дата-центру, а потім – знову малий сервер.

Resource pooling. Принцип незалежності від «заліза». Користувачеві невідомо і неважливо, на якому апаратному вузлі хмари і навіть на якому континенті зараз працюють його віртуальні машини. Користувач не повинен за жодних умов залежати від працездатності якого-небудь конкретного вузла.

Порівняння віртуального і хмарного хостингу

Віртуальний хостинг не витримує високих показників відвідуваності, а хмарний хостинг має можливість динамічного розподілу ресурсів. Це означає, що у разі виникнення підвищеного навантаження на один із серверів, хмарний хостинг може залучити обчислювальні ресурси менш завантажених серверів, що забезпечує стабільність і надійність роботи сайтів (рис. 11.1).

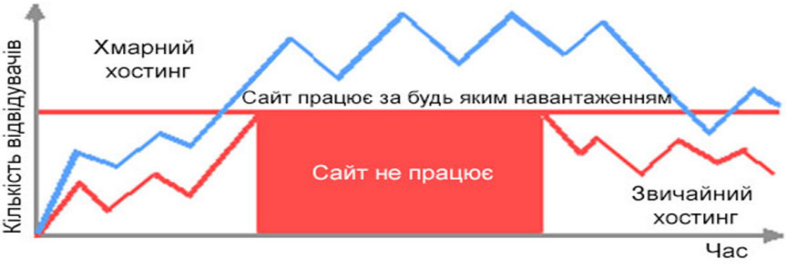


Рисунок 11.1. Порівняння працездатності віртуального та хмарного хостингів

Віртуальний виділений і хмарний хостинг

Віртуальний виділений сервер є більш потужною платформою, що має виділену оперативну пам'ять, процесорний час і обсяг твердого диска зі своєю ОС. Є можливості встановлення необхідного програмного забезпечення, проводити його налаштування і сайт не буде залежати від сайтів сусідів по фізичному серверу.

Однак, навантаження на сервер варіюється в часі: наприклад, вранці VS може простоювати, а ввечері будуть значні навантаження. Хмарний хостинг характеризується погодинною тарифікацією, а також зміною тарифного плану в ручному або автоматичному режимі. Якщо на даний момент відвідувачів сайту мало, доцільно перейти на дешевший трафік, а коли показники відвідуваності починають зростати, можна взяти навіть цілий кластер з кількох серверів (рис. 11.2).

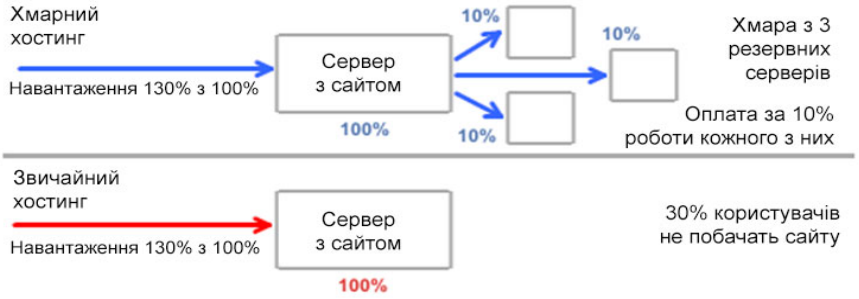


Рисунок 11.2. Порівняння працездатності віртуального виділеного та хмарного хостингів

Виділений фізичний сервер і хмарний хостинг

Хмарний хостинг робить існування виділеного фізичного сервера абсолютно недоцільним – навіщо оплачувати послуги виділеного сервера, якщо хмарний можна розігнати до рівня цього сервера за лічені секунди, а потім, за необхідності, зменшити.

Отже, хмарний хостинг має багато переваг над іншими видами хостингу, але слід пам'ятати те, що чим більше ресурсів споживає сайт, тим більше коштів доведеться платити, а розрахувати обсяг споживання ресурсів заздалегідь практично неможливо.

Колокація

Колокація є популярною опцією в організаціях, які хочуть отримати високу ступінь контролю над програмним і апаратним забезпеченням сервера без істотних витрат на організацію дата-центру на своїй території.

Забезпечення інфраструктури. Зазвичай, хостер, який надає послуги колокації, забезпечує місце для встановлення сервера, всі необхідні під’єднання (електрика і мережа) і відповідну інфраструктуру, а клієнт використовує власне апаратне і програмне забезпечення.

Ефективність використання. Колокація дозволяє клієнтам накопичувати досвід роботи з серверами, який потім можна використовувати для організації власного дата-центру або для надання послуг хостингу.

Технічна підтримка. Більшість провайдерів забезпечують підтримку колокації за схемою 24/7/365. В дата-центрі команда провайдера стежить за енергетичним забезпеченням, системами зв'язку і безпеки. Часто провайдери пропонують безпосередню підтримку клієнтів у обслуговуванні серверів за телефоном, вебчатом або інших засобів зв'язку.

Безпека і достатність технічних ресурсів для сервера. Ці послуги можуть суттєво відрізнятися в залежності від провайдера послуг. Звичайна компанія може пропонувати лише серверну стійку, з'єднання з Інтернет і просте забезпечення електроживленням, а потужні хостери можуть мати окремі системи UPS для кожного сервера, локальну систему електроживлення, спеціальні вентиляційні установки. Найпростіший сервіс колокації може

пропонувати систему безпеки, яка буде виглядати як звичайний замок на двері, а кращі провайдери мають багаторівневу систему безпеки для збереження обладнання і даних клієнта.

11.2. Основні параметри хостингу

Серверне програмне забезпечення

Одним із головних аспектів при виборі хостингу є операційна система, яку встановлено на сервері. Саме від цього фактора залежить вибір програмного забезпечення, яке буде підтримувати безперебійну роботу сервісів.

Unix-клони. На сьогодні, Unix-хостинг є найпопулярнішим у світі. Практично у всіх вітчизняних та зарубіжних провайдерів на серверах встановлено програмне забезпечення для Unix-платформ. Відвідувачі, в яких на комп'ютері встановлено операційну систему Windows, зможуть нормально відвідувати сайт.

Windows. Windows-хостинг це розміщення сайтів на серверах під управлінням ОС Windows Server. У першу чергу, це надає можливість працювати з сайтами, які потребують підтримки технологій ASP.net і Ms SQL. Обидві технології розробляються Microsoft і підтримуються виключно Windows-серверами. Дане ПЗ є платним, тому часто Windows-хостинг є дорожчим, ніж Unix-хостинг.

Підтримка мов програмування

Сучасні сайти є достатньо складними комплексами. Розробники мають широкі можливості для вибору засобів створення динамічних сайтів та забезпечення доступності такого змісту, незалежно від змін у сучасних технологіях програмного і апаратного забезпечення. Для цього хостинг має підтримувати поширені мови програмування, зокрема: Java, PHP, Ruby on Rails, Python, ASP.net.

Практично всі хостинги підтримують мову PHP, завдяки її швидкості, простоті, надійності, незалежності від платформи і можливості роботи з файлами, базами даних MySQL, зображеннями, мультимедіа тощо. Мова PHP добре справляється з власними завданнями і підтримує багато розширень.

Технологію ASP.net підтримує лише провайдер, який пропонує Windows-хостинг. Компанія Microsoft для популяризації Windows-server створила ASP-конструкції, які дають певні переваги над Unix-платформою, втім це спірне питання.

Використання баз даних

Якщо сайт передбачає зберігання великої кількості даних, до яких повинен бути динамічний доступ (облікові записи користувачів, карти, новини, оголошення, форуми, блоги тощо), то хостинг має підтримувати використання баз даних. Потужною і найбільш популярною базою даних є MYSQL з широкими можливостями обробки та зберігання вебдодатків.

Панель керування

Панель керування хостингом – інструмент, за допомогою якого користувач отримує доступ до свого сайту в мережі, щоб встановлювати, оновлювати чи змінювати сайт.

Провайдери можуть застосовувати різні варіанти панелей керування:

Власна панель керування використовується провайдерами досить часто. Такі панелі спрощують роботу для початківців, роблячи її більш зручною та зрозумілою. Хостинг із власною панеллю керування має меншу собівартість за рахунок того, що провайдер не оплачує ліцензію. Але, якщо для стандартних платних панелей керування в Інтернеті можна знайти багато інформації щодо способів вирішення проблем, то для цього випадку, вирішити проблеми можна виключно за допомогою технічної підтримки або інструкцій, що існують лише на сайті провайдера, який розробляв панель.

cPanel – одна з популярних панелей керування хостингом. Її широко використовують як західні, так і вітчизняні провайдери. До переваг панелі відносять стабільну роботу з поширеними технологіями (Apache, MySQL, PHP і т. д.), багатомовний інтерфейс.

ISP Manager – це платна панель керування, яку розробила російська компанія ISPsystem. Панель має кілька мовних інтерфейсів і популярність за кордоном. До основних особливостей можна віднести легку роботу зі стандартними налаштуваннями сервера, що робить її більш вигідною для початківців, а також

підтримку API-інтерфейсу. Інтерфейс є достатньо простим, зрозумілим та зручним.

DirectAdmin – це панель керування, яка створена канадською компанією JBMC Software. Панель є стабільною і швидкою, працює зі всіма популярними хостинг-технологіями особливо з Unix-системами. Русифікованого інтерфейсу наразі панель не має.**Parallels Plesk** – це панель керування від швейцарської компанії Parallels Inc. Дана панель найбільш підходить досвідченим користувачам та професіоналам, для роботи зі складними проєктами. Однаково добре працює і з Windows-серверами, і з Unix.

Встановлення та підтримка систем управління контентом (CMS)

Wordpress – це безкоштовна CMS, яка використовується в основному для створення блогів. Особливих вимог до хостингу вона не має і, зазвичай, проблем з її розгортанням не виникає.

Joomla – популярна система управління сайтом, яка підходить практично для будь-яких сайтів і приваблює користувачів своєю простотою. Joomla споживає багато ресурсів сервера, на якому зберігається сайт, тому, при великих навантаженнях провайдери можуть відключати такі сайти.

Drupal – це система управління сайтом, яка добре підходить для складних сайтів (портали, Інтернет-магазини тощо) і справляється з великими навантаженнями на сервер. Хостинг Drupal-сайтів є менш вимогливим до ресурсів і, відповідно, є виграш у ціні. Багато сучасних провайдерів не лише підтримують роботу даної CMS, а й безкоштовно надають її останні русифіковані версії.

Typo3 – це популярна безкоштовна система управління сайтом, не вибаглива до хостингу, використовує мову PHP, традиційні бази даних (MySQL, Oracle Database, PostgreSQL), а також може працювати з серверами Apache і IIS. Підтримується більшістю популярних вітчизняних провайдерів.

MODx – це безкоштовне середовище розробки та система управління сайтом, яке здебільшого використовується професіоналами і дозволяє розробляти сайти різної складності. У порівнянні з іншими CMS, створює менше навантаження на ресурси хостингу. Таким чином, її можна сміливо рекомендувати для великих проєктів з великою відвідуваністю. Єдиним

вагомим недоліком можна назвати проблеми в роботі сайту при використанні більше 5000 сторінок.

Доступ за FTP-протоколом

Після створення сайту на локальному комп'ютері його потрібно перенести на сервер. Одним із способів перенесення файлів є використання протоколу FTP для завантаження файлів з локального комп'ютера на віддалений сервер і навпаки для резервного копіювання. Хостер надає користувачу ідентифікатори для доступу через зовнішні FTP-клієнти або пропонує доступ через файловий менеджер, який знаходиться на панелі керування хостингом.

Можливості ознайомлення з хостингом

Часто компанії платного хостингу пропонують потенційним замовникам послуг скористатися безкоштовним тестом на певний період часу. Після закінчення зазначеного терміну користувач приймає рішення: чи підходять йому послуги обраної хостингової компанії.

Безкоштовний тестовий період. Хостер надає користувачеві послуги безкоштовно, але на певний час. На час пробного періоду можуть встановлюватися деякі обмеження, зазвичай, заборона доступу до певних опцій та часові обмеження, тобто один провайдер може дати безкоштовно спробувати свій хостинг на 5 днів, а інший на 30.

Moneyback період. Послуга Moneyback передбачає повернення грошей, якщо клієнту не сподобався хостинг протягом певного періоду часу (15-30 днів, іноді 60-90).

Системи безпеки та захисту ресурсів

Безпека хостингу залежить насамперед від власників сайтів, оскільки за статистикою близько 35% зламів припадає на ті випадки, коли логіни/паролі крадуть не з серверу хостинг-провайдера, а з комп'ютера веброзробника. Засобом захисту від цього є сучасне платне антивірусне програмне забезпечення.

Захист від DOS-атак

Загрози у вигляді DDoS-атак (Distributed denial of service. Розподілена атака типу «Відмова в обслуговуванні») можуть привести до серйозних наслідків,

коли на сайт-жертву одночасно надходять запити від величезної кількості спеціально заражених комп'ютерів (ботів). Їх кількість може обчислюватися десятками і сотнями тисяч. Сайт свідомо перевантажують безглузд https://www.facebook.com/groups/1179293055426737/ими запитами, непотрібною інформацією, невідомими пакетами даних, що призводить до постійних зависань, збоїв, змін параметрів роботи системи, а іноді й до повного припинення діяльності сайту на невизначений термін. У результаті – псується репутація компанії і втрачаються великі кошти.

Хостинг-компанія має забезпечити якісний захист від DDoS-атак. IT-фахівці повинні правильно налаштувати міжмережний екран і апаратуру, що блокує проникнення шкідливих об'єктів. Постійний моніторинг вхідного і вихідного трафіку, цілодобове стеження за статистикою на серверах і вивчення її в динаміці. Це відбувається шляхом побудови спеціальних графіків. Система запам'ятовує, як повинні виглядати графіки при нормальній роботі. Всі аномальні графічні відхилення моментально розпізнаються. Відбувається автоматичне блокування надходження підробленої інформації.

Система резервного копіювання даних

У реальності завжди існує можливість виникнення непередбачуваного фактора в складній інформаційній системі. Це може бути спроба злому, необережна дія користувача, що запустив вірус, некоректна робота пакету оновлення програмного забезпечення, несправність обладнання або помилка в діях адміністратора. А результатом впливу таких факторів стає втрата важливих даних, які можуть накопичуватися протягом багатьох років.

Більшість хостинг-провайдерів пропонують послуги BackUp – сервісу щоденного резервного копіювання баз даних, файлів сайту, пошти, FTP-акаунтів і багатьох інших параметрів хостингу. Сайт та його налаштування зберігаються в окремому місці, а за необхідності сайт можна повернути до останньої збереженої версії. Копіювання даних може здійснюватися на поточний і BackUp (додатковий) сервер, що розташований окремо від серверів провайдера або в іншому дата-центрі. Це проводиться на випадок, якщо щось трапиться з сервером на якому зберігається сайт.

11.3. Опції хостерів

Продаж доменів

Популярним для хостинг-компаній є заробіток на продажі доменів, оскільки користувачу зручно разом із покупкою хостингу реєструвати домен для сайту.

Надання домену за цінами, які значно вищі за середньоринкові, заробляючи на різниці.

Надання домену за цінами нижче середньоринкових чи іноді безкоштовно як бонус. У цьому разі формальним власником домену буде хостер, який викуповує права на домени і вже від себе надає право користування ними клієнтам. Клієнт, що купив домен, потрапляє в залежність від власного провайдера, який теоретично може в будь-який момент відмовити в домені.

Рекомендується уважно читати додаткові умови акції, а також вивчати договір-оферту.

Паркування доменів

Паркінг або паркування домену надає користувачам можливість зберегти зареєстрований домен без вебсторінки. Розвиток подібного бізнесу пов'язаний із реєстрацією багатьох доменів, які не використовуються і знаходяться в стадії очікування завершення терміну своєї реєстрації. Більшість таких доменів реєструвалося «про запас», але термін їх використання не перевищує 365 днів. Перед веброзробником стоїть вибір:

1. Продовжити домен, щоб його не втратити.
2. Втратити домен і виділені на його реєстрацію гроші витрачаються марно.

Паркінг застосовується також для:

- Використання спеціальних сервісів, які не мають інтерфейсу користувача.

- Підвищення відвідуваності певних сайтів. Реєструються доменні імена, які збігаються за назвою з продуктом, послугою чи найменуванням відомих сайтів. В назві допускаються друкарські помилки або транскрипція назви офіційного сайту. В цьому випадку орієнтуються на інтуїтивну поведінку відвідувачів.
- Використання минулої історії доменів, на яких раніше перебували сайти. Багато сайтів мають постійних відвідувачів, які переходять на сторінку по пам’яті або з закладок.

Безлімітний хостинг

Безлімітний хостинг передбачає відсутність обмежень на основні параметрів хостингу: дисковий простір, кількість сайтів/доменів, поштових акаунтів, трафіку тощо. Такий вид хостингу популярний виключно за кордоном, а на вітчизняному ринку переважають провайдери, в яких можна побачити різноманітність тарифних планів, і пов'язаних з ними обмежень основних параметрів.

Абузостійкий хостинг

Абузостійкий хостинг надає можливість розміщувати інформацію будь-якого характеру без загрози, що хостингова компанія зможе без попередження видалити інформацію при першій скарзі (abuse). В якості «інформації будь-якого характеру» можуть виступати заборонений контент, неліцензовані продукти, порнографія, спам, несанкціонований продаж ліків тощо.

Абузостійкі сервери розташовані в країнах, де подібні матеріали дозволено офіційно, наприклад, в Панамі, Малайзії та Китаї, де закрити mp3-архів піратських копій не можуть навіть державні влади. Звичайно, послуги абузостійкого хостингу коштують значно дорожче віртуального.

Реселінг (recelling) – перепродаж хостингу

Реселінг хостингу (реселер-хостинг) – це розділення ресурсів фізичного або віртуального сервера з метою їх подальшого перепродажу під іншим брендом. При цьому реселер може або придбати сервер і розділити його на хостингові пакети самостійно, або скористатися готовим рішенням (спеціальним тарифом) від хостингової компанії.

Апаратне забезпечення і організацію його роботи (обслуговування фізичного сервера, співпраця з дата-центром) хостинговий провайдер бере на себе, а реселери оплачують виділені їм ресурси за фіксованою оплатою. Їх основним завданням є залучення якомога більшої кількості клієнтів на виділені ресурси, з метою отримання доходу.

Реселерам не потрібно отримувати ліцензію щодо телекомунікаційних послуг зв'язку та обладнання, але вони можуть використовувати власний бренд для продажу хостингу. Реселер сам здійснює підтримку користувачів. Якщо виникають труднощі технічного характеру в підтримці клієнтів, то цю опцію можна перекласти на системних адміністраторів головного хостера (умови і вартість обговорюються окремо).

Реселінгом часто займаються вебстудії, оскільки багато клієнтів, що замовляють розроблення сайту, погоджуються з подальшим технічним супроводом сайту: розміщення на хостингу, реєстрація доменного імені, технічна підтримка, антивірусний захист, внесення змін до сайту. Як правило, багато пересічних клієнтів не стануть самостійно шукати хостинг і скористаються повним комплектом, а вебстудія отримує додаткові дивіденди від замовника.

11.4. Критерії вибору хостингу та тарифного плану

Основні опції тарифів

Вибір платного варіанту хостингу – це засіб нормального функціонування і подальшого розвитку проєкту, особливо якщо він комерційний. Після рішення розмістити сайт в Інтернеті на платній основі постає задача вибору хостингової компанії та оптимального тарифу. Обраний тарифний план повинен повністю задовольняти потреби і не містити зайвих функцій, за які потрібно переплачувати.

Тарифні плани будь-яких хостингових компаній різняться за набором основних послуг, обсягом доступного дискового простору, максимально можливою кількістю розміщуваних сайтів. Найпростіший і дешевий тариф, зазвичай,

носить назву типу «Мінімальний», «Базовий», «Початковий», а найдорожчий – «Максимальний», «Безлімітний», «Професійний».

Втім, слід бути обізнаним в основних опціях тарифних планів, що можуть бути зазначені або про них можна дізнатися через службу технічної підтримки або з відгуків клієнтів даної компанії.

Дисковий простір. Для кожного тарифного плану хостинг-провайдер надає, як правило, обмежений дисковий простір, оскільки окрім текстової інформації, веброзробник може завантажувати на сервер аудіо та відео файли, архіви тощо. На вітчизняному ринку хостинг-провайдери часто надають наступні обсяги: 1 GB; 3 GB; 6 GB; 10 GB, на закордонному, зазвичай, обмежень обсягу немає.

Кількість сайтів для розміщення. Хостинг-провайдер може обмежувати кількість вебсайтів, які можуть бути розміщені на хостингу.

Кількість поштових акаунтів. Надання одного або кількох поштових акаунтів типу ваше_і'мя@ваш_домен. Якщо ще надається можливість отримувати/відправляти пошту через POP3/SMTP, то для роботи з поштою можна користуватися поштовими клієнтами (Outlook, The Bat тощо). Багато хостинг-провайдерів надають можливість фільтрації спаму, переадресацію пошти до іншої поштової скриньки.

Пропускна здатність каналу зв'язку. Це кількість даних, що передаються або приймаються за одиницю часу. Для хостингу ця характеристика є важливою, особливо якщо передбачається вивантажування багатьох файлів із сайту.

Смуга пропускання, по якій проходить зовнішній трафік, має бути достатньою, щоб міг пройти весь трафік, але надмірні характеристики можуть призвести до зайвих грошових витрат.

Необмежений трафік. Провайдер не обмежує обсяг інформації, яку можна завантажити з сайту щомісяця і не стягує додаткову плату за перевищення ліміту трафіку. В тарифних планах з обмеженнями хостери вимагають додаткової плати за трафік, що перевищує обмеження, в деяких компаніях сайт від'єднують до поповнення оплаченого періоду, зазвичай, до кінця місяця або доки клієнт не оплатить ці перевищення. Як правило обмеження на трафік

встановлюють провайдери, що мають проблеми із завантаженням серверів і каналів і намагаються таким чином обмежити трафік.

Швидкість каналу. Від цього параметра залежить швидкість завантаження сторінок сайту. Якщо проєкт розраховано виключно на місцеву аудиторію, тоді достатньо швидкості каналу в 10 Мбіт/сек. у місцевого хостинг-провайдера. Але якщо проєкт розраховано на широку аудиторію країни чи світу, тоді мова йде як мінімум про 100 Мбіт/сек.

Завантаженість каналів і серверів провайдера безпосередньо впливає на доступність сторінок в Інтернет. Якщо канали хостинг-провайдера будуть завантажені на 100%, а ресурси сервера на 90%, тоді виникають проблеми з доступністю, зависанням, повільною швидкістю завантаження сайту.

Uptime – показує час безперебійної роботи сервера у відсотках від загального часу, без збоїв або інших неприємностей, пов'язаних з припиненням роботи з технічних проблем. Найвище значення Uptime, зазвичай, становить 99,9%. 100% Uptime – це ідеал, який не можна досягти, оскільки відбувається багато процесів і критичні оновлення потребують перевантаження серверів. Це неминуче і до того ж, можуть виникнути різні форс-мажорні обставини, які вплинуть на цей показник.

Uptime сервера є важливим показником його надійності, оскільки від нього залежить доступність сайту та відвідуваність його користувачами і ботами. Роботи пошукових систем регулярно перевіряють доступність сайтів і сайт, який з вини хостингу не працював, викидається з пошукової бази роботів. Якщо Uptime становить 99% – сайт може повноцінно функціонувати, але це не гарантується при Uptime нижче за 98%.

Гарантіями хостинг-провайдерів є безперебійна робота серверу та повернення грошей як компенсацію, у разі зменшення зазначеного Uptime.

Показники для вибору хостингової компанії

Забезпечення надійної та безперебійної роботи

Моніторинг. Цілодобове спостереження за роботою сервера і негайне відновлення працездатності у разі виникнення проблем. Сайт має бути доступним 24 години на добу.

Технічна підтримка. Якісна, за можливістю цілодобова технічна підтримка, від якої залежить швидкість вирішення різного роду питань, проблем, вимог, пов'язаних з сайтом.

Резервне копіювання інформації. Збереження інформації у вигляді щоденних копій всієї інформації клієнтів на окремій машині. Поламка комп'ютерної техніки може статися будь-коли і ризик втрати цінної інформації є великим.

Потужна система статистики. Більшість хостинг-провайдерів надають доступ до лог-файлів, а також програми аналізу статистики відвідувань.

Відношення до клієнтів

Все включено. За такою схемою працює хостинг-провайдер, який надає все, що потрібно для повноцінної роботи сайту. Якщо для роботи потрібні підтримка основних мов програмування, під'єднання баз даних, SSI, FTP-доступ, поштові скриньки, редиректи тощо, то це повинно бути в арсеналі провайдера. Хороший провайдер завжди готовий встановити додаткові засоби для роботи, наприклад, модулі мов або сервіси.

Прийнятна ціна. Ціни мають відбивати реальну вартість послуг, враховуючи витрати, вигоди, інтереси клієнтів тощо.

Порядність і надійність. Хостинг-провайдер на своєму сайті має розмістити договір про надання послуг, в якому чітко прописано відповідальність самого провайдера і зазначено пункт про повернення грошей – за ненадані або неякісно надані послуги.

Обмежений термін дії підписаного договору. Довгостроковий договір може бути проблемою у разі зміни хостингу.

Відповідальність. Захист інформації клієнтів за допомогою антивірусних програм і програм, що блокують спроби зламу сайтів.

Додаткові бонуси. Для залучення клієнтів надаються кращі умови, знижки, подарунки тощо, наприклад, безкоштовний домен при замовленні хостингу.

Список клієнтів на сайті хостингу. Якщо список клієнтів хостингу має значні розміри, то можна зробити висновки, що даний сервіс користується довірою у клієнтів і має хорошу функціональність.

Відгуки користувачів. Хороший хостинг-провайдер має у своєму активі лише позитивні відгуки своїх клієнтів. Звичайно не варто на 100% довіряти відгукам, що опубліковані на сайті самого провайдера, їх слід пошукати на форумах і соціальних групах.

Конкуруючий сайт. Однією з кращих порад при виборі хостингу є така: «Дізнайся, який хостинг використовує конкуруючий сайт і купи аналогічний або краще». Дізнатися на якому хостингу знаходиться сайт-конкурент можна за допомогою whois-сервісу (Who is – «хто такий?»). Даний сервіс є у всіх без винятку реєстраторів доменів.

11.5. Вибір хостингу відповідно до типу вебресурсу

Сайт-візитка

Сайт-візитка – це найпростіший варіант вебресурсу, який використовується для представлення компанії або особистого бренду. Він складається з кількох сторінок з основною інформацією: контакти, послуги, портфоліо. Для такого сайту достатньо віртуального хостингу на Unix-платформі з 100 МБ дискового простору. Якщо сайт не використовує бази даних та програмування, можна вибрати тариф без підтримки цих технологій, що здешевить обслуговування.

Лендінг (Landing Page)

Лендінг – це односторінковий сайт, створений для рекламної кампанії або просування конкретного продукту чи послуги. Основне завдання – конверсія відвідувачів у покупців або підписників. Оскільки такі сайти зазвичай мають мінімум контенту і не використовують бази даних, їм достатньо простого хостингу з 100-500 МБ пам’яті. Багато хто використовує конструктори сайтів (Tilda, Wix, LPGenerator), що спрощує роботу.

Блог

Блог – це вебресурс для публікації статей, фото та відео. Найчастіше блоги створюють на системах управління контентом (CMS), таких як WordPress чи Joomla, тому хостинг має підтримувати відповідну платформу. Основний контент блогу – це текст, тому обсяг пам’яті може варіюватися від 100 до 500

МБ. Якщо блог міститиме багато мультимедійних матеріалів, варто обрати тарифний план із 1 ГБ і більше.

Корпоративний сайт

Для компанії важливо мати надійний вебресурс, який працюватиме без перебоїв. Тому хостинг для корпоративного сайту повинен мати високий UpTime і стабільну підтримку. Такий сайт часто використовує динамічний контент, каталоги продукції, модулі новин, що вимагає підтримки мов програмування та баз даних. Важливим фактором є допустиме навантаження на сервер, щоб забезпечити стабільну роботу сайту навіть при великій кількості відвідувачів.

Інтернет-магазин

Хостинг для інтернет-магазину вибирається залежно від платформи, на якій працюватиме сайт (наприклад, OpenCart, Magento, WooCommerce). Для стабільної роботи потрібна підтримка новітніх технологій, мов програмування та баз даних. Інтернет-магазин має динамічний контент, тому обов'язковою є щоденна резервна копія (Backup). Для зберігання зображень товарів рекомендується мінімальний дисковий простір від 1 ГБ. Багато хостинг-провайдерів пропонують спеціальні тарифи для e-commerce сайтів.

Портал

Портали – це багатофункціональні сайти з великим обсягом контенту, які відвідує велика аудиторія. Для їхньої роботи зазвичай використовується виділений сервер або хмарний хостинг. Останній є надійним рішенням, оскільки навантаження розподіляється між кількома серверами, що зменшує ризик перевантаження. Оплата здійснюється за використання ресурсів, тому чим більше відвідувачів і складніший функціонал сайту, тим вищою буде вартість хостингу.

Форум

Форум – це вебресурс для спілкування користувачів. Він повинен працювати безперебійно та витримувати велике навантаження. Тому хостинг для форуму має бути відмовостійким та підтримувати стабільне збереження даних. На початковому етапі можна використовувати хороший віртуальний хостинг, але

якщо кількість відвідувачів перевищує 7-8 тисяч на добу, варто розглянути перехід на виділений сервер. Важливим критерієм є якість резервного копіювання, щоб у разі збою можна було швидко відновити роботу сайту.

Медіа-сайт (новинний, онлайн-журнал)

Новинні сайти та онлайн-журнали потребують швидкого оновлення контенту та високої надійності. Оскільки такі сайти містять багато текстів, фото і відео, потрібно вибирати хостинг з дисковим простором від 5 ГБ та високим рівнем захисту. Часто використовується CMS WordPress або спеціалізовані рішення для медіа-проєктів.

Сайт із штучним інтелектом (AI-powered вебресурс)

Вебресурси, що використовують штучний інтелект, стають дедалі популярнішими. Це можуть бути чат-боти, системи рекомендацій, генерація контенту, голосові помічники чи аналітичні сервіси. Такі сайти потребують потужних серверів для обробки даних у реальному часі.

11.6.«Чорні списки» хостерів

В кожній компанії є правила користування, порушення яких може призвести до розірвання договору або серйозних проблем з законом. Щоб не опинитися в «чорному списку» провайдера, слід ознайомитися з основними обмеженнями.

Нелегальний зміст сайту

Інформація на сайті може мати незаконний характер, наприклад, порнографічна колекція фотографій, відео або тексти аналогічного змісту; різні афери; азартні ігри на гроші. Це може спричинити відмову хостинг-провайдера у подальшій співпраці, і акаунт буде закрито. Якщо такий сайт виявиться в поле зору правоохоронних органів, компанія-провайдер буде зобов'язана надати їм особисті дані клієнта. За розміщення заборонених матеріалів відповідальність буде нести саме клієнт хостингу.

Слід заздалегідь дізнатися про те, яка інформація є нелегальною саме на території країни, в якій розміщено хостинг. Наприклад, в Україні на хостингах

заборонено розміщувати сайти, що містять порнографічну тематику, а законодавство Німеччини та Голландії дозволяє.

В Україні, як і в ряді інших країн, під забороною знаходяться торрент-сайти. А на канадських хостингах можливе розміщення adult-контенту, торрент-трекерів, онлайн-кінотеатрів. Нелегальними будуть вважатися тільки сайти, що поширюють спам, проявляють хакерську активність у будь-яких її проявах, застосовують шахрайські схеми з платіжними картами і їх реквізитами.

Спам

Практично кожна хостинг компанія веде сувору політику, яка спрямована на захист від спаму. Неконтрольовану масову розсилку небажаних листів з сайту клієнта буде відстежено, а інформацію про це – передано у відповідні правозахисні організації. З цієї причини хостинг-провайдер буде змушений розпрощатися з клієнтом. Власник сайту не завжди навмисне застосовує спам, часто сайт зламують і використовують у своїх корисних цілях шахраї. В таких випадках можна сміливо подавати апеляцію і намагатися довести, що сайт став випадковою жертвою. Це допоможе реабілітуватися в очах провайдера і продовжити хостинг згідно з договором.

Порушення авторських прав

У розвинених країнах, зокрема в США, закон обмежує використання авторської музики, назв, імен, відео та іншого. Багато хостинг-компанії, не бажаючи проблем із законом, при виявленні на своєму сервері порушення авторських прав, просто відмовлять у співпраці, навіть не чекаючи офіційного рішення від служителів закону. Крім того, використання незаконних копій музики або фільмів має тенденцію накопичуватися, займаючи на сервері величезне місце, що згубно для сервера. Хостинг-провайдери відслідковують подібні речі не лише через порушення законних прав, але і з метою збереження стабільної роботи сервера.

Несанкціоноване завантаження скриптів

Стосується це насамперед клієнтів, чиї сайти знаходяться на звичайному віртуальному хостингу. Несанкціоновані скрипти швидко перевантажують сервер, і при їх виявленні провайдер може відмовити у співпраці. Спочатку

клієнт отримує попередження, після якого слід виправити зазначені порушення, щоб не втратити хостинг.

Хакерство

Найлегший спосіб потрапити до чорного списку хостинг-провайдера – вчиняти незаконні дії, що спрямовані на злам або видалення Інтернет-ресурсів. Якщо буде помічено спробу зламу з метою отримання доступу до серверу хостинг-провайдера, компанія застосує негайних заходів для захисту своїх даних, а також видалення акаунта.

Якщо з клієнтського сайту відбуваються хакерські атаки на сервер, то це вірний шанс потрапити в бан. Компанії-провайдери нещадно борються з такими атаками. Причому розплатою може виявитися не лише потрапляння до «чорного списку», але і більш серйозні наслідки.

Будь-які незаконні дії з боку клієнтського сайту негайно призводять до видалення зі списку клієнтів хостинг-компанії. Тому, в обов'язковому порядку перед тим, як підписувати угоду з провайдером, слід детально вивчити все, що дозволено і що неприпустимо. Від правомірності дій власника безпосередньо залежить подальша доля розвитку його сайту.

11.7.Хмарні технології

Хмарні обчислення (Cloud Computing) – це технологія розподіленої обробки даних, в якій комп'ютерні ресурси і потужності надаються користувачеві як Інтернет-сервіс, тобто робочий майданчик на віддаленому сервері. Наприклад, якщо користувач працює з електронною поштою на сайті-сервісі (наприклад, gmail), який цю пошту дозволяє використовувати чи обробка зображення в браузері через сервіс Picasa, то це є використання хмарного сервісу.

Різниця полягає виключно в методі зберігання і обробки даних. Якщо всі операції відбуваються на комп'ютері користувача (з використанням його потужностей), то це – не «хмара», а якщо процес відбувається на сервері в мережі, то це «хмарні технології» – різні апаратні, програмні засоби, методології та інструменти, що надаються користувачеві, як Інтернет-сервіси, для реалізації своїх цілей, завдань, проєктів.

Терміни «хмарні технології»/«хмарний сервіс», з їх загальноприйнятим графічним представленням, у вигляді «хмарок», тільки плутає користувачів, насправді їх структуру, можна легко зрозуміти, якщо уявити її у вигляді такої піраміди (рис. 11.3).

Основа піраміди – «інфраструктура» – це набір фізичних пристроїв (сервери, тверді диски тощо), над нею надбудовується «платформа» – набір послуг і верхівка – програмне забезпечення, що доступне за запитом користувачів.

Хмарні обчислення – це певний базис-вектор, отриманий у результаті синтезу цілого ряду технологій і підходів.

Хмарні технології – це набір засобів, що виконують обчислення за допомогою віддалених серверів і програм без безпосереднього залучення ресурсів комп'ютера користувача. Можливо, в майбутньому комп'ютери будуть представляти один лише екран з мікро-процесором, а всі обчислення і потужності будуть розташовані і виконуватися віддалено на серверах «хмари» (рис. 11.4).

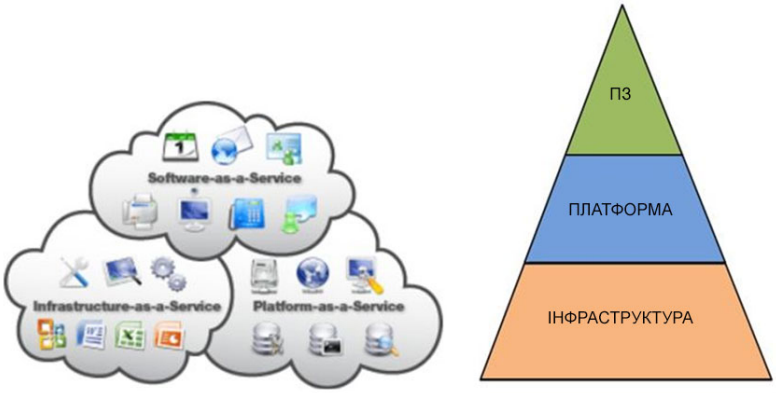


Рис.11.3. Пираміда хмарних сервісів

Рис.11.4. Спроможності хмарних обчислень

Послуги, що надаються хмарними системами

Все, що стосується Cloud computing (далі CC), зазвичай прийнято називати aaS – «as a Service», тобто «як сервіс», або «у вигляді сервісу».

На даний час концепція передбачає надання наступних типів послуг своїм користувачам:

Storage-as-a-Service («зберігання як сервіс»)

Найпростіший з CC-сервісів, що представляє собою дисковий простір на вимогу. Послуга Storage-as-a-Service дає можливість зберігати дані в зовнішньому сховище, в «хмарі». Для користувача воно буде виглядати, як додатковий логічний диск або папка. Сервіс є базовим для інших, оскільки входить до складу практично кожного з них. Прикладом може служити Google Drive та інші схожі сервіси.

Database-as-a-Service («база даних як сервіс»)

Послуга більше для адмінів, бо надає можливість працювати з базами даних, подібно так, як би СУБД було встановлено на локальному ресурсі. У цьому випадку значно легше розділяти проєкти між різними виконавцями та

заощадити на комп'ютерному обладнанні та ліцензіях, необхідних для грамотного використання СУБД у великій чи середньої організації.

Information-as-a-Service («інформація як сервіс»)

Дає можливість віддалено використовувати будь-які види інформації, яка може змінюватися щохвилини або навіть щомиті.

Process-as-a-Service («управління процесом як сервіс»)

Віддалений ресурс, який може зв'язати воєдино кілька ресурсів (таких як послуги або дані, що містяться в межах однієї «хмари» або інших доступних «хмар»), для створення єдиного бізнес-процесу.

Application-as-a-Service («додаток як сервіс»)

Також називається, Software-as-a-Service («ПЗ як сервіс»). Позиціонується як «програмне забезпечення на вимогу», яке розгорнуто на віддалених серверах і кожен користувач може отримувати до нього доступ за допомогою Інтернету, причому всі питання оновлення та ліцензій на дане забезпечення регулюється постачальником даної послуги. Оплата, в даному випадку, відбувається за фактичне використання останнього. Як приклад можна навести Google Docs, Google Calendar і т. п. онлайн-програми.

Platform-as-a-Service («платформа як сервіс»)

Користувачеві надається комп'ютерна платформа із встановленою операційною системою і певним програмним забезпеченням.

Integration-as-a-Service («інтеграція як сервіс»)

Це можливість отримувати з «хмари» повний інтеграційний пакет, включаючи програмні інтерфейси між додатками і управління їх алгоритмами. Сюди входять відомі послуги та функції пакетів централізації, оптимізації та інтеграції корпоративних додатків (EAI), але вони надаються як «хмарний» сервіс.

Security-as-a-Service («безпека як сервіс»)

Даний вид послуги надає можливість користувачам швидко розгортати продукти, що вимагають безпечне використання вебтехнологій, електронного листування, локальної мережі. Користувачі даного сервісу мають змогу економити на розгортанні та підтримці своєї власної системи безпеки.

Management / Governace-as-a-Service («адміністрування та управління як сервіс»)

Дає можливість керувати і задавати параметри роботи одного або багатьох «хмарних» сервісів. Це в основному такі параметри, як топологія, використання ресурсів, віртуалізація.

Infrastructure-as-a-Service («інфраструктура як сервіс»)

Користувачеві надається комп'ютерна інфраструктура, зазвичай віртуальні платформи (комп'ютери), пов'язані в мережі, які він самостійно налаштовує під власні цілі.

Testing-as-a-Service («тестування як сервіс»)

Дає можливість тестування локальних або «хмарних» систем із використанням тестового ПЗ з «хмари» (при цьому жодного устаткування або забезпечення на підприємстві, не потрібно).

Для наочності, узагальнимо сервіси архітектури «хмара», в одну схему на де наведено класифікацію сервісів, за типом послуг (рис. 11.5):

Рис. 11.5. Сервіси хмарної архітектури

Категорії «хмар» (за формою власності)

Публічна хмара – це IT-інфраструктура, яка використовується одночасно багатьма компаніями і сервісами. Користувачі не мають можливості управляти і обслуговувати дану «хмару», відповідальність з цих питань покладено на власника ресурсу. Абонентом пропонованих сервісів може стати будь-яка компанія та індивідуальний користувач.

Прикладами можуть служити онлайн-сервіси: Amazon EC2, Google Apps / Docs, Microsoft Office Web.

Приватна хмара – це безпечна IT-інфраструктура, що контролюється і експлуатується в інтересах однієї організації. Організація може керувати приватною «хмарою» самостійно або доручити це завдання зовнішньому підряднику. Інфраструктура може розміщуватися або в приміщеннях замовника, або у зовнішнього оператора (або частково у замовника і частково у оператора).

Гібридна хмара – це IT-інфраструктура, що використовує найкращі якості публічної і приватної хмари при вирішенні поставленого завдання. Часто такий тип застосовується, коли організація має сезонні періоди активності, тобто, як тільки внутрішня IT-інфраструктура не справляється з поточними завданнями, частина потужностей перекидається на публічну «хмару» (наприклад, великі обсяги статистичної інформації), а також для надання доступу користувачам до ресурсів підприємства через публічну «хмару».

Можливості хмарних обчислень

Доступ до особистої інформації з будь-якого комп'ютера, що підключений до Інтернету.

Можливість працювати з інформацією з різних пристроїв (ПК, планшети, телефони і т. п.).

Незалежність від операційної системи комп’ютера користувача – вебсервіси працюють в браузері будь-яких ОС.

Одну інформацію можна переглядати і редагувати одночасно з різних пристроїв.

Багато платних програм є безкоштовними (або дешевшими) вебдодатками.

Запобігання втрати інформації, вона зберігається в хмарних сховищах.

Завжди актуальна і оновлена інформація.

Використання останніх версій програм і оновлень.

Можливість об’єднання інформації з іншими користувачами

Легко ділитися інформацією з людьми в будь-якій точці земної кулі.

Недоліки:

Необхідність постійного з'єднання. Для отримання доступу до послуг «хмари» необхідно постійне з'єднання з Інтернетом.

Програмне забезпечення та його «кастомизация». Є обмеження по ПЗ, яке можна розгортати на «хмарах» і надавати його користувачеві. Користувач має обмеження у використовуваному забезпеченні та іноді не має можливості налаштувати його під свої власні цілі.

Конфіденційність. Конфіденційність даних, що зберігаються в публічних «хмарах», в даний час, викликає багато суперечок, але в більшості випадків експерти сходяться в тому, що не рекомендується зберігати найбільш цінні для компанії документи на публічній «хмарі», оскільки в даний час немає технології, яка б гарантувала 100% конфіденційність даних.

Безпека. «Хмара» саме по собі є достатньо надійною системою, однак при проникненні в неї зловмисник отримує доступ до величезного сховища даних. Ще один мінус, – це використання систем віртуалізації, в яких, як гипервізора, використовуються ядра стандартних ОС (наприклад Windows), що дозволяє використовувати віруси та вразливості системи.

Дороге обладнання. Для побудови власної хмари необхідно виділити значні матеріальні ресурси, що не вигідно щойно створеним і малим компаніям.

Подальша монетизація ресурсу. Цілком можливо, що компанії надалі вирішать брати плату з користувачів за надані послуги.

ПИТАННЯ ДО РОЗДІЛІВ

1. Концепція WEB

1. Що таке Інтернет? Які основні принципи його побудови?
2. Опишіть відмінності між Інтернетом та WWW.
3. Які основні терміни використовуються у веброзробці?
4. Що таке клієнт у комп'ютерній мережі?
5. Яка роль серверів у мережі Інтернет?
6. Поясніть принцип взаємодії «клієнт–сервер».
7. Які види провайдерів існують? Які їх функції?
8. Що таке хостинг-провайдер? Які послуги він надає?
9. Як здійснюється передача даних між клієнтом і сервером?
10. Що таке IP-адреса? Які типи IP-адрес існують?
11. Чим відрізняється IPv4 від IPv6?
12. Яка роль WWW у сучасному Інтернеті?
13. Що таке вебсайт і з яких елементів він складається?
14. Які типи послуг надаються через Інтернет? Наведіть приклади.
15. Що таке гіперпосилання? Яка їхня функція у вебдокументах?
16. Поясніть, що таке дзеркало сайту.
17. Що таке служба DNS? Які її основні функції?
18. Яка структура доменного імені? Що таке домени 1, 2, 3 рівнів?
19. Що таке делегування у DNS? Яка його роль?
20. Чим відрізняються URI, URL, URN? Наведіть приклади.
21. Поясніть поняття User-Agent та його роль у запитах.
22. Що таке spoofing User-Agent, які є ризики?
23. Які основні протоколи використовуються для роботи в Інтернеті?
24. Поясніть принцип роботи протоколу HTTP.
25. Які ще протоколи, окрім HTTP, використовуються у веброзробці? Наведіть приклади та їх призначення.

2. Вебвидавництво

1. Що таке вебвидавництво? Які його основні етапи?
2. З чого починається створення вебсайту?
3. Які види хостингу існують? Їх переваги і недоліки.
4. Чим відрізняються статичний і динамічний сайт?
5. Що таке CMS і які їх основні функції?
6. Які способи публікації сайту у мережі Інтернет існують?

7. Які критерії вибору хостингу ви знаєте?
8. Що таке доменне ім’я та як його зареєструвати?
9. Які технічні вимоги до хостингу для сучасного сайту?
10. Як організувати резервне копіювання сайту?
11. Поясніть процедуру перенесення сайту на інший хостинг.
12. Що таке “паркування” домену?
13. Як забезпечити безперервність роботи сайту при зміні хостингу?
14. Що таке SSL-сертифікат, і чому він важливий для вебсайту?
15. Як оцінити швидкість та надійність роботи хостингу?
16. Поясніть поняття “реселінг” у сфері хостингу.
17. Як протестувати роботу сайту після його публікації?
18. Які юридичні питання виникають при публікації сайту?
19. Які інструменти автоматизації публікації сайтів ви знаєте?
20. Які вимоги щодо безпеки при вебвидавництві?
21. Які метрики використовують для оцінки ефективності вебсайту?
22. Поясніть, що таке CDN і для чого він використовується.
23. Що таке “landing page” і як вона створюється?
24. Як реалізується інтеграція сайту з соціальними мережами?
25. Які є шляхи просування і реклами вебсайтів?

3. Цикл розробки вебсайту

1. Назвіть основні етапи життєвого циклу розробки сайту.
2. Що таке технічне завдання (ТЗ) для створення сайту?
3. Яку роль відіграє аналіз цільової аудиторії при розробці сайту?
4. Що таке бриф і як його використовують у проектуванні сайту?
5. Які дані повинні міститися у брифі на розробку вебсайту?
6. Опишіть процес проектування структури сайту.
7. Які принципи формування навігації на сайті?
8. Що таке прототипування та його місце у розробці?
9. Як визначити функціональні та нефункціональні вимоги до сайту?
10. Які підходи існують до розробки макета дизайну сайту?
11. Поясніть етапи верстки сайту (HTML, CSS).
12. Які системи керування контентом (CMS) найпоширеніші сьогодні?
13. Які існують підходи до наповнення сайту контентом?
14. Які види тестування сайтів застосовуються?
15. Що таке “реліз” сайту і як його організувати?
16. Які є методи оцінки якості розробленого сайту?
17. Чому важливо враховувати SEO на етапі розробки?

18. Які існують інструменти командної роботи над сайтом?
19. Що таке “редизайн” і коли він необхідний?
20. Як організувати зворотний зв’язок із користувачами сайту?
21. Які ключові помилки зустрічаються при розробці вебсайтів?
22. Яка роль документації у циклі розробки?
23. Як визначити вимоги до масштабованості сайту?
24. Що таке “backlog” у проекті створення сайту?
25. Як організовується підтримка і супровід вебсайту?

4. Програмне забезпечення розробника

1. Які текстові редактори використовують веброзробники?
2. Що таке IDE та у чому їх переваги над простими редакторами?
3. Які можливості дають WYSIWYG-редактори?
4. Які недоліки мають WYSIWYG-редактори?
5. Які основні функції браузера для розробника?
6. Які інструменти розробника є у сучасних браузерах?
7. Що таке система контролю версій і які її типи?
8. У чому різниця між централізованими і розподіленими системами контролю версій?
9. Що таке Git і як він використовується в розробці?
10. Які основні команди Git потрібно знати кожному розробнику?
11. Як організована робота з гілками (branch) у Git?
12. Що таке pull request і merge conflict у Git?
13. Як забезпечити спільну роботу кількох розробників над проектом?
14. Які сервіси хостингу репозиторіїв найбільш популярні?
15. Для чого використовуються локальні сервери (наприклад, XAMPP, WAMP)?
16. Які є інструменти для роботи з базами даних?
17. Як налаштовується інтеграція редактора з системою контролю версій?
18. Що таке CI/CD та для чого його використовують?
19. Які засоби автоматизації тестування використовуються у веброзробці?
20. Як організовується автоматичне розгортання вебсайтів?
21. Які є засоби для моніторингу продуктивності сайту?
22. Що таке баг-трекер і яку роль він відіграє у розробці?
23. Як працюють інструменти профілювання коду?
24. Які є засоби для колективної розробки документації?
25. Чим відрізняється термінальна (CLI) робота з інструментами від графічної?

5. Основи HTML

1. Що таке HTML і яка його роль у веброзробці?
2. Яка структура стандартного HTML-документа?
3. Що таке DOCTYPE і навіщо він потрібен?
4. Які основні елементи містить секція <head>?
5. Які функції виконує тег <body>?
6. Що таке атрибут у HTML? Наведіть приклади.
7. Які види тегів існують у HTML?
8. Що таке семантична верстка?
9. Яка роль елементів <header>, <footer>, <nav>, <main>?
10. Що таке DOM-модель HTML-документа?
11. Які основні теги для форматування тексту?
12. Як створити заголовок різного рівня у HTML?
13. Які правила побудови гіперпосилань?
14. Як вставити зображення на вебсторінку?
15. Що таке абсолютна та відносна адресація у HTML?
16. Як створюються списки (маркеровані, нумеровані, визначень)?
17. Які основні правила створення HTML-форм?
18. Які теги використовуються для створення таблиць?
19. Як працюють блокові та рядкові елементи HTML?
20. Що таке метатеги та для чого вони потрібні?
21. Як підключати зовнішні ресурси до HTML-документа?
22. Що таке "alt"-атрибут у зображеннях і для чого він використовується?
23. Які нові можливості з'явилися у HTML5?
24. Що таке адаптивна розмітка сторінки у HTML?
25. Які засоби валідації HTML-коду ви знаєте?

6. Основи CSS

1. Що таке CSS та яка його основна роль?
2. Які способи підключення CSS існують?
3. Що таке селектор у CSS? Які є типи селекторів?
4. Поясніть поняття “каскадування” у CSS.
5. Що таке специфічність селектора?
6. Як застосовуються класи та ідентифікатори у CSS?
7. Що таке інлайнові, внутрішні та зовнішні стилі?
8. Як задати фон сторінки або елемента?
9. Які властивості використовуються для оформлення тексту?

10. Як працюють псевдокласи у CSS?
11. Що таке "float" та як його використовувати?
12. Які способи позиціювання елементів на сторінці ви знаєте?
13. Що таке Flexbox і які його переваги?
14. Поясніть роботу CSS Grid.
15. Що таке адаптивний (responsive) дизайн у CSS?
16. Які медіа-запити (media queries) використовують для адаптації?
17. Як організувати анімацію у CSS?
18. Що таке transition і transform у CSS?
19. Які засоби оптимізації CSS-коду ви знаєте?
20. Які існують CSS-фреймворки та для чого вони потрібні?
21. Як організувати спадкування властивостей у CSS?
22. Що таке перемикачі (toggle) на CSS?
23. Як працюють шрифти та їх підключення у CSS?
24. Що таке змінні у CSS (CSS Variables)?
25. Які помилки допускають новачки при верстці з CSS?

7. Дизайн

1. Що таке стандарт HTML5 і як він вплинув на сучасний вебдизайн?
2. Які існують типи дизайну сайтів?
3. Що таке чуйний (responsive) та адаптивний (adaptive) дизайн? У чому різниця?
4. Що таке модульна сітка і для чого вона використовується?
5. Які принципи вибору колірної схеми для сайту?
6. Як створюється сучасний макет сайту?
7. Що таке UX/UI дизайн і які їх завдання?
8. Яка роль іконок та піктограм у вебдизайні?
9. Як правильно підібрати шрифти для сайту?
10. Що таке вебанімація і для чого вона застосовується?
11. Які типи графічних форматів використовуються для веб?
12. Поясніть різницю між растровою і векторною графікою.
13. Що таке SVG, Canvas, WebP, GIF? Які їх переваги і недоліки?
14. Як організовується навігаційне меню?
15. Які особливості має “landing page” дизайн?
16. Як створити call-to-action елементи на сайті?
17. Що таке accessibility (доступність) у вебдизайні?
18. Як забезпечити зручність користування сайтом для людей з інвалідністю?

19. Які основні тренди сучасного вебдизайну?
20. Що таке дизайн-система і для чого вона потрібна?
21. Як перевірити дизайн сайту на відповідність стандартам?
22. Які онлайн-інструменти використовуються для прототипування?
23. Що таке A/B тестування у вебдизайні?
24. Як оцінити ефективність дизайну сайту?
25. Які помилки часто допускають початківці у вебдизайні?

8. Типографіка

1. Що таке типографіка і чому вона важлива для вебсайту?
2. Які типи шрифтів існують у вебдизайні?
3. Що таке гарнітура та кегль шрифту?
4. Які формати шрифтів підтримуються браузерами?
5. Що таке кодування тексту і які його види використовуються у веброзробці?
6. Що таке Unicode, UTF-8, ASCII?
7. Які особливості верстки тексту для багатомовних сайтів?
8. Як забезпечити коректне відображення кирилиці на сайті?
9. Які вимоги до інтерліньяжу і кернінгу у вебтипографіці?
10. Що таке вебшрифти і як їх підключати?
11. Які є популярні сервіси для підключення шрифтів?
12. Що таке "fallback" шрифт і для чого він потрібен?
13. Як вибирати оптимальний розмір шрифту для різних пристроїв?
14. Які поради для читабельності тексту на сайті?
15. Що таке псевдографіка і де її можна застосувати?
16. Як відобразити спеціальні символи у HTML?
17. Що таке BOM у кодуванні тексту?
18. Як організувати багаторівневу структуру заголовків у тексті?
19. Які є обмеження щодо кількості шрифтів на одній сторінці?
20. Як впливає типографіка на загальне враження від сайту?
21. Що таке “легкість читання” тексту і як її забезпечити?
22. Які є типові помилки у вебтипографіці?
23. Які існують CSS-властивості для оформлення тексту?
24. Що таке line-height та letter-spacing у CSS?
25. Які засоби забезпечення адаптивної типографіки існують?

9. Вебавторство (верстка)

1. Що таке верстка у веброзробці?

2. Чим відрізняється таблична та блокова верстка?
3. Які переваги має блокова верстка?
4. Що таке “семантична верстка” і чому вона важлива?
5. Які теги використовуються для створення таблиць?
6. Як здійснюється перевірка коректності HTML/CSS-коду?
7. Які основні вимоги до якості верстки?
8. Як організувати структуру сайту за допомогою блоків?
9. Що таке модель блоків у CSS?
10. Як використовувати властивості width, height, padding, margin, border?
11. Які принципи побудови адаптивної верстки?
12. Що таке абсолютне та відносне позиціювання елементів?
13. Як працює властивість z-index?
14. Що таке “плаваючі” (float) елементи і коли їх застосовують?
15. Які сучасні підходи до верстки (Flexbox, Grid)?
16. Що таке “схлопування” margin-ів?
17. Як реалізувати багаторівневу навігацію за допомогою верстки?
18. Які засоби оптимізації верстки існують?
19. Як уникати дублювання коду при верстці?
20. Що таке “респонсивна” верстка?
21. Як працюють фреймворки для верстки (Bootstrap, Foundation)?
22. Які інструменти використовують для тестування верстки на різних пристроях?
23. Як забезпечити доступність (accessibility) верстки?
24. Які є типові помилки при верстці вебсторінок?
25. Як організувати інтеграцію верстки з JavaScript-функціоналом?

10. Підготовка завантаження та опублікування

1. Що таке толерантний код і чому він важливий для веброзробки?
2. Як перевірити посилання на сайті перед публікацією?
3. Які вимоги до оптимізації зображень для сайту?
4. Які етапи проходить сайт перед завантаженням на хостинг?
5. Як перевірити роботу сайту у різних браузерах?
6. Що таке favicon і як його додати на сайт?
7. Як організувати резервне копіювання при публікації?
8. Які інструменти використовуються для розгортання сайтів?
9. Як забезпечити відкат (rollback) до попередньої версії сайту?
10. Що таке “гаряче” оновлення сайту?
11. Як перевірити швидкість завантаження сторінок?

12. Що таке кешування і як воно використовується на сайті?
13. Які основні причини помилок при публікації сайту?
14. Як організувати моніторинг працездатності сайту?
15. Які існують способи відстеження збоїв та повідомлення про них?
16. Як працюють засоби аналітики вебсайтів (Google Analytics)?
17. Що таке sitemap.xml і для чого він потрібен?
18. Які існують вимоги до robots.txt?
19. Як перевірити індексацію сайту пошуковими системами?
20. Як організовується “перенаправлення” сторінок (redirects)?
21. Що таке SSL/HTTPS і як його підключити до сайту?
22. Які питання безпеки слід враховувати при публікації сайту?
23. Як уникнути дублювання контенту при завантаженні сайту?
24. Які основні дії після запуску сайту в Інтернеті?
25. Які методи використовують для SEO-оптимізації перед запуском сайту?

11. Хостинг

1. Які типи хостингу існують? Дайте коротку характеристику кожному.
2. Що таке віртуальний хостинг і які його переваги та недоліки?
3. Що таке виділений сервер (dedicated server)?
4. Поясніть принцип роботи VPS/VDS хостингу.
5. Що таке хмарний (cloud) хостинг?
6. Що таке колокація і коли її використовують?
7. Які основні параметри потрібно враховувати при виборі хостингу?
8. Як обрати оптимальний тарифний план?
9. Які додаткові опції пропонують хостинг-провайдери?
10. Які критерії безпеки важливі при виборі хостингу?
11. Що таке “абузостійкий” хостинг?
12. Що таке “чорний список” хостерів і за що туди потрапляють?
13. Як організована підтримка клієнтів у хостинг-компаніях?
14. Які особливості хостингу для різних типів сайтів (блог, магазин, форум)?
15. Як перенести сайт на інший хостинг?
16. Що таке паркування домену?
17. Що таке SSL і як його підключити на хостингу?
18. Які є способи оплати хостингу?
19. Як організовується масштабування ресурсів на хостингу?
20. Які ризики пов’язані з дешевим хостингом?

21. Що таке реселінг хостингу?
22. Які хостинги популярні в Україні та світі?
23. Які особливості мають хмарні технології для сайтів?
24. Які є сервіси для перевірки якості хостингу?
25. Як впливає вибір хостингу на SEO сайту?

12. Юридичні аспекти та захист авторських прав

1. Які основні юридичні аспекти слід враховувати при створенні сайту?
2. Що таке авторське право у веброзробці та які його основні положення?
3. Які закони регулюють вміст сайтів в Україні?
4. Що таке доменне ім’я з точки зору права?
5. Як визначити власника доменного імені?
6. Що таке “договір-оферта” для інтернет-сайтів?
7. Що таке конфіденційність та політика приватності на сайті?
8. Які вимоги до зберігання персональних даних користувачів?
9. Що таке GDPR і чи застосовується він до українських сайтів?
10. Як уникнути порушення авторських прав при використанні зображень?
11. Що таке Creative Commons та інші відкриті ліцензії?
12. Як оформити права на контент, створений для сайту?
13. Що таке порушення авторських прав у сфері веброзробки?
14. Які наслідки розміщення нелегального контенту на сайті?
15. Які санкції можливі за порушення прав інтелектуальної власності?
16. Які є механізми вирішення спорів щодо авторських прав у мережі?
17. Як відбувається блокування сайтів за рішенням суду?
18. Що таке доменний спір і як його вирішують?
19. Що таке плагіат і як його виявляють у вебсередовищі?
20. Як захистити власний сайт від копіювання контенту?
21. Які правила реклами на вебсайтах регулюються законом?
22. Що таке відповідальність за коментарі та контент користувачів на сайті?
23. Які вимоги до розміщення файлів cookie на сайті?
24. Що таке угода користувача (Terms of Service)?
25. Які є поради щодо юридичної безпеки сайту для власника та користувачів?

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Microsoft. Visual Studio Code – Code Editing. Redefined [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com – Дата звернення: 12 лютого 2025 р.
2. Outlines.css. Google Docs [Електронний ресурс]. – Режим доступу: https://drive.google.com/file/d/1anDiqoAPDbuvpMrqXo7lDj2ZmUdogcdZ/view?usp=sharing – Дата звернення: 12 лютого 2025 р.
3. Що таке хмарний хостинг (Cloud VPS) | VPS.ua. Купить VPS хостинг в Україні. VDS і віртуальні сервери VPS.ua [Електронний ресурс]. – Режим доступу: https://vps.ua/ukr/info/what-is-cloud-hosting/ – Дата звернення: 14 лютого 2025 р.
4. Using Git – GitHub Docs [Електронний ресурс]. – Режим доступу: https://docs.github.com/en/get-started/using-git – Дата звернення: 14 лютого 2025 р.
5. Які бувають рівні провайдерів і який краще обрати бізнесу [Електронний ресурс] // GigaTrans – Інтернет для Вашого офісу. – Режим доступу: https://gigatrans.ua/ua/news/kakie-buvayut-urovni-provayderov-i-kakoy-luchshe-vubirat-biznesu – Дата звернення: 14 лютого 2025 р.
6. Про охорону прав на знаки для товарів і послуг : Закон України від 15.12.1993 № 3689-12 [Електронний ресурс]. – Офіційний вебпортал парламенту України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/3689-12#Text – Дата звернення: 14 лютого 2025 р.
7. UANIC – Український Мережевий Інформаційний Центр [Електронний ресурс]. – Режим доступу: http://uanic.net/ – Дата звернення: 14 лютого 2025 р.
8. Web Templates | HTML5 Website Templates | Web Graphics [Електронний ресурс] // TemplateMonster. – Режим доступу: http://www.templatemonster.com – Дата звернення: 12 лютого 2025 р.
9. W3C Public CVS Repository [Електронний ресурс]. – Режим доступу: http://dev.w3.org/geo/api/spec-source.html – Дата звернення: 12 лютого 2025 р.

10. FLIF – Free Lossless Image Format [Електронний ресурс]. – Режим доступу: http://flif.info/ – Дата звернення: 14 лютого 2025 р.

11. An image format for the Web | WebP [Електронний ресурс] // Google for Developers. – Режим доступу: https://developers.google.com/speed/webp – Дата звернення: 14 лютого 2025 р.

12. GIF Graphics Interchange Format, Version 89a [Електронний ресурс] // Home | Library of Congress. – Режим доступу: https://www.loc.gov/preservation/digital/formats/fdd/fdd000133.shtml – Дата звернення: 14 лютого 2025 р.

13. Find Icons with the Perfect Look & Feel [Електронний ресурс] // Font Awesome. – Режим доступу: http://fontawesome.io/icons – Дата звернення: 14 лютого 2025 р.

14. Інформатика. Комп'ютерна техніка. Комп'ютерні технології : підручник / за ред. В. А. Баженова, Г. А. Шинкаренка. – Київ, 2023. – 496 с.

15. Microsoft Typography documentation – Typography [Електронний ресурс] // Microsoft Learn. – Режим доступу: https://learn.microsoft.com/en-us/typography/ – Дата звернення: 14 лютого 2025 р.

16. Create your own @font-face kits » font squirrel [Електронний ресурс] // Font Squirrel. – Режим доступу: http://www.fontsquirrel.com/tools/webfont-generator – Дата звернення: 14 лютого 2025 р.

17. Browse fonts – Google Fonts [Електронний ресурс]. – Режим доступу: https://fonts.google.com/ – Дата звернення: 14 лютого 2025 р.

18. Sublime Text 3 – Sublime Text [Електронний ресурс]. – Режим доступу: http://www.sublimetext.com/3 – Дата звернення: 14 лютого 2025 р.

19. Sunsetting Atom [Електронний ресурс] // The GitHub Blog. – Режим доступу: https://atom.io – Дата звернення: 14 лютого 2025 р.

20. Основи веб та вебдизайн, програмування на боці клієнта : лабораторний практикум з навчальної дисципліни "Вебтехнології та вебдизайн" для студентів напряму підготовки 6.050101 "Комп'ютерні науки" / В. В. Огурцов, Д. В. Гриньов, О. В. Щербаков. – Харків : ХНЕУ ім. С. Кузнеця, 2015. – 208 с.

21. С. В. Баран. Основи web-програмування : навчальний посібник. – Кривий Ріг : Державний університет економіки і технологій, 2023. – 316 с.

22. A modern, open source code editor that understands web design [Електронний ресурс] // Brackets. – Режим доступу: http://brackets.io – Дата звернення: 14 лютого 2025 р.

23. Вебтехнології та вебдизайн : навчальний посібник / О. Г. Трофименко, О. Б. Козін, О. В. Задерейко, О. Є. Плачінда. – Одеса : Фенікс, 2019. – 284 с.

24. The W3C Markup Validation Service [Електронний ресурс]. – Режим доступу: http://validator.w3.org – Дата звернення: 14 лютого 2025 р.

25. Strict-Transport-Security – HTTP [Електронний ресурс] // MDN Web Docs. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security – Дата звернення: 14 лютого 2025 р.

26. Why and how to secure your website with the HTTPS protocol [Електронний ресурс] // Google Search Central Blog | Google for Developers. – Режим доступу: https://developers.google.com/search/blog/2018/12/why-how-to-secure-your-website-https – Дата звернення: 14 лютого 2025 р.

Навчальне видання

ЗАВОЛОДЬКО Ганна Едвардівна
КАСІЛОВ Олег Вікторович
БРОНІН Сергій Вадимович

ОСНОВИ ВЕБРОЗРОБКИ

Навчально-методичний посібник
для студентів – бакалаврів та магістрів спеціальності
122 «Комп'ютерні науки», 123 «Комп'ютерна інженерія»
денної та заочної форм навчання

Відповідальний за випуск проф. Пустовойтов П .Е.

Роботу до видання рекомендувала доц. Крилова В. А.

Верстка: доц. Заволодько Г. Е.

В авторській редакції

План 2025 р., поз. 66

Гарнітура Aptos. Ум. друк. арк. 11,2

Видавничий центр НТУ «ХПІ».

Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.

61002, м. Харків, вул. Кирпичова, 2.

Електронне видання


contact
blog
web server
home page
internet
browser
hyperlink
domain
web
development
layout
HTML
URL
email
CSS
HTTP
PHP
F3
F4