Босько В.В., Константинова Л.В., Марченко К.М., Улічев О.С.

WEB-ПРОГРАМУВАННЯ

ЧАСТИНА 1 (FRONTEND)

НАВЧАЛЬНИЙ ПОСІБНИК

МІНІСТЕРСТО ОСВІТИ І НАУКИ УКРАЇНИ
ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Босько В.В., Константинова Л.В., Марченко К.М., Улічев О.С.

WEB-ПРОГРАМУВАННЯ

ЧАСТИНА 1 (FRONTEND)

Навчальний посібник

Кропивницький
2022

УДК 004.42
ББК 32.973.4
W 37

Рекомендовано вченою радою Центральноукраїнського національного технічного університету, протокол №6 від 24 січня 2022 року

Рецензенти:

Мелешко Є. В. доктор технічних наук, професор, доцент кафедри кібербезпеки та програмного забезпечення Центральноукраїнського національного технічного університету;

Павленко М. А. доктор технічних наук, професор, начальник кафедри математичного та програмного забезпечення АСУ Харківського університету Повітряних Сил ім. Івана Кожедуба.

Босько В.В., Константинова Л.В., Марченко К.М., Улічев О.С.

W 37 Web-програмування. Частина 1 (frontend) : навч. посіб. – Кропивницький: ЦНТУ, 2022. – 208 с.

Навчальний посібник представляє теоретичний та практичний матеріал про інструментальні засоби розробки WEB додатків. Розглянуто основні інструменти для розробки динамічних додатків: скриптову мову розмітки гіпертексту HTML5, мову стилю сторінок CSS та клієнтський фреймворк Bootstrap для розробок зовнішнього вигляду та інтерактивності веб-сторінок. Також розглянуто засоби мови програмування JavaScript для створення сценаріїв вебсторінок.

Призначений для студентів, що навчаються за спеціальностями «Комп'ютерна інженерія», «Комп'ютерні науки» та «Кібербезпека». Для полегшення засвоєння наведеного матеріалу надаються ілюстрації та приклади.

УДК 004.42
ББК 32.973.4

ЗМІСТ

ВСТУП

З розвитком інформаційних технологій сучасна людина не уявляє життя без Інтернету. Кожен день ми черпаємо інформацію з всесвітньої мережі, користуємося електронною поштою, виконуємо пошук необхідної інформації, дізнаємось про новини та спілкуємося в соціальних мережах з друзями. Велика заслуга у цьому веб-програмістів.

Веб-розробка - це процес створення вебсайта або вебдодатку. Термін включає розробку додатків електронної комерції, веб-дизайн, програмування для веб на стороні клієнта й серверу, а також конфігурування веб-серверу. Розробка сайту з нуля розбивається на такі етапи: дизайн сайту (шаблон); фронтенд розробка; бекенд розробка; робота з базами даних; адміністрування сайту.

Даний посібник призначений для студентів-програмістів, що навчаються за спеціальностями «Комп'ютерна інженерія», «Комп'ютерні науки» та «Кібербезпека». Також посібник буде корисним для усіх бажаючих навчитися створювати інтерактивні вебсайти та поглибити свої знання в області Інтернет технологій. В ньому розглядаються компоненти фронтенд розробки.

В першій частині представлено технологію HTML (англ. Hyper Text Markup Language - мова розмітки гіпертекстових документів) - стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML обробляється браузером та відтворюється на екрані у звичному для людини вигляді.

Також в першому розділі представлено технологію CSS (англ. Cascading Style Sheets або скорочено CSS) - спеціальна мова, що

використовується для опису сторінок, написаних мовами розмітки даних. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

В другому розділі приділена увага сучасному фреймворку Bootstrap - це безкоштовний набір інструментів з відкритим кодом, призначений для створення вебсайтів та вебдодатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних вебсайтів і вебдодатків.

В третій частині розглянуто можливості мови програмування JavaScript – об’єктно-орієнтованої мови програмування сценаріїв. В даний час використовується в основному для створення вбудованих у веб-сторінки сценаріїв, дозволяючи повністю керувати як власне сторінками, так і браузерами, в яких вони відкриті. Таким чином JavaScript використовується в основному для створення інтерактивних сторінок та вебдодатків.

HTML DOM (Document Object Model) розглянуто в четвертому розділі посібника. Практично, в браузерах XHTML DOM реалізується у вигляді об‘єктно-орієнтованого програмного інтерфейсу (API), де об‘єктами є вузли вебдокумента, властивостями – їх атрибути, а методи визначають логіку обробки даних вузлів.

Представлений в навчальному посібнику матеріал надасть читачам теоретичні та практичні рекомендації і допоможе стати затребуваним та досвідченим фахівцем у галузі фронтенд розробки.

РОЗДІЛ 1

HTML5 ТА CSS ЯК ГОЛОВНІ ІНСТРУМЕНТИ РОЗРОБКИ ФРОНТ-ЕНД ЧАСТИНИ ВЕБСАЙТУ

1.1 Технології створення веб-застосунків

Одним із ключових моментів в розвитку всесвітньої павутини є веб-розробка – процес створення вебсайту або вебдодатку. Термін включає розробку додатків електронної комерції, веб-дизайн, програмування для web на стороні клієнта й серверу, а також конфігурування веб-серверу. Основними етапами веб-розробки є:

- проектування сайту або вебдодатку;
- створення макетів сторінок;
- наповнення;
- обслуговування працюючого сайту або його програмної основи;
- подальше просування сайту в мережі та підняття його рейтингу.

Однією з головних існуючих технологій для стоврення сайтів є використання HTML (англ. Hyper Text Markup Language - мова розмітки гіпертекстових документів) - стандартна мова розмітки веб-сторінок в Інтернеті [1]. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді. HTML дозволяє відзначити, де в документі повинен бути заголовок або абзац за допомогою тега HTML, а потім надає Web-браузеру інтерпретувати ці теги. Наприклад, один Web-браузер може розпізнавати тег початку

абзацу й представляти документ у потрібному вигляді, а інший не має такої можливості.

HTML - теги можуть бути умовно розділені на дві категорії:

– Теги, що визначають, як саме буде відображатися Web-браузером тіло документа в цілому.

– Теги, що описують загальні властивості документа, такі як заголовок чи хто є автор документа.

HTML-документи можуть бути створені за допомогою будь-якого текстового редактора або спеціалізованих HTML-редакторів і конвертерів. Вибір редактора, який буде використовуватися для створення HTML-документів, залежить виключно від поняття зручності й особистих вподобань кожного розробника. Основна перевага HTML полягає в тому, що ваш документ може бути переглянутий на Web-браузерах різних типів і на різних платформах.

Каскадні таблиці стилів (англ. Cascading Style Sheets або скорочено CSS) - спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних [2][3][4].

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML[5], але формат CSS може застосовуватися до інших видів XML-документів.

Таблицю стилів CSS можна вмонтувати прямо в HTML - сторінку - це внутрішня таблиця стилів. Або ж її можна створити в окремому файлі, і вже потім приєднати посилання на нього до потрібної HTML-сторінки - це зовнішня таблиця стилів. Зовнішню таблицю необхідно підключити до основного HTML-документу за допомогою спеціальних тегів: <link rel="stylesheet" type="text/css" href="/style.css>, де style.css - це ім'я файлу, що містить таблицю

CSS. Завдяки цьому, стиль, описаний у зовнішній таблиці CSS, можна використовувати повторно скільки завгодно разів.

Таким чином, для початку створення вебдодатків в першу чергу варто вивчити HTML і CSS (актуальні версії CSS 3, HTML 5). Ця зв'язка дозволяє змінювати розміщення, оформлення елементів на HTML-сторінці. Але перед тим, як поринути в мови розмітки/програмування, необхідно познайомитись з основними поняттями у веб-розробці, а також з корисними інструментами, які допоможуть у роботі[6][7][8].

1.2 Клієнт-серверна архітектура Інтернету

Для передачі даних використовуються комп’ютери. Вони являються найбільш важливими апаратними компонентами системи Одні комп’ютери виступають в ролі *серверів* – це означає, що вони надають послуги іншим комп’ютерам. Комп’ютери, які користуються їх послугами, називаються клієнтами. На рисунку 1.1 - зображено клієнт-серверну архітектуру. Клієнт виконує два завдання – представлення й запити. Представлення також називається – інтерфейсом; він просто дає клієнту можливість взаємодіяти з комп’ютером і перетворює дані, отримані від сервера, в формат зрозумілий користувачу. Клієнт також виконує запити на сервер, і сервер відповідає на запити, відправлені клієнтом [9]. Це є простий опис – на практиці він набагато складніший [10]. Для того щоб зрозуміти як працює система клієнт-сервер, використаємо для прикладу Інтернет. Коли ми використовуємо Інтернет, ми використовуємо багаторівневу архітектуру. Але зупинимося на дворівневій архітектурі (клієнт – один сервер).

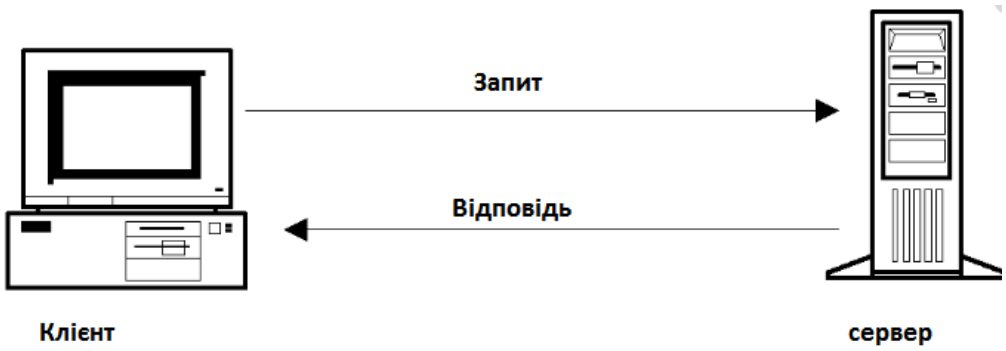


Рисунок 1.1 – Клієнт-серверна архітектура Інтернету

Далі, якщо ми хочемо скористатися, наприклад, пошуковою системою google.com. Ви вже підключені до Інтернету й запустили Web-браузер. Тому просто необхідно ввести в потрібному місці адресу й натиснути кнопку Enter. Це буде приклад, як клієнт виконує задачу представлення. В браузері є місце, де можна ввести адресу, й метод віддати команду щоб перейти на цю адресу. В дійсності, коли ми натискаємо клавішу Enter, браузер відправляє нас на Web-сервер Google запит домашньої сторінки Google. Домашня сторінка – це файл, написаний на мові HTML[11][12]. Web-сервер відповідає на запит нашого браузера, відправляючи через Інтернет дані на наш комп'ютер (клієнт). На нашому комп'ютері ці дані повинні утворити домашню сторінку пошукової системи. Web-браузер приймає дані (в вигляді HTML-файлу) і конвертує код мови HTML, як ви це вже бачите на екрані комп'ютера. Ще одна частина функції представлення: відображення інформації в придатній формі. Відповідно клієнт (комп'ютер із запущеною програмою браузера) дає нам зручну можливість створити запит на сервер, а потім відправити цей запит через Інтернет на Web-сервер Google. Сервер після цього відправляє запитані дані через Інтернет на Ваш комп'ютер, де Web-браузер перетворює їх на Web-сторінку. Цей процес графічно зображено на рисунку 1.2.

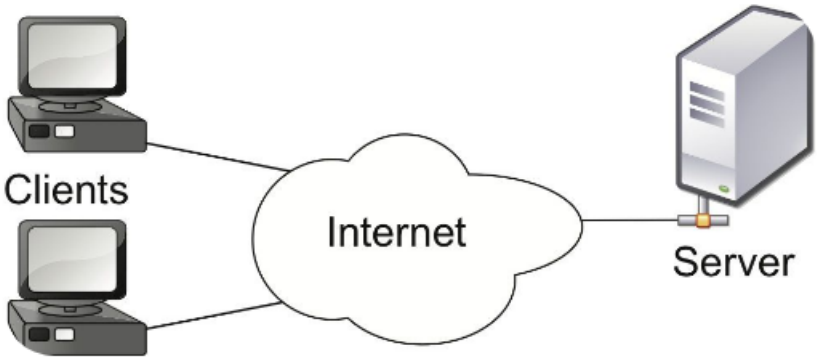


Рисунок 1.2 – Приклад клієнт-серверної архітектури Інтернету

TCP/IP протокол

Інтернет - це мережа, яка об'єднує велику кількість пристроїв (персональних комп'ютерів, смартфонів, смарт TV та ін.). Ці пристрої спілкуються між собою (обмінюються даними), використовуючи TCP/IP протокол.

Протокол TCP (Transmission Control Protocol) розбиває передану інформацію на порції та нумерує їх. За допомогою протоколу IP (Internet Protocol) ці частини передаються одержувачу. Далі TCP перевіряє, чи всі частини отримані, а також розміщує їх в потрібному порядку й збирає в єдине ціле.

HTTP (Hyper Text Transfer Protocol) - протокол передачі даних, що використовується в комп'ютерних мережах. Якщо без зайвих ускладнень, то це проста текстова мова, яка дозволяє двом комп'ютерам спілкуватися один з одним. До прикладу, ноутбук чи смартфон (далі просто клієнт) звертається до сервера використовуючи HTTP і чекає відповідь. Сервер обробляє запит і повертає клієнту тією ж мовою відповідь. Запит - це свого роду текстове повідомлення, яке створюється клієнтом. Тільки

формується він відповідно до спеціальних правил формату, який відомий як HTTP.

Широко використовуються наступні HTTP-методи:

- GET — отримати ресурс із серверу;
- POST — створити ресурс на сервері;
- PUT — оновити ресурс на сервері;
- DELETE — видалити ресурс із серверу.

Веб-браузер

Веб-браузер це програмне забезпечення для перегляду сторінок, але крім цього сучасні браузери мають вбудований потужний набір інструментів для розробників (англ. Developer Tools). Зупинимось на двох важливих і корисних інструментах, які варто розібрати в першу чергу.

Інспектор

Інспектор дозволяє вам побачити, як виглядає HTML на сторінці, а також те, яка CSS застосовується до кожного елемента на цій сторінці.

Щоб дослідити конкретний елемент на сторінці, достатньо навести на нього курсор миші й натиснути праву кнопку маніпулятора миші комп’ютера. В контекстному меню, що з'явиться, слід вибрати пункт "Перевірити" (англ. Inspect), як на рисунку 1.3:

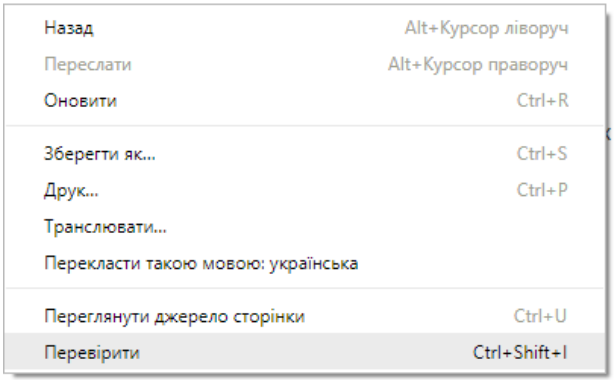


Рисунок 1.3 – Інспектор браузера Firefox

За допомогою інспектора можна змінювати HTML і CSS та одразу ж переглядати результати змін в браузері (рисунок 1.4).

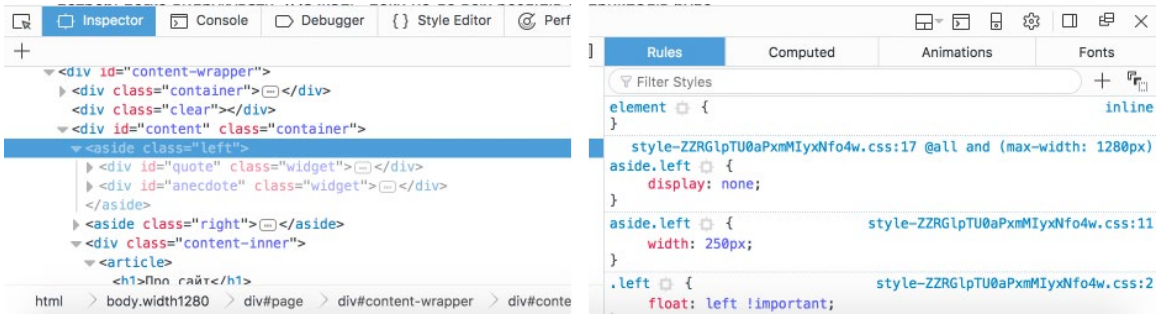


Рисунок 1.4 – Інспектор елементів в браузері Firefox

1.3 Консоль JavaScript

Щоб відкрити панель консолі, наприклад, в браузері Chrome, натисніть комбінацію клавіш Ctrl + Shift + J(Windows / Linux) або ж Cmd + Opt + J (Mac OS).

Консоль JavaScript є неймовірно корисним інструментом для пошуку помилок в JavaScript коді, дослідження коду, який не працює як очікується. Це досягається за рахунок того, що помилки та попередження, що виникають на веб-сторінці, виводяться в консоль.

Метод console.log() показує повідомлення в консолі браузера. Наприклад,

```
<script>
console.log(100 - 99);
</script>
```

Виведе в консоль - 1.

Якщо потрібно, команди JavaScript можуть бути виконані одразу ж в консолі (рисунок 1.5).

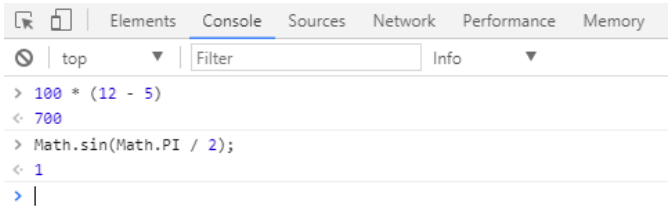


Рисунок 1.5 – Виконання команд в консолі JavaScript

Веб-сервер

Веб-серверами можуть бути комп'ютери або спеціальні програми, які виконують роль сервера. Коли користувач намагається отримати HTML-документ через рядок вводу адреси, то браузер посилає запит через протокол передачі даних HTTP. Коли запит досягає потрібного веб-сервера (залізо), сервер HTTP (програмне забезпечення) передає запитуваний документ назад, також через HTTP.

HTTP (Hyper Text Transfer Protocol) — протокол передачі даних (гіпертекстових документів). Веб-сервер використовує протокол HTTP для сполучення з клієнтом через TCP/IP-мережу.

HTTPS (Hyper Text Transmission Protocol, Secure — протокол захищеної передачі гіпертекстових документів) — HTTP в сукупності з SSL (Secure Sockets Layer) — протоколом захищених сокетів.

Веб-сервер може бути статичним або ж динамічним. Статичний сервер просто надсилає потрібні файли в браузер. Динамічний також вміє надсилати файли в браузер, але на ньому встановлене додаткове програмне забезпечення, яке перед відправкою в браузер змінює вихідні файли. По суті, на льоту генерується відповідь — виконуються обчислення, беруться дані з бази тощо.

Apache і Nginx

Apache - найбільш популярний веб-сервер у світі. Проте чимало високонавантажених вебсайтів використовують Nginx або комбінує їх. Наприклад, Nginx приймає запити і, в разі статичного файлу (зображення, файл CSS, JavaScript або XML) відразу ж віддає його вміст, а в разі, наприклад, PHP-скрипта, відправляє його до сервера Apache, який вже вміє обробляти PHP.

Локальний HTTP-сервер

Якщо на початку роботи для фронтенд розробки можна обійтись без локального HTTP-сервера, то для бекенд розробки він необхідний одразу ж.

Сучасний сайт являє собою не просто набір HTML-документів, але і включає в себе безліч технологій, бази даних та багато іншого.

Для вивчення серверних технологій не зручно та й не ефективно використовувати справжній доступний в мережі Інтернет сервер, тому варто встановити необхідний комплект програм на локальний комп'ютер і розробляти все на ньому.

Найбільш популярною зв'язкою таких програм є веб-сервер Apache, мова програмування PHP і система керування базами даних MySQL.

Cookies

Cookies - механізм, який може використовувати веб-сервер для отримання та збереження інформації про клієнта. Використовуючи cookies, веб-сервер "позначає" комп’ютер-клієнт, записуючи на нього відповідні дані. Під час візитів далі з цього ж комп’ютера до веб-серверу факт попереднього відвідування може бути враховано. Наприклад, при звертанні до веб-серверу реклама на його веб-сторінках може змінюватися залежно від номера відвідування та інтересів певного користувача.

1.4 Фронтенд і бекенд

Веб-розробку умовно можна розділити на дві частини - фронтенд (англ. front-end) і бекенд (англ. back-end). До фронтенду (клієнтська частина) слід віднести HTML-верстку зі CSS-стилями і JavaScript, а до бекенду - серверну частину, яку зазвичай пишуть на Python, PHP, Ruby тощо [9]. Простими словами можна пояснити це наступним чином. Те, що кінцевий користувач бачить в браузері (що найбільше кидається йому в очі) - це фронтенд. А, те, що сховане від людських очей - це бекенд.

Якщо для серверної частини існує чималий список мов програмування, то для клієнтської частини мова JavaScript поза конкуренцією.

Поділ фронтенду

Фронтенд нерідко також розділяють на дизайн (верстання) та розробку. Фронтенд дизайнер (англ. Front-end Designer) більшою мірою займається створенням користувацьких інтерфейсів (англ. User Interface). Він знає добре HTML та CSS. Добре володіє інструментами створення макетів, має відчуття прекрасного. Часом недолюблює JavaScript, зате вправно використовує такі надбудови як jQuery. По суті отримує задоволення від реалізації дизайн рішень. Фронтенд розробник (англ. Front-end Developer) більше уваги приділяє написанню коду на JavaScript. Фреймворки, алгоритми, парадигми програмування тощо не є для нього чимось незрозумілим. Добре знає HTML та CSS. По суті має хист до програмування й отримує задоволення від цього процесу.

Fullstack веб-розробник

Fullstack Developer - це універсальний програміст, який може сам з нуля розробити функціональний продукт. Такий фахівець розуміється як у програмно-апаратній частині сервісу, так і в інтерфейсі користувача. Термін fullstack описує розробника, який

однаково добре справляється з написанням фронтенду й бекенду. Рівень "fullstack" передбачає добре розуміння кожного рівня стеку.

Фреймворки

Фреймворк (англ. framework) - це набір всіляких бібліотек (інструментів) для швидкої розробки повсякденних (рутинних) завдань. Головна мета фреймворку - надати програмісту зручне середовище для проекту з великим і добре розширюваним функціоналом.

Переваги фреймворків:

- фреймворки максимально полегшують роботу розробників, зменшуючи час розробки;

- завдяки фреймворкам, код виходить структурованим, зрозумілим і доступним для повторного використання;

- фрейморки використовують популярні шаблони проектування (наприклад, Singleton, MVC).

Якщо в проекті ви використовуєте фреймворк, то більша частина коду й структура проекту будуть базуватись на цьому каркасі. Ви отримаєте, як інструмент, добре спроектовану систему, оминаючи чимало підводних каменів, про які ви навіть можете не підозрювати на початку вивчення веб-розробки.

З іншого боку, якщо ви вирішили створювати сайт, сервіс з нуля, то від вас вимагається вищий рівень кваліфікації, адже спроектувати більш-менш серйозний проект - діло не з легких.

Редактори коду

Є чимало різних варіантів і з часом кожен підбере той редактор, який буде найзручнішим для нього. Але перед цим слід уточнити, що умовно редактори можна поділити на текстові редактори й IDE (англ. Integrated Development Environment).

В той час як перші є легкими, інші (наприклад, Eclipse, NetBeans або AptanaStudio) - важкі, вимогливі до ресурсів, проте мають значно більше корисних функцій.

Вимоги до сучасного редактора коду

– Підсвічування синтаксису мови програмування, на якій пишуть код.

– Підказки коду, авто-завершення.

– Можливість відлагоджувати код (англ. debugging).

– Розширюваність за допомогою плагінів.

– Можливість працювати з Git (або іншою системою керування версіями файлів).

– Підтримка препроцесорів, якщо ви їх використовуєте (Less/Sass тощо).

– Наявність вбудованого FTP-клієнта, SSH тощо.

Нижче наведено невеличкий список сучасних редакторів коду.

Текстові редактори

Atom від GitHub

Atom позиціонують, як текстовий редактор, створений для потреб сучасних розробників. Завдяки відкритому API має чимало послідовників та силу-силенну плагінів. Atom - безкоштовний і працює на Windows, Mac OS та Linux.

Brackets від Adobe

Brackets зосереджений на потребах веб-дизайнерів з вбудованою підтримкою HTML, CSS та JavaScript. Brackets - безкоштовний, легкий у використанні, швидкий і має багато додаткових плагінів. Працює на Windows, Mac OS та Linux.

Notepad++

Notepad++ - безкоштовний, невимогливий до ресурсів і дуже швидкий редактор коду, який, щоправда, доступний лише для Windows. Даний редактор призначений для тих, хто полюбляє простий інтерфейс. Базовий функціонал Notepad++ можна доповнювати за допомогою плагінів.

SublimeText (частково платний)

SublimeText - це надзвичайно потужний текстовий редактор для коду, який можна розширити за допомогою плагінів. Привабливий його витончений інтерфейс, набір корисних функцій та вражаюча продуктивність. Редактор платний, але нема обмеження на те, як довго ви можете використовувати пробну версію. SublimeText доступний на Windows, Mac OS та Linux.

Он-лайн редактори

Останнім часом швидкими темпами розвиваються он-лайн редактори:

Cloud9 — https://aws.amazon.com/cloud9/

ICEcoder — https://icecoder.net/

Codio — https://www.codio.com/

Codeanywhere — https://codeanywhere.com/

Їх не потрібно встановлювати, вони працюють в браузері як GoogleDocs.

Emmet

Emmet (раніше називався ZenCoding) — інструмент для веб-розробників, який дозволяє швидше писати HTML та CSS (або будь-який інший структурований формат коду). Можливість досягається за рахунок використання спеціальних комбінацій коду і клавіш. Після чого короткі позначення, які схожі на CSS селектори, трансформуються в HTML-код.

Emmet використовується як розширення (плагін) до вашого улюбленого текстового редактора. Наприклад, в редакторі, який

підтримує даний інструмент, вводиться короткий запис і тиснути клавішу Tab (рисунок 1.6):

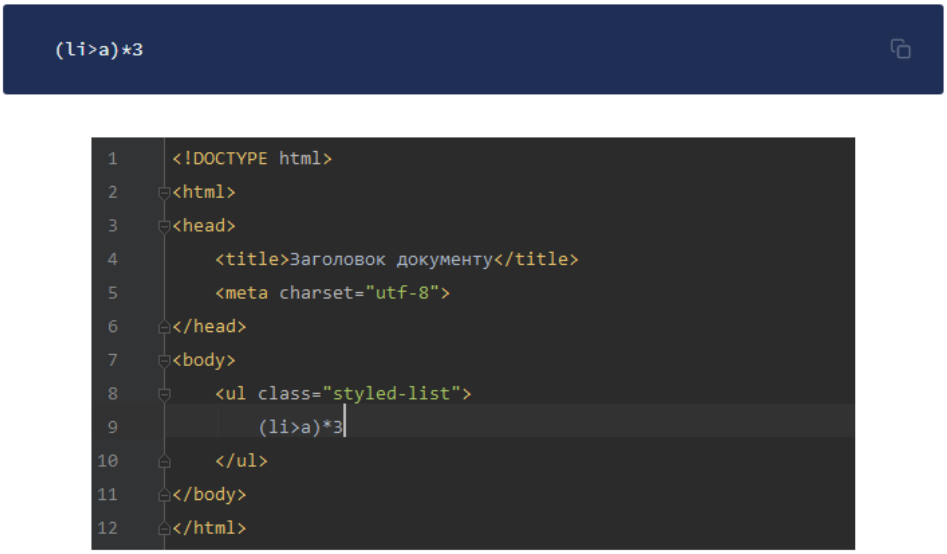


Рисунок 1.6 – Приклад використання плагіну Emmet

Після чого короткий код розгортається у відповідний HTML (рисунок 1.7).

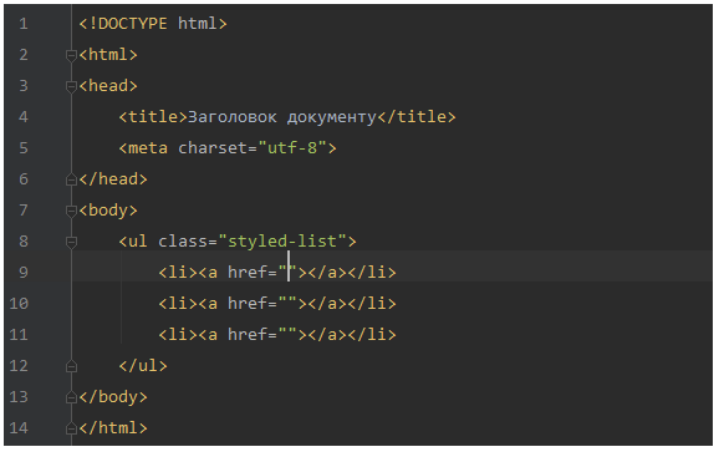


Рисунок 1.7 – Розгорнуте представлення коду

1.5 Створення першої сторінки

Увесь код при роботі з HTML5, CSS, JS, PHP, MySQL в даному посібнику буде написаний в інтегрованому середовищі IDE компанії JetBrains – PhpStorm (дана IDE для студентів навчальних

закладів розповсюджується безкоштовно) тому рекомендується встановити дану систему або подібну, для виконання завдань і прикладів, які будуть наведені в посібнику. Також, на перший час буде достатньо й текстового редактора для програмістів з відкритим кодом NotePad++ (теж має функцію підсвічування коду й помилок). Також усі приклади будуть протестовані в браузері Google Chrome, а для роботи з базами даних та PHP встановлено локальний сервер Open Server [13][14].

Адреси для скачування:

IDE PHPStorm - https://www.jetbrains.com/

Open Server Panel - https://ospanel.io/

HTML є стандартною мовою розмітки для створення веб-сторінок.

– HTML означає HyperText Markup Language (мова гіпертекстової розмітки);

– HTML описує структуру веб-сторінок за допомогою розмітки;

– HTML-елементи є будівельними блоками сторінок HTML;

– Елементи HTML представлені тегами;

– Теги HTML позначають фрагменти вмісту, а саме, header (заголовок), paragraph (параграф), table (таблиця), тощо;

– Браузери не показують теги HTML, але використовують їх для відображення вмісту web-сторінки. Метою веб-браузера (Chrome, Firefox, Opera, Internet Explorer, Safari) є читання документів HTML та їх відображення. Браузер використовує теги HTML щоб визначити, як відобразити веб-сторінку.

Тепер запустимо програмне забезпечення PhpStorm і створимо перший HTML-документ. Для цього необхідно перейти на

вкладку File/New../HTML File/ й у відкритому вікні написать ім’я нового файлу: Index. Так буде системою створено головний файл потрібної сторінки Index.html п’ятої версії. Слід зазначити, що усі документи HTML повинні мати розширення файлу html.

HTML-сторінку (суміш тегів і тексту) зберігають у звичайних текстових файлах з розширенням .html (рідше .htm). Наприклад, example.html чи test1.html.

Результат виконання представлено на рисунку 1.8.

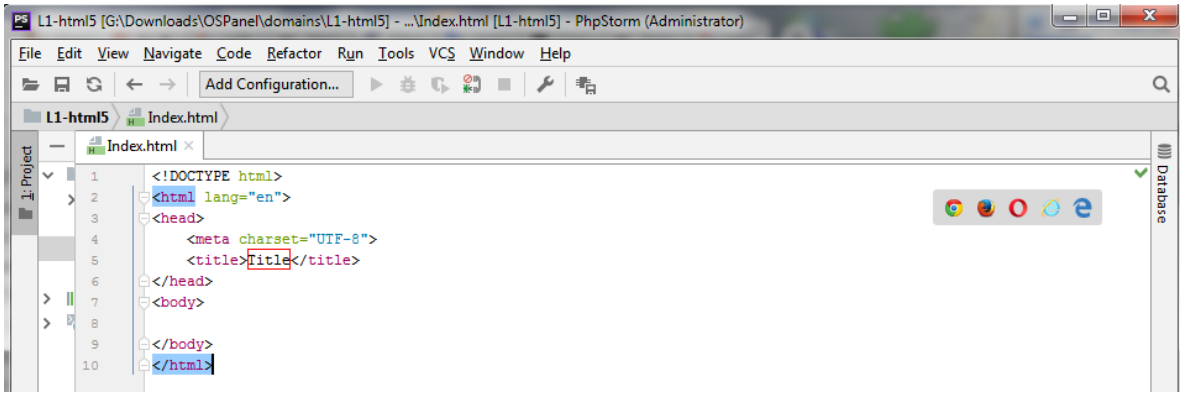


Рисунок 1.8 – Створення першої HTML-сторінки

На рисунку 1.9 зображено застосування деяких тегів для побудови HTML-сторінки. Пояснення прикладу та деяких тегів наводиться далі.

Рисунок 1.9 – Застосування тегів для побудови сторінки

Пояснення прикладу:

– Оголошення <!DOCTYPE html> визначає цей документ, як HTML5;

– Елемент <html> є кореневим елементом HTML-сторінки;

– Елемент <head> містить мета-інформацію про HTML документ;

– Елемент <title> вказує назву HTML документа;

– Елемент <body> містить видимий вміст HTML-сторінки;

– Елемент <h1> визначає великий заголовок на HTML-сторінці;

– Елемент <p> визначає абзац (параграф) в HTML документі.

Результат виконання можна побачити відкривши збережений файл в браузері (рисунок 1.10).

Рисунок 1.10 - Результат виконання в браузері Google Chrome

Теги HTML

HTML-теги - це назви елементів, оточені кутовими дужками: <назва тегу> Тут йде зміст... </назва тегу>.

Теги бувають парні та непарні (елементи, створені парними тегами, називаються контейнерами).Теги зазвичай йдуть парами, наприклад, і .

Першим тегом у парі є початковий тег, другий - кінцевий тег. Кінцевий тег записується як початковий тег, але перед ім'ям тегу ставлять косу рису (/слеш).

Початковий тег називається - відкриваючим тегом, а кінцевий тег - закриваючим тегом.

Примітка: у браузері відображається лише вміст у розділі <body>.

Об'ява <!DOCTYPE>

Об'ява <!DOCTYPE> являє собою визначення типу документа і допомагає браузерам правильно відображати веб-сторінки. Вона повинна з'являтися тільки один раз у верхній частині сторінки (перед будь-якими тегами HTML).

Об’ява <!DOCTYPE> не чутлива до регістру.

Об’ява <!DOCTYPE> для HTML5 виглядає так:<!DOCTYPE html>.

Елементи HTML-коду

Елемент в HTML зазвичай складається з початкового тегу та кінцевого тегу, вміст якого вставлений між ними:

<Початковий тег> Вміст розміщується тут ... </Кінцевий тег>.

Елемент HTML - це все від початкового тегу до кінцевого тегу:

Текст параграфу може бути тут... .

Непарні або одиничні теги не мають закриваючого тегу. Наприклад, тег, яким позначають картинку:

<img src="/path/to/logo.png" alt="Логотип">.

У прикладі вище в тезі img такими атрибутами є src (повний шлях, за яким можна знайти зображення) та alt (текст-підказка, яка відобразиться в браузері, якщо картинку не вдасться показати). Атрибут дозволяє задати додаткову інформацію про тег.

Важливою частиною будь-якого HTML-документу є посилання. Вставити його можна наступним чином:

Текст посилання

В атрибуті href слід вказати URL-адресу.

Вкладені елементи HTML

Елементи HTML можуть бути вкладеними (елементи можуть містити елементи). Всі документи HTML складаються з вкладених елементів HTML. Наприклад:

```
<html>
<body>
<p>Hello World!
<strong>My first paragraph</strong>
</p>
</body>
</html>
```

Цей приклад містить чотири HTML елементи.

Пояснення прикладу

Елемент <html> визначає весь документ. Він має початковий тег <html> і кінцевий тег </html>. Вміст (контент) елемента - це інший елемент HTML (елемент <body>).

Елемент <body> визначає тіло документа. Він має початковий тег <body> і кінцевий тег </body>. Вміст (контент) елемента - це два інші елементи HTML (і).

Елемент визначає параграф. Він має початковий тег і кінцевий тег . Вміст (контент) елемента: My first paragraph.

Порожні елементи HTML

Елементи HTML без вмісту називаються порожніми елементами.

 - це порожній елемент без закриваючого тегу (тег
 визначає розрив рядка):

```
<p>This is a <br>paragraph with a linebreak.</p>
```

HTML5 не вимагає закриття порожніх елементів. Але якщо ви хочете більш суворої перевірки, або якщо вам потрібно зробити ваш документ таким, що читається XML-парсерами, ви повинні закрити всі HTML-елементи належним чином.

Теги HTML не чутливі до регістру: <P> (у великому регістрі) означає те саме, що і (у маленькому регістрі).

Стандарт HTML5 не вимагає використання тегів в нижньому регістрі, але W3C рекомендує нижній регістр для написання HTML-коду й вимагає нижнього регістру для більш жорстких типів документів, таких як XHTML.

1.6 Семантична структура HTML5 сторінки

Семантичні елементи HTML5 описують свій сенс або призначення як для браузерів, так і для веб-розробників.

До появи стандарту HTML5 вся розмітка сторінок здійснювалася переважно за допомогою елементів , яким привласнювали класи class або ідентифікатори id для наочності розмітки (наприклад,). З їх допомогою в HTML-документі розміщували верхні й нижні колонтитули, бічні панелі, навігацію й багато іншого.

Стандарт HTML5 [1] надав нові елементи для структурування, групування контенту й розмітки текстового вмісту. Нові семантичні елементи дозволили поліпшити структуру веб-сторінки, додавши смислове значення укладеному в них вмісту (було , стало <header>). Для відображення зовнішнього вигляду елементів не задано жодних правил, тому елементи можна стилізувати на свій розсуд. Для всіх елементів доступні глобальні атрибути.

Основна мета використання семантичних тегів це:

– поліпшення структури контенту;
– допомога пошуковим роботам;
– адаптація для різних пристроїв (наприклад, для пристроїв, що використовуються людьми зі слабким зором).

Тільки використання семантичних тегів не призведе до революції в роботі SEO (англ. search engine optimization). Як відомо, успішне SEO це сукупність багатьох і багатьох дрібних деталей. І це одна з таких малих деталей, яка поліпшить розуміння контенту сайту з боку будь-якого пошукача, що помітно внесе вклад у SEO зусилля.

На рисунку 1.11 наведено схему HTML5 сторінки і відповідні теги.

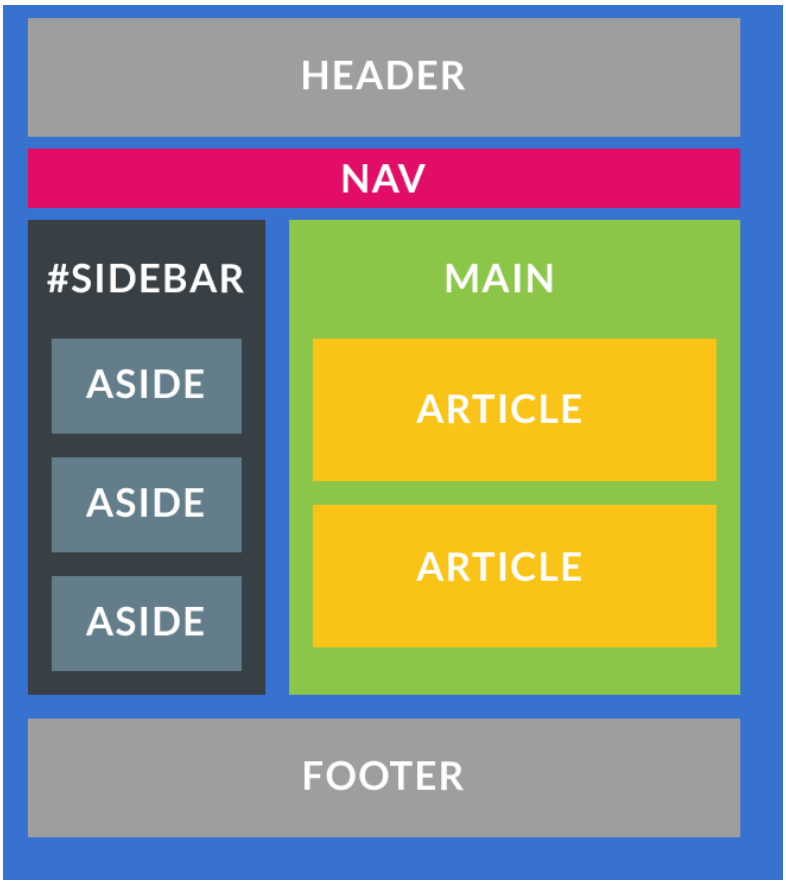


Рисунок 1.11 - Семантична структура для HTML5 сторінки

Розглянемо спочатку які є семантичні теги в п’ятій версії HTML. Нижче в таблиці наведено короткий опис семантичних тегів.

Таблиця 1.1 – Семантичні теги HTML5

Header	шапка документа або секції
Nav	основна навігація
Aside	допоміжний контент
Section	семантично відокремлена секція
Main	основний текст сторінки
Footer	підвал документа або секції
Article	стаття або новина
Figure	автономний контент (картинка і т.п.)
Figcaption	підпис для figure

Details	додаткові відомості
Summary	видимий підпис до details
Mark	виділений текст
Time	час / дата

Відповідно як можна змінити структуру сторінки порівняно з типовим представленням наведено на рисунку 1.12.

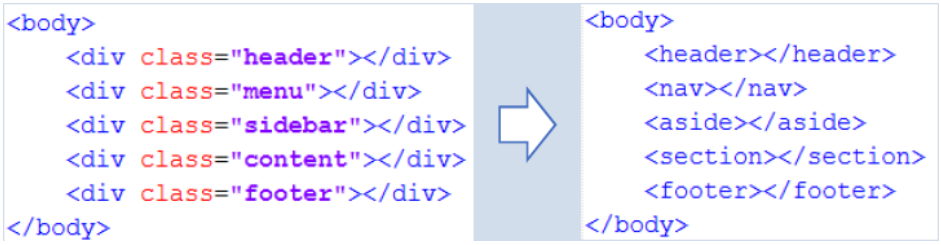


Рисунок 1.12 – Структура сторінки в типовому вигляді та з використанням семантичних тегів HTML5

Далі розглядається більш детально структура сторінки.

Заголовок сторінки

Шапка сторінки оформляється тегом <header>. Тут необхідно зауважити, що заголовок сторінки виконується тегом h1. Якщо у нас є ще й слоган поруч із заголовком, то поміщаємо його в тег p, div або span.

Слід зауважити, що в HTML5 тег h1 використовується для вказівки заголовка контейнера, в якому він знаходиться (це може бути header, section, article і т.д.)

До появи HTML5 тегів семантика була дещо іншою й відрізнялася. Так в HTML4 на сторінці міг бути тільки один заголовок <h1>! Як правило це був заголовок статті або заголовок сторінки (наприклад, якщо це сторінка рубрики на якій

відображаються кілька статей). Тег <h2> використовувався для підзаголовків, або для розділів головної статті. Тег <h3> для підрозділів і так далі.

```
<!-- Header сторінки -->
<header>
<h1>Site title</h1>
<p>site slogan</p>
</header>
```

Навігація на сторінці. <Nav>элемент

Оформлення головного меню сайту повинно бути обернуто в тег <nav>. Також слід пам'ятати що хорошою практикою вважається оформляти навігацію елементами списку.

Приклад:

```
<!-- Головна навігація по сайту -->
<nav>
<ul>
<li><a href="#">Home</a></li>
<li><a href="#">Portfolio</a></li>
<li><a href="#">Gallery</a></li>
<li><a href="#">Contacts</a></li>
</ul>
</nav>
```

Контент на сторінці

Основний вміст сторінки оформляється тегом main. Це може бути одна стаття, або кілька перев'ю статей, якщо мова йде про сторінку блога з декількома записами. Ніколи не кладіть сайдбар, хедер сторінки, футер або головну навігацію в тег main!

<!-- Основний вміст сторінки -->

<main>

...головний контент сторінки...

</main>

Оформлення статті

Тег article - служить для обгортки статей. Загалом цей тег містить в собі блок контенту, який може бути вийнятий з контексту сторінки, і використаний окремо в іншому місці. Це може бути стаття (повний текст статті або перев'ю), пост на форумі, й т.ін.

Сайдбар або колонка з відметами

Для кожного окремого елемента сайдбара використовуємо блок aside. Усередині нього заголовок створюємо тегом <h1>. Так колонка в сайдбарі може виглядати наступним чином:

```
<!-- Сайдбар -->
<div class="sidebar">
<!-- Віджет в сайдбарі -->
<aside>
<h1>Widget title</h1>
            ...
</aside>
<!-- Віджет в сайдбарі -->
<aside>
<h1>Останні записи</h1>
            ...
</aside>
<!-- Віджет в сайдбарі -->
<aside>
<h1> Популярні коментарі </h1>
            ...
</aside>
</div>
```

Тег section

Тег <section> - використовується для представлення групи або секції тематично пов'язаного контента. Його використання схоже на article з головною відмінністю в тому, що допускається відсутність сенсу вмісту всередині елемента <section> поза

контекстом самої сторінки. Рекомендується використовувати теги (<h1> - <h6>) для позначення теми секції.

Елементи <figure> і <figcaption>

Призначення елемента <figcaption> - додавання візуального пояснення до зображення. У HTML5 зображення й пояснення до нього може бути згруповано в елементі <figure>:

```
<figure>
<imgsrc='img_pulpit.jpg'  alt="The  Pulpit  Rock"
width="304" height="228">
<figcaption> Рис. 1 - Гора в Криму </figcaption>
</figure>
```

Підвал сайта - Footer

Підвал сайта оформляється тегом <footer>. Зазвичай, в "підвалі" розміщують інформацію про автора документа, посилання на умови використання тексту, інформація про авторські права, контактні дані й т.ін. В одному документі дозволяється визначати кілька елементів <footer>.

Приклад:

```
<!-- Підвал сайта -->
<footer>
<p class="copyright">© 2021 Copyright</p>
</footer>
```

Про інші семантичні елементи та їх використання детально можна ознайомитися за наступними посиланнями:

https://developer.mozilla.org/en-US/docs/Web/HTML

https://metanit.com/web/html5/1.1.php

1.7 Форми HTML5

HTML-форми є елементами управління, які застосовуються для збору інформації від відвідувачів вебсайту [4][5], ті поля, куди ми, наприклад, вводимо свої дані при реєстрації на будь - якому

сайті, форумі й т.д. Веб-форми складаються з набору текстових полів, кнопок, списків та інших елементів управління, які активізуються натисканням миші. Технічно форми передають дані від користувача віддаленого сервера. Для отримання й обробки даних форм використовуються мови веб-програмування, такі як PHP, Perl.

До появи HTML5 веб-форми представляли собою набір декількох елементів <input type = "text">, <input type = "password">, що завершуються кнопкою <input type = "submit">. Для стилізації форм в різних браузерах доводилося докладати чимало зусиль. Крім того, форми вимагали застосування JavaScript для перевірки введених даних, а також були позбавлені специфічних типів полів введення для вказівки інформації типу дат, адрес електронної пошти та URL-адрес.

HTML5-форми вирішують більшість цих поширених проблем завдяки наявності нових атрибутів, надавши можливість змінювати зовнішній вигляд елементів форм за рахунок CSS3[6].

Приклад форми створеної в HTML5 представлено на рисунку 1.13.

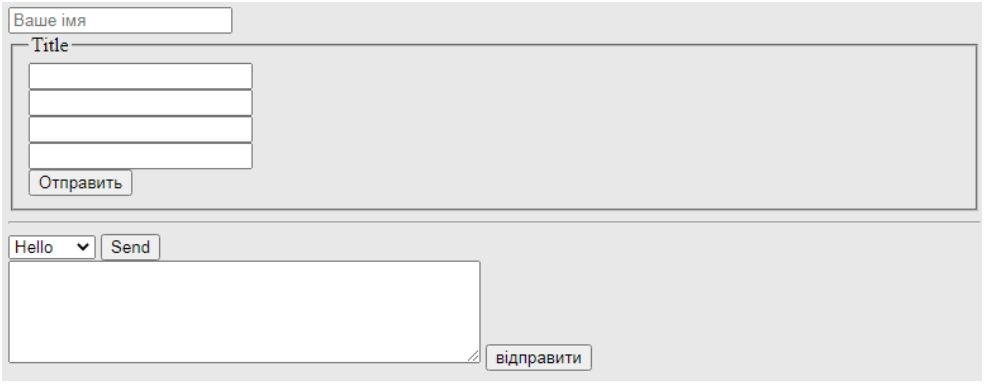


Рисунок 1.13 – Форма HTML5

Створення HTML5-форми

Элемент <form>

Основу будь-якої форми становить елемент <form> ..</ form>. Він не передбачає введення даних, тому, що є контейнером, що утримує разом всі елементи управління форми - поля. Атрибути цього елемента містять інформацію, загальну для всіх полів форми, тому в одну форму потрібно включати поля, об'єднані логічно.

Головні атрибути елемента <FORM>

Action - обов'язковий атрибут, який вказує url обробника форми на сервері, якому передаються дані. Являє собою файл (наприклад, action.php), в якому описано, що потрібно робити з даними форми. Якщо значення атрибута не буде вказано, то після перезавантаження сторінки елементи форми приймуть значення за замовчуванням.

У разі, якщо вся робота буде виконуватися на стороні клієнта сценаріями JavaScript, то для атрибута action можна вказати значення #.

Також можна зробити так, щоб заповнена відвідувачем форма приходила на вказану поштову адресу. Для цього потрібно внести такий запис:

<form action = "mailto: адреса вашої електронної пошти" enctype = "text / plain"></form>

Method - задає спосіб передачі даних форми.

Метод **get** передає дані на сервер через адресний рядок браузера.

При формуванні запиту до сервера всі змінні та їх значення формують послідовність виду www.site.ua/form.php?var1=1&var2=2. Імена та значення змінних приєднуються до адреси сервера після знака ? і розділяються між собою знаком &. Всі спеціальні символи й букви, відмінні від латинських, кодуються в форматі %nn, пробіл замінюється на +. Цей метод потрібно використовувати, якщо ви не

передаєте великих обсягів інформації. Якщо разом з формою передбачається відправка будь-якого файлу, цей метод не підійде.

Метод **post** застосовується для пересилання даних великих обсягів, а також конфіденційної інформації і паролів. Дані, що відправляються за допомогою цього методу, не помітні в заголовку URL, оскільки вони містяться в тексті листа.

<form action = "action.php" enctype = "multipart / form-data" method = "post"></ form>

Name - задає ім'я форми, яке буде використовуватися для доступу до елементів форми через сценарії, наприклад, name = "test".

Групування елементів форми

Елемент <fieldset> ... </ fieldset> призначений для групування елементів, пов'язаних один з одним, розділяючи таким чином форму на логічні фрагменти.

Кожній групі елементів можна привласнити назву за допомогою елемента <legend>, який йде відразу за відкриваючим тегом елемента <fieldset>. Назва групи проявляється зліва у верхній межі <fieldset>. Наприклад, якщо в елементі <fieldset> зберігається контактна інформація:

```
<form>
<fieldset>
<legend>Контактна інформація</legend>
<p><label for="name"> Имя <em>*</em></label><input
type="text" id="name">
</p>
<p><label for="email">E-mail</label><input
type="email" id="email"></p>
</fieldset>
<p><input type="submit" value="відправити"></p>
</form>
```

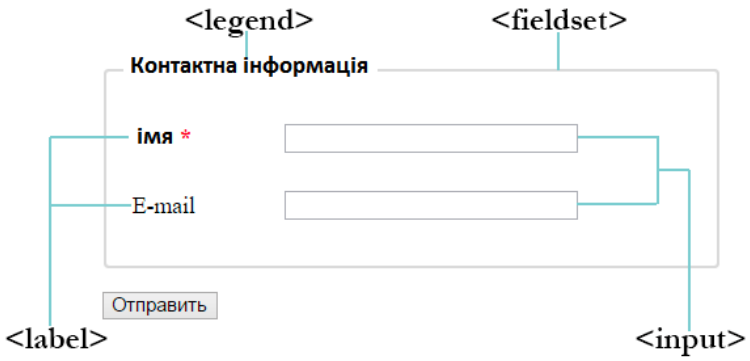


Рисунок 1.14 – Групування елементів форми за допомогою тегу «fieldset»

В таблиці 1.2 розглянуто атрибути елемента «FIELDSET».

Таблиця 1.2 - Атрибути елемента «FIELDSET»

Атрибут	**Значення/опис**
disabled	Якщо атрибут присутній, то група пов'язаних елементів форми, що знаходяться всередині контейнера <fieldset>, відключені для заповнення й редагування. Використовується для обмеження доступу до деяких полів форми, що містять раніше введені дані. Атрибут використовується без вказівки значення – <fieldset disabled>.
Form	Значення атрибута має дорівнювати атрибуту id елемента <form> в цьому ж документі. Вказує на одну або кілька форм, до яких належить дана група елементів.
Name	Визначає ім'я, яке буде використовуватися для посилання на елементи в JavaScript, або для посилання на дані форми після заповнення й відправки форми. Є аналогом атрибута id.

Створення полів форми

Елемент <input> створює більшість полів форми. Атрибути елемента відрізняються в залежності від типу поля, для створення якого використовується цей елемент. Далі за допомогою css-стилів можна змінити розмір шрифту, тип шрифту, колір та інші властивості тексту, а також додати, колір фону й фонове зображення. Ширина поля задається властивістю width.

Нижче в таблиці 1.3 наведено таблицю головних атрибутів елемента <input>.

Таблиця 1.3 - Головні атрибути елементу <input>

Атрибут	**Значення та опис**
accept	Визначає тип файлу, дозволений для відправки на сервер. Вказується тільки для <input type = "file">. Можливі значения: file_extention - дозволяє завантаження файлів із вказаним розширенням, наприклад, accept = ". gif", accept = ". pdf", accept = ". doc" audio /* - дозволяє завантаження аудіо файлів video /* - дозволяє завантаження відео файлів image /* - дозволяє завантаження зображень media_type - вказує на медіа-тип завантажених файлів.
Alt	Визначає альтернативний текст для зображень, лише для <input type = "image">.
checked	Атрибут перевіряє, чи встановлений флажок за замовчуванням при завантаженні сторінок для полей типу type = "checkbox" і type = "radio"

formmethod	Атрибут визначає метод, який браузер буде використовувати для відправлення даних на сервер. Задається лише для полів типу type = "submit" і type = "image". Перевизначає значення форм методу атрибута. Варіанти: get - значення за умовою. Дані з форм (параметр / ім'я) додаються в URL-адресу та відправляються на сервер: URL? Ім'я = значення & ім'я = значення post - дані форми відправляються у вигляді http-запиту.
List	Є посилання на елемент <datalist>, містить його ідентифікатор. Дозволяє надати користувачеві кілька варіантів вибору, коли він починає вводити значення у відповідному полі**.**
Name	Визначає ім'я, яке буде використовуватися для доступу до елементу <form>, до прикладу, у таблицях стилів css. Є аналогом атрибута id.
Required	Виводить повідомлення про те, що дане поле є обов'язковим для заповнення. Якщо користувач намагається надіслати форму, не заповнивши це поле, то на екрані відображається попереджувальне повідомлення. Використовується без значення атрибута.
Size	Задає видиму ширину поля в символах. Значення за замовчуванням - 20 символів . Працює з наступними типами полів: text, search, tel, url, email та password.
Value	Визначає текст, який відображається на кнопці. Не вказується для полів типу file.
Width	Значення атрибута містить кількість пікселів.

	Дозволяє задати ширину полів форм.
Type	– button - створює кнопку – checkbox - прапорець – перетворює поле вводу у флажок, який можна встановити або очистити – date - дозволяє вводити дату у форматі дд.мм.рррр – email - браузери, підтримуючі даний атрибут, будуть очікувати, що користувач вводить дані, відповідні синтаксису адреси електронної пошти. – file - дозволяє завантажувати файли з комп'ютера користувача. – image - створює кнопку, дозволяючи замість тексту на кнопці вставити зображення. – password - створює текстові поля у формі, при цьому вводячи користувачем символи, що змінюються на зірочки, маркети, або інші, встановлені браузером значки. – radio - створює перемикач - елемент управління у вигляді невеликого кола, який можна включити або виключити. – reset - створює кнопку, яка очищає поля форми від введених користувачем даних. – url - поле призначено для вказування URL-адреси.

Розглянемо можливості елемента <input> який використовується для створення більшості полів форм HTML5

Текстові поля, в яких можна написати один рядок

Текстове поле <input type="text" autocomplete="on" maxlength="5" placeholder="Маша" required size="100" spellcheck="true" name="myName">

Поле пароль<input type="password">

Поле емаіл<input type="email" multiple>

Поле url<input type="url">

Поле пошуку<input type="search">

Поле для телефону<input type="tel" required>

Результат виконання представлено на рисунку 1.15.

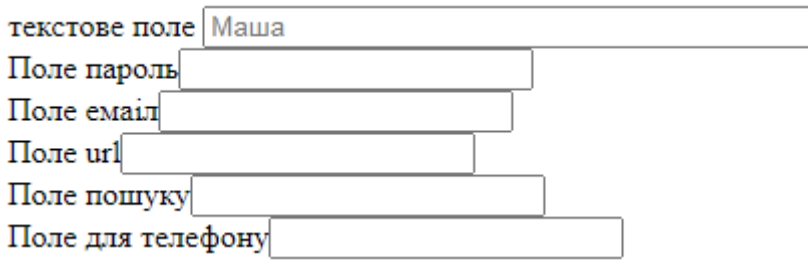


Рисунок 1.15 - Створення input полів вводу тексту

Input без вводу текста

Чекбокс1: <input type="checkbox" name="c1" checked>

Чекбокс2: <input type="checkbox" name="c1">

Чекбокс3: <input type="checkbox" name="c1">

Радіо-кнопка1<input type="radio" name="c1" value="1">

Радіо-кнопка2<input type="radio" name="c1" value="2">

Радіо-кнопка3<input type="radio" name="c1" value="3" checked>

Колір:<input type="color">

файл<input type="file" accept="image/jpeg,image/png" multiple>

Сховане поле<input type="hidden">

Числове поле<input type="number">

Ввод дати без часу<input type="date">

ввод дати з часом<input type="datetime">

Поле-діапазон<input type="range" min="0" max="1000"

step="100">

Контроль вводу дати<input type="datetime-local">

Місяць<input type="month">

Час<input type="time">

Неділя<input type="week">

Результат виконання представлено на рисунку 1.16.

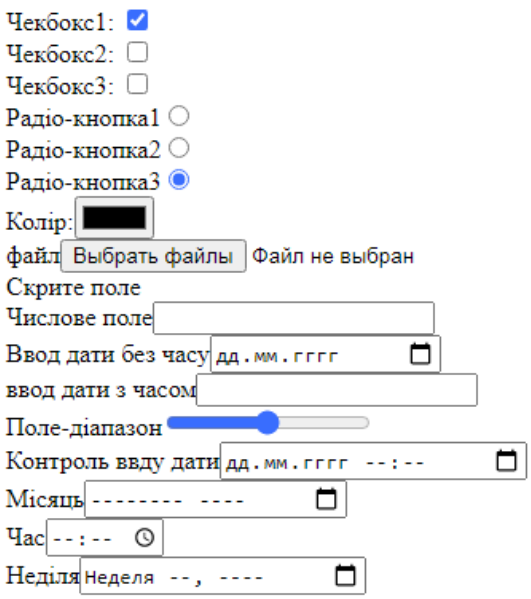


Рисунок 1.16 – Створення input полів без вводу тексту

Input-кнопки

Кнопка без поведінки <input type="button">

Графічна кнопка відправить<input type="image" alt="sub"

src="../img/sub.jpg" width="5%" height="5%">

Кнопка очищення форми<input type="reset">

Кнопка відправки форми<input type="submit" >

Результат виконання представлено на рисунку 1.17.

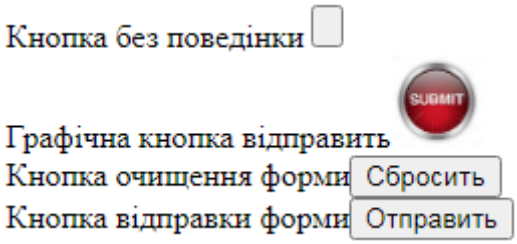


Рисунок 1.17 – Створення кнопок input

Текстові поля вводу

Елемент <textarea> ... </textarea> використовується замість елемента <input type = "text">, коли потрібно створити більші текстові поля. Текст, що відображається як вихідне значення, розміщується в середині.

Розміри поля встановлюються при використанні атрибутів cols - розміри по горизонталі, rows - розміри по вертикалі. Висоту поля можна задати властивістю height. Усі розміри розраховуються за розміром одного символу моноширинного шрифту.

Нижче в таблиці 1.4 наведено головні атрибути елемента <textarea>.

Таблиця 1.4 – Головні атрибути елемента <textarea>

Cols	Встановлює ширину через кількість символів. Якщо користувач вводить більше тексту, з'являється полоса прокрутки.
Rows	Вказує число, яке означає, скільки рядків повинно відображатися в текстовій області.
Maxlength	Значення атрибута задає максимальне число

	символів для вводу в поле.
Placeholder	Визначає коротку текстову підказку, яка описує очікуване значення.
Name	Задає ім'я текстового поля.
Form	Значення атрибута має бути рівнозначним атрибуту ідентифікатора елемента <form> в цьому ж документі. Визначає одну або кілька форм, які належать даному текстовому полю.

Список, що розгортається

Списки дають можливість розташувати велику кількість пунктів компактно. Списки, що розгортаються створюються за допомогою елемента <select> ... </select>. Вони дозволяють вибрати одне або декілька значень із запропонованої множини. За умовою в полі списку відображається його перший елемент.

Для додавання в список пунктів використовуються елементи <option> ... </option>, які розміщуються всередині <select>.

Для систематизації списків застосовується елемент <optgroup> ... </optgroup>, який створює заголовки в списках.

Для списків можливо змінити розмір шрифта, тип шрифта, колір та інші властивості тексту, а також додати границі, кольори та фонове зображення.

Прапорці та перемикачі у формах

Прапорці у формах задаються за допомогою конструкцій <input type = "checkbox">, а перемикач - за допомогою <input type = "radio">.

Прапорці, відрізняються від перемикачів, тим, прапорець існує сам по собі, перемикачі можуть з'являтися тільки у вигляді

списку (що означає, щонайменше два варіанти). Крім того, клацання по прапорцю є довільним, в той час як вибір одного з перемикачів є обов'язковим.

Якщо для прапорців вказаний атрибут checked, то після завантаження сторінки на відповідних полях форми прапорці вже будуть встановлені.

Елемент <label> застосовується при реалізації вибору за допомогою перемикачів та прапорців. Можливо вибрати потрібний пункт, просто натискаючи кнопку миші на тексті, пов'язаному з ним. Для цього потрібно розмістити <input> всередину елемента <label>.

Нижче наведено код форми побудованої з застосуванням тегів форми HTML5 та результат виконання на рисунку 1.18.

```
<body>
<header>
<h4>Контактна форма</h4>
<hr>
</header>
<main>
<form action="contact.php" method="post" >
<fieldset>
<legend>Контактна інформація</legend>
      Your name<br>
<input type="text" name="cf_name"><br>
      Your e-mail<br>
<input type="text" name="cf_email"><br>
      Message<br>
<textarea name="cf_message" rows="10"
cols="50"></textarea>
<br>
</fieldset>
<fieldset>
<legend>Перснальна інформація</legend>
```

```
          Your name<br>
<input type="text" name="cf_name"><br>
          Your e-mail<br>
<input type="text" name="cf_email"><br>
          Скільки Вам років <br>
<input type="number" min="1" max="100" step="1"><br>
          Phone Number<br>
<input type="tel" required><br>
          Одружений<input type="radio" name="c1"
value="1"><br>
          Неодружений<input type="radio" name="c1"
value="2"><br>
<br>
<input type="submit" value="Send">
<input type="reset" value="Clear">
</fieldset>
</form>
</main>
```

Рисунок 1.18 – Результат виконання коду побудови форми

1.8 CSS стилі в HTML

CSS (Cascading Style Sheets) – каскадні таблиці стилів, які застосовуються для візуального форматування документу в мовах розмітки. CSS використовується для того, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки [2][5].

CSS найчастіше зменшує об’єм коду, дозволяє виносити стилі в окремий файл, який можна повторно використовувати та ще багато іншого.

Головна ідея CSS в тому, щоб відокремити дизайн документу від його вмісту. Тобто CSS відповідає за оформлення й зовнішній вигляд, а HTML - за зміст і логічну структуру документу.

За бажанням можливо стилізувати теги, використовуючи атрибут *style*:

 Текст посилання

В коді вище ми задали для посилання червоний колір тексту та розмір шрифту 12 пікселів.

В разі, якщо у вас виникла необхідність стилізувати HTML-елемент, куди ефективнішим і кращим рішенням буде використовувати можливості мови CSS, замість використання style.

Наприклад, розглянемо наступний фрагмент HTML-документу:

В ньому декілька HTML-елементів - заголовок (h1) та два абзаци (p).

Спробуємо за допомогою CSS-коду трішки оформити ці елементи. Але спочатку потрібно підключити зовнішній файл з розширенням css до нашої сторінки index.html. Для цього застосуємо спеціальну конструкцію підключення **зовнішніх стилів**.

Виконується підключення за допомогою **елемента link**, який повинен розташовуватися всередині елемента **head.**

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

Зустрівши в HTML-документі цей тег, браузер завантажить із сайту CSS-файл і застосує стилі, що містяться в ньому. Шлях до файлу вказують в атрибуті href (у нашому випадку це styles.css).

Створимо відповідний файл за допомогою системи PhpStorm та виконаємо відповідне підключення. Тут слід відмітити, що таких файлів стилів може буде безліч, тому зразу необхідно створити відповідний каталог для стилів - css.

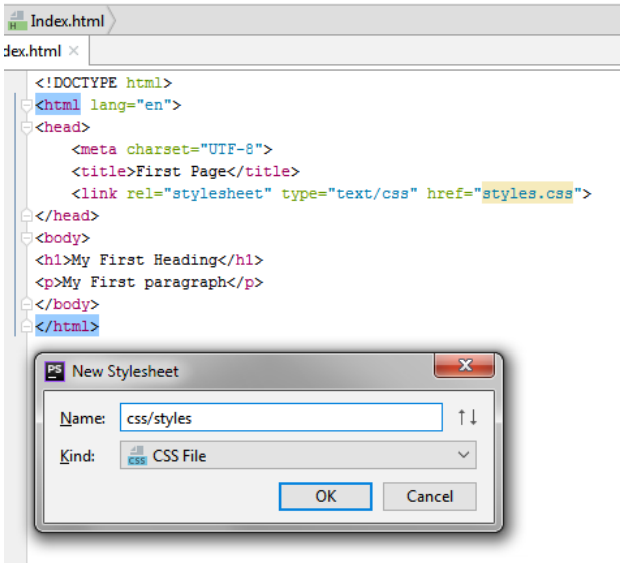


Рисунок 1.19 – Створення файлу стилів style.css

Виконаємо підключення стилів до файлу index.html:

```
<link rel="stylesheet" type="text/css" href="css/styles.css">
```

Код підключення наведено нижче.

```
<!DOCTYPE html>
<htmllang="en">
```

```
	<head>
	<meta charset="UTF-8">
	<title>FirstPage</title>
	<linkrel="stylesheet" type="text/css"
href="css/styles.css">
	</head>
	<body>
	<h1>My First Heading</h1>
	<h1>My two Heading</h1>
	<p>My First paragraph</p>
	</body>
	</html>
```

Таким чином ми даємо інструкцію браузеру, яким має бути шрифт для h1-заголовків й абзаців на цій сторінці, а також вказуємо колір для абзацу.

Якщо узагальнити, то CSS-правила складаються із **селектора** та **блоку оголошень**. Схематично це виглядає так:

```
	певний-селектор {
	властивість-1: значення-1;
	властивість-2: значення-2;
	...
	}
```

Селектор вказує на HTML-елемент(и), які ми намагаємось стилізувати. Тоді як в блоці оголошень ми зазначаємо CSS-властивості цих елементів і задаємо їм певні значення.

Блок оголошень слід "огорнути" фігурними дужками. Всередині цих дужок можна вказати одне або декілька оголошень, розділених між собою крапкою з комою.

Застосуємо відповідні стилі й подивимося не результат виконання. Для цього в файлі style.css використаємо відповідні стилі:

```
h1 {
      font-family: Georgia, serif;
         text-transform:uppercase;
       }
p {
      color:#f46155;
         font-family: "TimesNewRoman", Times, serif;
         text-align: justify;
         font-size: 20px;
      }
```

Результат виконання наведено на рисунку 1.20.

Рисунок 1.20 – Результат виконання підключених файлів стилів

Таким чином, для того, щоб застосувати стилі до HTML-документу, ми можемо обрати один з чотирьох способів, або ж комбінувати їх:

– застосувати зовнішні стилі за допомогою елемента link;

– додати CSS-блок за допомогою елемента style;

– вказати стиль конкретному HTML-елементу за допомогою HTML-атрибуту style (inline-стилі);

– використати @import (правило @import дозволяє імпортувати (завантажити) вміст CSS-файлу в поточну стильову таблицю).

Найчастіше використовується метод підключення за допомогою елемента *link*, який повинен розташовуватися всередині елемента *head*.

Селектори класів

Класи CSS - це чудовий інструмент, який розширює можливості створення стилів в рази. Для кращого розуміння ми будемо розглядати все на прикладах.

Отже, трохи вище ми застосували стиль для всіх тегів <h1> та <p> на веб-сторінці – абзац має шрифт TimesNewRoman і червоний колір , а для заголовка <h1> встановлено, що він буде оформлений великими літерами (text-transform:uppercase) і шрифт (font-family: Georgia).

У випадку, якщо знадобиться змінити колір одного з тегів <h1> на зелений, наприклад, то на допомогу приходять селектори класів. Все що необхідно зробити, це створити стиль, де селектор – придумане будь-яке ім'я класу. Наприклад, назвемо клас .greentext (назва класу починається крапкою перед назвою) і запишемо правило:

```
.greentext {
color: green;
}
```

Але це ще не все. Тепер, щоб змінити колір для одного з тегів <h1> на сторінці, потрібно відредагувати HTML-документ, застосувавши клас greentext до необхідного нам тегу. Записується це так:

```
<h1 class = "greentext"></ h1>
```

Створений клас можна застосовувати до будь-яких елементів веб-сторінки. Ви можете надавати стиль не тільки цілим заголовкам й абзацам, а й окремим фрагментам сторінки, наприклад, словам (використовуючи тег привласнюючи йому клас).

Запам'ятайте кілька правил написання класів:

- CSS перед назвою селектора класу обов'язково ставиться крапка (але при присвоєнні класу в HTML-документі ця крапка не потрібна);

- в назві класів можна використовувати тільки букви латинського алфавіту, дефіс, підкреслення, цифри;

- назва класу завжди повинна починатися з літери (правильні варіанти назв: .intro, .img-border, .nav_menu_01; неправильні: .2color,.-link, ._divider);

- назви класів CSS чутливі до регістру, тому класи на зразок .review і .Review будуть сприйматися як два окремих.

Селектори ID

Ідентифікатор визначає унікальну назву елемента. Записується він майже так само, як і клас, тільки в CSS замість крапки ставиться символ решітки #:

```
#footer {
width: 100%;
}
```

У HTML-документі ідентифікатор присвоюється за допомогою атрибута id:

```
<div id = "footer"></div>
```

Існує кілька відмінностей між ідентифікатором і класом:

-ID - це унікальна назва елемента на веб-сторінці, яка повинна зустрічатися на ній тільки один раз (наприклад, шапка сайту і підвал: id = "header" і id = "footer"), в той час як клас може назначатися до кількох елементів з метою відрізняти їх від інших;

- ідентифікатори зручні для JavaScript-розробників, оскільки дозволяють отримати швидкий доступ до елементу DOM з скриптів (багато в чому саме тому необхідно, щоб ID зустрічався на сторінці лише один раз);

- кожне правило CSS має свій пріоритет (від пріоритету залежить, яке з правил отримає більш високий пріоритет при виконанні). Ідентифікатор має більшу вагу, ніж клас, тому, якщо елементу присвоєно і ID, і клас, перевага віддається ID. приклад:

```
<pid = "text" class = "content"> текст </ p>
#text {
color: yellow;
}
.content {
color: blue;
}
```

У ID вищий пріоритет, тому колір тексту буде жовтим (yellow).

За допомогою ідентифікаторів можна ставити якірні посилання на певні елементи веб-сторінки. Досить привласнити цьому елементу id:

<h3 id = "description"> Опис </ h3>

І потім дати на нього посилання виду: http://site.com/category/page/#description.

Групові селектори

Четвертий тип селекторів CSS – групові селектори. Уявіть ситуацію, що вам необхідно зробити шрифт жирним для декількох елементів веб-сторінки - p, h1, h2, h3. Можна було б записати цей стиль окремо для кожного елемента:

```
p {
 font-weight: bold;
}
h1 {
 font-weight: bold;
}
h2 {
 font-weight: bold;
```

```
}
h3 {
 font-weight: bold;
}
```

Але такий код займає більше часу й місця, ніж це необхідно. Адже все можна записати набагато простіше й коротше, перерахувавши імена через кому та створивши таким чином груповий селектор:

```
p, h1, h2, h3 {
 font-weight: bold;
}
```

Звичайно, в перерахуванні можуть брати участь не тільки селектори тегів, а й класи, та ідентифікатори. А якщо з якоїсь причини вам необхідно створити стиль абсолютно для всіх елементів веб-сторінки, можна скористатися універсальним селектором CSS, для позначення якого використовується символ зірочки *:

```
* {
font-family: Geneva, Arial, sans-serif;
}
```

За наступним посиланням можна отримати більш детальну інформацію з застосування ідентифікаторів та класів[3][4][15]:

https://habr.com/ru/post/303174/

https://developer.mozilla.org/en-US/docs/Web

CSS Flexbox як інструмент побудови контейнерів сайту

CSS Flexbox (Flexible Box Layout Module) - модуль макета гнучкого контейнера - являє собою спосіб компонування елементів, в основі лежить ідея осі.

Flexbox складається з гнучкого контейнера (flex container) і гнучких елементів (flex items). Гнучкі елементи можуть

вибудовуватися в рядок або стовпчик, а вільний простір розподіляється між ними різними способами.

На даний час Flexbox вже підтримується практично усіма сучасними браузерами, включаючи Android та iOS.

Хоча за допомогою Flexbox технічно можна зверстати повний макет для сайту, він не призначений виключно для цієї мети. Швидше, він краще підходить для стилізації окремих контейнерів, таких як контейнер основного контенту, бічна панель (сайдбар), хедер та інших подібних розділів. Все ж сітки краще підходять для створення всього макету.

CSS Flexbox – це CSS3 веб модуль. Flex дозволяє автоматично організовувати відповідні елементи в контейнері залежно від розміру екрана (або пристрою) та дає змогу використовувати більш ефективний спосіб верстання, вирівнювання й розподілу вільного місця між елементами у контейнері, навіть коли їх розмір невідомий і/або динамічний.

Однією з найважливіших особливостей Flexbox є його здатність формуватися на основі його середовища перегляду. Контейнери flex можна регулювати за розміром, (як збільшувати, так і зменшувати) щоб уникнути надмірної монополізації простору. Більше того, цей лейаут краще обмежує потік контенту, ніж, наприклад, блокові та вбудовані типи дисплею, які, як правило, односпрямовані. Дійсно, можна не лише задати напрямок потоку флексу на рівні стилю, наприклад, праворуч, ліворуч, вгору або вниз; окремі елементи в такому контейнері також можуть бути автоматично перевпорядковані та перебудовані відповідно до наявного макета.

На рисунку 1.21 представлено загальну схему модуля Flexbox CSS.

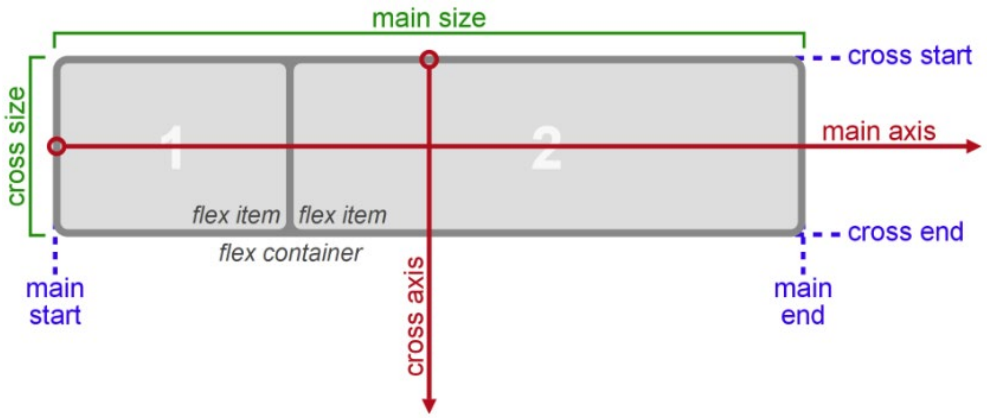


Рисунок 1.21 – Модуль Flexbox

Модуль Flexbox дозволяє вирішувати наступні завдання:

– Розташовувати елементи в одному з чотирьох напрямків: зліва направо, справа наліво, згори донизу або знизу догори.

– Перевизначати порядок відображення елементів.

– Автоматично визначати розміри елементів таким чином, щоб вони вписувалися в доступний простір.

– Вирішувати проблему з горизонтальним і вертикальним центруванням.

– Переносити елементи всередині контейнера, не допускаючи його переповнення.

– Створювати колонки однакової висоти.

– Створювати притиснутий до низу сторінки підвал сайту.

– Flexbox вирішує специфічні завдання - створення одновимірних макетів, наприклад, навігаційної панелі, оскільки flex-елементи можна розміщувати тільки по одній з осей.

Термінологія flexbox

Нижче в таблиці 1.5 наведено головну термінологію даної технології.

Таблиця 1.5 - Термінологія Flexbox

Flex container	Батьківський елемент, що містить всі фрагменти контенту. Використовуючи властивість CSS, контейнер можна визначити як flex або inline-flex.
Flex item	Будь-який дочірній елемент, що зберігається в флекс контейнері, вважається гнучким. Будь-який текст у межах контейнера загортається.
Axes((main axis/ cross axis)	Кожна гнучка коробка містить дві вісі: основну та поперечну. Основна вісь - вісь, на якій елементи вирівнюються відповідно один одного. Перехресна вісь перпендикулярна головній вісі.
Flex-direction	Встановлює основну вісь. Можливі аргументи left, right, center, space-between, space-around.
Justify-content	Визначає, як вміст розміщується на основній вісі на поточній лінії. Додаткові аргументи: ліворуч, праворуч, в центрі, пробіл, простір навколо**.**
Align-items	Визначає за замовчуванням, як гнучкі елементи розміщуються по поперечній вісі на кожній лінії.
Align-content	Визначає за замовчуванням вирівнювання ліній поперечної вісі.
Align-self	Визначає, як розташовується один елемент уздовж вісей. Це перевизначає будь-які значення за замовчуванням, встановлені командою «align-items».
Directions	Інструмент визначає, з чого почати розміщення елементів у флекс контейнері, починаючи з "main-start" та переходячи до "main-end". "Cross-start/cross-end інструмент визначає, де флекс лінії

	наповнюються контентом від "cross-start" до "cross-end".
Order	Розміщує елементи по групах і визначає, у якому порядку вони повинні бути поміщені в контейнер.
Flex-flow	Поєднує flex-direction і flex-wrap.
Lines	Флексові елементи можуть бути розміщені на одинарній лінії або на кількох рядках, як це визначено властивістю гнучкого пакетування, яка контролює як напрямок поперечної вісі, так і спосіб стягування ліній у контейнері.
Dimensions	«main size» і «cross size» - це, по суті, висота і ширина флекс контейнера, і стосуються головної та поперечної осей відповідно.
Flex-basis	Дозволяє визначати початкові ширину та висоту контейнера

Створення флекс-контейнера

Переведення контейнера в стан «флекс» є досить простим. Все, що необхідно - це змінити властивість дисплею на flex чи inline-flex як показано: display: flex; чи: display: inline-flex;

Після зміни властивості display на одну з вказаних вище, елемент стає флекс-контейнером, а вміст - флекс-контентом. Зміна властивості дисплею на «flex» перетворює контейнер на блоковий елемент, тоді як зміна на «inline-flex» перетворює контейнер на інлайн-елемент.

Код приклад побудови макету наведено нижче:

```
<div class="content">
<div class="blok">Блок інформації №1</div>
<div class="blok">Блок інформації №2</div>
```

```
<div class="blok">Блок інформації №3</div>
<article>Тут основний контент.</article>
</div>
```

Так щоб створити flexbox, нам потрібно визначити властивість display для потрібного контейнера. У нашому прикладі це властивість для батьківського елемента:

```
.content {
display: flex;
 }
```

Цей крок створить Flexbox на рівні блоку. В якості альтернативи (якщо потрібно застосувати inline стилі) можливо використати inline-flex замість flex.

Тепер додамо трохи CSS стилів, щоб протестувати роботу нашого створеного Flexbox.

Є багато властивостей, які можливо включити для Flexbox, щоб створити адаптивний макет.

Щоб розмістити флекс елементи в потрібному порядку, досить додати властивість order і короткий запис властивості flex.

```
.blok { order: 1;
flex: 1 1 30%;
          }
article { order: 2;
flex: 1 1 auto;
          }
```

Властивості order передається номер, щоб повідомити браузеру, який флекс елемент відображати перед іншими. Для order з номером 1 – флекс елемент буде показуватися першим. Для 2 – другим, для 3 – третім, і так далі. Також можливо використовувати негативні числа. Якщо ви раптом зрозумієте при верстуванні, що потрібно додати флекс елемент перед першим, який вже розміщений, ви можете встановити новому флекс елементу властивість order: -1;

Короткий запис flex складається з трьох властивостей:

– flex-grow – визначає, яку частину вільного простору може зайняти контейнер, в співвідношенні з іншими контейнерами. Це може бути тільки позитивне число;

– flex-shrink – властивість, яка визначає фактор стиснення flex елемента. Flex елементи будуть заповнювати контейнер за значенням flex-shrink, коли стандартна ширина flex елементів ширше, ніж flex контейнер. Негативні числа не мають впливу;

– flex-basis – початковий розмір флекс елемента до застосування будь-яких флекс розмірів і перед тим, як буде зайнято вільний простір, або при нестачі місця. Може бути задано в пікселях або відсотках.

Деякі пояснення до запису властивостей:

- flex: initial або flex: 0 1 auto – цей параметр робить розмір флекс елемента відносним вмісту, який знаходиться всередині нього. Він збільшується, якщо багато контенту й стискається, якщо контенту небагато.

- flex: auto або flex: 1 1 auto – будь-який з цих параметрів дозволяє флекс елементу стискатися і збільшуватися в міру необхідності, щоб підлаштовуватися під будь-який розмір екрану.

- flex: none або flex: 0 0 auto – це відключає гнучкість розміру й встановлює розмір флекс елемента фіксованим і нерегульованим для користувача при будь-якому розмірі екрану.

- Відносний гнучкий розмір з **flex:1%px**. Позитивне число спочатку встановлює частину вільного місця, яке флекс елемент займає щодо інших флекс елементів. Другий номер дозволяє зменшити розмір елемента на менших екранах. Третє значення в пікселях (або відсотках) встановлює початковий розмір елемента flex, але майте на увазі, що це значення також відключає властивість **flex-**

basis, що означає, що цей початковий розмір не гарантується. Якщо для відображення цього початкового розміру є достатньо місця, то він показується, але якщо місця недостатньо, то він не відображається. Це особливо вірно, якщо порядок показу елемента знаходиться нижче в списку, а інші попередні елементи вище в порядку займають більшу частину простору. Прикладом цього параметра буде **flex: 2 1 0%.**

Щоб будь-який з цих параметрів спрацював, потрібно спочатку встановити розмір flexbox із зазначенням висоти і ширини.

Ґрунтуючись на прикладі CSS-стилів вище, ось як можуть виглядати стилі CSS для flexbox з додаванням розміру:

```
.content { display: flex;
width: 75%;
height: 450px; }
```

За допомогою цих стилів всі флекс-елементи в прикладі відображаються в одному рядку. Оскільки ми хочемо відобразити флекс-елементи в один ряд, а основний контент під ними, мені потрібно додати властивість flex-flow і встановити його значення для перенесення вмісту в рядках. Щоб виконати цю функцію, нам потрібно додати цю властивість для flexbox контейнера коли один або кілька флекс елементів займають собою всю ширину flexbox, інші флекс елементи, що йдуть після них, переміщуються на новий рядок.

```
.content { display: flex;
flex-flow: row wrap; /* флекс елементи заповнюють собою рядок (row) */
width: 75%; height: 450px; }
```

Далі, щоб розуміти, де який блок, і які він має розміри, додамо кілька додаткових стилів. Також, весь текстовий контент ми вкладемо в тег «p» і додамо цьому тегу кілька стилів:

```
<div class="content">
```

```
<div class="blok"><p>Блок інформації №1</p></div>
<div class="blok"><p>Блок інформації №2</p></div>
<div class="blok"><p>Блок інформації №3</p></div>
<article><p>Тут контент</p></article>
</div>
```

Нижче наведено відповідні стилі:

```
.content { display: flex;
flex-flow: row wrap;
/* флекс елементи заповнюють собою рядок (row) */
width: 75%;
height: 450px;
margin: auto; background-color: #333; }
 .blok {
order: 1;
 flex: 1 1 30%; background-color: #ccc; margin:
5px; }
article { order: 2;
    flex: 1 1 auto;
    background-color: #bbb;
    border: 5px solid #333;
    padding: 7px; }
 p {
    color: #fff;
    padding: 15px;
font-size: 22px;
 }
```

!!!Не забувайте, що якщо додати властивість min-width flex-боксу або флекс елементам, це може привести до того, що Flexbox не працюватиме належним чином. Також, для адаптивності дизайну, краще використовувати ширину основного контейнера з використанням відсотків. Якщо задати жорсткі значення в пікселях, для маленьких екранів адаптивність не спрацює.

Результат наведеного коду представлено на рисунку 1.22.

Рисунок 1.22 – Застосування властивості flex для побудови контейнера

Далі розглянемо приклад створення простої адаптивної сторінки за допомогою властивості *display:flex* та з використанням медіа запитів *@media screen:*

Код файлу index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Flexbox</title>
    <link rel="stylesheet" type="text/css"
href="css/styles.css">
</head>
<body>
<section class="sample">
    <div class="container">
        <div class="row">
            <div class="item">1</div>
            <div class="item">2</div>
            <div class="item">3</div>
            <div class="item">4</div>
            <div class="item gr1">5</div>
            <div class="item gr2">6</div>
```

```
                    <img src="../img/cat2.jpg">
                </div>
            </div>
        </section>
        <ul class="flex-container">
            <li class="flex-item flex1">7</li>
            <li class="flex-item flex2">8</li>
        </ul>
        </body>
        </html>
```

Тепер напишемо код файлу стилів styles.css адаптивної сторінки.

```
        body {
            background-color:#f5fffa;
        }
        .row {
            /*max-width:1280px;
           margin: 0 auto;
           padding: 0 30px; */
            display:flex; /*ставимо елементи в рядок і кожний елемент за розміром буде дорівнювати контенту .item flex-basis:auto;*/
        }
        .item {
            padding: 20px 0px;
            text-align:center;
            background-color:#bfbe58;
            width:25%; /*розтягнемо порівну на всю ширину*/
        }
```

Якщо div добавити то відповідно вони займуть 100%/n, а ми хотіли по 25 на 4, а це все тому, що стоїть властивість за замовчуванням перенос не дозволено й всі вони будуть в одній лінії – ужимаються, тому змінимо це, тепер кожен елемент займе 25% ширини.

```
.row {
    flex-wrap:wrap;    /*контейнер    з    переносом
рядків*/
}
/*але в елементвів сітки повинні бути відступи*/
.item {
    margin:15px; /*відступи між елементами*/
}
```

Але 4 наші елементи тепер не поміщаються в рядок, тому що формула до жорстко прописаної ширини + margin 25%*4 ... + 15px*8 відступів, в фреймоврку bootstrap робиться сітка на властивостях padding, але ми застосовуємо функцію css, а саме з % віднімимо піксельну.

```
.item {
    width: calc(25% - 30px);
}
```

Тут ми бачимо, що по краях отримали непотрібні відступи в батьківському елементі, властивість margin «обнулять» не зручно й тому всі використовують від’ємні маржини притиснувши елементи один до одного.

```
.row {
    margin:  0  -15px;  /*компенсація відступів по
краях*/
}
```

Але з’явилося протиріччя з рядком 7 коду це не допустимо й відповідно властивості повинні бути в додатковій обгортці й саме задамо таку властивість для .container.

```
.container {
    max-width:1280px;
    margin: 0 auto;
    padding: 0 30px;
    border: 1px solid chocolate;
}
```

Далі спробуємо добавити відступи, наприклад, для тексту всередині блоків:

```
.item {
    padding: 20px 15px;
}
```

І знову вся наша система порушена, а все тому, що діє властивість за замовчуванням бокс-размір розраховується як width + padding + margin, застосуємо:

```
.item {
    box-sizing:border-box;
}
```

Це є парадигма, щоб padding і margin розраховувалися всередині елемента для колонок. А якщо елемент потрібно розташувати не на чотирьох а на довільну кількість колонок, то можливо зробити таким чином:

Число 3 це кількість колонок, що буде займати.

```
.item {
    width: calc(100% / 12 * 3 - 30px);
}
```

Далі зробимо сторінку адаптивною. Для цього потрібно підключити медіа запити:

```
@media screen and (max-width: 920px){
    .item {
         width: calc(100% / 12 * 6 - 30px);
    }
}
@media screen and (max-width: 520px){
    .item {
         width: calc(100% / 12 * 12  - 30px);
    }
}
.row {
    justify-content:center;
```

```
    }
    img {
        width:100%;
        height:100%;
    }
    /*flex-grow: коефіцієнт росту;
        flex-shrink: коефіцієнт стискання*/
    .flex-container {
        padding: 0;
        margin: 0;
        list-style: none;
        display: flex;
        /*flex-direction:column;*/
    }
    .flex-item {
        background: tomato;
        padding: 10px;
        border: 5px solid red;
        color: white;
        font-weight: bold;
        font-size: 2em;
        text-align: center;
    }
    .flex1 { flex: 1 1 300px;
        order:1;
    }
    /*flex-grow:1, flex-shrink: 1, flex-basis: 300px.*/
    .flex2 { flex: 1 1 300px;
        order:-1;
    }    /*flex-grow:2,  flex-shrink:  2,  flex-basis:
300px.*/
    /*Розміри flex-grow при 1 : 2  , відповідно 1/3 і
2/3 на елемент 0 - заборона росту */
```

Результат виконання представлено на рисунках 1.23, 1.24 в залежності від зміни розміру екрану пристроїв.

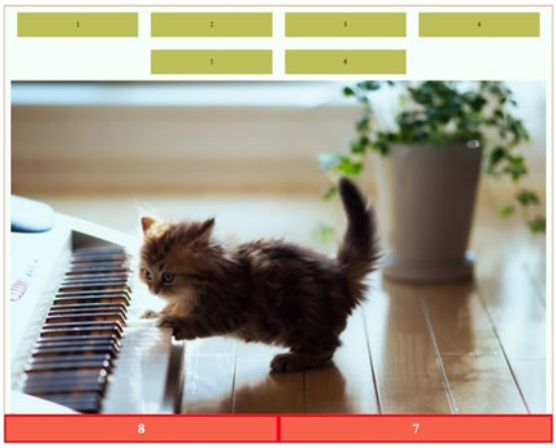


Рисунок 1.23 – Media screen max-width: 920px

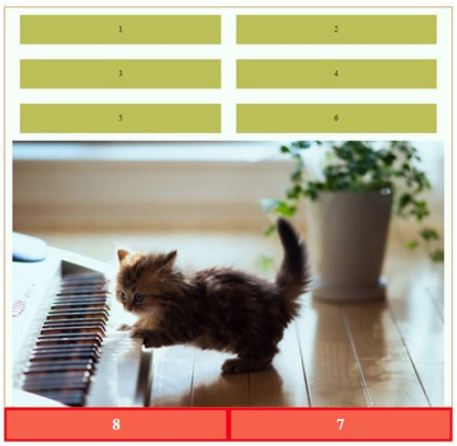


Рисунок 1.24 – Media screen max-width: 520px

1.9 CSS-препроцесори

CSS-препроцесори (наприклад, Less, Sass і Stylus) розширюють можливості стандартного CSS. Дозволяють використовувати змінні, вкладені правила, mixins, вбудований імпорт тощо. Що в свою чергу допомагає тримати CSS-стилі добре організованими, робить їх більш зрозумілими й компактними. CSS-препроцесори ще називають динамічними мовами стилів. Браузер не може інтерпретувати синтаксис Less/Sass/Stylus, тому вихідний код повинен бути скомпільований в CSS одним з трьох способів:

- в браузері за допомогою JavaScript-інструменту;
- на стороні сервера з використанням мови програмування;
- локально на машині за допомогою програми.

Sass використовує розширення .scss (напр., style.scss), Less - .less, a Stylus -.styl.

Писати код за допомогою CSS препроцесора, використовуючи його потужні можливості - справді захоплююче.

Розглянемо деякі можливості та переваги використання трьох різних препроцесорів – Sass, Less, та Stylus.

Препроцесор компілює код в CSS, що працює однаково у всіх браузерах.

CSS препроцесор – це програма, яка компілює написаний код (з використанням спеціального синтаксису) в чистий CSS код. При цьому, препроцесор надає розробнику нові можливості та функції. Препроцесори мають тисячі особливостей, і ми розглянемо деякі найбільш оригінальні властивості препроцесорів.

Синтаксис

Найважливіше, при використанні CSS препроцесора - це розуміти синтаксис. На щастя, він є однаковий для всіх трьох CSS препроцесорів, які ми розглядаємо.

Sass & Less

Як Sass, так і Less використовують стандартний CSS синтаксис. Це робить їх надзвичайно простими для розуміння. Sass використовує розширення .scss, Less – .less. Простий Sass чи Less файл виглядає так:

```
/* style.scss or style.less */
h2 {
  color: #0982C1;
}
```

Це є звичайний CSS код, який чудово компілюється, як в Sass, так і в Less. Важливо зауважити, що Sass також використовує старий синтаксис, при якому опускаються коми та фігурні дужки:

```
h2
 color: #0982c1
```

Синтаксис Stylus є більш різноманітним. Використовуючи розширення файлу .styl, Stylus допускає стандартний CSS

синтаксис, але також пропонує інші варіанти, де дужки, двокрапки та крапки з комою є необов'язковими. Наприклад:

```
/* style.styl */
h1 {
  color: #0982C1;
}
/* без дужок */
h1
  color: #0982C1;
/* без крапок та двокрапок */
h1
  color #0982C1
```

Також можна використовувати декілька варіантів в одному стилі, тобто наступний код буде скомпільований без помилок:

```
h1 {
  color #0982c1
}
h2
  font-size: 1.2em
```

Змінні

Змінні можна оголошувати й використовувати у всій таблиці стилів. Вони можуть приймати будь-яке, використовуване в CSS значення (наприклад, кольори, цифри чи текст).

Sass

Sass змінні оголошуються з символом $ перед ім'ям, команда привласнення – двокрапка(:). Наприклад:

```
$mainColor: #0982c1;
$siteWidth: 1024px;
$borderStyle: dotted;
body {
  color: $mainColor;
  border: 1px $borderStyle $mainColor;
  max-width: $siteWidth;
}
```

Less

Less змінні точно такі, як і Sass, але назва змінної починається з символу «@» :

```
@mainColor: #0982c1;
@siteWidth: 1024px;
@borderStyle: dotted;
body {
  color: @mainColor;
  border: 1px @borderStyle @mainColor;
  max-width: @siteWidth;
}
```

Stylus

Оголошуються Stylus змінні без всяких символів перед ім'ям, хоча дозволяється використання символу $. Не вимагається наявність крапки з комою в кінці, знак привласнення – дорівнює (=). Важливо зауважити, що змінні з символом @ Stylus компілює, але вони не є дійсні. Такого краще уникати.

Компільований CSS

При компіляції будь-якого, з вище наведених прикладів, ми отримаємо однаковий CSS код. Якщо застосувати фантазію, можна побачити наскільки корисними при написанні CSS бувають змінні. Наприклад, щоб змінити один колір в кількох місцях, вам більше не потрібно буде витрачати багато часу, щоб знайти й переписати його в кожному місці, а оголосивши на початку змінну з кольором, можна буде це зробити всього лише за одне виправлення. Приклад CSS після компіляції:

```
body {
  color: #0982c1;
  border: 1px dotted #0982c1;
  max-width: 1024px;
}
```

Вкладеність

При використанні вкладеності в звичайному CSS, доводиться багато разів переписувати ім'я батьківського елемента. Це доволі незручно.

```
section {
  margin: 10px;
}
section nav {
  height: 25px;
}
section nav a {
  color: #0982C1;
}
section nav a:hover {
  text-decoration: underline;
}
```

Препроцесор вирішує цю проблему: ми можемо писати стиль дочірнього елемента всередині дужок батьківського компонента.

Sass, Less, & Stylus

Всі три препроцесори використовують однаковий синтаксис для вкладених селекторів. Також, за допомогою символу & можна посилатися на батьківський селектор.

```
section {
  margin: 10px;
  nav {
    height: 25px;
    a {
      color: #0982C1;
&:hover {
        text-decoration: underline;
      }
    }
```

```
  }
}
```

Скомпільований CSS код з прикладу вище:

```
section {
  margin: 10px;
}
section nav {
  height: 25px;
}
section nav a {
  color: #0982C1;
}
section nav a:hover {
  text-decoration: underline;
}
```

Домішки

Домішки це функції, в яких можна окремо виділити певні властивості й повторно використовувати їх кілька разів у всьому CSS файлі. Коли домішка викликається з CSS селектора, препроцесор розпізнає аргумент домішки, й стилі, що розташовані всередині домішки, застосовуються до селектора.

Sass:

```
@mixin error($borderWidth: 2px) {
  border: $borderWidth solid #F00;
  color: #F00;
}
.generic-error {
  padding: 20px;
  margin: 4px;
  @include error();
}
.login-error {
  left: 12px;
  position: absolute;
```

```
  top: 20px;
  @include error(5px);
}
```

Less:

```
.error(@borderWidth: 2px) {
  border: @borderWidth solid #F00;
  color: #F00;
}
.generic-error {
  padding: 20px;
  margin: 4px;
  .error();
}
.login-error {
  left: 12px;
  position: absolute;
  top: 20px;
  .error(5px);
}
```

Stylus:

```
error(borderWidth= 2px) {
  border: borderWidth solid #F00;
  color: #F00;
}
.generic-error {
  padding: 20px;
  margin: 4px;
  error();
}
.login-error {
  left: 12px;
  position: absolute;
  top: 20px;
  error(5px);
}
```

Всі три препроцесори скомпілюють вищенаведений код в однаковий CSS файл.

```
.generic-error {
  padding: 20px;
  margin: 4px;
  border: 2px solid #f00;
  color: #f00;
}
.login-error {
  left: 12px;
  position: absolute;
  top: 20px;
  border: 5px solid #f00;
  color: #f00;
}
```

Спадкування

При написанні звичного CSS коду, ми могли б використовувати наступний код, щоб застосувати той самий стиль до кількох елементів відразу:

Корисні посилання CSS :

p,

ul,

ol {

}

Це працює чудово, але якщо окремо потрібно буде додати стиль до деяких елементів, необхідно буде створити ще по одному селектору для кожного. Це складніше підтримувати і код є доволі незрозумілим. Щоб уникнути цього, можна використовувати спадкування. Спадкування - це здатність одних CSS селекторів успадковувати властивості інших.

Sass & Stylus

```
.block {
  margin: 10px 5px;
  padding: 2px;
}
p {
  @extend .block; /* Успадкує стиль з '.block' */
  border: 1px solid #EEE;
}
ul, ol {
  @extend .block; /* Успадкує стиль з '.block' */
  color: #333;
  text-transform: uppercase;
}
```

Компільований CSS (Sass & Stylus)

```
.block, p, ul, ol {
  margin: 10px 5px;
  padding: 2px;
}
p {
  border: 1px solid #EEE;
}
ul, ol {
  color: #333;
  text-transform: uppercase;
}
```

Less не підтримує спадкування, як Sass та Stylus. Замість додавання декількох селекторів до одного набору властивостей, він розглядає спадкування, як домішку без аргументів та імпортує стилі в кожен селектор. Недоліком є те, що властивості повторюються в компільованому CSS.

Імпорт

У CSS, імпортування є несхвальним, оскільки воно вимагає кількох HTTP запитів. Проте, імпорт, використовуючи препроцесори, працює зовсім інакше. Якщо ви імпортуєте файл, використовуючи будь-який з трьох препроцесорів, вони включать імпортований файл в CSS код протягом компіляції, створюючи при цьому лише один CSS файл. Майте на увазі, що імпортування звичайного .css файлу виконується так: @import ‘file.css’;. Домішки й змінні також можна імпортувати та використовувати в основному коді. Імпортування надає можливість створювати структури файлів.

Sass, Less, & Stylus:

```
body {
  background: #EEE;
}
@import "reset.css";
@import "file.{type}";
p {
  background: #0982C1;
}
```

Компільований CSS:

```
@import "reset.css";
body {
  background: #EEE;
}
p {
  background: #0982C1;
}
```

Операції

Препроцесор надає можливість використання математичних розрахунків в CSS коді.

Sass, Less, & Stylus:

```
body {
```

```
  margin: (14px/2);
  top: 50px + 100px;
  right: 100px - 50px;
  left: 10 * 10;
}
```

Практичне застосування

Особливо необхідно використовувати препроцесор при створенні Vendor Prefixes – це економить багато часу й зусиль. Приклад реалізації:

Sass

```
@mixin border-radius($values) {
  -webkit-border-radius: $values;
     -moz-border-radius: $values;
          border-radius: $values;
}
div {
  @include border-radius(10px);
}
```

Less

```
.border-radius(@values) {
  -webkit-border-radius: @values;
     -moz-border-radius: @values;
          border-radius: @values;
}
div {
  .border-radius(10px);
}
```

Компільований CSS буде виглядати наступним чином:

```
div {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
```

```
    border-radius: 10px;
}
```

Висновки

Кожен CSS препроцесор (Less[16], Stylus[17] і Sass[18]), має свій унікальний спосіб досягнення тої самої мети – використання корисних, функцій CSS, що не підтримуються, зберігаючи сумісність браузера й чистоту коду.

Потрібно спробувати якомога більше препроцесорів, щоб вибрати найбільш ефективний для себе.

Повний список функцій препроцесорів можна знайти в документації за адресами:

http://lesscss.org/

https://stylus-lang.com/

https://sass-lang.com/

Контрольні запитання:

1. Що представляє собою процес веб-розробки?
2. Що представляє собою HTML?
3. Що представляє собою CSS?
4. Як використовують сервери механізм Cookies?
5. Що називають фронтендом?
6. Що представляє собою бекенд?
7. Що представляє собою фреймворк?
8. Які вимоги до сучасного редактора коду?
9. Що представляють собою HTML-теги?
10. Що представляє собою елемент HTML?
11. Що означають елементи: <html>, <body>, <p>?
12. Що означає DOCTYPE у HTML-документі?
13. Що представляє собою заголовок HTML-документа?
14. Яку частину називають тіло HTML-документа?

15. Що визначають <header>, <nav>, <section>, і <footer> елементи?

16. Яка основна мета використання семантичних тегів?

17. Що представляють собою HTML-форми?

18. Які методи передачі даних форми використовуються?

19. За допомогою якого елемента виконується підключення стилів до файлу?

20. З чого складаються CSS-правила?

21. Як визначаються селектори класів?

22. Що представляють собою селектори ID?

23. Що представляє собою CSS Flexbox?

24. Які функції виконують CSS-препроцесори?

РОЗДІЛ 2
ФРЕЙМВОРК BOOTSTRAP 4

2.1 Знайомство з Bootstrap

Створення свого власного міні-фреймворка - це задача, з якою стикається кожен фронтенд-розробник. Зазвичай він складається з тих правил і функцій, які повторювалися у всіх недавніх проектах. Після того, як вони будуть зібрані в одну бібліотеку, їх буде легше разом підключити до нового проекту й користуватися готовими рішеннями. У бібліотеку може входити сітка з колонками, в яких знаходиться будь-який контент, стандартні правила для спрайтів, зовнішніх відступів, заголовків і т.д.

Якщо в розробці проекту бере участь кілька професійних фронтенд-розробників, то подібні фреймворки потрібно стандартизувати. Якщо так, то перевага віддається вже стандартизованим фреймворками. Одним із найбільш популярних зараз є - фреймворк Bootstrap[7][8].

Створений командою розробників в компанії Twitter, Bootstrap спочатку використовувався для власних продуктів компанії і був названий «Twitter Bootstrap», але потім став самостійним рішенням. Ось чому слово «Twitter» було в подальшому відкинуто.

Bootstrap - це CSS / HTML фреймворк для створення вебсайтів. Іншими словами, це набір інструментів для створення веб-макета. У нього є ряд переваг, що робить його найпопулярнішим серед інших подібних фреймворків.

Переваги Bootstrap:

– Швидкість роботи - створення макетів з Bootstrap займає менше часу завдяки великому набору готових до використання елементів.

– Гнучкість - додавання нових елементів не порушує загальну структуру завдяки сітці, що динамічно змінюється.

– Легка змінюваність - правка стилів досягається завдяки додаванню нових CSS правил, які скасовують існуючі. При цьому, вам не потрібно використовувати атрибути типу !Important.

– Велика кількість шаблонів (буде розглянуто далі).

– Величезне співтовариство прихильників / розробників.

– Широкий спектр застосування - Bootstrap використовується для створення тем майже для будь-якої CMS (Magento, Joomla, WordPress або будь-якої іншої), включаючи односторінкові Лендінги.

– Зрозуміла та доступна офіційна документація.

– Bootstrap особливо популярний серед тих, хто займається створенням так званих «Лендингів» (посадочних / цільових сторінок).

Шаблони Bootstrap

Шаблони в Bootstrap дозволяють вам змінювати вже модифіковані елементи під ваші потреби. Багато розробників пропонують використовувати їх власні шаблони (платні або безкоштовні). Підключаються шаблони Bootstrap дуже просто: після підключення самого Bootstrap додається виклик CSS шаблону.

Вміст фреймворка

Якщо застосовувати Bootstrap, то даний фреймворк дозволить істотно заощадити час розробки фронтенд частини

проекту завдяки великій кількості готових компонентів. Пізніше розглянемо основні компоненти, якими користуються майже всі фронтенд розробники. Слід тут зазначити, що Bootstrap - це, так би мовити, набір з трьох фреймворків: CSS / HTML, JS компоненти та ікончасті шрифти.

Сітка

Наявність сітки є базовою вимогою для гарного макета. Сітка - це потужний інструмент для розташування блочного контенту й вкладених елементів. За допомогою префіксів ви можете вказати, яким чином повинні відображатися блоки, в залежності від типу пристрою, на якому проглядається вебсайт.

Наприклад, клас «col-xs-» буде використовуватися для мобільних телефонів з шириною екрану менше 768 пікселів, а клас «col-lg-» - для пристроїв з шириною екрану більш 1170 пікселів. Bootstrap розділяє ширину батьківського блоку на 12 рівних блоків, які ми можемо використовувати як завгодно. Деякі блоки можуть комбінуватися, щоб отримати, наприклад, три колонки: дві 25% «col-lg-3» й одна 50% «col-lg-6».

Візуально сторінка може бути представлена в будь-якому бажаному вигляді:

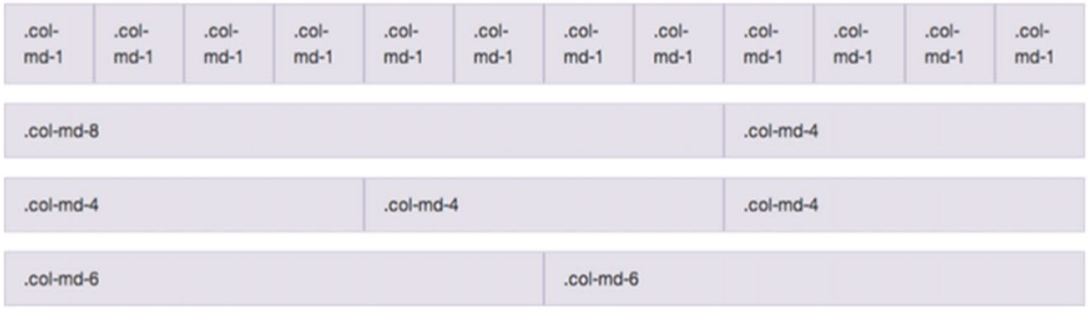


Рисунок 2.1 – Візуальне представлення сторінки за допомогою Bootstrap

!!! Якщо ви вийдете за межі 12 блоків, то решта буде наведена нижче, під іншими блоками, тому потрібна увага.

Ось приклад блоків, які будуть однаково відображатися на всіх пристроях. Давайте розділимо екран на три рівні частини:

```
<div class="row">
<div class="col-sm-4">First column</div>
<div class="col-sm-4">Second column</div>
<div class="col-sm-4">Third column</div>
</div>
```

Вище наведено приклад трьох рівних за шириною колонок, які будуть відображатися горизонтально на планшетах і більших екранах. На екрані мобільного кожен блок буде розтягнутий і кожний займе всю ширину екрану.

Результат роботи даного коду в парі з роботою фреймворку представлено нижче на рисунку 2.2а та 2.2б.

Рисунок 2.2а - Розміщення блоків для пристроїв шириною екрана > 576px

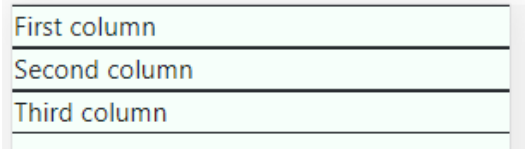


Рисунок 2.2б – розміщення блоків для пристроїв Extra small < 576px

Рисунок 2.2 – Автоматична зміна розміщення блоків на різних пристроях

Типографіка

На додаток до можливості змінювати блоки на макеті й структуру сторінки Bootstrap дозволяє форматувати параметри

шрифту: абзаци, цитати, заголовки, підзаголовки, різні розміри тексту, вставки коду й ін. У більшості випадків не потрібно змінювати налаштування тексту за замовчуванням, тому що всі зовнішні відступи, заголовки, інтервали між рядками (інтерліньяж) і т. ін. вже ретельно підібрані.

Велика увага приділяється семантиці: основний заголовок може бути заданий у вигляді тега <h1> heading </ h1>, але також і у вигляді heading - обидва варіанти будуть виглядати однаково, але другий може бути використаний будь-яку кількість разів на сторінці.

На рисунку 2.3 представлено, як будуть виглядати заголовки у фреймворку за замовчуванням:

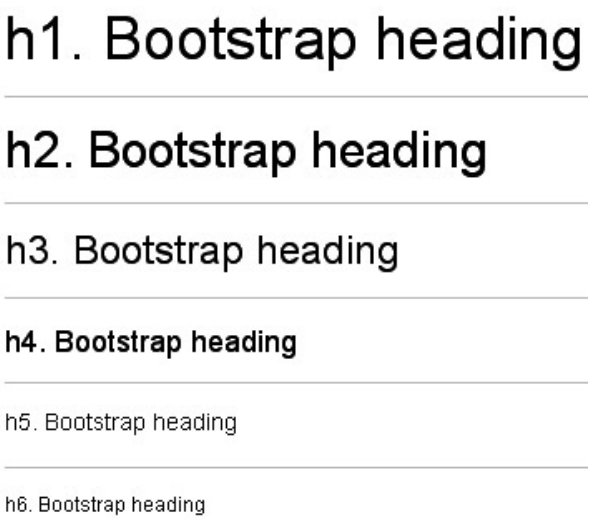


Рисунок 2.3 – Відображення заголовків у фреймворку Bootstrap

Сповіщення (Алерти)

Будь-яке сповіщення може бути представлено в 4 стандартних форматах: позитивний, інформаційний, що попереджає, негативний. Ось як вони виглядають:

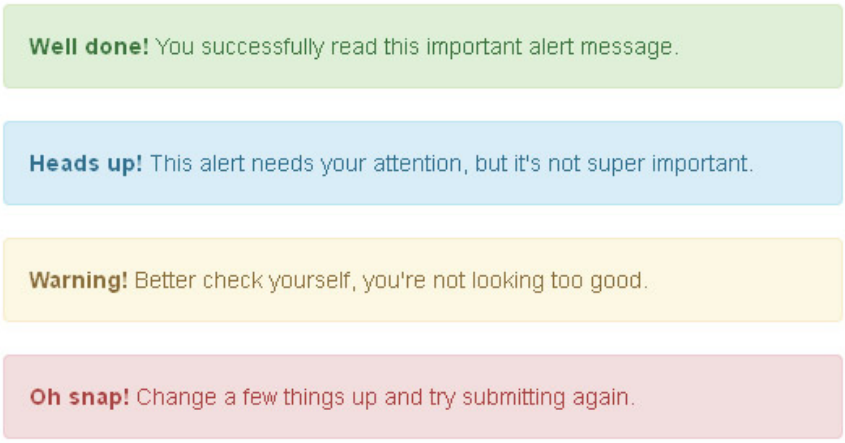


Рисунок 2.4 – Відображення сповіщень у фреймворку Bootstrap

Для того щоб відформатувати будь-яке повідомлення, потрібно додати два класи для потрібного об'єкту:

```
<div class="alert alert-success" role="alert">Well done</div>
<div class="alert alert-info" role="alert">Heads up! </div>
<div class="alert alert-warning" role="alert">warning! </div>
<div class="alert alert-danger" role="alert">Oh snap! </div>
```

Також Bootstrap дозволяє вам форматувати діалогові вікна, спливаючі вікна (pop-up) і спливаючі підказки (tooltip).

Навігація

Навігація зазвичай є одним з основних елементів на вебсайті та їй приділяється окрема увага при розробці макета. Правильне проектування макета, створення й оформлення елементів навігації для їх належного функціонування - це зазвичай одне з найбільш складних завдань. Навігації в Bootstrap приділено особливу увагу: фреймворк містить дизайни для вкладок (табів), посторінкової

навігації (пагінацію), бічних меню, «хлібних крихт», основного меню, панелі інструментів (тулбара) і т.д.

Дуже легко зробити так, щоб основне меню виглядало наступним чином (рисунок 2.5):

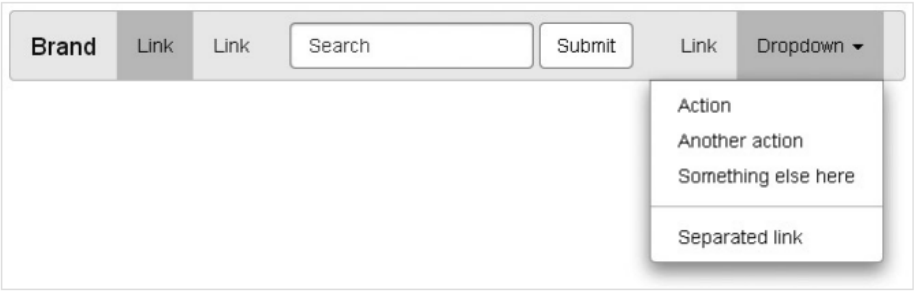


Рисунок 2.5 – Можливості побудови меню на Bootstrap

Пагінація

На рисунку 2.6 зображено, як виглядає пагінація.

Рисунок 2.6 – Пагінація на Bootstrap

Нижче наведено код.

```
<nav>
<ul class="pagination">
<li><a href="#"><span aria-
hidden="true">&laquo;</span><span class="sr-
only">Previous</span></a></li>
<li><a href="#">1</a></li>
<li><a href="#">2</a></li>
<li><a href="#">3</a></li>
<li><a href="#">4</a></li>
<li><a href="#">5</a></li>
```

```
      <li><a href="#"><span aria-
hidden="true">&raquo;</span><span class="sr-
only">Next</span></a></li>
      </ul>
      </nav>
```

2.2 Форми

Текстові поля й блоки (textarea), кнопки, мітки (label), радіокнопки, чекбокси, та випадаючі списки - для всіх цих елементів в Bootstrap вже є підготовлені стилі. Ви можете створити вертикальні й горизонтальні роздільники в заголовках, а також підсвітити частину форми, якщо виникнуть якісь попередження або помилки.

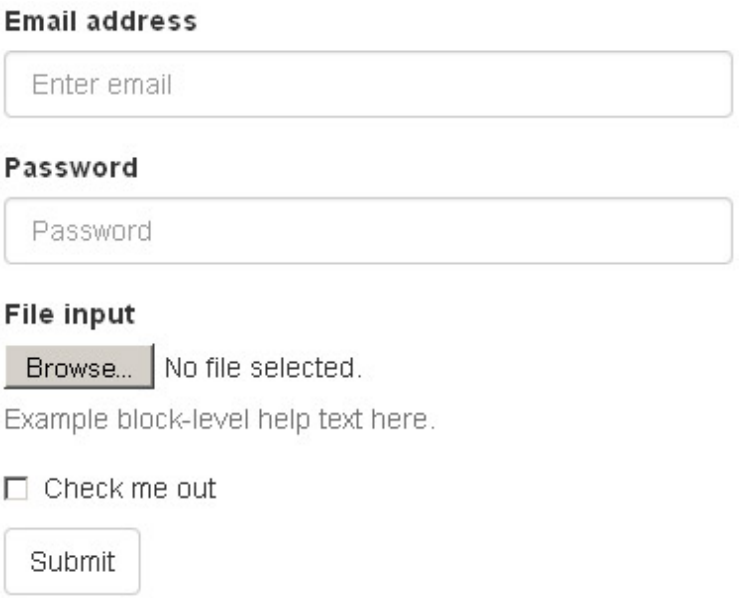


Рисунок 2.7 – Можливості форм Bootstrap

Ви можете зробити горизонтальне розташування блоків, розташування всіх полів в один рядок і багато іншого.

Кнопки

Щоб створити кнопку, потрібно вказати потрібний набір класів. Кілька типів кнопок зображено на рисунку 2.8.

Рисунок 2.8 - Кнопки

Нижче наведено необхідний код.

```
    <button type="button" class="btn btn-
default">Default</button>
    <button type="button" class="btn btn-
primary">Primary</button>
    <button type="button" class="btn btn-
success">Success</button>
    <button type="button" class="btn btn-
info">Info</button>
    <button type="button" class="btn btn-
warning">Warning</button>
    <button type="button" class="btn btn-
danger">Danger</button>
    <button type="button" class="btn btn-
link">Link</button>
```

Таблиці

Таблиці створюються шляхом додавання класу «table». І ми отримуємо дуже акуратну таблицю, що на рисунку 2.9.

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

Рисунок 2.9 - Створення таблиць

Іконочний шрифт

Іконочний шрифт дозволить забути про спрайт для іконок і дуже значно облегшує життя розробника. Єдине обмеження полягає в тому, що до однієї іконки можна застосувати тільки один колір.

Рисунок 2.10 - Іконки bootstrap

Вам надається вибір з 200 іконок, і ви можете додати їх на вебсайт таким чином:

Іконочні шрифти знаходяться в теці fonts (шрифти), і якщо він дійсно потрібен, переконайтеся в тому, щоб він залишився в цій теці. Необхідно зберегти структуру папок, щоб все працювало коректно.

Багато компонентів вимагають використання JavaScript. Точніше - їм потрібні jQuery, Popper.js та наші власні плагіни. Для активації плагінів розмістіть наступний кусок коду <script> у кінці сторінки, безпосередньо перед закритим </body>. Спочатку jQuery, потім Popper.js, потім наші.

Ми будемо використовувати міні-збірку jQuery, хоча можна використати і повну.

2.3 Адаптивний дизайн Bootstrap

Адаптивний дизайн (англ. Responsive Web Design) об'єднує в собі три методики - гнучкий макет на основі сіток, гнучкі зображення та медіазапити. Гнучкість макета базується на використанні відносних одиниць вимірювання замість фіксованих піксельних значень. Проблема гнучких зображень вирішується за допомогою правила img (width: 100%; max-width: 100%;) для всіх картинок на сайті. Це правило гарантує, що зображення ніколи не будуть ширше, ніж їхні контейнери і ніколи не перевищать своїх справжніх розмірів на великих екранах. Медіазапити змінюють стилі на підставі характеристик пристрою, пов'язаних з відображенням контенту, включаючи тип, ширину, висоту, орієнтацію й характеристики екрану. За допомогою медіазапитів створюється адаптивний дизайн, в якому до кожного розміру екрану застосовуються відповідні стилі.

Адаптивний мета-тег

Bootstrap розроблявся спочатку як мобільний (mobile first), тобто його налаштування попередньо оптимізовані під мобільні пристрої, а потім за допомогою медіа-запитів підганяється масштаб компонентів як необхідно на інших пристроях. Необхідно помістити цей код між тегами <head>:

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

Після вставки даного коду сторінка, що відкрита в браузері, буде підлаштовуватися до розміру екрану й блоки будуть вже адаптивні. Це головний мета-тег, який відповідає за адаптивність сторінки.

Компоненти JavaScript

На додаток до стилів в Bootstrap є правила поведінки для модальних вікон, слайдерів, вспливаючіх підказок, та інших

інтерактивних елементів на сторінці. Щоб керувати цими компонентами, необхідно підключити бібліотеку jQuery до файлу bootstrap.js.

Початок роботи

Ви можете почати використовувати Bootstrap 4 на своєму вебсайті, включивши його з CDN (Content Delivery Network - марежа доставки контенту) або завантаживши з вебсайту getbootstrap.com

Але для початку роботи з Bootstrap слід завантажити файли, які складають його бібліотеку й розібратися в їх структурі.

Підключення за допомогою CDN

Зайдіть на офіційний сайт фреймворку за адресою getbootstrap.com та оберіть в правому верхньому куті необхідну версію Bootstrap. В нашому випадку це v4.6.x. Потім перейшовши в пункт:

CSS

Copy-paste the stylesheet <link> into your <head> before all other stylesheets to load our CSS.

Скопіюйте необхідний код підключення фреймворку натиснувши кнопку copy:

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css" integrity="sha384-B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l" crossorigin="anonymous">

Отриманий код необхідно вставити між тегами <head>після мета-тега viewport.

Потім увімкніть CDN-версії jQuery і Popper.js (Bootstrap 4 використовує jQuery і Popper.js для використання таких компонентів JavaScript, як модали, спливаючі підказки, спливаючі

вікна і т.ін) перед мінімізованим Bootstrap JavaScript, якщо ви використовуєте скомпільовану версію JavaScript.

Також зайшовши на вкладку Starter template можна одразу скопіювати стартовий код і вставити в проект.

Нижче наведені деякі компоненти, які вимагають JQuery:

– Використовується для закриття попереджень

– Переведення стану за допомогою кнопок і прапорців / перемикачів

– Карусель для слайдів, елементів керування та індикаторів

– Dropdowns (для ідеального позиціонування використовується Popper.js)

– Відкриття і закриття модалів

– Для згортання навбара

– Підказки та спливаючі підказки (для ідеального позиціонування використовується Popper.js)

Приклад стартової сторінки можна отримати перейшовши на сайті фреймврку на відповідну вкладку *starter template.*

Код, що отримали скопіюйте в буфер обміну та скопіюйте в новий проект (index.html).

В даному випадку підключено фреймворк за допомогою CDN.

CDN – це мережа доставки контенту, яка допомагає збільшити швидкість завантаження файлів на сторінках сайту за допомогою декількох серверів, розташованих віддалено один від одного.

2.4 Завантаження Bootstrap (локальне підключення)

Для вище запропонованого методу (CDN – підключення фреймворку) для використання нам буде необхідно постійне підключення до мережі Інтернет.

Тому, якщо ми хочемо використовувати фреймворк без постійного підключення нам необхідно скачати одну із версій, запропоновану на сайті перейшовши на відповідну вкладку Download за адресою[8]:

https://getbootstrap.com/docs/4.6/getting-started/download/

Файлова структура

Як тільки скомпільована версія Bootstrap 4 буде завантажена, розархівуйте ZIP-файл, і ви побачите наступну структуру файлів / каталогів:

Рисунок 2.11 – Структура каталогів Bootsrap

Як видно, скомпільовані CSS і JS (bootstrap.*), А також скомпільовані й мінімізовані CSS і JS (bootstrap.min.*).

Розглянемо, для чого призначені файли Bootstrap:

- bootstrap.css - головний файл Bootstrap, що визначає оформлення css елементів бібліотеки;
- bootstrap.min.css - містить ті самі стилі, що і попередній файл, але стиснутий для швидкого завантаження;
- bootstrap - theme.css - файл перевизначає стандартні стилі оформлення елементів Bootstrap, можна використовувати як часткову заміну bootstrap.css;
- bootstrap - theme.min.css - містить ті самі стильові описи, що і попередній файл, але стиснутий для швидкого завантаження;
- bootstrap-theme.css.map - дозволяє працювати з файлами так, начебто вони не були стиснуті;
- bootstrap.css.map - дозволяє працювати з файлом bootstrap.min.css, так наче він не був стиснутий;
- bootstrap.js - файл, що відповідає за динамічні можливості бібліотеки;
- bootstrap.min.js - стиснутий файл bootstrap.js;
- glyphicons - halflings - regular.eot - у даному файлі знаходяться векторні іконки бібліотеки для браузера Internet Explorer;
- glyphicons - halflings - regular.svg - у даному файлі знаходяться всі шрифти й іконки, що в попередньому, але у форматі векторної графіки svg;
- glyphicons - halflings - regular.ttf - стандартний файл шрифтів;
- glyphicons - halflings - regular.woff - стиснутий файл шрифтів.

Для початку роботи слід підключити bootstrap.css і bootstrap.js. Також для повноцінної роботи бібліотеки слід підключити JQuery. Приблизно так повинен виглядати початковий документ (рисунок 2.12), заснований на Bootstrap:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Bootstrap</title>
  <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
 </head>
 <body>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  <script src="bootstrap/js/bootstrap.min.js"></script>
 </body>
</html>
```

Рисунок 2.12 – Початковий документ побудований на Bootstrap

Завантажений файл дистрибутива поміщаємо в теку з нашим проектом та перейменовуємо її в Bootstrap.

Після розархівування відповідного файлу дистрибутива фреймворку ми отримаємо дві теки CSS та JS.

Для режиму розробки підключаємо не стиснуту версію, для цього перед тегом </head> прописуємо даний код:

<link rel="stylesheet" href="bootstrap/css/bootstrap.css">

Далі необхідно підключити файли роботи з JS, для цього підключивши відповідний файл перед закриваючим тегом </body>:

<script src="bootstrap/js/bootstrap.js"></script>

Щоб перевірити чи підключився фреймворк запустимо наш проект і зайдемо в консоль браузера (перед цим закоментувавши файли підключення CDN). Буде видно, що для нашого тега h1 підключаються відповідні правила з файлу bootstrap.css.

Далі продовжимо знайомитися з фреймворком вивчаючи його сітку як головний елемент, на якому будується вся робота фреймворку.

2.5 Модульна сітка Bootstrap

Кожний фреймворк CSS, зазвичай, пропонує свою сітку для використання. Це те, навіщо нам потрібний фреймворк. Це те, за допомогою чого ми створюємо (верстаємо) наш адаптивний рівний красивий сайт і один з головних компонентів який використовується найчастіше, (кнопки, каруселі, модальні вікна це все вторинні продукти) це наш каркас для роботи.

Для знайомства з сіткою перейдемо в розділ нашого фреймворку Layout підрозділ Grid [8]. Посилання відповідне знаходиться за адресою: https://getbootstrap.com/docs/4.6/layout/grid/.

Сіткова система використовує flexbox для мобільних пристроїв, щоб створювати макети будь-якої форми та розміру завдяки системі дванадцяти стовпців, п’яти адаптивним рівням за замовчуванням, змінним і комбінаціям Sass та десяткам попередньо визначених класів.

Як це працює

Сіткова система Bootstrap використовує ряд контейнерів, рядків та стовпців для розміщення та вирівнювання вмісту. Вона побудована на флексбоксі й повністю гнучка.

Для кращого розуміння перед вивченням сітки Bootstrap обов’язково розгляньте розділ посібника по флексбоксам.

Розглянемо приклад сітки на прикладі сітки Excel. На звичайному аркуші екселя можна побачити звичайну сітку – це колонки й комірки, що утворені на перетині рядів і стовпців. Результатом буде сітка. Сітка фреймворку Bootstrap складається з дванадцяти стовпців (рисунок 2.13):

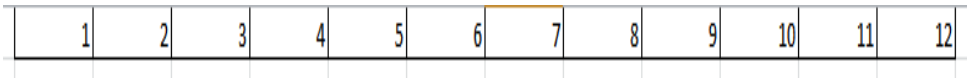


Рисунок 2.13 – Сітка фреймворку Bootstrap

Це кількість комірок в одному ряді. За замовчуванням їх дванадцять, хоча це можна змінити за допомогою Sass змінних та перекомпілювавши відповідні файли.

Чому саме дванадцять? Це оптимально практично для усіх задач. Все тому що оперувати дванадцятьма колонками дуже легко оскільки дане число ділиться на багато чисел 1..2..3..4..6..12. Так ми можемо визначити скільки колонок ми хочемо бачити в одному ряді. А саме, скільки комірок буде відведено під одну колонку. Наприклад, якщо це весь екран, то ми вкажемо дванадцять комірок в одному ряді, якщо менше (наприклад, сайт із сайдбаром і контентом) – вкажемо, наприклад, чотири комірки під сайдбар а вісім під контент (все що залишилося 12-4=8).

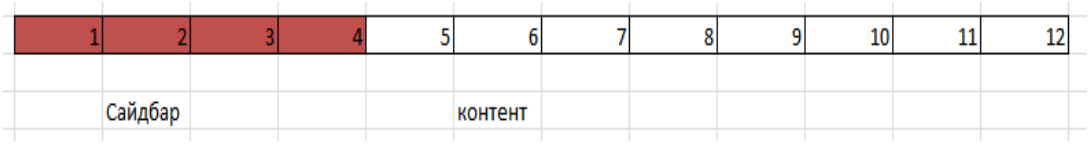


Рисунок 2.14 – Розподіл сітки між блоками по формулі 1:2

Якщо це вже трьох колонковий сайт і ми хочемо розділити його на три частини, то структуру необхідно змінити, наприклад, на відповідну, що зображено на рисунку 2.15.

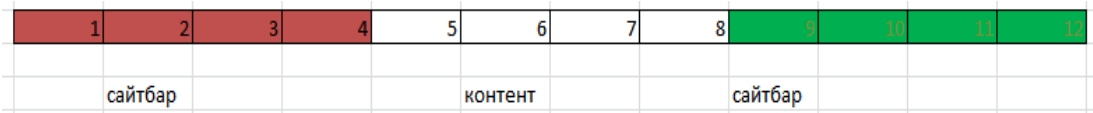


Рисунок 2.15 – Розподіл сітки на 3 рівні колонки

Якщо ми створюємо, наприклад, лендингову сторінку (landing Page), то під контент ми виділяємо всю доступну ширину (12 стовпців). Це дуже гнучко та зручно.

Є три головні класи роботи з сіткою:

– class="container "

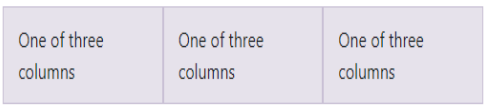
– class="row"

– class="col"

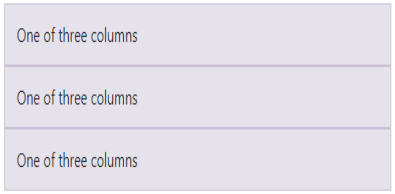
Розглянемо приклад, наведений в документації:

```
<div class="container">
<div class="row">
<div class="col-sm">
      One of three columns
</div>
<div class="col-sm">
      One of three columns
</div>
<div class="col-sm">
      One of three columns
</div>
</div>
</div>
```

Результат виконання даного коду наведений на рисунку 2.16, в залежності від ширини екрану відповідного пристрою.



а) Середні, великі та надвеликі пристрої



б) Мобільні пристрої

Рисунок 2.16 – а) Персональні комп’ютери, б) Мобільні пристрої

Наведений приклад створює три стовпці однакової ширини на малих, середніх, великих та надвеликих пристроях за допомогою наших попередньо визначених класів сітки. Ці стовпці центруються на сторінці з батьківським класом .container.

Контейнери забезпечують можливість центрування та горизонтального розміщення вмісту вебсайту. Використовується .container для змінної ширини (за замовчуванням клас .container є чутливим контейнером фіксованої ширини, що означає, що його максимальна ширина max-width змінюється в кожній точці зупинки.) або .container-fluid для ширини: 100% (для створення контейнера повної ширини, що охоплює всю ширину області перегляду).

Чутливі контейнери з'явилися в Bootstrap з версії 4.4. Вони дозволяють вказувати клас шириною 100%, поки не буде досягнута зазначена точка зупинки, після чого ми застосовуємо max-width для кожної з вищих точок зупинок. Наприклад, .container-sm має мати ширину в 100% до тих пір, поки не буде досягнута точка зупинки sm, де він буде масштабуватися вже за допомогою md, lg і xl. Приклад наведено нижче:

ширина 100% до маленької (small) точки зупинки

ширина 100% до середньої (medium) точки зупинки

ширина 100% до большої (large) точки зупинки

ширина 100% до екстра великої (Extra large) точки зупинки

Всередині контейнера повинен розміщуватися ряд (class="row"):

Ряди - це обгортки для стовпців. А всередині стовпців знаходиться клас col. У макеті сітки вміст повинен розміщуватися всередині стовпців, і лише стовпці можуть бути безпосередніми нащадками рядків.

Завдяки flexbox стовпці сітки без заданої ширини автоматично розміщуватимуться як стовпці однакової ширини. Наприклад, чотири екземпляри .col-sm автоматично матимуть ширину 25%.

Класи стовпців вказують кількість стовпців, які потрібно використовувати з можливих 12 на рядок. Отже, якщо потрібні три однакові за шириною стовпці, можете використовувати .col-4.

Ширину стовпців встановлюють у відсотках, тому вони завжди є похідними від розмірів батьківського елемента.

Щоб зробити сітку гнучкою, існує п’ять точок (breakpoints), в залежності від розмірів екрана: extra small, small, medium, large, та extra large.

Ці точки (брейкпоінти) сітки базуються на медіа-запитах мінімальної ширини, (наприклад, .col-sm-4 застосовується до малих, середніх, великих та надвеликих пристроїв, але не до першої точки xs).

Можливо використовувати заздалегідь визначені класи сітки (наприклад .col-4) або комбінації Sass для більш семантичної розмітки.

Існують обмеження та помилки навколо flexbox, як-от неможливість використовувати деякі елементи HTML як гнучкі контейнери.

2.6 Класи сітки

Система Grid Bootstrap 4 має п'ять класів:

.col- (малі пристрої - ширина екрану менше 576 пікселів)

.col-sm- (малі пристрої - ширина екрану дорівнює або більше, ніж 576 пікселів)

.col-md- (середні пристрої - ширина екрану, що дорівнює або перевищує 768 px)

.col-lg- (великі пристрої - ширина екрану дорівнює або більше, ніж 992 px)

.col-xl- (XLarge пристрої-ширина екрану, що дорівнює або перевищує 1200 px)

Наведені вище класи можна комбінувати для створення більш динамічних і гнучких макетів.

Кожен клас масштабується вгору, так що якщо потрібно встановити однакову ширину sm і md, ви повинні вказати тільки sm.

Системні правила сітки

Деякі правила системи сітки Bootstrap 4:

– Рядки повинні бути розміщені в межах .container (фіксованої ширини) або .container-fluid (повна ширина) для правильного вирівнювання та заповнення;

– Використання рядків для створення горизонтальних груп стовпців;

– Вміст повинен бути поміщено в стовпці, й тільки стовпці можуть бути безпосередніми нащадками рядків;

– Попередньо визначені класи .row, як і .col-sm-4 доступні для швидкого створення макетів сітки;

– Стовпці створюють проміжки між вмістом стовпців за допомогою заповнення. Це заповнення зміщення в рядках для

першого й останнього стовпця за допомогою негативного поля на.rows;

– Стовпці сітки створюються шляхом вказування числа 12 доступних стовпців, які необхідно охопити. Наприклад, три рівних стовпці будуть використовувати три .col-sm-4;

– Ширина стовпців у відсотках, тому вони завжди є гнучкими і мають розмір відносно батьківського елементу;

– Відстань між колонками задається за допомогою padding.

– Найбільша різниця між Bootstrap 3 і Bootstrap 4 полягає в тому, що BootStrap 4 тепер використовує Flexbox, а не float. Одна з великих переваг з Flexbox є те, що стовпчики сітки без заданої ширини будуть автоматично як однаковою шириною і рівної висоти. Приклад: три елементи з .col-sm - кожному автоматично буде 33,33%.

Контейнери

Це базовий елемент в Bootstrap і вони необхідні при використанні нашої стандартної Grid системи. Контейнери використовуються для розміщення в них вмісту, доповнень й (іноді) центрування вмісту всередині них. Хоча контейнери можуть бути вкладеними, більшість макетів не вимагають вкладеного контейнера.

У Bootstrap використовується три різних типи контейнерів:

– .container, який встановлює максимальну ширину max-width в кожній точці зупинки;

– .container-fluid, ширина якого width: 100% на всіх точках;

– .container- {breakpoint}, тобто ширина width: 100% до зазначеної точки.

У таблиці 2.1 показано, як максимальна ширина max-width кожного контейнера .container і .container-fluid порівнюється з вихідними в кожній точці зупинки.

Таблиця 2.1 – Розміщення контенту на сторінці в залежності від параметрів пристроїв

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
.container	100%	540px	720px	960px	1140px
.container-sm	100%	540px	720px	960px	1140px
.container-md	100%	100%	720px	960px	1140px
.container-lg	100%	100%	100%	960px	1140px
.container-xl	100%	100%	100%	100%	1140px
.container-fluid	100%	100%	100%	100%	100%

2.7 Медіа запити

Для визначення основних параметрів сітки можна використовувати медіа запити Less:

Для невеликих пристроїв і планшетів з екраном до 768px:

@media (min-width: @screen-sm-min) { ... }

Для средніх екранів 992px и вище:

@media (min-width: @screen-md-min) { ... }

Для великихекранів 1200px и вище:

@media (min-width: @screen-lg-min) { ... }

Також в ці медіа запити можно включати max-width, для того щоб обмежити CSS для більш вузького набору пристроїв.

@media (max-width: @screen-xs-max) { ... }

@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }

@media (min-width: @screen-md-min) and (max-width: @screen-md-max) { ... }

@media (min-width: @screen-lg-min) { ... }

Головна структура сітки

Нижче наведено код базової структури сітки Bootstrap 4.

```
<div class = "container">
< div class = "row">
<div class = "col-*-*"></div>
<div class = "col-*-*"></div>
</div>
<div class = "row">
<div class = "col-*-*"></div>
<div class = "col-*-*"></div>
<div class = "col-*-*"></div>
</div>
<div class = "row">...</div>
</div>
```

Приклад Grid-системи Bootstrap 4

```
<div class="container-fluid">
<!-- Control the column width, and how they should appear on different devices -->
<div class="row">
<div class="col-sm-6" style="background-color:yellow;">50%</div>
<div class="col-sm-6" style="background-color:orange;">50%</div>
</div>
<br>
<div class="row">
<div class="col-sm-4" style="background-color:yellow;">33.33%</div>
<div class="col-sm-4" style="background-color:orange;">33.33%</div>
<div class="col-sm-4" style="background-color:yellow;">33.33%</div>
```

```
      </div>
      <br>
      <!-- Or let Bootstrap automatically handle the
layout -->
      <div class="row">
      <div class="col-sm" style="background-
color:yellow;">25%</div>
      <div class="col-sm" style="background-
color:orange;">25%</div>
      <div class="col-sm" style="background-
color:yellow;">25%</div>
      <div class="col-sm" style="background-
color:orange;">25%</div>
      </div>
      <br>
      <div class="row">
      <div class="col" style="background-
color:yellow;">25%</div>
      <div class="col" style="background-
color:orange;">25%</div>
      <div class="col" style="background-
color:yellow;">25%</div>
      <div class="col" style="background-
color:orange;">25%</div>
      </div>
      </div>
      </div>
```

Параметри сітки (Grid option)

Хоча Bootstrap використовує ems або rems одиниці для визначення більшості розмірів, пікселі (px) використовується для брейкпоінтів сітки та ширини контейнера. Це тому, що ширина області визначається у пікселях і не змінюється залежно від розміру шрифту.

Нижче в таблиці 2.2 наведено основні точки (breakpoint) сітки Bootstrap-4 в залежності розмірів екрана пристроїв.

Таблиця 2.2 – Робота Grid системи Bootstrap 4

	Дуже маленькі пристрої (<576px)	**Маленькі пристрої (≥576px)**	**Середньої величини пристрої (≥768px)**	**Великі пристрої (≥992px)**	**Дуже великі пристрої (≥1200px)**
Поведінка сітки	Горизонтально за всіх часів	Згорнуто, щоб почати, горизонтально над контрольними точками	Згорнуто, щоб почати, горизонтально над контрольними точками	Згорнуто, щоб почати, горизонтально над контрольними точками	Згорнуто, щоб почати, горизонтально над контрольними точками
Максимальна ширина контейнера	Ні (авто)	540 px	720 px	960 px	1140 px
Класи класів	.col-	.col-SM-	.col-MD-	.col-LG	.col-XL-
Кількість стовпців	12	12	12	12	12
Ширина ринви	30px (15 px на кожній стороні стовпця)	30px (15 px на кожній стороні стовпця)	30px (15 px на кожній стороні стовпця)	30px (15 px на кожній стороні стовпця)	30px (15 px на кожній стороні стовпця)
Вкладені	Да	Да	Да	Да	да
Колонка заказа	Да	Да	Да	Да	да

Нижче наведено приклад створення класичної (середньої сітки). Для створення середньої модульної сітки використовується клас .col - md- *, який відображається на мобільних пристроях та невеликих планшетах у стиснутому вертикальному стані, а на комп'ютері стає горизонтальною.

У наступному прикладі зображено один горизонтальний рядок з 12 модулів (максимально можливих). Зверніть увагу, що всі модулі розміщені в контейнері з класом .row , який визначає горизонтальне розташування модулів і зв'язує їх в єдине ціле. Кажному модулю задається клас col-md- 1, визначаючи ширину модуля рівною приблизно 81px і займає місце в модульній сітці рівне одному модулю. Якщо було б вказано в класі col-md-2, то модуль став би шириною приблизно 162 і займав два модулі сітки. Якщо в підсумку модулі стануть більше 12, то наступні модулі переходять на новий ряд, чого краще не допустити.

```
<div class="container">
<div class="row">
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
<div class="col-md-1">.col-md-1</div>
</div>
</div>
```

У наступному прикладі створено рядок з 2 модулів, ширина кожного приблизно 570px і займає 6 модулів.

```
<div class="container">
<div class="row">
<div class="col-md-6">.col-md-6</div>
<div class="col-md-6">.col-md-6</div>
</div>
</div>
```

Для створення гумового (гнучкого) макета слід використовувати клас .container-fluid (займе всю доступну ширину екерана):

```
<div class="container-fluid">
<div class="row">
<div class="col-md-6">.col-md-6</div>
<div class="col-md-6">.col-md-6</div>
</div>
</div>
```

Сітка для мобільних пристроїв(<576px) та комп'ютерів

Можна одночасно застосовувати класи для різних типів пристроїв. У наступному прикладі створено рядок з двох модулів. При перегляді на комп'ютері два модуля розташовуються поруч col-md-8 і col-md-4. При перегляді на мобільному пристрої перший модуль буде займати всю область col-12, а другий col-6 перейде на новий рядок, і буде займати половину області перегляду.

```
<div class="container">
<div class="row">
<div class="col-sm-12 col-md-8">.col-12 .col-md-8</div>
<div class="col-sm-6 col-md-4">.col-6 .col-md-4</div>
</div>
</div>
```

Сітка для мобільних пристроїв, планшетів і комп'ютерів

До попереднього прикладу можна додати ще клас для зовсім невеликих пристроїв, наприклад, смартфонів. Розглянемо приклад нижче. При розмірі екрану більше 768 px будуть використовуватися класи col-md, які складуть рядок з двох модулів по 8 і 4 модуля відповідно, якщо встановлено фіксований контейнер, то сукупна ширина дорівнюватиме 720px.

При розмірі екрану в діапазоні від 768px до 576px будуть застосовані класи col-sm, які так само створять два модуля один з 10 колонок, інший з двох, і їх загальна ширина становитиме 750px.

При розмірі екрану менше 576px будуть застосовані класи col-, які створять два модуля, один буде займати весь рядок, а інший буде перенесений на новий рядок, і буде займати 6 модулів. Ширина в цьому випадку буде вираховуватися автоматично в залежності від ширини екрану.

Приклад побудови :

```
<div class="container">
<div class="row">
<div class="col-12 col-sm-10 col-md-8">col-12 col-sm-10 col-md-8</div>
<div class="col-6 col-sm-2 col-md-4">col-6 col-sm-10 col-md-4</div>
</div>
</div>
```

Перенос модулів

Як уже згадувалося, якщо модулів більше ніж 12, то останній модуль колонок переноситься цілком на новий рядок. У наступному прикладі, у рядку, який повинен бути однорядковим, з'являється другий рядок за рахунок колонки з чотирьох модулів, яка не поміщається в дванадцять дозволених:

```
<div class="container">
```

```
      <div class="row">
      <div class="col-xs-9">.col-xs-9</div>
      <div class="col-xs-4">.col-xs-4<br> 9 + 4 = 13
</div>
      </div>
      </div>
```

Ці чотири модуля будуть перенесені цілком на новий рядок, оскільки не поміщаються в попередній рядок.

Відступи в модулях

Для створення відступів використовуються класи offset-md-*. У наступному прикладі у першого модуля буде створено відступ ліворуч, рівний 2 модулів і в сумі всі модулі будуть займати 8 модулів.

```
      <div class="container">
          <div class="row">
               <div class="col-md-2 offset-md-2">
               .col-md-2</div>
               <div class="col-md-2">.col-md-2</div>
               <div class="col-md-2">.col-md-2</div>
          </div>
      </div>
```

У наступному прикладі у першого і другого модуля буде створено відступ ліворуч, рівний 2 модулів і в сумі всі модулі будуть займати 10 модулів.

```
      <div class="container">
          <div class="row">
      <div class="col-md-2 offset-md-2">.col-md-2</div>
      <div class="col-md-2offset-md-2">.col-md-2</div>
      <div class="col-md-2">.col-md-2</div>
          </div>
      </div>
```

Вкладені модулі

Модулі можна вкладати один в одного. У наступному прикладі в модуль з класом top вкладено рядок з трьома модулями.

```
<div class="container">
    <div class="col-sm-12">
         Головний модуль:
    <div class="row">
<div class="col-md-2 offset-md-1">.col-md-2</div>
<div class="col-md-2 offset-md-1">.col-md-2</div>
<div class="col-md-2 offset-md-1">.col-md-2</div>
    </div>
</div>
```

Порядок розташування модулів

Класи .order - використовуються для контролю над візуальним порядком контенту. Ці класи «чутливі», отже можливо задати порядок за допомогою order брейкпоїнту (наприклад, .order-1.order-md-2).

```
<div class="container">
<div class="row">
<div class="col">
Перший, але не впорядкований
</div>
<div class="col order-12">
другий, але не впорядкований
</div>
<div class="col order-2">
Третій, але перший
</div>
</div>
</div>
```

Результат зображено на рисунку 2.17:

Рисунок 2.17 – Результат прикладу з классом .order -

Також існує чутливий клас .order-first, який швидко змінює порядок одного елемента застосуванням властивості order: -1. Цей клас також може застосовуватися з нумерованими класами order- *.

```
<div class="container">
<div class="row">
<div class="col">
Перший, але не впорядкований
</div>
<div class="col">
другий, але не впорядкований
</div>
<div class="col order-first">
Третій, але перший
</div>
</div>
</div>
```

Результат зображено на рисунку 2.18.

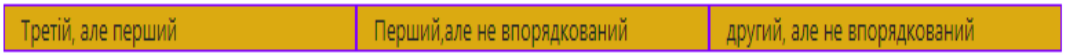


Рисунок 2.18 – Результат прикладу з класом .order-first

Для прикладу використання стоворюємо трьохблочний макет та робимо розташування модулів в залежності від пристроїв, що використовуються:

```
<div class="container">
<div class="row">
<div class="col-md-6 r1">Content</div>
<div class="col-md-3 r2">Sidebar-1</div>
<div class="col-md-3 r1">Sidebar-2</div>
</div>
</div>
```

Результат зображено на рисунку 2.19.

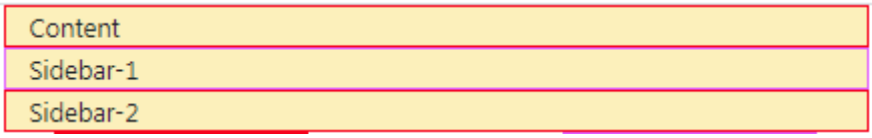


Рисунок 2.19 – Результат прикладу

Як видно, на мобільних пристроях ми отримаємо три блока розташованих в класичному порядку, як це повинно бути на мобільних пристроях, але для сайтів, які працюють на десктопах контент, як правило, розташовується за центром між лівим і правим сайдбаром. Застосуємо клас .order, що автоматично змінить розташування модулів на пристроях medium та більших.

```
<div class="container">
<div class="row">
<div class="col-md-6 order-md-2 r1">Content</div>
<div class="col-md-3 order-md-1 r2">Sidebar-1</div>
<div class="col-md-3 order-md-3 r1">Sidebar-2</div>
</div>
</div>
```

В результаті отримали таке розміщення блоків для пристроїв >768px (рисунок 2.20).

Рисунок 2.20 – Результат прикладу

!!!Якщо якийсь блок буде без вказаного номера то він стане першим, оскільки він не бере участі в сортуванні

Створення мультирядів однакової ширини

Для створення блоків однієї ширини, для різної кількості рядів, виконується застосуванням класу .w-100 у місці, де необхідно

змістити вниз на новий рядок наступний блок. Також можна зробити їх адаптивними, застосовуючи .w-100 разом з адаптивними класами .col.

Наступний приклад створить чотири адаптивних блоки й вони будуть розташовані в два ряди по два блоки в кожному:

```
<div class="row">
<div class="col">col</div>
<div class="col">col</div>
<div class="w-100"></div>
<div class="col">col</div>
<div class="col">col</div>
</div>
```

Результат виконання представлено на рисунку 2.21.

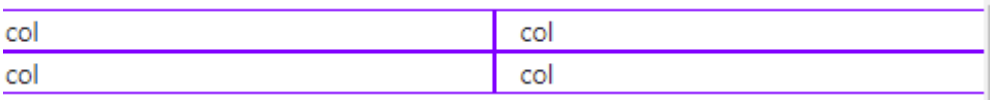


Рисунок 2.21 – Результат прикладу з адаптивними блоками

Застосування класу .col

Для блоків, які виглядають і розташовуються однаково на всіх пристроях будь-якого розміру, використовуйте класи .col і .col-*. Визначте іменований клас з цифрою, коли вам потрібний блок певного розміру, у всіх інших випадках вільно користуйтеся .col:

Приклад:

```
<div class="row">
<div class="col">col</div>
<div class="col">col</div>
<div class="col">col</div>
<div class="col">col</div>
</div>
<div class="row">
<div class="col-8">col-8</div>
<div class="col-4">col-4</div>
```

```
</div>
```

Результат виконання на усіх пристроях (рисунок 2.22):

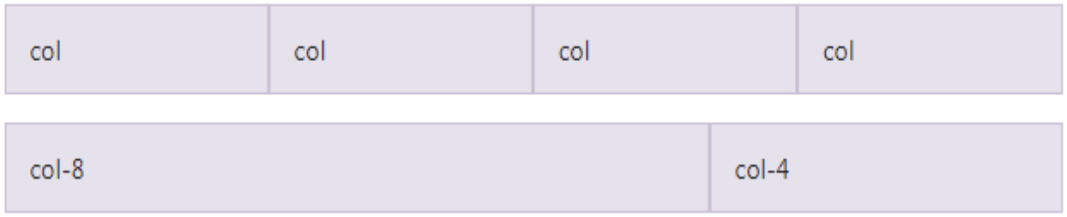


Рисунок 2.22 – Результат роботи прикладу

Відступи в модулях

За замовчуванням в модулях є відступи по 15px в середині блоків. Для того, щоб це скинути (наприклад створення галереї і рисунок повинен займати весь блок) використовується клас *no-gutters* для класу .row, який видаляє негативний margin з .row і горизонтальний padding з усіх колонок.

```
.no-gutters {
margin-right: 0;
margin-left: 0;
Приклад:
<div class="container">
<div class="row no-gutters">
<div class="col">
Перший, але не впорядкований
</div>
<div class="col order-12">
Другий, але не впорядкований
</div>
<div class="col order-2">
Третій, але перший
</div>
</div>
</div>
```

Результат прикладу представлено на рисунку 2.23.

Рисунок 2.23 – Результат прикладу

2.8 Адаптивна ширина блоків

Використовуйте класи col- {breakpoint} -auto для створення колонок, що змінюють свою ширину за шириною вмісту.

Нижче наведено приклад створення такого блока (колонок).

```
<div class="container">
<div class="row justify-content-md-center">
<div class="col col-lg-2">
1 із 3
</div>
<div class="col-md-auto">
Створення адаптивної ширини колонки (блока для
тексту або рисунка)
</div>
<div class="col col-lg-2">
3 із 3
</div>
</div>
<div class="row">
<div class="col">
1 із 3
</div>
<div class="col-md-auto">
Створення адаптивної ширини колонки
</div>
<div class="col col-lg-2">
3 із 3
</div>
</div>
</div>
```

Вирівнювання рядів по вертикалі

Для прикладу спочатку створимо ряд довільної висоти застосувавши правило css:

```
.flex-col{
min-height:10rem;
background-color:#7abaff;
}
```

При цьому слід запам’ятати, що колонки автоматично вирівнюються одна до одної (це дуже добре вирішує давню проблему верстування у разі необхідності зробити колонки однакової висоти – раніше потрібно було емулювати висоту, наприклад, за допомогою JS).

Для вирівнювання є три класи які слід застосувати для ряду .row для вертикального вирівнювання:

– align-items-start - контент прижатий до верху;

– align-items-center - контент в центрі блока;

– align-items-end - контент прижатий до низу:

Приклад:

```
<div class="row flex-col align-items-start">
<div class="col">col-1</div>
<div class="col">Lorem</div>
</div>
<br>
<div class="row flex-col align-items-center">
<div class="col">col-1</div>
<div class="col">Lorem</div>
</div>
<br>
<div class="row flex-col align-items-end">
<div class="col">col-1</div>
<div class="col">Lorem</div>
</div>
```

```
<br>
```

Результат зображено на рисунку 2.24.

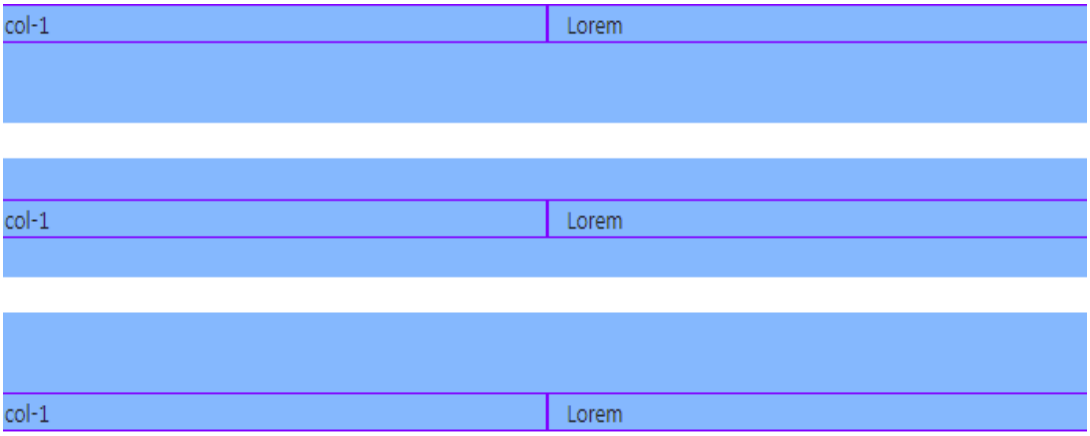


Рисунок 2.24 – Результат прикладу

Але інколи необхідно вирівняти не всі однаково. Для цього слід застосувати класи не для ряду, а для колонок клас .col. Теж є три класи.

– align-self-start
– align-self-center
– align-self-end

Приклад:

```
<div class="row flex-col">
<div class="col align-self-start">col-1</div>
<div class="col align-self-center">col-2</div>
<div class="col align-self-end">col-3</div>
</div>
```

Результат на рисунку 2.25.

Рисунок 2.25 – Результат прикладу вирівнювання блоків

Вирівнювання рядів по горизонталі

Для вирівнювання блоків по горизонталі є також три класи, які вказуються для всього ряда .row:

- justify-content-start - прижаті до лівої частини ряду;
- justify-content-center – центрування;
- justify-content-end - прижаті до правої частини ряду.

Приклад:

```
<div class="row justify-content-end">
<div class="col-3 r1">col-1</div>
<div class="col-3 r2">col-2</div>
<div class="col-3 r1">col-3</div>
</div>
```

Результат зображено на рисунку 2.26.

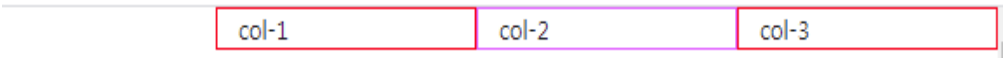


Рисунок 2.26 – Вирівнювання рядів по горизонталі

Також є ще два класи, які застосовуються доволі часто:

- justify-content-around - блоки розташовані рівномірно по осі X;
- justify-content-between - перший блок ліворуч, другий праворуч, а інші рівномірно розподілили доступне місце.

Приклад:

```
<div class="row justify-content-around">
<div class="col-3 r1">col-1</div>
<div class="col-3 r2">col-2</div>
</div>
<br>
<div class="row justify-content-between">
<div class="col-3 r1">col-1</div>
<div class="col-3 r2">col-2</div>
</div>
```

Результат зображено на рисунку 2.27.

Рисунок 2.27 – Результат прикладу

2.9 Допоміжні класи (Utilites)

В Bootstrap є дуже багато модулів. В документації знаходяться в розділі Utilites. Розглянемо тільки головні.

Клас Float (обтікання)

Сторона вирівнювання («обтікання» до певної сторони).

Використовується float для будь-якого елементу, через будь-яку точку брейк-поінту, використовуючи float-утиліти.

Ці класи розміщують елемент ліворуч чи праворуч, або вимикають «обтікання» до певного краю, грунтуючись на поточному розмірі області видимості, використовуючи властивість float в CSS.

– До лівого краю на всіх областях видимості

– До правого краю на всіх областях видимості

– Без боку вирівнювання на всіх областях видимості

Colors

Класи для підкраски тексту, кнопок та інших елементів. В таблиці 2.3 ліворуч наведено головні класи кольорів, а праворуч результат відображення. Класи з префіксом text використовуються для кольорів тексту.

Таблиця 2.3 - Головні класи кольорів

.text-primary .text-secondary .text-success .text-danger .text-warning .text-info .text-light .text-dark .text-body .text-muted .text-white .text-black-50 .text-white-50	.text-primary .text-secondary .text-success .text-danger .text-warning .text-info .text-light .text-dark .text-body .text-muted .text-white .text-black-50 .text-white-50

Колір фона

Для блоків використовуються схожі класи тільки з префіксом bg (наприклад, bg-primary). Нижче в таблиці 2.4 наведено головні класи bg. Компоненти посилання будуть затемнюватися по наведенню, як і класи тексту. У класах фону не задається атрибут color, тому в деяких випадках знадобиться клас.text- *.

Таблиця 2.4 - Головні класи bg

<divclass="p-3 mb-2 bg-primary text-white">.bg-primary	.bg-primary
<divclass="p-3 mb-2 bg-secondary text-white">.bg-secondary	.bg-secondary
<divclass="p-3 mb-2 bg-success text-white">.bg-success	.bg-success
<divclass="p-3 mb-2 bg-danger text-white">.bg-danger	.bg-danger
<divclass="p-3 mb-2 bg-warning text-dark">.bg-warning	.bg-warning
<divclass="p-3 mb-2 bg-info text-white">.bg-info	.bg-info
<divclass="p-3 mb-2 bg-light text-dark">.bg-light	.bg-light
<divclass="p-3 mb-2 bg-dark text-white">.bg-dark	.bg-dark
<divclass="p-3 mb-2 bg-white text-dark">.bg-white	.bg-white
<divclass="p-3 mb-2 bg-transparent text-dark">.bg-transparent	.bg-transparent

Клас Flex

Якщо Вам потрібний контейнер Flex (який застосовується для управління компоновкою, вирівнюванням і калібруванням стовпців сітки, навігації, компонентів і т.ін., з повним набором гнучких утиліт flexbox) то не обов'язково використовувати сітку Bootstrap. Для цього є спеціальні класи емулятори.

Нижче наведено головні класи Flex:

– .d-flex,

– .d-inline-flex,

– .d-sm-flex,
– .d-sm-inline-flex,
– .d-md-flex,
– .d-md-inline-flex,
– .d-lg-flex,
– .d-lg-inline-flex,
– .d-xl-flex,
– .d-xl-inline-flex.

Приклади використання й створення контейнерів наведено нижче. Так утиліти класу .d (скорочено .display) використовуються для створення гнучких контейнерів та трансформації прямих дочірних елементів в гнучкі.

```
<div class="d-flex p-2 bd-highlight bg-info">I'm a flexbox container!</div>
<div class="d-inline-flex p-2 bd-highlight bg-primary">I'm an inline flexbox container!</div>
```

Результат зображено на рисунку 2.28.

Рисунок 2.28 – Результат прикладу роботи Flex-класів

Більш детально даний клас представлено на сайті Bootstrap офіційної документації розділ Utilites/Flex.

Embed (вставка адаптивного відеоконтента)

Для вставки на сторінку відео використовуються два класи:

– embed-responsive;
– embed-responsive-16by9.

Приклад вставки відео наведено в коді нижче. А на рисунку 2.29 результат виконання даного кода з відео, що розташоване на youtube.com.

```
      <div class="container">
      <div class="row">
      <div class="col-md-8 offset-md-2">
      <div class="embed-responsive embed-responsive-
16by9">
      <iframe class="embed-responsive-item"
src="https://www.youtube.com/embed/zpOULjyy-n8?rel=0"
allowfullscreen></iframe>
      </div>
      </div>
      </div>
      </div>
```

Рисунок 2.29 – Вставка відеоконтенту

Display (керування переглядом модулів)

Для керування переглядом модулів (це може бути необхідно коли ми на якомусь із пристроїв бажаємо, наприклад, сховати якесь меню або контент) для різних пристроїв використовуються спеціальні класи, які описані в таблиці 2.5.

Таблиця 2.5 - Класи керування переглядом

Screen Size	Class
Hidden on all	`.d-none`
Hidden only on xs	`.d-none .d-sm-block`
Hidden only on sm	`.d-sm-none .d-md-block`
Hidden only on md	`.d-md-none .d-lg-block`
Hidden only on lg	`.d-lg-none .d-xl-block`
Hidden only on xl	`.d-xl-none`
Visible on all	`.d-block`
Visible only on xs	`.d-block .d-sm-none`
Visible only on sm	`.d-none .d-sm-block .d-md-none`
Visible only on md	`.d-none .d-md-block .d-lg-none`
Visible only on lg	`.d-none .d-lg-block .d-xl-none`
Visible only on xl	`.d-none .d-xl-block`

В даній таблиці представлено класи, які призначені для відображення або приховування блоків сторінки сайту по брейкпоінтам. Так, наприклад, в таблиці клас .d-none повністю приховує блок, але поряд клас .d-sm-block показує на яких пристроях його все таки буде показано. Або, наприклад:

.d-sm-none .d-md-block – не показує блок на sm пристроях, але показує на пристороях md і більших.

Також можна маніпулювати спочатку показом блоків, а потім приховувати блок: **.d-block .d-sm-none.**

Приклад:

```
<div class="container">
<div class="row">
<div class="col bg-primary d-lg-none">hide on lg and wider screens</div>
<div class="col bg-info d-none d-lg-block">hide on screens smaller than lg</div>
</div>
</div>
```

Так в результаті виконання коду ми побачимо, що перший блок буде показано на екранах менших lg (на них буде показано другий блок) і навпаки, другий буде показаний тільки на екранах lg та xl, а на менших - показано перший блок.

Тепер зробимо навпаки:

Показувати на малих екранах менших lg

Показувати на великих екранах lg та el

Далі це все можна комбінувати відкриваючи й приховуючи блоки на потрібних брейкпоінтах. Для прикладу побудуємо сторінку з трьома колонками:

Отже маємо три колонки які показуються на всіх екранах.

```
<div class="container">
<div class="row">
<div class="col-md-3 r1">Sidebar-left</div>
<div class="col-md-6 r2"> Content</div>
<div class="col-md-3 r1">Sidebar-right</div>
</div>
</div>
```

І якщо, наприклад, на мобільних пристроях хочемо приховати правий Sidebar-right, то потрібно встановити йому відповідний клас (не показувати блок на всіх пристроях крім sm та більших):

```
<div class="container">
<div class="row">
<div class="col-md-3 r1">Sidebar-left</div>
<div class="col-md r2"> Content</div>
<div class="col-md-3 d-none d-sm-block r1">Sidebar-
right</div>
</div>
</div>
```

2.10 Компоненти Bootsrap

Тут ми розглянемо головні компоненти, які є в четвертій версії Bootstrap (більшість з них є і в попередніх версіях). Всі вони описані в документації з посиланням:

https://getbootstrap.com/docs/4.6/components/alerts/

Alert

Застосовується для повідомлень змін (наприклад товар добавлено /не добавлено в кошик і т.ін.).

Для прикладу візьмемо зелений компонент (alert-success) і розглянемо його роботу.

```
<div class="alert alert-success" role="alert">
```

```
        A simple success alert—check it out!
    </div>
```

Результат зображено на рисунку 2.30.

Рисунок 2.30 – Результат прикладу використання alert-success

Даний компонент можна ускладнити, якщо додати іконку закрития після виводу повідомлення на екран застосувати до нього клас .dismissing та додати кнопку, після настискання якої повідомлення буде закрито (також за бажанням можна застосувати два додаткові ефекти fade show – плавного закриття повідомлення)

```
    <buttontype="button"class="close"data-
dismiss="alert"aria-label="Close">
    <spanaria-hidden="true">&times;</span>
    </button>
    Приклад:
    <div class="alert alert-success alert-dismissible
fade show
    " role="alert"> A simple success alert—check it
out!
    <button type="button" class="close" data-
dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
    </button>
    </div>
```

!!!Для роботи даного компонента повинен бути підключений jquery

Modal

Показ інформації в модальних (діалогових) вікнах, лайтбоксах та сповіщень для користувачів. Модальні вікна

побудовані за допомогою HTML, CSS та JavaScript. Вони розташовані над усим іншим у документі та видаляють прокрутку з <body>, щоб замість цього прокручувався модальний вміст.

Клік на «модальному вікні» автоматично його закриває.

Bootstrap одночасно підтримує лише одне модальне вікно. Вкладені модальні вікна не підтримуються. За можливості розміщуйте свій модальний HTML у позиції верхнього рівня, щоб уникнути потенційних перешкод з боку інших елементів. Детальний опис наведено в офіційній документації за адресою:

https://getbootstrap.com/docs/4.6/components/modal/.

На рисунку 2.31 наведено приклад вигляду модального вікна Bootstrap 4 та відповідний код.

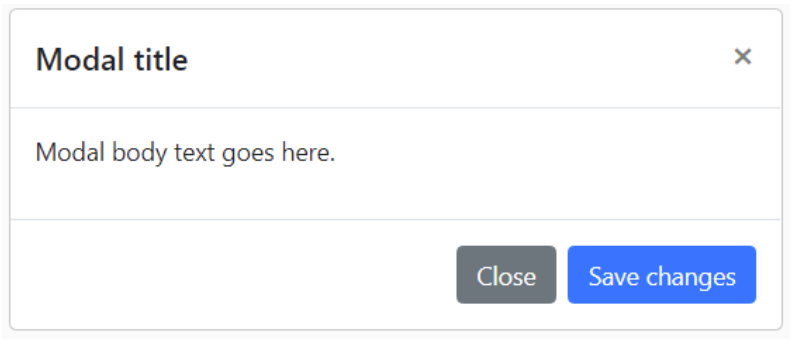


Рисунок 2.31 – Вигляд модального вікна

```
<div class="modal" tabindex="-1">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<h5 class="modal-title">Modal title</h5>
<button type="button" class="close" data-dismiss="modal" aria-label="Close">
<span aria-hidden="true">&times;</span>
</button>
</div>
<div class="modal-body">
```

```
        <p>Modal body text goes here.</p>
        </div>
        <div class="modal-footer">
        <button type="button" class="btn btn-secondary"
data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save
changes</button>
        </div>
        </div>
        </div>
        </div>
```

Щоб побачити виконання коду необхідно замінити lass="modal" на class="modal2" оскільки за замовчуванням дане модальне вікно буде сховане за допомогою класа modal.

Відкриття вікна відбувається за допомогою кнопки, що зображено на рисунку 2.32.

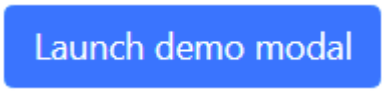


Рисунок 2.32 – Вигляд кнопки для відкриття

Зв'язок відкриття з кнопкую відбувається по data-target="#exampleModal відповідно в блоці побудови модального вікна id="exampleModal". Так створюється зв'язок кнопки з модальним вікном. За натисканням кнопки відбувається зміна класу class="modal" (сховане вікно) на id="exampleModal" – показувати модальне вікно. Також можна застосувати плавність показу вікна за допомогою класу fade для модального вікна.

```
        <buttontype="button"class="btn    btn-primary"data-
toggle="modal"data-target="#exampleModal">
        Launch demo modal
        </button>
```

Нижче наведено код вставки побудови модального вікна.

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
Launch demo modal
</button>
<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<h5 class="modal-title" id="exampleModalLabel">Modal title </h5>
<button type="button" class="close" data-dismiss="modal" aria-label="Close">
<span aria-hidden="true">&times;</span>
</button>
</div>
<div class="modal-body">
...
</div>
<div class="modal-footer">
<button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
<button type="button" class="btn btn-primary">Save changes</button>
</div>
</div>
</div>
</div>
```

Також можна змінювати розмір модального вікна за допомогою Optional sizes. Можливі опції наведено на рисунку 2.33.

Дані опції потрібно встановити для class="modal-dialog". Наприклад: <divclass="modal-dialog modal-sm">...

Size	Class	Modal max-width
Small	.modal-sm	300px
Default	None	500px
Large	.modal-lg	800px
Extra large	.modal-xl	1140px

Рисунок 2.33 – Розміри модальних вікон

Cards

Картки Bootstrap забезпечують гнучкий контейнер вмісту з різними варіантами та опціями. В них доступні контекстні кольори, потужні параметри відображення. Дана функціональність доступна через класи модифікаторів для карток. Даний модуль, який можна корегувати під власні потреби, зображено на рисунку 2.34.

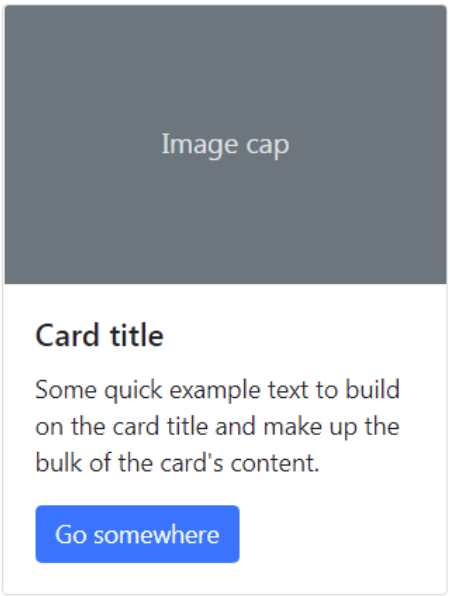


Рисунок 2.34 – Модуль Cards

Також дуже часто даний модуль поміщають в сітку Bootstrap або, наприклад, в модальне вікно.

```
<divclass="card"style="width: 18rem;">
<imgsrc="..."class="card-img-top"alt="...">
<divclass="card-body">
<h5class="card-title">Card title</h5>
<pclass="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
<ahref="#"class="btn btn-primary">Go somewhere</a>
</div>
</div>
Приклад:
<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
<div class="modal-dialog modal-lg">
<div class="modal-content">
<div class="modal-header">
<h5 class="modal-title " id="exampleModalLabel">Modal title</h5>
<button type="button" class="close" data-dismiss="modal" aria-label="Close">
<span aria-hidden="true">&times;</span> </button>
</div>
<div class="modal-body">
<div class="container">
<div class="row">
<div class="col-md-6 ">
<div class="card" >
<img src="img/port.jpg/" class="card-img-top" alt="...">
<div class="card-body">
```

```
      <h5 class="card-title">Card title</h5>
      <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
      <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
      </div>
      </div>
      <div class="col-md-6 ">
      <div class="card">
      <img src="img/port.jpg/" class="card-img-top" alt="...">
      <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
      <a href="#" class="btn btn-primary">Go somewhere</a>
      </div>
      </div>
      </div>
      </div>
      </div>
      </div>
      </div>
      <div class="modal-footer">
      <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
      <button type="button" class="btn btn-primary">Save changes</button>
      </div>
      </div>
      </div>
```

В результаті отримаємо дві картки вбудовані в модальне вікно, як показано на рисунку 2.35.

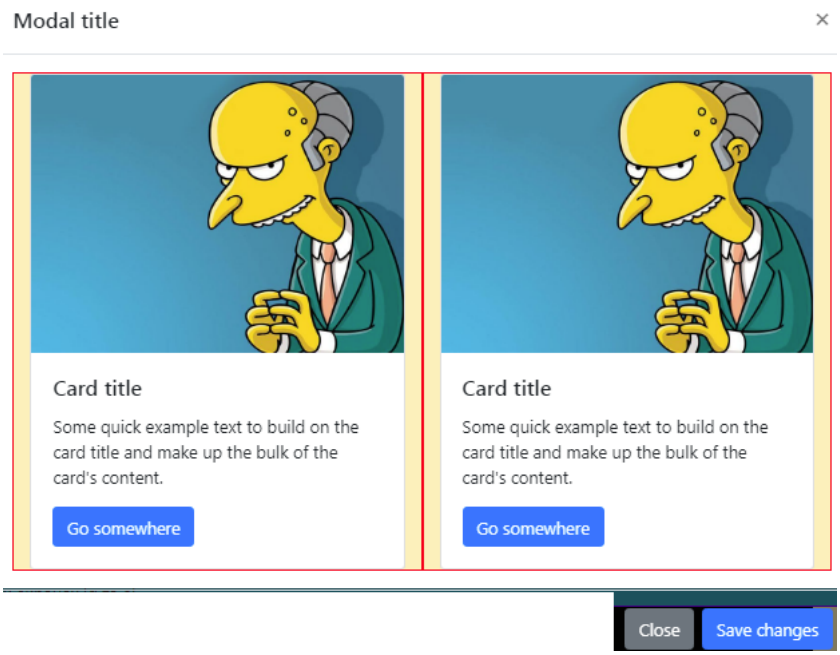


Рисунок 2.35 – Вставка модуля cards в модальне вікно

Також модальними вікнами можна керувати програмно за допомогою JavaScript. Для цього за допомогою компонента кнопки (btn) в Bootstrap створюємо відповідну кнопку на нашій сторіці:

<button class="btn btn-success">open</button>

Також визначаємо клас для показу нашої картки, наприклад, open-cart:

<button class="btn-success open-cart">open</button>

Для того, щоб обробити «клік» за кнопкою нам потрібно додати відповідний код на JQuary:

```
<script>
    $('.open-cart').on('click',function () {
        $('#exampleModal').modal();
    })
</script>
```

Також є класи для дій, які відбуваються після або перед закриттям модального вікна, що наведені в таблиці 2.6.

Таблиця 2.6 – Типи подій та їх опис

Event Type	Description
show.bs.modal	This event fires immediately when the `show` instance method is called. If caused by a click, the clicked element is available as the `relatedTarget` property of the event.
shown.bs.modal	This event is fired when the modal has been made visible to the user (will wait for CSS transitions to complete). If caused by a click, the clicked element is available as the `relatedTarget` property of the event.
hide.bs.modal	This event is fired immediately when the `hide` instance method has been called.
hidden.bs.modal	This event is fired when the modal has finished being hidden from the user (will wait for CSS transitions to complete).
hidePrevented.bs.modal	This event is fired when the modal is shown, its backdrop is `static` and a click outside the modal or an escape key press is performed with the keyboard option or `data-keyboard` set to `false`.

Наприклад :

```
$('#myModal').on('hidden.bs.modal', function
(event) {
// do something...
})
Дадамо даний код до нашого попереднього скрипта:
<script>
   $('.open-cart').on('click',function () {
      $('#exampleModal').modal();
   })
   $('#exampleModal').on('hidden.bs.modal',
function (event) {
      alert('модальне вікно закрито');
```

```
    })
</script>
```

Content/Images

Для створення адаптивного зображення, яке підлаштовується під розмір батьківського блоку, використовується клас img-fluid. В результаті використання цього класу до зображення застосовуються властивості, що призводить до масштабування зображення:

max-width:100%;

height: auto;

<imgsrc="..."class="img-fluid"alt="...">

Також зображення можна поміщати в рамки та робити для них ревю. Для цього слід підключити клас .img-thumbnail.

```
<div class="container">
<div class="row">
<div class="col-md-12">
<img src="img/port.jpg/" class="img-fluid img-thumbnail " alt="...">
</div>
</div>
<div>
Для вирівнювання зображень слід застосувати вже відомі класи:
<imgsrc="..."class="rounded float-left"alt="...">
<imgsrc="..."class="rounded float-right"alt="...">
```

2.11 Таблиці (Tables)

Таблиці в Bootstrap створюються так само, як і в класичному HTML, але при додаванні певних класів вони змінюють свій зовнішній вигляд і стають більш привабливими й зручними для читання. Для цього використовується спеціальний клас table.

Усі стилі таблиць успадковуються в Bootstrap 4, тобто будь-які вкладені таблиці будуть стилізовані так само, як і батьківська таблиця.

Для прикладу побудуємо таблицю яка наведена в офіційній документації:

```
<table class="table table-bordered table-hover">
<thead class="table-dark">
<tr>
<th scope="col">#</th>
<th scope="col">First</th>
<th scope="col">Last</th>
<th scope="col">Handle</th>
</tr>
</thead>
<tbody>
<tr>
<th scope="row">1</th>
<td>Mark</td>
<td>Otto</td>
<td>@mdo</td>
</tr>
<tr>
<th scope="row">2</th>
<td>Jacob</td>
<td>Thornton</td>
<td>@fat</td>
</tr>
</tbody>
</table>
```

Так в даному прикладі за допомогою класу table створюється таблиця, заголовок виділено чорним кольором клас: table-dark. За допомогою класу table-bordered додано горизонтальні межі таблиці. Якщо потрібно сховати межі таблиці, то для цього є відповідний

клас: borderedless. Для включення підсвітки при наведенні курсором використовується клас table-hover.

Адаптивність таблиць

Для створення адаптивної таблиці слід її розміщувати в контейнері з класом .table - responsive, що призводить до появи горизонтальної прокрутки на екранах менше 768px або на точці брейкпоінту, якщо він вказаний (table-responsive-xl).

Для цього необхідно таблицю «обернути» в даний клас:

```
<divclass="table-responsive">
<tableclass="table">
...
</table>
</div>
```

Більш детально ознайомитись з даними класами можна в офіційній документації:

https://getbootstrap.com/docs/4.6/content/tables/

2.12 Компоненти меню

Компонент Navbar і його елементи

Панель навігації в Bootstrap будується за допомогою елемента Navbar. В неї можна додавати елементи форм, кнопки, текст, посилки.

Компонент має два режими відображення:

– десктопний (звичайний) - виводяться всі елементи меню;

– мобільний (згорнутий) - відображається бренд і кнопка «Гамбургер».

Режим відображення меню залежить від ширини viewport браузера. Точка перелому задається за допомогою css-класу navbar-expand - {sm ... xl}.

Для «NavBar» не обов'язковий контейнер, але якщо необхідно обмежити ширину всього меню, то можна його загорнути в .container.

```
<divclass="container">
<navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
.........
</nav>
</div>
```

Якщо ж потрібно обмежити ширину тільки вмісту, а саме меню відображати на всю ширину сторінки, то .container слід помістити в середину .navbar.

```
<navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
<divclass="container">
.........
</div>
</nav>
```

Для установки кольору оформлення меню використовуються ще два додаткових класи: navbar- {...} і bg- {...}. Для установки кольору фону до bg- {...} необхідно додати будь-яку з восьми колірних схем - primary, secondary, dark, light, warning, danger, info, success. Клас navbar- {...} відповідає за колір тексту й підтримує два варіанти - navbar-dark і navbar-light.

Для позиціонування меню фреймворк пропонує три додаткових класи:

– fixed-top - фіксує меню в верхній частині сторінки;

– fixed-bottom - фіксує меню в нижній частині сторінки;

– sticky-top - приклеює меню зверху при прокручуванні сторінки.

```
<navclass="navbar fixed-top navbar-expand-lg
navbar-dark bg-dark">
```

```
      ..........
      </nav>
      <navclass="navbar fixed-bottom navbar-expand-lg
navbar-dark bg-dark">
      ..........
      </nav>
      <navclass="navbar sticky-top navbar-expand-lg
navbar-dark bg-dark">
      ..........
      </nav>
```

Елементи меню

Компонент «NavBar» складається з декількох підкомпонентів, які можна додавати або виключати в міру необхідності.

Назва або логотип компанії

```
      <navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
      <!-- Текстова посилка -->
      <a class="navbar-brand" href="#">Brand</a>
      ..........
      </nav>
      <!-- *** -->
      <navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
      <!-- Посилка з логотипом -->
      <a class="navbar-brand" href="#">
      <imgsrc="/images/logo.png" width="50" height="50"
alt="...">
      </a>
      ..........
      </nav>
      <!-- *** -->
      <navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
```

```
      <!-- Посилка з логотипом та текстом-->
      <a class="navbar-brand" href="#">
      <imgsrc="/images/logo.png" class="d-inline-block
align-top" width="50" height="50" alt="...">
      Brand
      </a>
      ..........
      </nav>
      <!-- *** -->
      <navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
      <!—Як  заголовок без посилки-->
      <spanclass="navbar-brand mb-0 h1">Brand</span>
          ..........
      </nav>
```

Кнопка для відкриття меню

Кнопка для відкриття меню має css-клас .navbar-toggler і призначена для показу і приховування вмісту меню .collapse.navbar-collapse на маленьких екранах:

```
      <navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
      ..........
      <buttonclass="navbar-toggler" type="button" data-
toggle="collapse"
      data-target="#navbar-example" aria-
controls="navbar-example"
      aria-expanded="false" aria-
label="Togglenavigation">
      <spanclass="navbar-toggler-icon"></span>
      </button>
      ..........
      </nav>
```

Блок вмісту меню

Блок вмісту меню має css-класи .collapse.navbar-collapse і ховається в контрольній точці, яка задається класом navbar-expand-{sm ... xl}. Може бути показано і приховано за кліком на кнопці .navbar-toggler:

<navclass="navbar navbar-expand-lg navbar-dark bg-dark">

.....

<divclass="collapse navbar-collapse" id="navbar-example">

.....

.....

</nav>

Блок посилань меню

Блок посилань меню має css-клас .navbar-nav і складається з посилань і вкладених випадаючих списків з посиланнями:

```
      <navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
         ..........
      <divclass="collapse navbar-collapse" id="navbar-
example">
      <ulclass="navbar-nav mr-auto">
      <liclass="nav-itemactive">
      <a class="nav-link" href="#">
                     Перше посилання <spanclass="sr-
only">(активна)</span>
      </a>
      </li>
      <liclass="nav-item">
      <a class="nav-link" href="#">Друге посилання</a>
      </li>
      <liclass="nav-itemdropdown">
```

```
      <a class="nav-link dropdown-toggle" href="#"
id="navbarDropdown"
      role="button" data-toggle="dropdown" aria-
haspopup="true"
      aria-expanded="false">
      Випадаюче меню
      </a>
      <divclass="dropdown-menu" aria-
labelledby="navbarDropdown">
      <a class="dropdown-item" href="#">Третя</a>
      <a class="dropdown-item" href="#">Четверта</a>
      <divclass="dropdown-divider"></div>
      <a class="dropdown-item" href="#">П"ята</a>
      </div>
      </li>
      <liclass="nav-item">
      <a class="nav-linkdisabled" href="#">Не активна</a>
      </li>
      </ul>
      ..........
      </div>
      ..........
      </nav>
```

!!! Для робити меню, що розгортається обов'язково повинні бути підключені бібліотеки JQ, JS, POPPER у відповідному порядку:

```
      <scriptsrc="bootstrap/js/jquery-3.5.1.slim.min.js"></script>
      <scriptsrc="bootstrap/js/bootstrap.js"></script>
      <scriptsrc="bootstrap/js/popper.min.js"></script>
```

Форма всередині меню

```
      <navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
      ..........
```

```
      <divclass="collapse navbar-collapse" id="navbar-
example">
      ..........
      <formclass="form-inline my-2 my-lg-0">
      <inputclass="form-control mr-sm-2" type="search"
placeholder="Search"
      aria-label="Search">
      <buttonclass="btn btn-outline-info my-2 my-sm-0"
type="submit">Search</button>
      </form>
      </div>
      ..........
      </nav>
```

Текст всередині меню

Клас .navbar-text - дозволяє додавати текст в меню.

```
      <navclass="navbar navbar-expand-lg navbar-dark bg-
dark">
      <spanclass="navbar-brand mb-0 h1">Brand</span>
      <buttonclass="navbar-toggler" type="button" data-
toggle="collapse"
      data-target="#navbar-example" aria-
controls="navbar-example"
      aria-expanded="false" aria-
label="Togglenavigation">
      <spanclass="navbar-toggler-icon"></span>
      </button>
      <divclass="collapse navbar-collapse" id="navbar-
example">
      <ulclass="navbar-nav">
      <liclass="nav-item">...</li>
      <liclass="nav-item">...</li>
      <liclass="nav-item">...</li>
      </ul>
      <!-- Текст всередині меню -->
```

```
      <spanclass="navbar-text mx-
auto">Navbartextwithaninlineelement</span>
      <formclass="form-inline">
      ..........
      </form>
      </div>
      </nav>
```

Контрольні запитання:

1. Що представляє собою Bootsrap?

2. Які переваги Bootstrap?

3. Назвіть головні компоненти Bootstrap.

4. Яку роль відіграє сітка для макета?

5. В яких форматах може бути представлено відображення сповіщень у фреймворку Bootstrap?

6. Які види дизайнів для навігації містить Bootstrap?

7. Скільки класів має система Grid Bootstrap 4?

8. Які типи контейнерів використовується у Bootstrap?

9. Який клас слід використовувати для створення гумового (гнучкого) макета?

10. Який клас слід використовувати для створення відступів?

11. Для чого використовуються класи .order?

12. Які класи застосовують для блоків, які виглядають і розташовуються однаково на всіх пристроях будь-якого розміру?

13. Для яких цілей використовують клас *.no-gutters* для класу .row?

14. Який клас застосовують для створення колонок, що змінюють свою ширину за шириною вмісту?

15. Які є три класи для вирівнювання блоків по горизонталі, які вказуються для всього ряда .row?

16. Які два класи використовуються для вставки на сторінку відео?

17. Для чого застосовується компонент Alert?

18. Що представляють собою модальні вікна побудовані за допомогою HTML, CSS та JavaScript?

19. Що забезпечують картки Bootstrap (Cards)?

20. Який клас застосовують для створення адаптивного зображення, яке підлаштовується під розмір батьківського блоку**?**

21. За допомогою якого класу створюються таблиці в Bootstrap?

22. Який базовий елемент в Bootstrap?

23. За допомогою чого будується панель навігації в Bootstrap?

24. Які два режими відображення має компонент навігації в Bootstrap?

25. З чого складається блок посилань меню компоненту «NavBar»?

26. Що дозволяє виконувати клас .navbar-text?

РОЗДІЛ 3.
JAVASCRIPT

3.1 Вступ до JavaScript (JS)

JavaScript (JS) - об’єктно-орієнтована мова програмування сценаріїв. Оновлення окремих блоків веб-сторінки, інтерактивні карти, анімування графіки, прокручування відео в медіапрогравачі та не тільки можна реалізувати за допомогою цієї мови.

JavaScript має надзвичайно багато застосувань [6][10]. Можна розпочати з малого: створити "каруселі", галереї зображень, динамічні макети сторінок, відповіді на натиски кнопок, тощо. Із досвідом, стає можливим створювати ігри, 2D та 3D графіку, складні застосунки з використанням баз даних та багато іншого!

JavaScript є мовою, що інтерпретується (англ. interpreted programming language) [19]. Це означає, що їй потрібен інтерпретатор для роботи. Веб-браузер - один з таких інтерпретаторів.

Через подібність JavaScript з Java люди часто їх плутають. Це звісно ж різні мови.

(Справа в тому, що при створенні цієї мови, інженери Netscape вирішили зробити йому додаткову рекламу. А однією з найпопулярніших і перспективних мов на той час вважалась Java).

JavaSscript є скрізь

В наш час важко уявити сучасний сайт без JavaScript. Тому, якщо компанія має вебсайт, то спеціаліст з JS їй потрібен. Мова JavaScript є обов'язковою умовою для більшості вакансій у веб сфері й не тільки [9].

З появою Node.js ви можете розробляти на JavaScript не лише клієнтську частину, але й серверну. В таких умовах

розробник(и) спроможні писати проект лише на одній мові, що має свої плюси.

Використовуючи такі інструменти, як Electron, React Native та інші, JavaScript дозволяє створювати програми для настільних комп'ютерів, мобільні додатки, веб-додатки тощо.

3.2 Он-лайн редактори

Щоб почати експериментувати з JavaScript, можна використовувати он-лайн редактори, які дозволяють одразу в браузері писати фрагменти HTML/CSS/JavaScript коду:

- jsfiddle.net;
- jsbin.com;
- codepen.io.

Зручно, бо не потрібно нічого встановлювати й результат доступний звідусіль.

Властивості JS

- Перша властивість - виконання на клієнтському комп'ютері. Якщо ви знаєте, як взагалі працює інтернет, то вам повинно бути відомо, що веб-сторінки, які відображає браузер, створюються (або просто зберігаються) на іншому комп'ютері, так званому сервері. Браузер надсилає запит на сервер і отримує у відповідь HTML-код сторінки. В цьому випадку браузер називається клієнтом. Головне - розуміти, що після того, як сторінка віддана браузеру, сервер вже не може змінити її вміст.

У випадку ж з JavaScript програми, а точніше - скрипти - виконуються прямо в браузері. Це дає таким скриптам можливість отримувати доступ до завантаженої сторінки, і змінювати її.

- Друга властивість - вбудованість. Для того щоб виконувати скрипти, написані на мові JavaScript, не потрібно ніяких

додаткових програм - все необхідне для роботи скрипта вже є в браузері.

Відразу потрібно сказати, що в різних браузерах JavaScript поводиться трохи по-різному. Це не стосується самої мови - одні й ті ж конструкції будуть виконуватися однаково. Вся справа в засобах, які браузер надає скрипту - так, наприклад, добре скрипти працюють в браузері MozillaFireFox, але видають помилки в інших браузерах, наприклад, в AppleSafari. Що правда, ці браузери вже не є такими популярним в наш час і весь код написаний на JS будемо розглядати в найбільш популярному браузері GoogleChrome.

Незважаючи на те, що мова JS найчастіше використовується для невеликих вставок в сторінки, це мова програмування досить потужна. Вона практично не поступається вже згаданій мові Java. Так, наприклад, на цій мові можна створювати (звичайно, не без участі браузера) програми, які мало чим відрізняються від звичних, так званих десктопних.

Щоб не бути голослівними, наведемо приклади:

– Google.com. Ця фірма використовує мову JavaScript в безлічі своїх продуктів. Так, наприклад, відома пошта Gmail практично повністю побудована на JS. А такі проекти, як GoogleMaps або GoogleDocs взагалі неможливі без застосування JavaScript.

– ExtJS.com. Тут можна подивитися, на що здатний JS - найкращим прикладом буде WebDesktop, цілком написаний на цій мові.

– Крім наведених сайтів, можна було б назвати ще тисячі, які в тій чи іншій мірі використовують JS. Але, краще це буде Ваш власноруч написаний сайт).

Інструменти для роботи з JS

За історично сформованною традицією, вивчення будь-якої мови програмування розпочинають з так званого «HelloWorld», тобто програми, яка не робить нічого, крім виведення простого повідомлення. Ми не будемо в посібнику відступати від традиції, але спочатку поговоримо про інструменти які будуть корисні при розробці програм на JS.

Для початку, потрібно мати редактор з підсвічуванням синтаксису. Вибір його - справа смаку. Наприклад, декому у великих проектах зручно використовувати Eclipse IDE або PHPShtorm, а в проектах невеликих - NotePad++. Можливо використовувати той редактор, який більше подобається (хоча, «Блокнот» зі стандартної поставки Windows не рекомендується).

Другий інструмент, без якого дуже складно обійтися, - це відлагоджувач коду, вбудований в GoogleChrome (доступ до якого відбувається натисканням кнопки F12). Саме в консолі розглянемо, як писати перший код на JS та відлагоджувати його.

Як пишеться код на JS

Після вивчення структури, контенту, макета та стилю веб-сторінок (які було розглянуто в попередніх частинах посібника) необхідно на сторінку додати інтерактивності.

У наш час сторінка, на яку можна тільки дивитися, нікому не цікава. Хороші сторінки повинні бути динамічними та інтерактивними, й вони повинні по-новому взаємодіяти з користувачами. Саме для цього й потрібена мова JavaScript. Для початку давайте подивимося, яке місце JavaScript займає в екосистемі веб-сторінок (рисунок 3.1):

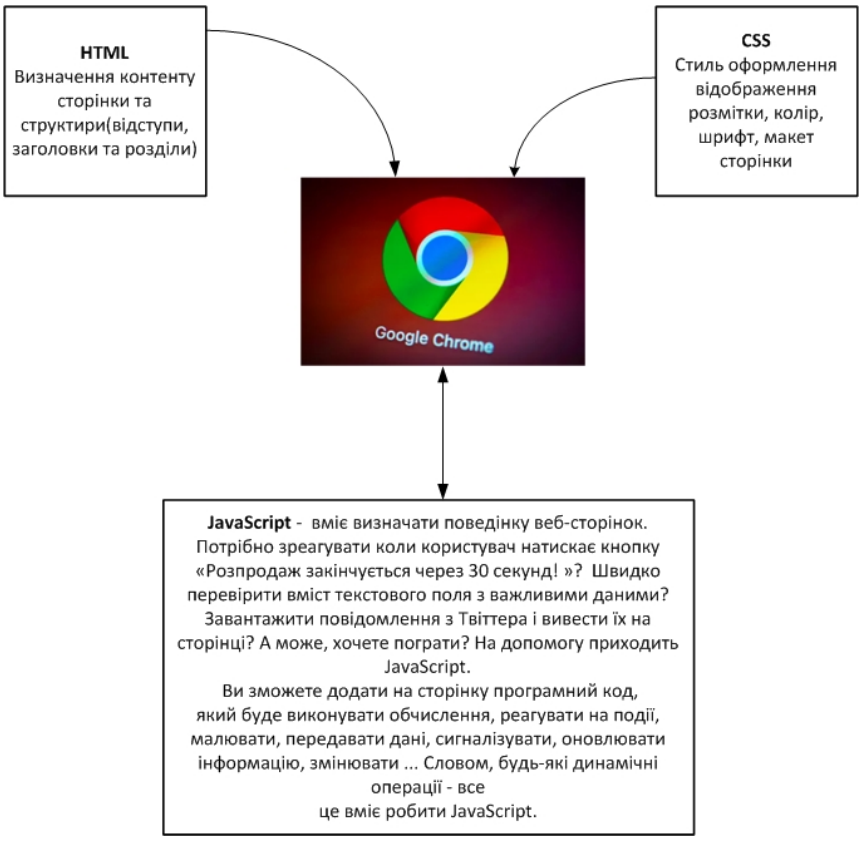


Рисунок 3.1 - Місце JavaScript у програмуванні веб сторінок

JavaScript займає особливе місце в світі програмування. Типова класична програма з'являється наступним чином. Пишеться код, компілюється, проводиться компоновку й встановлюється на комп'ютер.

Мова JavaScript куди більш гнучка й динамічна. Програміст включає код JavaScript прямо в сторінку, а потім завантажує її в браузер. Далі браузер сам зробить все необхідне для виконання написаного коду.

3.3 Написання та виконання коду

Сторінка створюється, зазвичай: з контентом HTML і стильовим оформленням CSS. На сторінку додається код JavaScript. За аналогією з HTML і CSS всі компоненти можна розмістити в одному файлі або ж виділити код JavaScript в окремий файл, який

включається в сторінку. Браузер виконує код, як тільки зустріячає його на сторінці.

Включення коду на HTML сторінку

Виконується за допомогою елемента <script>…. </script>

Візьмемо одну веб-сторінку і визначимо для неї динамічну поведінку в елементі <script>.

```
<!DOCTYPE html>
<htmllang="en">
<head>
    <metacharset="UTF-8">
    <title>Title</title>
    <script>
    alert('HelloWorld!')
    </script>
</head>
<body>
</body>
</html>
```

В розділ <head> сторінки було додано елемент <script> та в ньому записано фрагмент коду на мові програмування JS.

Після збереження і завантаження сторінки буде виконано код, а саме викликається модальне вікно з повідомленням 'HelloWorld!'. Як це наведено на рисунку 3.2.

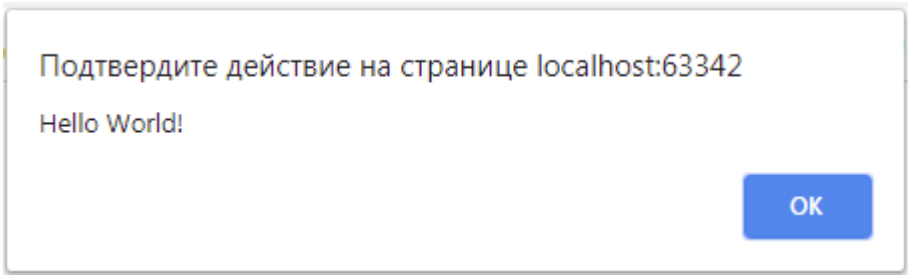


Рисунок 3.2 – Виведення повідомлення в модальне вікно засобами JS

Код JavaScript складається з команд. Кожна команда описує невелику частину операції, що виконується, а весь набір команд визначає поведінку сторінки. Команда виконує невелику частину роботи, наприклад, визначає змінні в яких буде зберігатися інформація.

Іншим варіантом підключення JavaScript є створення окремого файла скрипта. Для цього перейдіть до тестового сайту та створіть нову теку з іменем "scripts" (без лапок). Тоді, всередині нової теки для скриптів, що з'явилась, створіть новий файл з назвою main.js. Збережіть його в теці scripts.

Далі, у файлі index.html введіть наступний елемент в новому рядку просто перед закриваючим тегом </body> :

<script src="scripts/main.js"></script>

Загалом, це те ж саме, що і <link> елемент для CSS - він додає JavaScript до сторінки, тож це може впливати на HTML (також на CSS, та будь що інше на сторінці).

Тепер додайте цей код до файлу main.js:

Var myHeading = document.querySelector('h1');

myHeading.textContent = 'Helloworld!';

Наприкінці, переконайтесь, що файли HTML та JavaScript збережені та приєднані до index.html в браузері.

Результатом виконання коду, наведеного вище буде заміна тексту заголовка на "Helloworld!", використовуючи JavaScript. Це було виконано за допомогою функції querySelector() (en-US) щоб отримати посилання на заголовок та зберегти його у змінній myHeading. Дуже схоже до того, що ми робили, використовуючи CSS селектори. Коли є потреба щось зробити з елементом, необхідно спочатку його вибрати.

Після цього, привласнюється значення змінної myHeading властивості textContent (en-US) (яка містить контекст заголовка), а саме "Helloworld!".

!!!Причиною, чому <script> елемент вставляють ближче до низу HTML файлу, є те, що HTML завантажується браузером в порядку того, як він прописаний в файлі. Якщо JavaScript завантажено першим, а очікується, що він має впливати на HTML під ним, він може не працювати, якщо JavaScript завантажується раніше, ніж HTML, над яким він має здійснювати якісь маніпуляції. Тож, писати підключення JavaScript наприкінці коду HTML це найкраща стратегія.

Змінні та значення

Змінні - контейнери, у яких можна зберігати значення. Можливо розпочати з оголошення змінної за допомогою ключового слова var, після якого вказавши ім'я, яким потрібно назвати змінну.

Змінні потрібні для всіх цікавих у програмуванні речей. Якби змінні не могли змінюватись, то ви не могли б робити ніяких динамічних речей, як то персоналізовані вітальні повідомлення чи зміна відображуваних в галереї зображень.

В JavaScript використовуються змінні, які призначенні для зберігання значень.

Крім чисел, рядків і логічних значень змінні можуть зберігати й інші дані, але незалежно від виду даних всі змінні створюються за одними правилами.

Цифри 0..9, букви латинського алфавіту, нижнє підкреслення та знак долара. Не можна розпочинати з цифри.

Вибір імен змінних

У змінної є ім'я і значення.

- Ім'я змінної має починатися з літери, підкреслення або знака долара.
- Потім можуть слідувати літери, цифри, підкреслення і знаки долара - в будь-якій кількості.
- Як правило, змінну визначають зі слів із великих літер. Наприклад: FootSize (верблюжа нотація).
- Також в JS може бути довільна довжина змінних.

JavaScript це мова чутлива до регістру - myVariable, це не те саме, що myvariable чи MYvariable. Якщо ви маєте проблеми з кодом, можливо, варто перевірити регістр.

У JavaScript є три види оголошення змінних:

– var
– let
– const

Найчастіше використовують var.

Слово **let**- оголошує змінну з блоковою областю видимості. Це дозволяє оголосити локальну змінну з обмеженою поточним блоком { } коду областю видимості. На відміну від **var**, яке оголошує змінну глобально або локально в усій функції незалежно від області блоку.

let var1 [= value1] [, var2 [= value2]] [..., varN [= valueN]];

Приклад:

```
var x=5;
if(x>3){
    lety='недвійка'; //зміна x буде видимою лише у блоці if {...}
    console.log(y);
    }
alert(y); //ReferenceError:  y isnot defined - змінної y не існує
```

Приклад різниці між var і let:

```
function LET(a){
    if(a<0){
         let a=0;
         console.log(a); //0
    }
    console.log(a); //-9
}
function VAR(a){
    if(a<0){
         var a=0;
         console.log(a);  //0
    }
VAR(-9);
```

LET(-9);

Подвійне оголошення змінної у одному ж блоці призведе до синтаксичної помилки коду:

let x=5;

alert(x);

let x=12; //SyntaxError: Identifier'x'hasalreadybeendeclared

Оголошення змінної такої ж самої назви у різник блоках не призводить до помилки:

```
if(10>1){
let x=5; //оголошення змінної x в межах блоку if {...}
alert(x);
}
let x=12;
alert(x);
```

Слово **const** - оголошує константу, доступну лише для читання.

constnames = values [, name2 = value2 [, ... [, nameN = valueN]]];

Параметри:

name - назва константи;

values - значення константи, може містити будь який тип даних.

Оголошення const створює посилання на значення, яке змінити не можна. Константа - це та сама змінна, тільки її присвоєння не можна перезаписати після оголошення (це може призвести до виникнення помилки або браузер просто проігнорує зміну значення й у константи значення не зміниться). Також константу не можна повторно оголошувати.

!!!Константи з'явилися у ECMAScript 6 відповідно старі браузери їх не підтримують!!!

Константи зручно використовувати, тоді коли необхідно, щоб значення змінної не змінювали в подальшому в коді.

Зверніть увагу, що в константі в JavaScript не можливо змінити присвоєне посилання, але можливо змінити значення, на яке посилається константа. Прикладом цього може бути присвоєння об'єкта константі й зміна значення об'єкта:

```
const a={test:'js'};
a.test='JavaScript';
alert(a.test);
```

Змінні JavaScript задаються в наступному форматі й можуть бути оголошені в будь-якому місці коду скрипта:

var cnt;

Можливе завдання початкового значення змінної при її оголошенні:

var name = "значення";

Можна звертатись до змінної, просто вказавши її ім'я:

myVariable;

Після надання змінній значення, ви можете пізніше змінити його:…

var myVariable = 'Bob';

myVariable = 'Steve'; …

JavaScript слабко типізована мова. А саме, якщо це число, то до нього відносяться і цілі числа, і дробові, з плаваючою точкою і т.д.

Основні три типи: Boolean() (використовуються у випадку, коли змінна приймає 2 значення false або true), Number (число), String (текст). Та три допоміжні типи: Undefined (невизначений тип), null (абсолютно пуста змінна) і Object (об’єкти – окрема розмова).

Також змінні можуть змінювати типи.

Тип змінної задається, як правило, форматом її значення. У процесі виконання програми скрипта та сама змінна може мати різний тип залежно від значення, що в ній зберігається. Описана нами змінна name очевидно буде мати рядковий тип. У деяких випадках можемо створювати типізовані змінні для породження екземплярів того або іншого класу об'єктів:

today=newDate();

Тут породжений екземпляр об'єкта Date, що дозволяє здійснювати роботу з датою.

Оператор виведення типу змінних alert(typeof змінна).

Приклад створення змінних:

var win=3;

Так команда з ключовим словом var створює змінну з іменем win та привласнює числове значення 3.

var name = 'Тарас';

Змінній з іменем name привласнюється послідовність символів‘Тарас’, яка називається рядком.

var siElig=false;

Змінній siElig привласнюється значення false (логічна змінна).

var losers;

Змінна оголошена без знаку дорівнюється і власне значення, якщо змінну планується якось використовувати далі в програмі.

Змінні можуть містити значення різних типів (таблиця 3.1).

Таблиця 3.1 – Типи змінних в JS

Тип	**Пояснення**	**Приклад**
String	Послідовність символів, відома як рядок. Щоб визначити, що значенням є рядок, ви мусите занести його в лапки.	varmyVariable = 'Bob';
Number	Число. Без лапок.	varmyVariable = 10;
Boolean	Значення True/False. Слова true та false є спеціальними словами в JS, та не потребують лапок.	varmyVariable = true;
Array	Масив. Дозволяє зберігати багато значень за одним посиланням.	varmyVariable = [1,'Bob','Steve',10]; Виклик значень елементів здійснюється так: myVariable[0], myVariable[1], і т. д.
Object	Загалом, будь-що. Все у JS є об'єктом і може міститись у змінній. Пам'ятайте це.	varmyVariable = document.querySelector('h1'); Всі попередні приклади теж.

3.4 Ключові слова

Не варто плутати JavaScript і використовувати як імена змінних вбудовані ключові слова, такі як var, function або false. Ці імена теж випадають зі списку. В таблиці 3.2 наведено ключові слова JS і використовувати їх в якості змінних не бажано.

Таблиця 3.2 – Ключові слова в мові JS

Break	Do	implements	public	try
case	else	import	return	typeof
catch	enum	in	static	var
class	export	instanceof	super	void
const	extends	interface	switch	while
continue	false	let	this	with
debugger	finally	new	throw	yield
default	for	package	true	
delete	function	private		
	if	protected		

Область видимості змінних

У JavaScript змінні за областю видимості поділяються на: глобальні й локальні.

Глобальна змінна - це змінна, яка доступна для всього коду веб-сторінки.

Локальна змінна - це змінна, яка доступна в певній функції або блоку коду. Якщо об'являти змінну за допомогою var, то змінна локальна в функції, якщо ж за допомогою let, const, то змінна локальна у блоці { ... }.

Глобальна змінна існує доки завантажена веб-сторінка, якщо перезавантажити сторінку, то змінна обновиться. А локальна змінна існує доки виконується функція або є блок.

```
//глобальна змінна
var a=2;
function test(){
//локальна змінна, що доступна лише в блоці функції
var b=2;
}
let s='глобальна змінна s';
if(true){
    let s='локальна змінна s';
    alert( s );
}
alert( s );
const s='глобальна константа s';
if (true){
    const s='локальна константа s';
    alert( s );
}
alert( s );
```

Якщо оголосити локальну змінну, то в глобальній області при зверненні до змінної виникає помилка:

```
function test(){
var a='ЯваСкрипт'; //локальна змінна
alert( a );
}
test(); // ЯваСкрипт
alert('зміна a: '+a); //ReferenceError: a is not
defined
if (true){
const a=2; //локальна константа, яка доступна в
блоці if{...}
    alert(a);
```

```
}
alert(a); //ReferenceError: a is not defined
```

3.5 Синтаксис JS

- Кожна команда завершується символом ";":

x = x + 1;

- Однорядковий коментар починається з двох косих рис (//). Коментарі містять інформацію про код для вас і інших розробників. У програмі вони не виконуються:

// Це коментар. Зайві пропуски дозволені (майже скрізь) x = 2233.

Ви можете залишати коментарі у JavaScript коді, так само як у CSS:

/*

Все, що всередині, є коментарем.

*/

- Рядки повинні знаходитися в подвійних лапках (або одинарні або подвійні, але виберіть щось одне – будьте послідовні):

"You rule!"

'And so do you!'

- Логічні значення true і false записуються без лапок:

rockin = true;

- Змінним не обов'язково привласнювати значення при оголошенні:

vaidth;

- Мова JavaScript, на відміну від розмітки HTML, враховує регістр символів. Іншими словами, Counter і counter – різні змінні.

3.6 Оператори виведення даних в JavaScript

Будь-яка мова програмування немислима без операторів виводу. JavaScript не є виключенням. Виведення даних на екран може відбуватися різними способами. При цьому оператори виводу оптимізовані для найбільш зручного застосування. Найбільш простим є застосування оператора alert(). Аргументом оператора може бути будь-який рядковий вираз. Якщо аргумент має нерядковий тип, то він переводиться в рядковий. Результатом виконання оператора alert є виведення на екран діалогового вікна, вмістом якого є отримане значення. При цьому діалогове вікно буде очікувати натискання користувачем кнопки ОК. Тільки після виконання цієї дії виконання програми й відображення сторінки буде продовжено. Виведення за допомогою вікна оператора alert досить зручно використовувати для контролю значень змінних на тому або іншому етапі виконання програми, тобто при налагодженні.

Також можна застосовувати виведення на консоль, як це наведено в наступному прикладі:

```
console.log('Hello World!');
```

Результат виконання даної команди можемо побачити в консолі - натиснувши F12 та перейшовши на вкладку Console (рисунок 3.3).

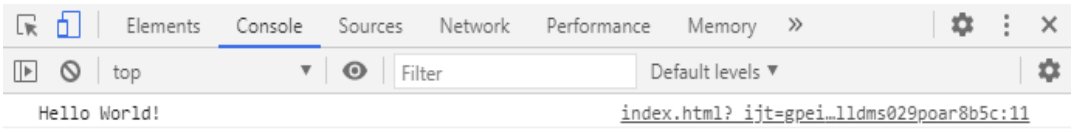


Рисунок 3.3 – Результат виконання оператора console.log

Консоль представляє собою вікно, що динамічно генерується, в яке можливо вводити та виводити інформацію й виконувати програми JavaScript.

Слід зазначити, що функція alert() є методом об'єкта window, що описує поточне вікно браузера. Тому синтаксично більш коректно викликати цю функцію в такий спосіб: window.alert("Текст повідомлення").

Іншим способом виводу інформації на екран є виведення у тіло документа. Організовується він за допомогою оператора write, що є методом об'єкта document, що описує поточний документ, завантажений у вікно.

Оператор document.writeln() відрізняється від оператора document.write() тим, що переносить позицію виводу на новий рядок. Наприклад:

document.write('HelloWorld!');

Виведення тексту відбувається з поточними атрибутами, які мають місце на момент виклику того або іншого оператора виведення. Вираз, що є аргументом оператора виведення, може містити будь-яку рядкову константу, а також містити в собі різні теги HTML. При виведенні подібного виразу ці теги будуть інтерпретуватися відповідним чином. Все це дозволяє будувати HTML-код на льоту, залежно від тих або інших параметрів.

Іноді при виведенні на екран засобами JavaScript виникають проблеми з кодуванням шрифту. Щоб ці проблеми не турбували, необхідно, щоб у настройках браузера була обрана опція автоматичного визначення кодування документа.

Оператори в JavaScript

Оператор, це математичний символ, що видає результат, оснований на двох значеннях (чи змінних). Нижче будуть наведені

деякі з напростіших операторів з деякими прикладами, які можна перевірити у JavaScript консолі.

Оператори введення даних в JavaScript

JavaScript надає нам кілька способів організації введення даних. Перший - використання методу prompt як об'єкта window. Він має наступний синтаксис:

d = window.prompt("Текст повідомлення", "Значення за замовчуванням");

Приклад:

a = prompt('Введіть скільки Вам років');

b = prompt("Введіть рік народження", "1990");

console.log(a, b);

У результаті виконання таких команд на екрані по черзі з’являться модальні вікна запиту, де користувачеві буде представлено запрошення на введення числа років та після введення з’явиться друге модальне вікно з запрошенням ввести рік народження, та текстом за замовчуванням. Після вводу введене значення привласнюється змінним a та b. Якщо користувачем не введено нічого, то змінній b буде присвоєно значення виразу "1990". Це значення буде виведено у вікні запиту й підсвічено так, що, натиснувши кнопку ОК, користувач уведе це значення, а, натиснувши будь-яку іншу кнопку, може приступити до введення своєї інформації. Останній вираз є необов'язковим елементом синтаксису оператора prompt.

Введення значень булевого типу зручніше за все здійснювати за допомогою оператора window.confirm, що має синтаксис: b = confirm("Питання");

how_old_are_you = confirm(**'Вам є 18 років?'**);

console.log((how_old_are_you));

У результаті виконання такої команди на екрані з'явиться вікно із заданим питанням і двома кнопками. Залежно від натискання користувачем тої або іншої кнопки змінна одержить або значення true (кнопка ОК), або false (кнопка Cancel або Скасування).

3.7 Операції зі змінними

Зі змінними можна виконувати такі операції (дії): арифметичні оператори (додавання, віднімання, множення й ділення), модуль числа, інкремент, декремент, унарний плюс, унарний мінус.

Для усіх математичних операцій існує об'єкт Math.

Привласнення змінній значення у JavaScript відбувається за допомогою знаку = .

var x=5; //змінній x привласнюємо 5

Змінній можна привласнювати значення інших змінних:

var x=10, y=6, suma;

suma=x;

//змінній suma привласнюється значення змінної x, тобто suma=10

suma=x+y;

//змінній suma привласнюється значення x+y, тобто suma=10+6

Змінній також можна привласнювати функцію і викликати функцію за назвою змінної. Приклад привласнення змінній функції:

var x=function(a,b){ return(a-b)/2; }

alert(x(109,37));

Додавання +

Додавати змінні можна за допомогою знаку +. Знак додавання використовується як для типів Number, так і для String.

Синтаксис:

var1+var2

15+2;

1+1;

var x=15+6;

var y=4;

x=x+y;

alert(x);

Додавання (з'єднання) рядків також виконується за допомогою оператора +.

'Java'+'Script';

var s1='Текст 1', s2='текст2';

alert(s1+s2); //'Текст 1текст2'

Додавання рядка і числа:

var n=5, s='text';

alert(s+n);

JavaScript автоматично перетворює Number в String (викликається метод Number.toString()). Але інколи буває необхідність власноруч перетворити тип:

```
var s='5', n=6;
alert(n+s); //65
console.log(n.toString()+s); //65
alert(n+Number(s)); //11
var n=10, b=false;
console.log(n+b); //10
alert(n+true); //11
```

У JavaScript буває неточне додавання чисел з плаваючою крапкою**:**

alert(0.3+0.2); //0.5

alert(0.1+0.2); //0.30000000000000004

Інкремент ++

Інкремент збільшує значення змінної на 1. Інкремент виконується за допомогою ++[q].

Синтаксис:

var1++ або ++var1

```
var x=1;
x++;
alert(x); //2
```

Аналог інкременту

```
var x=1;
x=x+1;
alert(x); //0
var x=1;
x+=1;
alert(x); //2
```

Зверніть увагу, що x++ збільшує змінну x на 1, але при цьому повертає попереднє значення. Приклад:

```
var x=1;
alert(x++); //1 - змінну x збільшено на 1 але при цьому повертається значення 1 а не 2
alert(x); //2
// Постфіксний
var x=5 ;
y= x++;  //x=6, y=5
alert('x:'+x+' y:'+y);
// Префіксний
var a=3 ;
b=++a ;  //a=4, b=4
alert('a:'+a+' b:'+b);
```

Декремент --

Синтаксис:

var1-- або --var1

Декремент зменшує значення змінної на 1. Декремент виконується за допомогою знаку --.

```
var x=5;
x--;
alert(x);
var s='5';
s--;
alert(s--); //4
```

Зверніть увагу, що x-- зменшує зміну x на 1 але при цьому повертає попереднє значення. Приклад:

```
var x=1;
alert(x--); //1 - зміну x зменшено на 1 але при цьому вертається 1 а не 0
alert(x); //0
//Постфіксний
var x=5 ;
y= x--;  //x=4, y=5
alert('x:'+x+' y:'+y);
// Префіксний
var a=3 ;
b=--a ;  //a=2, b=2
alert('a:'+a+' b:'+b);
```

Зведення в ступінь **

Синтаксис:

```
var x=2, y=2;
alert(x ** y);
```

Модуль числа %

Модуль числа - залишок при діленні. У JavaScript модуль числа виконується за допомогою оператора %.

```
var x=15, y=6;
alert(x % y);
console.log(4 % 2); //0
var s='';
for(i=1;i<10;i++)
for(j=1;j<10;j++)s+='\n'+i+'%'+j+'='+(i%j);
alert(s);
```

3.8 Арифметичні операції в JS

Таблиця 3.3 – Арифметичні операції в JS

Оператор	Пояснення	Символ	Приклад
Додавання/ Конкатенація	Вживається для додавання двох чисел чи злиття двох рядків разом.	+	6 + 9; "Hello " + "world!";

Віднімання, множення, ділення	Як в математиці.	-, *, /	9 - 3;
			8 * 2; //Множення у JS позначається зірочкою
			9 / 3;

Присвоєння	Присвоєння змінній значення.	=	var myVariable = 'Bob';
Порівняння	Перевіряє чи дві змінні рівні та повертає true/false (Boolean) .	===	var myVariable = 3; myVariable === 4;//false
Заперечення	Повертає логічний	!, !==	Початковий

(обернений до порівняння)	вираз, протилежний значенню операнда; true стає false і т.п. Повертає true, якщо значення не рівні.		вираз true, але повертається false , оскільки це заперечене порівняння : var myVariable = 3; !(myVariable === 3); Ми перевіряємо "чи myVariable НЕ рівне 3". Повертає false, оскільки myVariable РІВНЕ 3. var myVariable = 3; myVariable !== 3;

Існує ще багато операторів, які можна використовувати для програмування [3]. Для довідки можна звернутися: https://developer.mozilla.org/uk/docs/Web/JavaScript/Reference/Operators щоб побачити повний список.

Контрольні запитання:

1. Що представляє собою JavaScript?
2. Які властивості JS?

3. Які є способи розміщення коду JavaScript на HTML-Сторінці?

4. За допомогою якого елемента виконується включення коду JS на HTML сторінку?

5. Як можна оголошувати змінні в JS?

6. Що оголошує слово let?

7. Чим let відрізняється від var?

8. Доки існує глобальна змінна?

9. Доки існує локальна змінна?

10. Який синтаксис в JavaScript?

11. Які оператори виведення даних в JavaScript?

12. Які оператори введення даних в JavaScript ?

13. Для яких операцій існує об'єкт Math у JS?

14. За допомогою якого оператору JS відбувається зведення в ступінь?

15. За допомогою якого оператору JS можна визначити модуль числа?

РОЗДІЛ 4.
DOM МОДЕЛЬ

4.1 HTML DOM (Document Object Model)

XHTML DOM – об’єктна модель Web-документа [2]. По суті це API для HTML та XML документів.

XHTML DOM – це стандартизований підхід до пошуку та маніпулювання вузлами та елементами Web-документа. DOM дозволяє виконувати такі задачі, як пошук вузлів за визначеним критерієм, додавати, вилучати та замінювати вузли Web-документа, звертатись до атрибутів та змінювати вміст елементів XHTML, маніпулювати стилями CSS, а також підключати оброблювачі подій. Практично, в браузерах XHTML DOM реалізується у вигляді об‘єктно-орієнтованого програмного інтерфейсу (API), де об‘єктами є вузли Web-документа, властивостями – їх атрибути, а методи визначають логіку обробки даних вузлів.

Іншими словами, HTML DOM - це стандарт того, як отримувати, змінювати, додавати або видаляти HTML елементи.

Вузли у XHTML DOM розділені на наступні типи:

1. Весь документ – це вузол-документ.
2. Кожний елемент XHTML – вузол-елемент.
3. Текстовий вміст елемента XHTML – це текстовий вузол.
4. Атрибут елемента XHTML – це вузол-атрибут.
5. Коментар – це вузол-коментар.

Після завантаження веб-сторінки браузер створює об'єктну модель документа (анг. Document Object Model) цієї сторінки.

Відповідно до типів вузлів надається визначений список властивостей та методів щодо їх обробки. Особливим вузлом є

вузол-елемент, стандартні властивості та методи якого варіюються в залежності від його призначення.

Наприклад, елементи XHTML-форми мають наступні додаткові властивості, як value, selectedIndex тощо. Як відомо, XHTML-документ являє собою ієрархічну структуру, яка дозволяє виконувати доступ до всіх вузлів Web-документа. Наприклад:

```
<html>
<head>
<title>Заголовок</title>
</head>
<body>
<h1>Привітання</h1>
<p id=”p1”>Текстовий <b>Напівжирний</b>фрагмент</p>
</body>
</html>
```

Тоді, подання цього документа у вигляді дерева буде таким, як на рисунку 4.1.

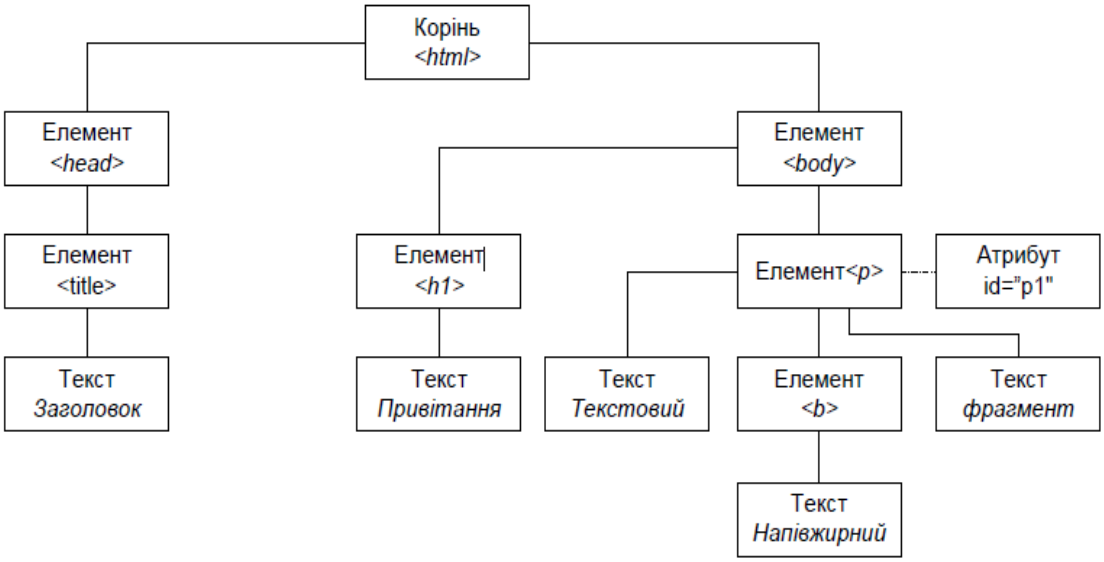


Рисунок 4.1 - Подання Web-документа у вигляді XHTML DOM

Виходячи із деревовидної моделі, доступними є такі дії, як отримання атрибута (attributes) елемента, його батьківського

елемента (parentNode), списку дочірніх елементів (childNodes), попереднього того ж рівня (previousSibling), наступного того ж рівня (nextSibling) та інші.

Серед методів відбору елементів документа можна виділити наступні:

1. getElementById – отримати елемент за його атрибутом id;

2. getElementsByTagName – отримати елементи за назвою тега;

3. getElementsByName – отримати елементи за їх атрибутом name.

Для маніпуляції елементами документа використовують наступні методи:

1. appendChild – додати елемент у кінець списку;

2. removeChild – вилучити елемент;

3. replaceChild – замінити елемент;

4. getAttribute/setAttribute – отримати/встановити значення атрибута та інші.

В основному DOM використовується разом з JavaScript. Тобто код пишеться на JavaScript, але він використовує DOM для доступу до веб-сторінки та її елементів. Проте, DOM створювався, щоб незалежно від конкретних мов програмування була можливість доступу до структурного подання документа через один API. Незважаючи на те, загострімо увагу на JavaScript, реалізації DOM можуть бути створені для будь-якої мови програмування.

Завдяки цій об'єктній моделі JavaScript отримує усі інструменти, необхідні для створення динамічного HTML:

– JavaScript може змінювати всі HTML елементи на сторінці;

– JavaScript може змінювати всі HTML атрибути на сторінці;

– JavaScript може змінювати всі CSS стилі на сторінці;

– JavaScript може видаляти існуючі HTML елементи і атрибути;

– JavaScript може додавати нові HTML елементи і атрибути;

– JavaScript може реагувати на всі існуючі HTML події на сторінці;

– JavaScript може створювати нові HTML події на сторінці.

4.2 Особливості програмування мовою JavaScript в DOM

Мова JavaScript – це інструмент маніпулювання вмістом XHTML-документа. Як було вже розглянуто , ця мова має об‘єктно-орієнтовані можливості, реалізована у вигляді інтерпретатора, вбудована у більшість сучасних Web-браузерів. Це означає, що немає необхідності у використанні інших засобів для виконання програм, крім браузера.

JavaScript має декілька важливих аспектів, які характеризують її як спеціалізовану мову для програмування Web-документів:

1. Мовний аспект. JavaScript має спрощений синтаксис, тому її відносять до так званих «скриптових» мов. Програми мовою JavaScript можуть бути розміщені як всередині XHTML-документа, так і в окремому файлі. Синтаксис операторів JavaScript схожий на синтаксис мов сімейства C, типи даних орієнтовані на специфіку Web-документа: рядки, масиви, об‘єкти та інші. Більше того, усі

типи даних реалізовані, як вбудовані об‘єкти із можливістю доступу до їх властивостей та методів маніпулювання даними цих типів.

2. Аспект DOM. JavasSript надає програмний інтерфейс доступу до усіх складових Web-сторінки: вузлів, елементів, атрибутів тощо. Цей інтерфейс реалізовано у вигляді вбудованих об‘єктів таких, як Document, Event, HTML Element та об‘єктів, специфічних для тегів XHTML: Form, Input, Body, Button, Frame, Image, Anchor тощо. Таким чином, реалізація DOM дозволяє працювати з вмістом Web-документа з одного боку як з XML-документом, з іншого – як з HTML-документом, враховуючи специфіку тегів та їх атрибутів.

3. Аспект браузера. Оскільки Web-документ відображається та оброблюється безпосередньо браузером, мова JavaScript має вбудовані об‘єкти для роботи зі специфічними для браузера компонентами, зокрема Window (вікно документа в цілому), Navigator (інформація про поточний браузер), Screen (інформація про екран користувача), History (дані про вже відвідані сторінки) та Location (інформація про поточну адресу URL сторінки). Ці об‘єкти дозволяють більш ефективно пристосувати відображення сторінки для конкретного браузера користувача.

4.3 Програмний інтерфейс DOM

В DOM всі HTML елементи визначені як об'єкти.

Програмний інтерфейс - це властивості й методи кожного об'єкта.

Властивість - це значення, які ви можете прочитати або встановити (наприклад, зміна вмісту елемента HTML).

Метод - це дія, що ви можете виконати (на кшталт додавання чи видалення елемента HTML).

У наступному прикладі змінюється вміст (innerHTML) елемента <p> з атрибутом id = "demo":

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<p id="demo"></p>
<script>
    document.getElementById("demo").innerHTML =
"Hello World!";
</script>
</body>
</html>
```

У наведеному прикладі getElementById - це метод, а innerHTML - властивість.

Метод getElementById

Найчастіший спосіб отримати доступ до елементу HTML - це використовувати ідентифікатор id елемента.

У попередньому прикладі метод getElementById використовує id = "demo", щоб знайти потрібний елемент.

Властивість innerHTML

Найпростіший спосіб прочитати вміст елемента - це скористатися властивістю innerHTML.

Властивість innerHTML корисна тим, що вона дозволяє прочитати або змінити вміст будь-якого елементу HTML, включаючи <html> і <body>.

4.4 DOM - об'єкт документа

Об'єкт документа document є батьком всіх інших об'єктів на веб-сторінці. Цей об'єкт і представляє вашу веб-сторінку.

Якщо є необхідність отримати доступ до якого-небудь елементу на HTML сторінці, то спочатку потрібно звернутися до об'єкта document.

У таблицях 4.1 - 4.2 представлено кілька прикладів, як можна використовувати об'єкт документа для доступу й маніпулювання HTML.

Таблиця 4.1 - Пошук елементів HTML

Метод	Опис
document.getElementById (id)	Пошук елемента за ідентифікатором id
document.getElementsByTagName(ім'я)	Пошук елемента за іменем тега
document.getElementsByClassName(ім'я)	Пошук елемента за іменем класу

Таблиця 4.2 - Додавання і видалення елементів

Метод	Опис
document.createElement (елемент)	Створює HTML елемент
document.removeChild (елемент)	Видаляє HTML елемент
document.appendChild (елемент)	Додає HTML елемент
document.replaceChild (елемент)	Замінює HTML елемент
document.write (текст)	Записує у зовнішній потік HTML

Таблиця 4.3 - Додавання обробника події

Метод	Опис
document.getElementById (id).onclick = function () {код}	Додає код обробника для події onclick

Таблиця 4.4 - Пошук об'єктів HTML

Властивість	Опис
document.applets	document.applets Повертає всі елементи <applet> (Заборонено в HTML5)
document.baseURI	Повертає абсолютний базовий URI документа
document.body	Повертає елемент <body>
document.cookie	Повертає куки документа
document.doctype	Повертає doctype документа
document.documentElement	Повертає елемент <html>
document.documentMode	Повертає режим, використаний браузером
document.documentURI	Повертає URI документа
document.domain	Повертає доменне ім'я сервера документа
document.embeds	Повертає всі елементи <embed>
document.forms	Повертає всі елементи <form>
document.head	Повертає елемент <head>
document.images	Повертає всі елементи <img>

document.implementation	Повертає реалізацію DOM
document.inputEncoding	Повертає кодування документа (набір символів)
document.lastModified	Повертає дату і час поновлення документа
document.links	Повертає всі елементи <area> і <a>, у яких є атрибут href
document.readyState	Повертає статус завантаження документа
document.referrer	Повертає URL-адресу документа, який завантажував поточний документ
document.scripts	Повертає всі елементи <script>
document.strictErrorChecking	Повертає, чи виконується перевірка помилок чи ні
document.title	Повертає елемент <title>
document.URL	Повертає повний URL документа
document.anchors	Повертає всі елементи <a>, в яких є атрибут name

DOM - Пошук елементів

Далі буде розглянуто, як знайти і отримати доступ до HTML елементам на HTML сторінці.

Пошук HTML елементів

Часто в JavaScript необхідно проводити певні маніпуляції з HTML елементами.

Щоб це зробити, спочатку потрібно знайти потрібний об'єкт. Знайти HTML елемент можна декількома способами:

- за ідентифікатором id;
- за іменем тега;
- за іменем класу;
- за селектором CSS;
- за розділами об'єктів HTML.

Пошук HTML елемента за ідентифікатором

Найпростіший спосіб знайти HTML елемент в DOM - це використовувати його ідентифікатор id.

У наступному прикладі ми шукаємо елемент з id = "intro":

```
var myElement = document.getElementById("intro");
```

Якщо елемент буде знайдений, то він буде повернутий у вигляді об'єкта (в змінну myElement).

Якщо елемент не буде знайдений, то змінна myElement буде містити значення null.

Пошук HTML елемента за іменем тега

У наступному прикладі ми шукаємо всі елементи <p>:

```
var x = document.getElementsByTagName("p");
```

У наступному прикладі спочатку відбувається пошук елемента з id = "main", а потім всіх елементів <p> всередині "main":

```
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
```

Пошук HTML елемента за іменем класу

Якщо потрібно знайти всі HTML елементи з однаковим іменем класу, то використовують метод getElementsByClassName ().

У наступному прикладі повертається список усіх елементів <p> з атрибутом class = "intro":

```
var x = document.getElementsByClassName("intro");
```

Пошук HTML елемента за CSS селектором

Якщо потрібно знайти всі HTML елементи, які підходять за заданим CSS селектором (id, імена класів, типи, атрибути, значення атрибутів і т.п.), використовується метод querySelectorAll ().

У наступному прикладі повертається список усіх елементів з атрибутом class = "intro":

var x = document.querySelectorAll("p.intro");

Приклад пошук HTML елемента за наборами HTML об'єктів

У наступному прикладі проводиться пошук елемента форми з атрибутом id = "frm1" в наборі об'єктів forms, і відображаються всі значення елементів:

```
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i < x.length; i++) {
    text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

Приклад пошук за назвою тега:

```
<head>
<script>
function f()
{
for (i in document.getElementsByTagName('P'))
document.getElementsByTagName('P')[i].style.backgro
undColor="red"
}
</script>
</head>
<body>
<p>Параграф 1</p>
<p>Параграф 2</p>
```

```
<p>Параграф 3</p>
<button onclick="f()">Розфарбувати</button>
</body>
```

В даному прикладі при натисканні кнопки виконується пошук усіх параграфів і для тих, що знайшлись встановлюється червоний колір фону.

Також доступні наступні HTML об'єкти (і набори об'єктів):

document.anchors

document.body

document.documentElement

document.embeds

document.forms

document.head

document.images

document.links

document.scripts

document.title

4.5 DOM - зміна HTML

HTML DOM дозволяє JavaScript змінювати вміст HTML елементів.

Зміна потоку виведення HTML

JavaScript може створювати динамічний HTML контент.

В JavaScript можна використовувати метод document.write () для прямого запису в потік виведення HTML:

```
<DOCTYPE html>
<html>
<body>
<script>
document.write(Date());
</script>
```

```
</body>
</html>
```

Зміна вмісту HTML елемента

Найпростіший спосіб змінити вміст HTML елемента - це скористатися властивістю innerHTML.

Синтаксис:

document.getElementById(id).innerHTML = новий HTML код

У наступному прикладі змінюється вміст елемента <p>:

```
<html>
<body>
<pid="p1">Привіт, світ!</p>
<script>
document.getElementById("p1").innerHTML = " Hello World!";
</script>
</body>
</html>
```

Пояснення прикладу:

Наведений в прикладі HTML документ містить елемент <p> з атрибутом id = "p1" Ми використовуємо HTML DOM, щоб отримати доступ до елементу з id = "p1". JavaScript змінює вміст (innerHTML) елемента на рядок "HelloWorld!"

У наступному прикладі змінюється вміст елемента <h1>:

```
<!DOCTYPE html>
<html>
<body>
<h1 id="id01">Старий заголовок</h1>
<script>
var element = document.getElementById("id01");
element.innerHTML = "Новий заголовок";
</script>
</body>
</html>
```

Зміна значення атрибута

HTML DOM дозволяє змінювати значення атрибутів HTML елементів.

Синтаксис:

document.getElementById(id).атрибут = нове значення

У наступному прикладі змінюється значення атрибута src елемента <img>:

```
<!DOCTYPE html>
<html>
<body>
<img id="myImage" src='smiley.gif'>
<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>
</body>
</html>
```

Наведений в прикладі HTML документ містить елемент <img> з атрибутом id = "myImage". Ми використовуємо HTML DOM, щоб отримати доступ до елементу з id = "myImage" JavaScript змінює значення атрибута src цього елемента з "smiley.gif" на "landscape.jpg".

4.6 DOM - зміна CSS

HTML DOM дозволяє JavaScript змінювати стиль HTML елементів.

Синтаксис:

document.getElementById (id) .style.властивість = новий стиль

У наступному прикладі змінюється стиль елемента <p>:

```
<html>
```

```
<body>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color = "blue";
</script>
</body>
</html>
```

4.7 Оброблювачі подій мовою JavaScript

Більшість дій, які виконує програма мовою JavaScript, реалізуються у вигляді реакції на визначені типи подій від користувача або браузера. Виділяють такі типи подій:

1. Події DOM. Такі події спрацьовують при натисканні, підведенні/відведенні миші над визначеним елементом, натисканні клавіш клавіатури над елементом, який знаходиться в фокусі, власне при отриманні/втраті фокусу, зміні вмісту елементів форм XHTML тощо.

2. Події завантаження. Виникають при закінченні завантаження сторінки, фрейму або зображення, а також при закритті сторінки.

3. Події вікна та інші події. До цієї групи відносять такі події, як зміна розміру екрану, закінчення завантаження даних з сервера тощо.

Додати оброблювач подій в JavaScript можна декількома способами:

1. Використання атрибуту onПодія безпосередньо для елементу DOM, наприклад: Гіперпосилання. В такому випадку значення атрибуту містить фрагмент коду JavaScript. Іншими атрибутами подій можуть бути, зокрема: ondblclick, onblur, onmousemove та інші. Такий спосіб є найпростішим, але використовується

найчастіше у випадках, коли треба виконати невеликий за обсягом код, наприклад, викликати функцію. Проте його використання призводить до небажаного змішування XHTML та JavaScript коду, що ускладнює подальшу підтримку програм.

2. Використання відповідної властивості об‘єкта DOM, наприклад: <body>

```
<p id='myEl'>Параграф</p>
<script>
document.getElementById('myEl').ondblclick =
function() {alert('Подвійне натискання')}
</script>
</body>
```

В наведеному прикладі слід звернути увагу на два важливі моменти. По-перше, оброблювач події в даному випадку являє собою саме функцію, а не рядок, як це було зроблено в попередньому випадку. По-друге, фрагмент JavaScript події описується після опису елемента параграфа, інакше елемент не буде знайдено.

У наступному прикладі змінюється стиль HTML елемента з id = "id1", коли користувач натискає на кнопку:

```
<!DOCTYPE html>
<html>
<body>
<h1 id="id1">Заголовок 1</h1>
<button type="button"
onclick="document.getElementById('id1').style.color =
'red'">
Натисни тут!
</button>
</body>
</html>
```

Код JavaScript може виконуватися під час виникнення будь-яких подій, на кшталт того, коли користувач натискає мишкою на HTML елемент.

Так, наприклад, щоб виконати де-який код при натисканні користувачем на елемент, необхідно додати код JavaScript в атрибут події HTML елемента:

onclick=код JavaScript

Приклади HTML подій:

- коли користувач клацає мишкою;
- коли веб-сторінка повністю завантажилась;
- коли зображення було завантажене;
- коли курсор мишки наводиться на елемент;
- коли поле вводу змінено;
- коли HTML форма відправлена;
- коли користувач натискає на клавішу клавіатури.

У наступному прикладі вміст елемента <h1> змінюється, коли користувач клацає на нього мишкою:

<h1 onclick="**this**.innerHTML = **'Ой!'**">Click on this text!</h1>

У наступному прикладі JavaScript функція викликається з обробника події:

```
<h1 onclick="changeText(this)">Натисніть на цей текст!</h1>
<script>
function changeText(id) {
    id.innerHTML = "Ой!";
}
</script>
```

HTML атрибути подій

Щоб призначити HTML елементу обробник події, потрібно використовувати відповідний атрибут події. У наступному прикладі

призначається обробник події onclick для елемента кнопки. При натисканні користувачем на кнопку, виконується функція displayDate:

<button onclick="displayDate()">Натискати тут!</button>

Події onload і onunload

Події onload і onunload спрацьовують, коли користувач заходить на веб-сторінку або залишає її. Подія **onload** може використовуватися, наприклад, для перевірки типу і версії користувальницького браузера і завантаження відповідного варіанту веб-сторінки, ґрунтуючись на цій інформації.

Ще події onload і onunload можуть використовуватися для роботи з файлами кукі.

<body onload="checkCookies()">

Подія onchange

Часто подія onchange використовується в поєднанні з функціями перевірки полів введення. Нижче показаний приклад, як можна використовувати подію onchange. Функція upperCase() буде викликатися, коли користувач змінює вміст поля введення:

<input type="text" id="fname" onchange="upperCase()">

Події onmouseover і onmouseout

Події onmouseover і onmouseout можуть використовуватися для виклику певних функцій тоді, коли користувач наводить курсор миші на HTML елемент або виводить його з елементу:

Приклад:

```
<div style="background-
color:#D94A38;color:#ffffff;font-weight:bold;font-
size:36px;padding: 8px 16px;"
onmouseover="this.innerHTML='Дякую!'"
onmouseout="this.innerHTML='Наведи на мене'">Навести
мишкою! </div>
```

Події onmousedown, onmouseup і onclick - це все група подій, пов'язаних з натисканням кнопки мишки. Коли натискається кнопка миші, спочатку виникає подія onmousedown, потім, коли кнопка миші відпускається, виникає подія onmouseup, і, нарешті, коли всі ці події закінчилися, виникає подія onclick.

DOM - Метод addEventListener

Додамо обробник події, який спрацьовує при натисканні користувачем на кнопку:

document.getElementById("myBtn").addEventListener("click", displayDate);

Метод addEventListener() приєднує обробник події до певного елементу. При цьому новий обробник події не переписує вже існуючі обробники подій.

Таким чином, ви можете додавати скільки завгодно обробників подій до одного елементу. При цьому це можуть бути обробники подій одного типу, наприклад, дві події натискання мишкою.

Ви можете додавати обробники подій до будь-якого об'єкта DOM, а не тільки HTML елементам, наприклад, до об'єкту вікна.

Метод addEventListener () дозволяє легко контролювати те, як обробник реагує на, так зване, "спливання" події.

Коли використовується метод addEventListener(), JavaScript відділяється від розмітки HTML, що покращує якість сприйняття при читанні скрипта і дозволяє додавати обробники подій навіть тоді, коли ви не можете контролювати розмітку HTML.

Щоб видалити обробник події, потрібно скористатися методом removeEventListener().

Синтаксис:

Елемент.addEventListener(подія, функція, useCapture);

Перший параметр - тип події (наприклад, "click" або "mousedown").

Другий параметр - функція, яка буде викликатися при виникненні події.

Третій параметр - логічне значення (true / false), що визначає, чи слід відправити подію далі ("спливання") або потрібно закрити цю подію. Цей параметр необов'язковий.

Зверніть увагу, що в імені події не використовується префікс "on" - "click" замість "onclick".

У наступному прикладі при натисканні користувачем на елемент з'являється вікно з повідомленням "Hello World!":

Елемент.addEventListener("click", function(){ alert("Hello World!"); });

Також, можна задати і зовнішню "іменовану" функцію:

елемент.addEventListener("click", myFunction);

function myFunction() {

alert ("Hello World!"); }

Метод addEventListener () дозволяє додавати кілька обробників подій до одного і того ж елемента не переписуючи вже існуючі обробники подій:

елемент.addEventListener("click", myFunction);

елемент.addEventListener("click", mySecondFunction);

Також, можна додавати обробники подій різних типів:

елемент.addEventListener("mouseover", myFunction);

елемент.addEventListener("click", mySecondFunction);

елемент.addEventListener("mouseout", myThirdFunction);

Додавання обробника подій до об'єкту window

Метод addEventListener () дозволяє додавати обробники подій до будь-якого об'єкта HTML DOM - HTML елементам, HTML

документу, об'єкту вікна (об'єкт window) та іншим об'єктам, що підтримують події як об'єкт xmlHttpRequest.

У наступному прикладі додається оброблювач події, який спрацьовує, коли користувач змінює розмір вікна браузера:

```
<p id="demo"></p>
<script>
window.addEventListener("resize", function(){
document.getElementById("demo").innerHTML = " Зміна розміру вікна";
});
</script>
```

Передача параметрів

Якщо необхідно передати параметри, то використовуйте "анонімну" функцію, яка викликає спеціалізовану функцію з параметрами:

элемент.addEventListener("click", function(){ myFunction(p1, p2); });

Метод removeEventListener()

Елемент.removeEventListener("mousemove", myFunction);

Модифікація елементів XHTML DOM

Будь-який Web-документ з огляду на інтерфейс програмування, являє собою ієрархічно побудовану структуру вкладених об‘єктів, доступ до яких забезпечується засобами XHTML DOM.

Базовим об‘єктом в даній об‘єктній моделі є window. На першому рівні він включає всі глобальні змінні програми, вбудовані в браузер об‘єкти history, screen, location та navigator, глобальні сервісні функції, а також об‘єкт document, який є проекцією тега <body>. Всередині document автоматично при завантаженні сторінки будується дерево таких об‘єктів, як Form, Link, Table тощо в залежності від наповнення XHTML-документа. Об‘єкт document

та інші елементи Web-документа, успадковані від об‘єктів DOMDocument та DOMElement відповідно. Вказані об‘єкти містять функціональні можливості щодо модифікації елементів XHTML DOM. Наведемо деякі приклади роботи із DOM.

Пошук за значенням атрибуту id:

```
<head>
<script>
function f()
{
var p = document.getElementById('p1')
p.parentNode.removeChild(p)
}
</script>
</head>
<body>
<p id='p1'>Параграф 1</p>
<p>Параграф 2</p>
<button onclick="f()">Вилучити</button>
</body>
```

В даному прикладі при натисканні на кнопку «Вилучити» знаходиться параграф з id=‘p1‘, присвоюється змінній p і потім вилучається зі списку його батьківського елемента (зникає з екрану). Метод getElementById на відміну від інших знаходить завжди один елемент або повертає null.

Пошук елементів за значенням атрибуту name:

```
<head>
<script>
function f()
{
var p = document.getElementsByName('r1')
for (i in p)
p[i].selected = ""
}
```

```
      </script>
      </head>
      <body>
      <form onsubmit="f();return false;">
      <input type="radio" name="r1" selected="selected"
/>V1
      <input type="radio" name="r1"/>V2
      <input type="radio" name="r1"/>V3
      <input type="submit" />
      </form>
      </body>
```

Даний приклад ілюструє роботу методу getElementsByName. При цьому створюється форма з трьома елементами типу «перемикач один з багатьох». При натисканні на кнопку Submit спрацьовує відповідний оброблювач: знаходяться всі перемикачі та в циклі для кожного відміняється його виділення.

4.8 Додавання елементів у Web-документ

Створення нових елементів відбувається завдяки використанню декількох методів: createElement (створити елемент), appendChild (додати елемент у кінець списку), insertBefore (вставити елемент перед визначеним елементом), insertAfter (вставити елемент після визначеного елемента). Розглянемо приклад:

```
      <head>
      <script>
      function f()
      {
      var d = document.getElementById('container')
      for (var i=0;i<3;++i){
      var p = document.createElement('p')
      p.innerHTML="Текст"+i;
      d.appendChild(p)
```

```
}
var a = document.createElement('a')
a.href = "#"
a.innerHTML="Гіперпосилання";
d.insertBefore(a,p)
}
</script>
</head>
<body onload="f()">
<div id="container"/>
</body>
```

В даному прикладі безпосередньо після завантаження сторінки динамічно створюються та додаються три параграфи з текстом Текст1, Текст2 та Текст3 відповідно. Після цього створюється новий об‘єкт типу гіперпосилання і вставляється перед останнім параграфом. Властивість innerHTML використовується для заповнення вмісту елемента. Слід зауважити, що використання змінної p після циклу є коректним, незважаючи на те, що оголошено її всередині циклу. Це обумовлено тим, що в Javascript найменша локальна область видимості – функція.

Вилучення елементів

Для вилучення елементів з DOM використовується функція removeChild:

```
<head>
<script>
function f()
{
var c = document.getElementById('container')
c.parentNode.removeChild(c)
}
</script>
</head>
<body onload="f()">
```

```
<div id="container">Вміст</div>
</body>
```

В цьому прикладі з документа вилучається елемент div безпосередньо при завантаженні сторінки. Слід зауважити, що вилучення відбувається зі списку батьківського елемента, що ілюструється у функції f().

Заміна елементів

Заміна елементів відбувається завдяки використанню метода replaceChild, наприклад:

```
<head>
<script>
function f()
{
var im1 = document.getElementById('im1')
var im2 = document.getElementById('im2')
var im_im2_old =
im1.parentNode.replaceChild(im1,im2)
im1.parentNode.insertBefore(im_im2_old,im1)
}
</script>
</head>
<body onload="f()">
<img src="i1.gif" id="im1" />
<img src="i2.gif" id="im2" />
</body>
```

В даному прикладі у функції f() спочатку отримуються вказівники на зображення im1 та im2. Після цього на місце im2 переноситься im1 (вказівник на старе значення im2 повертає функція), а останнім рядком вилучене значення im2 вставляється перед im1.

4.9 Обробка форм XHTML

Валідація даних форм

Часто в задачах обробки форм виникає необхідність у перевірці коректності введених користувачем даних у елементи форм. Зокрема, це відноситься до перевірки коректності введення дат, електронної адреси тощо. Для виконання валідації даних використовують події форми та її елементів, а також, як правило, регулярні вирази. Наведемо приклад перевірки введення рядка в текстове поле за шаблоном «тільки літери у нижньому регістрі та символ підкреслення».

```
<head>
<script>
function f()
{
var v = document.getElementById("a").value
var p=new RegExp("^[a-z_]+$","m")
var res = p.test(v)
if (!res) alert('Недопустимі символи')
return res
}
</script>
</head>
<body>
<form action="http://example.com" onsubmit="return
f()">
Поле [a..z_]<input type="text" id="a" value=""/>
<input type="submit" value="Перевірити"/>
</form>
</body>
```

В даному прикладі використовується оброблювач події форми - onsubmit, який перед відправленням форми перевіряє задану умову. Якщо результат перевірки негативний, виводиться

повідомлення та відправлення відміняється (return false). Функція f() виконує перевірку співпадіння введених даних із шаблоном регулярного виразу.

Створення та вилучення елемента форми

Наступний приклад ілюструє можливості додавання та вилучення елементів форми:

```
      <head>
      <script>
      function add()
      {
      var b = document.createElement("input")
      b.type="text"
      b.id="i"+document.getElementById("frm").childNodes.
length
      document.getElementById("frm").appendChild(b)
      }
      function del()
      {
      var cnt =
document.getElementById("frm").elements.length
      if (cnt<3) return
      var last =
document.getElementById("frm").elements[cnt-1]
      document.getElementById("frm").removeChild(last)
      }
      </script>
      </head>
      <body>
      <form id="frm">
      <input type="button" onclick="add()"
value="Створити"/>
      <input type="button" onclick="del()"
value="Вилучити останній"/>
      </form>
```

```
</body>
```

В даному прикладі виводяться дві кнопки, перша додає елемент уведення і встановлює йому атрибут «текстовий». Друга функція вилучає останній в списку елемент. Якщо елементів залишилось два (кнопки), то вилучення відміняється.

4.10 Обробка даних елементів форм

Наступний приклад ілюструє обробку даних елементів форми: для значень двох текстових полів виконується обчислення їх суми. У випадку, коли введені значення не числові, виводиться повідомлення про помилку:

```
<head>
<script>
function sum()
{
var a = document.getElementById("a").value
var b = document.getElementById("b").value
var res = document.getElementById("res")
if (isNaN(a) || isNaN(b)) res.value = "невірні
дані"; else res.value = Number(a)+Number(b)
}
</script>
</head>
<body>
<form id="frm">
<input type="text" id="a" value=""/><br/>
<input type="text" id="b" value=""/><br/>
<input type="text" id="res" readonly="readonly"
value="Результат"/><br/>
<input type="button" onclick="sum()"
value="Обчислити суму"/>
</form>
</body>
```

Функція isNaN() перевіряє чи є передане значення невірним числом.

Ключові моменти DOM:

– Об'єктна модель документа, або DOM, - внутрішнє поданням веб-сторінки в браузері.

– Браузер створює модель DOM сторінки в процесі завантаження і розбору розмітки HTML.

– Для отримання доступу до DOM в коді JavaScript використовується об'єкт document.

– Об'єкт document має властивості й методи, які можуть використовуватися для отримання доступу й модифікації DOM.

– Метод document.getElementById отримує елемент з DOM за ідентифікатором.

– Метод document.getElementById method повертає об'єкт element, що представляє елемент сторінки.

– Об'єкт element має властивості й методи, за допомогою яких можна прочитати вміст елемента й змінити його.

– Властивість innerHTML містить текстовий вміст елемента, а також всю його вкладену розмітку HTML.

– Щоб змінити вміст елемента, необхідно змінити значення його властивості innerHTML.

– При зміні елемента (за допомогою зміни його властивості innerHTML) результат негайно відображається на веб-сторінці.

– Метод getAttribute дозволяє отримувати значення атрибутів елементів.

– Метод setAttribute дозволяє задавати значення атрибутів елементів.

– Якщо код розміщується в елементі <script> в секції <head> сторінки, простежте за тим, щоб він не намагався змінювати DOM до того, як сторінка буде повністю завантажена.

– Властивість onload об'єкту window може використовуватись для призначення обробника події (або функції зворотного виклику) для завантаження сторінки.

– Обробник події властивості onload об'єкту window викликається відразу ж після того, як сторінка буде повністю завантажена.

– Існує багато різних подій, які можуть оброблятися з коду JavaScript за допомогою функцій-обробників.

Контрольні запитання:

1. Що представляє собою XHTML DOM?

2. Який об’єкт є батьком всіх інших об'єктів на веб-сторінці?

3. Чи HTML DOM дозволяє JavaScript змінювати стиль HTML елементів?

4. Який об’єкт використовується для отримання доступу до DOM в коді JavaScript?

5. Що можна виконати за допомогою методу document.getElementById?

6. Який найпростіший спосіб змінити вміст HTML елемента?

7. Що представляє собою об'єкт element?

8. Що дозволяє виконувати метод getAttribute?

9. Що дозволяє метод setAttribute?

10. На які типи поділяють події?

11. Який метод приєднує обробник події до певного елементу?

12. Як можна застосовувати події onload і onunload?

13. В яких випадках можуть використовуватись події onmouseover і onmouseout?

14. Які події пов’язані з натисканням кнопки миші?

15. Який метод дозволяє додавати кілька обробників подій до одного і того ж елемента не переписуючи вже існуючі обробники подій?

Список скорочень

HTML (Hypertext Markup Language) - це мова тегів, засобами якої здійснюється розмічання вебсторінок для мережі Інтернет.

CSS (англ. Cascading Style Sheets) - каскадні таблиці стилів, спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних.

TCP (Transmission Control Protocol) - протокол керування передачею – протокол орієнтований на роботу з підключеннями і передає дані у вигляді потоків байтів.

HTTP (Hyper Text Transfer Protocol) - протокол передачі даних (гіпер-текстових документів).

HTTPS (Hyper Text Transmission Protocol, Secure - протокол захищеної передачі гіпер-текстових документів) - HTTP в сукупності з SSL (Secure Sockets Layer) - протоколом захищених сокетів.

IDE (англ. Integrated Development Environment) - інтегроване середовище розробки.

SEO (англ. search engine optimization) - процес коригування HTML-коду, текстового наповнення (контенту), структури сайту, контроль зовнішніх чинників для відповідності вимогам алгоритму пошукових систем, з метою підняття позиції сайту в результатах пошуку в цих системах за певними запитами користувачів.

CDN (Content Delivery Network) - географічно розподілена мережева інфраструктура, що дозволяє оптимізувати доправлення та розповсюдження контенту кінцевим користувачам в мережі Інтернет.

API (Application Programming Interface) - інтерфейс прикладного програмування

JS (JavaScript) - об’єктно-орієнтована мова програмування сценаріїв.

HTML DOM (Document Object Model) - об’єктна модель Web-документу.

ANSI (American national standards institute) - Американський національний інститут стандартів.

WWW (World Wide Web) - всесвітня павутина.

Список використаних джерел

1. Дженніфер Роббінс HTML 5. Карманный справочник.- Диалектика, 2020.-192 с.
2. Мельник Р. Програмування веб-застосувань (фронт-енд та бек-енд) Видавництво: Львівська політехніка, 2018, 248с.
3. Resources for developers, by developers. MDN Web Docs [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web (дата звернення: 18.01.2022)
4. Metanit.com. Сайт о программировании [Електронний ресурс] – Режим доступу: https://metanit.com/web/html5/1.1.php (дата звернення: 18.01.2022)
5. Квинт И. HTML, XHTML и CSS на 100%. Издательство:Питер, 2010, 384с.
6. Никсон Р. Создаем динамические вебсайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. Питер, 2015, 688с.
7. Перепелица Ф.А. Эффективная разработка веб-сайтов. Bootstrap. – СПб: Университет ИТМО, 2015.– 71 с. https://books.ifmo.ru/file/pdf/1828.pdf
8. Build fast, responsive sites with Bootstrap [Електронний ресурс] – Режим доступу: https://getbootstrap.com/docs/5.1/getting-started/introduction (дата звернення: 18.01.2022)
9. Хьюго Ди Франческо Простыми словами о «фронтенде» и «бэкенде»: что это такое и как они взаимодействуют [Електронний ресурс] – Режим доступу: https://tproger.ru/translations/frontend-backend-interaction/ (дата звернення: 19.01.2022)

10. Пасічник О.Г., Пасічник О.В., Стеценко І.В. Основи веб-дизайну Видавництво: Вид група BHV, 2009, 336с. [Електронний ресурс] – Режим доступу: http://school1k24.at.ua/10CLASS_WEB/OsnovyWebDis.pdf (дата звернення: 20.01.2022)

11. Манако В., Манако Д., Данилова О., Войченко О.П. Основи будування сайтів Видавництво: Шкільний світ, 2006, 120с.

12. Дунаев В. В. Web-программирование для всех. Издательство: БХВ_Петербург, 2008, 560с.

13. Все необходимые инструменты для разработчиков и команд. JetBrains [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/ru-ru/ (дата звернення: 20.01.2022)

14. Open Server [Електронний ресурс] – Режим доступу: https://ospanel.io/ (дата звернення: 20.01.2022)

15. David Boureau Как называть css-классы [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/303174/ (дата звернення: 21.01.2022)

16. Less [Електронний ресурс] – Режим доступу:http://lesscss.org (дата звернення: 21.01.2022)

17. Stylus. Expressive, dynamic, robust CSS [Електронний ресурс] – Режим доступу:https://stylus-lang.com/ (дата звернення: 21.01.2022)

18. Sass. CSS with superpowers [Електронний ресурс] – Режим доступу:https://sass-lang.com/ (дата звернення: 21.01.2022)

19. Ричард Вагнер, Аллен Вайк. JavaScript. Энциклопедия пользователя. Издательство: ДиаСофт, 2001 г., 464 стр.

Навчальне видання

Босько Віктор Васильович

Константинова Лілія Володимирівна

Марченко Костянтин Миколайович

Улічев Олександр Сергійович

WEB-ПРОГРАМУВАННЯ
ЧАСТИНА 1 (FRONTEND)

Навчальний посібник

Українською мовою

Редактор: Константинова Л. В.