МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ МЕНЕДЖМЕНТУ
Кафедра економічної кібернетики, комп'ютерних наук та інформаційних технологій

ВЕБТЕХНОЛОГІЇ ТА ВЕБДИЗАЙН

Конспект лекцій
для здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» денної форми здобуття вищої освіти

Миколаїв
2024

УДК 004.777
В26

Друкується за рішенням науково-методичної комісії факультету менеджменту Миколаївського національного аграрного університету від 08 лютого 2024 року, протокол № 7.

Укладачі:

О. Ю. Пархоменко – канд. фіз-мат. наук, доцент, доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій, Миколаївський національний аграрний університет;

С. І. Тищенко – канд. пед. наук, доцент, доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій, Миколаївський національний аграрний університет;

Рецензенти:

І. П. Атаманюк – д-р техн. наук, професор, професор кафедри вищої та прикладної математики, Миколаївський національний аграрний університет;

Р. В. Дінжос – д-р техн. наук, професор кафедри фізики та математики, Чорноморський національний університет ім. Петра Могили.

Вебтехнології та вебдизайн : конспект лекцій / уклад. О. Ю. Пархоменко, В-26 С. І. Тищенко. Миколаїв : МНАУ, 2024. 43 с.

Конспект лекцій призначений для вивчення теоретичних та інструментальних аспектів вебтехнологій та вебдизайну. Містить навчальні матеріали з основних тем курсу «Вебтехнології та вебдизайн», що передбачені освітньо-професійною програмою «Комп'ютерні науки» першого (бакалаврського) рівня вищої освіти за спеціальністю 122 «Комп'ютерні науки», галузі знань 12 «Інформаційні технології. До кожної теми подаються докладні теоретичні відомості та практичні приклади їх застосування. Даний посібник буде корисним здобувачам та викладачам вищої освіти.

УДК 004.777

ЗМІСТ

Базові технології фронтенду та бекенду……4
Основи HTML……...18
Основи CSS……...37

БАЗОВІ ТЕХНОЛОГІЇ ФРОНТЕНДУ ТА БЕКЕНДУ

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

Веб-технології стрімко розвиваються і розробники мають широкі можливості створювати веб-вміст нового покоління. Сьогоднішній Інтернет є результатом безперервних зусиль відкритої веб-спільноти, яка розробляє новітні технології і домагається їх підтримки всіма браузерами.

Будь-який сучасний сайт чи додаток має дві частини, що тісно взаємодіють між собою. Ці частини зазвичай вимагають окремої розробки. Розробляти клієнтську та серверну частину може як один розробник, так і велика команда фахівців. Це залежить від складності веб-проекту та бюджету.

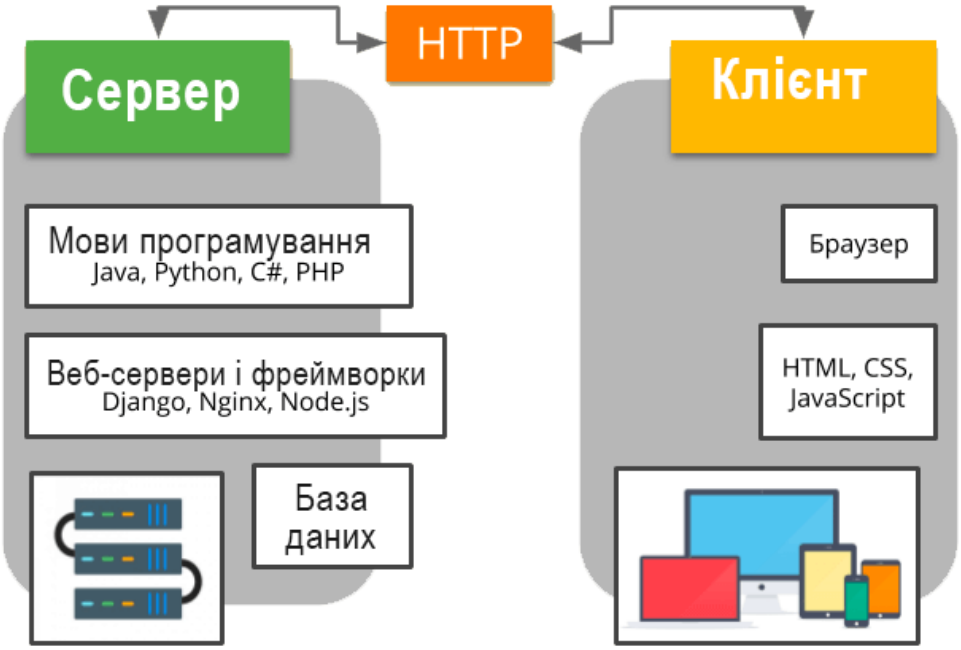

Рис.1. Взаємодія серверної та клієнтської частин веб-програми

Фронтенд розробка

Для написання клієнтського коду зазвичай використовують комбінацію HTML (відповідає за структурування змісту сторінки), CSS (надає веб-сторінці певного вигляду), JavaScript (клієнтська мова

програмування). Браузер розуміє код цих мов, тому, для відтворення сайту (додатку) не потрібно жодних налаштувань.

Розробка клієнтської частини додатків пов'язана з низкою фронтенд-фреймворків (Angular, React тощо).

Серверна частина (Backend)

Серверний компонент складається з 2 частин – логіки додатку (сервер додатку) та бази даних. Перший – головний центр управління веб-додатком, другий – місце, де зберігається інформація.

Розробка серверної частини додатків пов'язана з низкою мов програмування (PHP, Java, Python, C#, C++), а спрощення такої розробки досягається використанням бекенд-фреймворків (Django, Laravel тощо) та веб-серверів (Nginx , Node.js тощо).

Бази даних потрібні для зберігання масивів даних, які при запиті користувача виймаються та відображаються у веб-додатку. На практиці можуть використовуватися різні бази даних, найбільш популярні з яких: PostgreSQL, MySQL, MongoDB. Для роботи з базами даних існує багато бібліотек, орієнтованих на різні серверні мови програмування.

Базові інструменти фронтенд-розробки

HTML – мова розмітки тексту

Мова гіпертекстової розмітки HTML (HyperText Markup Language) є основним будівельним засобом для веб-сторінок, використовується для створення та візуального представлення веб-сторінок. Визначає структуру і описує зміст веб-сторінки в структурованій формі.

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

HTML дозволяє формувати на сторінці сайту текстові блоки, додавати до них зображення, організовувати таблиці, додавати до дизайну сайту звуковий супровід, організовувати гіперпосилання з переходом до інших розділів сайту або ресурсів Інтернету і компонувати всі ці елементи між собою. За допомогою HTML можна

створити як статичний так і динамічний сайт. Сторінки, які створено лише засобами HTML мають розширення .html.

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

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

На сьогодні актуальною є версія HTML5/

Елементи, що знаходяться всередині елементу <html>, утворюють дерево документа, так звану об'єктну модель документа DOM (Document Object Model).

DOM (Document Object Model)

Об'єктна модель документа – це модель документа як об'єкта, створюється веб-браузером у пам'яті пристрою на підставі коду HTML, що отримано від сервера.

1. Сервер створює HTML код сторінки або віддає його (якщо це простий статичний сайт).
2. Браузер отримує код, аналізує його та розбирає на елементи дерева.
3. За необхідності підключається JavaScript, якщо він використовується, щоб змінити поведінку тегів та їх вмісту залежно від впливу користувача.
4. DOM-дерево відображається у вкладці браузера у вигляді, який задуманий розробниками.

HTML-код, який пишуть програмісти – це лише текстовий файл певного формату, а DOM – результат дій браузера, який створює об'єкти при розборі текстових файлів.

Завдяки W3-консорціуму вироблено єдиний стандарт побудови та аналізу вмісту веб-сторінки. До цього різні браузери діяли по-своєму,

що створювало багато незручностей для розробників. Ознайомитись з актуальним DOM-стандартом можна на офіційному сайті.

DOM представляє документ у вигляді дерева, де кожен елемент сторінки представлений як вузол дерева. Ці вузли поєднуються в ієрархічну структуру елементів веб-сторінки, між якими задіяні "родинні стосунки". Відносини між множинними вкладеними елементами підрозділяються на батьківські, дочірні та сіблінгові (від англ. sibling - діти, що мають спільних батьків).

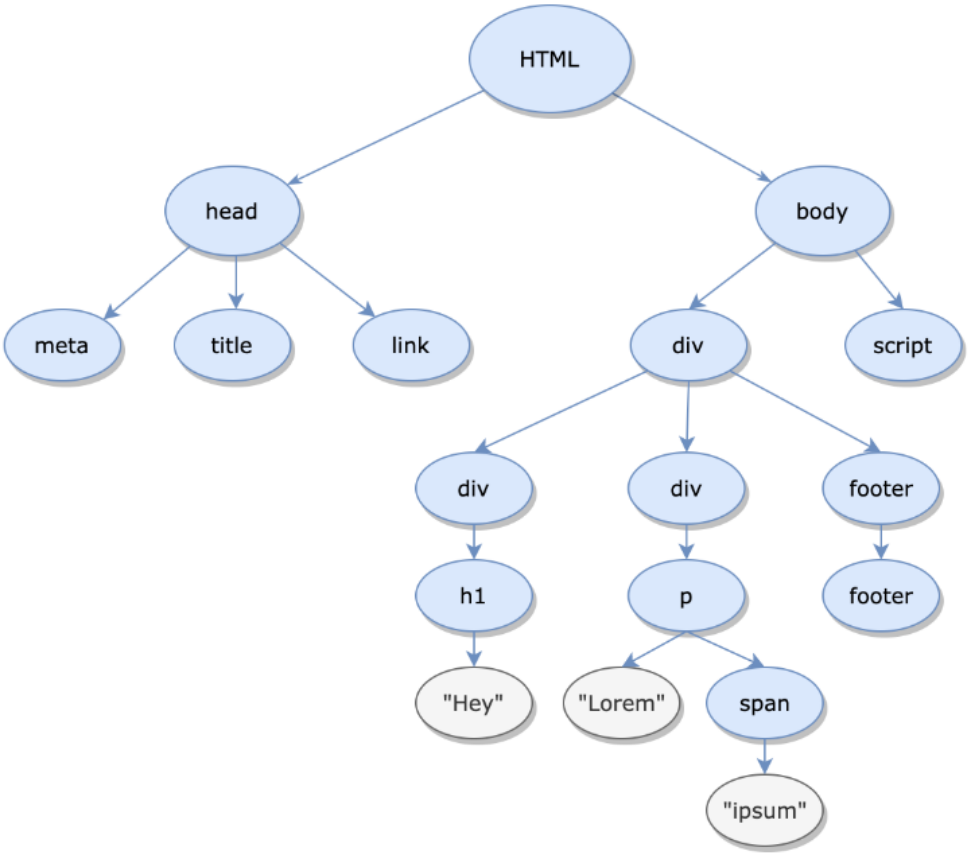

Рис. 2. Найпростіша DOM структура веб-сторінки

Основні структури, які використовуються в об'єктній моделі документа:

• **Дерево.** Веб-сторінка представляється як ієрархічне перевернуте дерево, що починається з головного елемента <html> і розширюється до низу. У кожного об'єкта дерева є батько і нащадки, якщо він знаходиться не в самому низу.

•**Батьківський елемент** - елемент, що пов'язаний з іншими елементами нижчого рівня і знаходиться на дереві вище за них. На рис.2 <html> є батьківським лише для <head> і <body>. Елемент

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

•**Дочірній елемент** - елемент, що розташований всередині іншого елементу. На рис.2 елементи <h1>, <h2>, <p> і <nav> є дочірніми по відношенню до <body>.

•**Сіблінговий елемент** - елемент, що має спільний батьківський елемент з даним, це елементи одного рівня. На рис.2 <head> і <body> є елементами одного рівня, так само як і елементи <h1>, <h2> і <p> є між собою сіблінговими.

Модель DOM є потужним інструментом для створення інтерактивних та динамічних веб-сторінок. Він дозволяє розробникам створювати складні користувацькі інтерфейси, керувати даними і забезпечувати взаємодію користувача з веб-сайтом. Однією з найпоширеніших мов програмування, які використовуються для роботи з DOM у веб-розробці є JavaScript.

DOM надає багато способів доступу до елементів веб-сторінки. Можна отримати доступ до елементів за їх тегом, класом, ідентифікатором або іншими атрибутами. За допомогою DOM можна змінювати вміст та атрибути елементів веб-сторінки. Це дозволяє динамічно оновлювати вміст сторінки без перезавантаження. DOM підтримує обробку подій, таких як клацання мишею, натискання клавіш та інші дії користувача.

CSS – каскадна таблиця стилів

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

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

Переваги CSS:

•Застосування кількох варіантів дизайну сторінки для перегляду на різних пристроїв, наприклад, для комп’ютера, планшету чи телефону.

•Зменшення часу завантаження сторінок сайту за рахунок перенесення правил опису даних до окремого CSS-файлу. В цьому випадку браузер завантажує лише структуру документа і дані, що містяться в HTML коді. CSS-файл з правилами опису цих даних завантажується браузером лише один раз і зберігається в кеші браузера.

•Простота подальшої зміни дизайну. Не потрібно виправляти кожну сторінку, достатньо лише змінити кілька правил у CSS-файлі.

•Додаткові можливості оформлення. Наприклад, за допомогою CSS-правил можна застосувати обтікання певного блоку текстом або зробити так, щоб меню фіксовано знаходилося в певному місці при перегортанні сторінки.

На сьогодні актуальною є версія CSS3.

JavaScript – мова сценаріїв

JavaScript – це фрагменти програмного коду (скрипти), що надають динаміки для певних елементів сторінки. Програмний код на JavaScript додається до HTML-сторінки і інтерпретується браузером в міру завантаження цього документа. За допомогою JavaScript можна динамічно реагувати на події, які пов'язані з діями відвідувача або змінами стану сторінки чи вікна.

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

Бекенд розробка. Програмування серверної частини

Веб-додатки і більшість веб-сайтів використовують програмування серверної частини, щоб за потребою динамічно відображати різні дані, які в основному витягуються з бази даних.

Бекенд веб-сайту — це серверна частина сайту, прихована від очей користувача. Це закладання певної логіки та написання серверних сценаріїв, що керують функціями і процесами сайту. Бекенд розробляється з використанням іншого стеку технологій, ніж у фронтенді.

Коди, задіяні у серверній частині та клієнтській частині значно відрізняються:

- Вони мають різні цілі та призначення.

- Як правило, вони не використовують однакові мови програмування. Виняток складає JavaScript, який можна використовувати на стороні сервера та клієнта.
- Вони виконуються у різних середовищах операційної системи.

Код, який виконується в браузері, відомий як код клієнтської частини, перш за все пов'язаний з зовнішнім виглядом і поведінкою веб-сторінки. Це включає вибір і стилізацію компонентів інтерфейсу користувача, створення макетів, навігацію, перевірку форм тощо.

Код клієнтської частини написаний з використанням HTML, CSS та JavaScript. Він запускається у браузері і практично не має доступу до базової операційної системи (включаючи обмежений доступ до файлової системи).

Програмування на стороні сервера в основному включає вибір вмісту, який повертається браузеру у відповідь на запити. Код на стороні сервера обробляє такі завдання, як перевірка надісланих даних та запитів, використання баз даних для зберігання та витягування даних та надсилання правильних даних до клієнта.

Код серверної частини може бути написаний на будь-якій мові програмування (PHP, Python, Ruby, C# і NodeJS). Код серверної частини має повний доступ до операційної системи сервера.

Дії у серверній частині

Програмування серверної частини дозволяє ефективно доставляти актуальну інформацію, складену для індивідуальних користувачів.

Ефективне зберігання та доставка інформації

Програмування серверної частини дозволяє зберігати інформацію в базі даних і динамічно створювати та повертати HTML чи інші типи файлів (наприклад, PDF, зображення тощо).

Сервер не обмежений у надсиланні інформації з баз даних і повертає результат виконання певних сценаріїв. Контент може бути цільовим щодо пристрою клієнта, який його отримує.

Налаштований контент для користувача

Сервери можуть зберігати та використовувати інформацію про клієнтів, щоб постачати індивідуальний контент. Глибокий аналіз звичок користувача може бути використаний для прогнозування їхніх інтересів та подальших налаштувань відповідей та повідомлень, наприклад, надання списку раніше відвіданих популярних місць на карті.

Контрольований доступ до контенту

Авторизація - центральна частина взаємодії користувача з веб-додатком. Програмування серверної частини дозволяє сайтам обмежувати доступ не авторизованим користувачам та надавати лише ту інформацію, яку користувачеві дозволено бачити.

Реальні приклади:

- Соціальні мережі, такі як Facebook, дозволяють користувачам повністю контролювати свої дані, але друзям дозволяти переглядати чи коментувати. Користувач визначає, хто може переглядати його дані та чиї дані з'являються на його стіні.
- Блог контролює доступ до контенту: статті видно всім, але тільки авторизовані користувачі можуть коментувати чи редагувати контент.
- На сайті онлайн-банкінгу доступ до контенту контролюється. Без авторизації на ньому практично нічого не можна зробити, окрім перегляду загальної та довідкової інформації. Після авторизації стають доступними всі дозволені фінансові операції.

Зберігання інформації про сесію/стан

Програмування серверної частини дозволяє розробникам використовувати сесії. Це механізм, що дозволяє серверу зберігати інформацію про поточного користувача сайту та надсилати різні відповіді на основі цієї інформації.

Наприклад, якщо користувач був попередньо авторизований, то виводити історію замовлень або зберегти прогрес простої гри, так щоб користувач міг повернутися на сайт продовжити з того місця, де він закінчив.

Повідомлення та засоби зв'язку

Через програмування серверної частини можуть надсилатися повідомлення електронною поштою, смс, миттєві повідомлення, відеозв'язок або інші засоби зв'язку.

- Facebook або Twitter надсилає повідомлення електронною поштою та смс-повідомлення, щоб повідомити про нові розмови.
- Amazon регулярно надсилає листи на електронну пошту, що пропонують товари, схожі на ті, які вже були куплені або переглядалися.
- Веб-сервер може надсилати повідомлення адміністратору сайту, попереджаючи його про те, що на сервері закінчується пам'ять або про підозрілу активність користувача.

• Найпоширеніший вид повідомлень – це підтвердження реєстрації. Якщо створити новий обліковий запис на великому порталі використовуючи адресу електронної пошти, то на пошту надходить лист, який підтверджує факт реєстрації або містить інформацію про необхідність активувати обліковий запис.

Аналіз даних

Веб-додаток може збирати багато даних про своїх користувачів: що вони шукають, що купують, що рекомендують, як довго залишаються на кожній сторінці. Програмування серверної частини може бути використане, щоб вдосконалити відповіді, що базуються на аналізі цих даних.

Наприклад, Google рекламує товари на підставі попередніх пошуків. Facebook формує стрічку новин і рекламу, орієнтуючись на користувача чи його друзів. Rozetka буде пам’ятати список переглянутих, вподобаних і куплених товарів.

Сучасні засоби веб-технологій

Сучасні розробники зазвичай пишуть свій код, використовуючи бібліотеки та веб-фреймворки, які для розробки клієнтської і серверної частин є різними.

Бібліотеки — це набори попередньо написаних фрагментів коду, які можна легко інтегрувати до існуючого коду проекту. Таким чином, бібліотека є спеціалізованим інструментом для конкретних вузьких потреб, а не універсальною машиною для підготовки всього проекту.

Бібліотека, зазвичай, скорочує час розробки приблизно на 20%, дозволяючи не турбуватися про дрібниці, втім, слід врахувати певні особливості:

•Баг в реалізації бібліотеки може викликати складності в його знаходженні та способи усунення.

•Немає гарантії, що розробники бібліотеки оперативно випустять оновлення або виправлять недоліки.

•Оновлення може змінити доступ до API бібліотеки, що може спричинити значні зміни у коді.

Більшість популярних бібліотек поширюються як Open Source. Популярних комерційних бібліотек досить мало, крім складних вузькоспеціалізованих напрямів.

Фреймворки є шаблонами для створення веб-сайту або веб-додатку. Вони забезпечують структуру, де можна розмістити весь проект. Шаблони фреймворку створюють структуру з певними

виділеними областями для вбудовування коду. Отже, фреймворки JavaScript - це повні набори інструментів для формування та налаштування веб-сайту або додатків.

Фреймворк знаходиться на більш високому рівні абстракції у порівнянні з бібліотекою і дозволить без зайвих зусиль розробляти близько 80% застосування. Але слід врахувати:

•Останні 20% можуть викликати чималі труднощі через обмеження, що накладаються фреймворком.

•Оновлення фреймворку може не мати підтримки попередніх версій.

•Основний код і концепції рідко задовольняють розробників в своєму первісному вигляді. Вони завжди знайдуть «кращий» спосіб зробити що-небудь.

Інструменти

Інструмент - це допоміжний засіб розробки, але він не є невід'ємною частиною проекту. Інструменти містять системи збирання, компілятори, транслятори, механізми розгортання, тестування та інше.

Інструменти спрощують процес розробки. Наприклад, використання препроцесору Sass замість чистого CSS, оскільки він надає можливість використовувати цикли, функції, локальні змінні і багато іншого. Браузери не розуміють Sass/SCSS синтаксис, тому код переводиться в CSS.

Open Source

Open Source – це програмне забезпечення, яке постачається для кінцевого користувача з відкритим вихідним кодом. Тобто програму можна доопрацювати під власні завдання без порушення авторських прав розробників вихідного ПЗ. Рішення розповсюджується під ліцензіями GNU/Linux, MIT та інші.

Саме поняття виникло як альтернатива пропрієтарного ПЗ, коли комерційні компанії закривали доступ до вихідного коду.

Програмне забезпечення, яке розповсюджується як Open Source продукт, має ряд особливостей:

• Програмні продукти постачаються абсолютно безкоштовно до кінцевого споживача.

- Розробляються різними фахівцями, які уважно відстежують код програми, оперативно усуваючи помилки чи вразливості, що можуть спричинити критичні наслідки.
- Більшість продуктів Open Source сумісні з різними сімействами операційних систем, тобто вони кросплатформні.
- Спільнота розробників вільного програмного забезпечення відкрита для зворотного зв'язку від користувачів. Кожен з них може зробити пропозицію щодо покращення або додавання нової функції.
- Оновлення Open Source рішень відбувається значно частіше, ніж комерційні продукти. Кінцевий користувач отримує актуальні виправлення миттєво, як тільки вони фіксуються.
- Активний розвиток вільного ПЗ пожвавлює конкуренцію серед комерційних організацій, що сприяє підвищенню якості програм.
- Як комерційну основу розробники використовують у своїх рішеннях опцію добровільних пожертв. Користувачі можуть за власним бажанням підтримати програмістів фінансовими коштами шляхом переведення на електронний гаманець.
- Більшість Open Source рішень створюються під операційні системи сімейства Unix/Linux, що зменшує ризик зараження комп'ютера чи сервера вірусом. Шкідливе ПЗ переважно пишеться під Windows.

Для розміщення відкритих проектів з контролем версій популярною є онлайн платформа Github з використанням Git. Git — розподілена система контроля версій, яка надає можливість розробникам відслідковувати зміни в файлах и колективно працювати над одним проектом. Хоча Github більше відомий як платформа для розробки open source проектів, ресурс також надає можливість використання приватних репозиторіїв.

CSS-бібліотека Bootstrap

Bootstrap є найпопулярнішою бібліотекою, що вільно поширюється через офіційний сайт getbootstrap.com. Він містить готові стилі і скрипти, для застосування яких достатньо прописати необхідні класи і атрибути html-елементів. За допомогою Bootstrap-сітки легко адаптувати будь-який сайт і добре відображати його на будь-яких пристроях.

- Файли Bootstrap з готовим написаним кодом HTML I CSS під’єднуються до сайту в елементі head, після чого стають доступними їх можливості.

• Bootstrap ідеально підходить при роботі в команді. Верстка на Bootstrap при належному розумінні відбувається в 3-5 разів швидше, а однаковість коду дозволить іншому розробнику вносити правки.

• В Bootstrap закладено багато компонентів, все, що може знадобитися при розробці типових сайтів, наприклад, випадне меню, кнопки, таби, індикатори стану, хлібні крихти, списки, заголовки тощо. Присутній іконковий шрифт, вставляти іконки на веб-сторінки просто, опис є в офіційній документації.

• Bootstrap має велику спільноту прихильників і наявність хорошої документації. Завдяки такій поширеності Bootstrap з'явилося багато шаблонів, де вже перероблено дизайн всіх основних елементів. На основі таких шаблонів можна робити сайти, лише незначно щось змінюючи.

JS-фреймворки

JS-фреймворки - це інструменти для побудови динамічних додатків на Javascript. Розробники використовують JS-фреймворки там, де неможливо/складно/довго виконувати завдання звичайними засобами. У переважній більшості випадків, фреймворки використовуються для написання односторінкових додатків.

Односторінковий додаток (Single Page Application, SPA) - це веб-додаток чи веб-сайт, в якому необхідний код - HTML, JavaScript, та CSS - завантажується разом із сторінкою або динамічно довантажується за потребою, зазвичай, у відповідь на дії користувача. Сторінка не оновлюється і не перескеровує користувача на іншу сторінку в процесі роботи з нею. Взаємодія з односторінковим додатком часто здійснюється через динамічний зв'язок з веб-сервером.

Переваги побудови програми на JS-фреймворку:

•Ефективність. Проекти, які раніше розроблялися місяцями і мали сотні рядків коду, зараз можуть бути реалізовані значно швидше з добре структурованими готовими шаблонами і функціями. Коду стає помітно менше і він чистіше, що позитивно відбивається на швидкості розробки, а також підтримки та усунення помилок в коді програми.

•Безпека. Кращі JavaScript фреймворки мають фірмову систему безпеки і підтримуються великою спільнотою.

•Витрати. Більшість фреймворків є з відкритим кодом і безкоштовними. Оскільки вони допомагають програмістам швидше розробляти власні рішення, підсумкова ціна веб додатку буде нижчою. Наявність структури передбачає модульність додатку, а це надає

можливість простіше працювати над додатком кільком розробникам одночасно.

•Можливість швидко створити мобільний або настільний кросплатформний додаток з веб-версії.

Додатків на js-фреймворках створено багато і цей сегмент швидко розвивається.

Популярні JavaScript фреймворки та бібліотеки

• **Angular** – це JavaScript-фреймворк від Google, сумісний з більшістю поширених редакторів коду. Angular призначений для створення динамічних односторінкових веб-додатків (SPA) та прогресивних веб-додатків (PWA). Angular є одним із найпопулярніших фронтенд-фреймворків.

• **Vue.js** —фреймворк із відкритим вихідним кодом для односторінкових додатків. Він використовує модель розробки на основі компонентів та дозволяє приєднувати компоненти до проекту. Vue.js — приклад бібліотеки, більше схожої на фреймворк. Він пропонує багато шаблонів та патернів, які застосовуються при розробці.

• **Ember.js** – це фреймворк для розробки односторінкових, мобільних та десктопних додатків. Інструменти Ember дозволяють проектувати середовище розробки, а його командний рядок надає інструменти для автоматизації сценаріїв.

• **React** — бібліотека з відкритим вихідним кодом для створення динамічних інтерфейсів, що розроблена Facebook. Застосовується для створення веб-додатків з множинними динамічними компонентами. React заснований на JavaScript та JSX і дозволяє створювати HTML-елементи для багаторазового використання. React також включає React Native, спеціальне кросплатформве середовище для розробки мобільних додатків.

• **Node.js** – це серверна платформа з відкритим вихідним кодом, що створена на основі Google Chrome JavaScript Engine. Це одне з найбільш завантажених кросплатформових середовищ для виконання коду JavaScript. Node.js - це асинхронна, однопоточна, неблокуюча модель введення/виводу, яка робить її легкою та ефективною.

• **jQuery** призначений для управління HTML-документами. Бібліотека має простий API для управління подіями та розробки анімації у браузерах. jQuery застосовується для управління об'єктною моделлю документа (DOM), а також є інструментом розробки плагінів.

Вона також поставляється з легшою крос-браузерною бібліотекою jQuery UI для побудови графічного інтерфейсу.

Контрольні питання

1. Для чого призначена мова розмітки HTML?
2. Перелічити переваги, що надає використання стилів CSS
3. Які функції покладено на мову клієнтського програмування JavaScript?
4. Для чого використовують мови серверного програмування?
5. Перелічити переваги від використання мови HTML5.
6. Які функції покладено на різні рівні специфікацій CSS?
7. Який набір функціоналу може містити бібліотека?
8. Що собою представляє JS-фреймворк?
9. Перелічити популярні JS-фреймворки.
10. Назвати переваги від використання бібліотеки jQuery?

Використані джерела

1. Огляд мови розмітки HTML для початківців. URL: https://timeweb.com/ru/community/articles/chto-takoe-html
2. HTML-довідник. URL: https://schoolsw3.com/html/
3. CSS-довідник. URL: https://schoolsw3.com/css/
4. JS-довідник. URL: https://schoolsw3.com/js/
5. HTML Book. URL: https://html5book.ru/html-html5/
6. Керівництво по оформленню коду HTML/CSS. URL: https://proglib.io/p/obuchenie-veb-razrabotke-rukovodstvo-po-oformleniyu-koda-html-css-2020-12-06
7. Мови програмування для серверної веб-розробки. URL: https://andreyex.ru/programmirovanie/7-luchshih-yazykov-programmirovaniya-dlya-servernoj-veb-razrabotki/
8. Селектори CSS. Види, групування та специфічність. URL: https://itchief.ru/html-and-css/selectors
9. Інструменти та веб-ресурси для веб-розробки. URL: https://techblog.sdstudio.top/blog/instrumenty-i-veb-resursy-dlia-veb-razrabotki-100-plus
10. Кращі фреймворки CSS. URL: https://proglib.io/p/9-luchshih-freymvorkov-css-aktualnyh-v-2021-godu-2021-03-10
11. Кращі JS-фреймворки та тенденції веб-розробки. URL: https://medium.com/nuances-of-programming/лучшие-javascript-фреймворки-и-тенденции-веб-разработки-в-2021-году-2a35e348a12

ОСНОВИ HTML

HTML - не є мовою програмування, це мова розмітки, яка використовується для розміщення елементів (тексту, картинок, кнопок, таблиць тощо) на сторінці, тобто, для складання структури. Сторінка може мати складну або просту структуру, все залежить від досвіду веб-розробника.

HTML складається з ряду елементів, які використовують для того, щоб різні частини сторінки мали певний вид або спрацьовували певним способом. За призначення, вигляд та розташування елементу відповідають спеціальні оператори (теги), які представляють собою команду, що поміщена в кутові дужки. Теги, зазвичай, є скороченою назвою або абревіатурою того завдання, що має виконати браузер наприклад:

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

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

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

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

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

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

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

Одинарні теги HTML

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

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

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

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

Парні теги HTML

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

Наприклад, для того, щоб певний текст відобразився в браузері як рядок, слід застосувати теги, що позначають елемент-параграф (рис. 1):

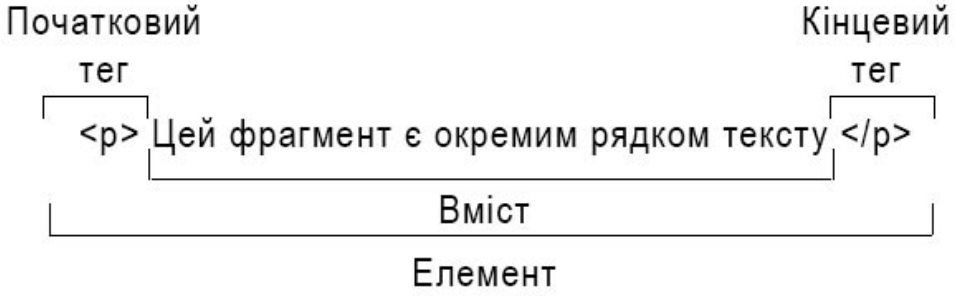

Рис. 2.1. Склад HTML-елементу

Основними частинами елемента є:

- **Початковий тег:** Він складається з назви елемента ( ) і є ознакою початку елемента, з цього моменту тег починає впливати на той вміст, що слідує за ним.
- **Кінцевий тег:** виглядає як і початковий, але містить слеш перед назвою тега ( ). Він вказує на закінчення елементу. Закінчення в парних тегах є обов'язковими, інакше браузер може невірно відобразити вміст.
- **Вміст:** в цьому випадку вмістом є простий текст.
- **Елемент:** початковий тег + вміст + кінцевий тег = елемент.

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

Цей фрагмент є окремим рядком тексту

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

Цей фрагмент є окремим рядком тексту

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

Теги для коментарів

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

Тег для коментаря починається з кутової дужки, знак оклику, два дефіси, текст коментаря, два дефіси, закінчується кутовою дужкою.

<!-- текст коментаря -->

Коментарі в HTML не відображаються на сторінці в браузері користувачеві, їх можна побачити лише в коді веб-сторінки. Для зручності слід коментувати не лише початок певного блоку коду, а й його завершення, наприклад:

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

Структура HTML-документа

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

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

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

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

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

•**<body> елемент** містить весь контент, який бачать відвідувачі - текст, зображення, відео, анімація тощо.

Елементи, що знаходяться всередині елементу <html>, утворюють дерево документа, так звану об'єктну модель документа, **DOM (Document Object Model)**. Елемент <html> є кореневим елементом, body і head – структурними елементами.

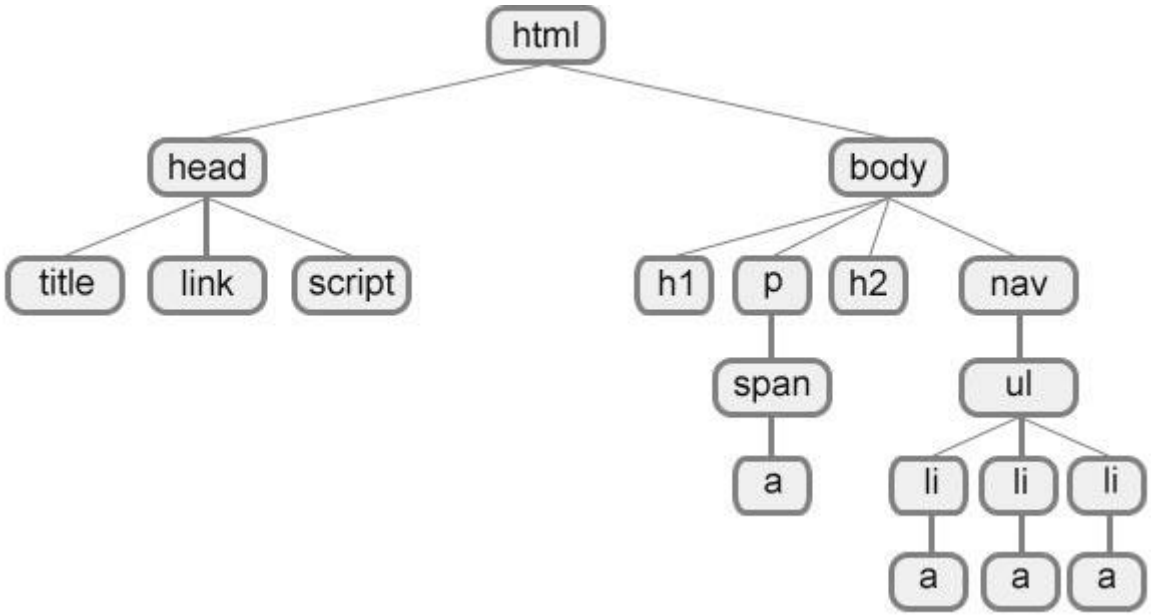

Рис. 3. Найпростіша структура веб-сторінки

У взаємодії елементів веб-сторінки задіяні "родинні стосунки" між елементами. Відносини між множинними вкладеними елементами підрозділяються на батьківські, дочірні та сестринські.

•**Батьківський елемент** - елемент, що пов'язаний з іншими елементами нижчого рівня, і що знаходиться на дереві вище за них. На рис.2 <html> є батьківським лише для <head> і <body>. Елемент <body> є батьківським для всіх елементів, що містяться в ньому: <h1>, <p>, , <nav> і т.д. Тег <p> є батьківським тільки для .

•**Дочірній елемент** - елемент, що розташований всередині іншого елементу. На рис.2 елементи <h1>, <h2>, <p> і <nav> є дочірніми по відношенню до <body>.

•**Сестринський елемент** - елемент, що має загальний батьківський елемент з даним, так звані елементи одного рівня. На рис.2 <head> і <body> - елементи одного рівня, так само як і елементи <h1>, <h2> і <p> є між собою сестринськими.

Елемент <head>

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

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

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

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

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

•**Елемент <link>.** Даний елемент визначає відношення між поточною сторінкою та іншими документами. Таких елементів на сторінці може бути кілька. Наприклад, вказати зв’язок сторінки з файлом зі стилями (окремий файл з розширенням .css), тоді посилання на файл зі стилями буде виглядати так <link href="style.css">. Обов’язковим є вказування атрибуту rel, де вказується як цей документ буде розпізнано браузером (<link rel="stylesheet" href=”style.css”> - документ є таблицею стилів, <link rel="icon" href="favicon.png"> - документ є фавіконкою). Додатково вказують тип документу атрибутом type (<link rel="stylesheet" type="text/css" href="style.css">, <link rel=" icon" type="image/x-icon" href="favicon.png">)

•**Елемент <script>.** Дозволяє приєднувати до документа різні сценарії. Закриваючий тег є обов'язковим, при цьому текст сценарію

може розташовуватися або всередині цього елемента, або в зовнішньому файлі. Якщо текст сценарію розташований в зовнішньому файлі, то він під'єднується за допомогою атрибутів елементу, наприклад: <script src=" scripts.js"></script>

Елемент <body>

Всередині елементу <body> міститься інформація, яка відображається в браузері. В таблиці https://html5book.ru/html-tags/ наведено повний перелік елементів, що підтримуються HTML5.

За призначенням теги можна умовно розподілити:

- **Контейнерні:** div, article, aside, header, footer, span, p, h1-h6, ul, ol
- **Функціональні:** a, img, table, form, input, audio, video, canvas
- **Фрейми:** iframe

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

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

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

Універсальними атрибутами є

- **title** Вказує додаткову текстову інформацію про елемент, що відображається у спливаючій підказці над елементом, наприклад: **<div title=”Червоний блок”>**.
- **style** Вказує код CSS, що застосовується для оформлення даного елементу, наприклад: **<div style=”width: 100%; background:red; color:#ffffff;”>**.
- **class** Вказує назву класу для елемента, сам клас описано в таблиці стилів, наприклад: ** **. Атрибут *class* застосовний до всіх HTML-елементів. Кожному конкретному елементу можна привласнити **кілька значень class**. Множинні значення *class* записуються через пробіл, наприклад, ** **. Назви класу повинні складатися з букв, цифр, дефісів і нижніх підкреслень і починатися тільки з літери.
- **id** Вказує унікальний ідентифікатор елемента, наприклад: ** **. Атрибут *id* застосовний до всіх HTML-елементів. Кожному конкретному елементу можна привласнити **лише одне**

значення id. Назви id повинні складатися тільки з букв, цифр, дефісів і нижніх підкреслень і повинні починатися тільки з літер.

Умовно елементи можна поділити на групи

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

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

Блокові елементи

Загальні характеристики блокових елементів

•На початку і в кінці блокових елементів, автоматично ставиться перенесення рядка.

•За замовченням блокові елементи займають всю ширину батьківського елементу.

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

•В блокових елементах можна керувати зовнішніми і внутрішніми відступами.

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

•Всередині блокових елементів можна розміщувати інші блокові елементи, а також рядкові елементи.

Типові блокові елементи

- ** ** Універсальний блоковий елемент.
- **<article>, <aside>, <header>, <footer>** Семантичні блоки для позначення структурних ділянок сторінки.
- **<p>, <h1>-<h6>, <blockquote>** Елементи для HTML тексту.
- **<hr>** Горизонтальна лінія.
- **<ol>, ** Встановлює нумерований (маркірований) список.
- **<table>** Створює таблицю.
- **<form>** Встановлює форму на веб-сторінці.

Рядкові елементи

Загальні характеристики рядкових елементів

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

•В рядкових елементах не можна керувати шириною і висотою: width, height.

•В рядкових елементах можна встановлювати лише внутрішні відступи (padding) та зовнішні бічні відступи (margin).

•Рядкові елементи НЕ займають всю ширину батьківського елементу.

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

•Всередині рядкових елементів можна розміщувати інші рядкові елементи, якщо розмістити там блоковий елемент, то він розірве логічний потік і розміститься з нового рядка.

Типові рядкові елементи

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

**** Універсальний рядковий елемент.

**
** Розірвання рядка.

<img> Уставляння зображення.

HTML текст

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

Грамотно відформатований текст надає для пошукових систем відомості, які фрагменти тексту мають важливий зміст, за якими з них слід ранжирувати веб-сторінку в пошуковій видачі.

Елемент абзацу <p>

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

Елементи заголовків <h1>-<h6>

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

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

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

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

Елементи для форматування тексту

- **** задає грубіше накреслення шрифту, вказуючи браузеру на важливість тексту. Є сучасною альтернативою для попереднього елементу <b>.
- **** відображає текст нахиленим шрифтом (курсивом). Є сучасною альтернативою для елементу <i>, який на сьогоднішній день охоче застосовують для розміщення іконок. Для нахилу тексту можна використати елемент <cite>.
- **<q>** додавання зовнішніх лапок до фрагменту.
- **<code>** зміна шрифту на більш технічний шрифт.
- **<del>** перекреслений текст.
- **<ins>** підкреслений текст.
- **<sub>** використовується для створення нижніх індексів. Зсуває текст нижче рівня рядка, зменшуючи його розмір.
- **<sup>** використовується для створення ступенів. Зсуває текст вище рівня рядка, зменшуючи його розмір.
- **<pre>** дозволяє вивести текст на екран, зберігши початкове форматування. Пробіли і переноси рядків при цьому не видаляються.
- **<blockquote>** Використовується для оформлення фрагменту особливим блоком, наприклад, цитати, виділяючи його відступами і переносами рядків.
- **
** Переносить текст на наступний рядок, створюючи розрив рядка.
- **<hr>** Використовується для поділу контенту на веб-сторінці. Відображається у вигляді горизонтальної лінії.

HTML посилання

HTML посилання складають основу Веб, вони пов'язують між собою різні веб-сторінки.

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

Замовити книжки у магазині

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

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

Уставляння гіпертекстових посилань в HTML-документ

Uniform Resource Locator, URL

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

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

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

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

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

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

Найбільш поширені методи доступу: http (https), mailto, tel, viber.

- **Метод http (https)** надає доступ до веб-сторінки за протоколом HTTP, наприклад:
НУ "Львівська політехніка"
- **Метод mailto** запускає сеанс поштового зв'язку із зазначеним адресатом і хостом, наприклад:
Iryna.Y.Yurchak@lpnu.ua
- **Метод tel** забезпечує з’єднання з програмою для здійснення телефонних дзвінків. Такий метод спрацьовує в мобільних пристроях і користувач може відразу позвонити за вказаним номером, наприклад:
(067)947-61-21
- **Метод viber** здійснює з’єднання з месенжером Viber, наприклад:
<a href="viber://chat?number=+380679476121">

Абсолютний URL

Абсолютний шлях містить всю інформацію, що необхідна браузеру для знаходження файлу: назву протоколу, доменну або IP-

адресу комп'ютера (хосту), папку (шлях до файлу) та ім'я файлу. Наприклад,

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

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

https://www.site.ru/file.html

У разі відсутності назви файлу буде завантажуватися веб-сторінка, яка задана за замовченням у налаштуваннях веб-сервера (зазвичай, це index.*):

http://www.site.ua/

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

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

Відносний URL

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

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

Текст посилання

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

Текст посилання

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

Текст посилання

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

Текст посилання

Зовнішні посилання

Посилання на веб-сторінки інших сайтів є зовнішніми, для них завжди вказують Абсолютний URL, наприклад:

Фільми англійською мовою

Посилання на розділи поточної сторінки

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

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

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

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

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

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

Або

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

Посилання на сторінки одного сайту

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

Коли браузер не знаходить у посиланні протокол https://, він виконує пошук зазначеного документа на тому ж сервері. Відносні

посилання описують шлях до потрібного файлу відносно розташування поточного документа.

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

Оформлення гіпертекстових посилань

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

Оформлення гіпертекстових посилань.

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

Урок 1

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

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

Урок 1

Елементи, до яких застосовують посилання

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

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

Атрибути посилань

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

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

•**nofollow** - забороняє роботам пошукової системи переходити за посиланнями на даній сторінці або за конкретними посиланнями.

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

_self - сторінка завантажується до поточного вікна (вкладки). Значення за замовченням;

_blank - сторінка відкривається в новому вікні (вкладці) браузера;

_parent - сторінка завантажується у фрейм-батько;

_top - сторінка завантажується в повне вікно браузера.

HTML зображення

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

Уставляння зображень в HTML-документ

Елемент <img>

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

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

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

Для елементу <img> доступні наступні атрибути:

•**src** - вказує URL-адресу зображення (де знаходиться зображення).

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

•**width** - задає ширину зображення. Прийняті значення: пікселі або відсотки.

•**height** - задає висоту зображення. Прийняті значення: пікселі або відсотки.

Адреса зображення

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

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

Або через відносний шлях від документа або кореневого каталогу сайту:

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

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

Розміри зображення

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

Формати графічних файлів

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

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

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

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

•Растровий формат **WebP**. Веб-формат від Google, що створено в 2010 році. Використовує потужні способи оптимізації зображень для зменшення ваги файлу, підтримує прозорість і анімацію. На сьогоднішній день формат підтримують популярні браузери (Chrome,

Firefox, Edge), у застарілих браузерів зображення в цьому форматі не відобразиться.

•Растровий формат **APNG.** Формат зображення, що засновано на форматі PNG від Mozilla в 2008році. Дозволяє зберігати анімацію, а також підтримує прозорість. Дозволу з боку Portable Network Graphics формат APNG не отримав, тому його неможна віднести ані до веб-стандарт, ані до стандарту PNG та libpng. Тому, робота з даним форматом може бути зі збоями.

•Растровий формат **ICO.** Формат зберігання фавіконок, що відображаються на закладці в браузері або поруч з адресою сайту у сторінці видачі результатів пошукової системи. На сьогоднішній день формат вважається застарілим, а фавіконки застосовують у форматах png та svg. Найбільш вживані розміри фавіконок: 16, 32, 48, 57, 72, 114, 256 пікселів.

HTML списки

HTML списки представляють набір згрупованих абзаців тексту, помічених значками (маркірований список) або цифрами (нумерований список). Елементи списку додаються за допомогою елементу <li> (List Item - елемент списку). Для елементу <li> доступні різні атрибути, що дозволяють змінити нумерацію або маркірування обраного елемента списку.

Маркірований список

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

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

- Microsoft
- Google
- Apple
- IBM

Нумерований список

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

Для тегу <ol> доступні наступні атрибути:

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

•**type** - задає тип нумерації для використання в списку (у вигляді букв або цифр). Прийняті значення:

1 - значення за замовчуванням, десяткова нумерація.

A - нумерація списку в алфавітному порядку, заголовні букви (A, B, C, D).

a - нумерація списку в алфавітному порядку, малі літери (a, b, c, d).

I - нумерація римськими великими цифрами (I, II, III, IV).

i - нумерація римськими малими цифрами (i, ii, iii, iv).

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

5. Microsoft
6. Google
7. Apple
8. IBM

Список визначень

Списки визначень описуються за допомогою елементу <dl>, який містить пари термін-визначення. Для додавання терміну застосовується блоковий елемент <dt>, а для уставляння визначення - <dd>.

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

```
<dl>
    <dt>Національний університет «Львівська політехніка»:</dt>
        <dd>ІКНІ</dd>
    <dt>Кафедри</dt>
        <dd>САПР</dd>
        <dd>ІСМ</dd>
        <dd>АСУ</dd>
</dl>
```

Національний університет «Львівська політехніка»:
ІКНІ
Кафедри
САПР
ІСМ
АСУ

Вкладений список

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

```
<ul>
```

•Пункт 1.

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

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

Більш докладну інформацію про HTML елементи можна дізнатися у специфікації мови та довідниках для веб-розробників.

Контрольні питання

1. Що собою представляють теги HTML?
2. Навести відмінності між одинарними та парними тегами.
3. Навести найпростішу структуру веб-сторінки.
4. Перелічити універсальні атрибути для тегів HTML.
5. Пояснити різницю між блоковими та рядковими елементами HTML.
6. Навести приклади посилання з абсолютним та відносним шляхом.
7. Які атрибути застосовують до посилань в HTML-документі?
8. Які атрибути застосовують до зображень в HTML-документі?
9. Які графічні формати підтримує HTML?
10. Перелічити відомі HTML-списки.

Використані джерела

1. Вступ у HTML. URL: https://developer.mozilla.org/ru/docs/Learn/HTML/Введение_в_HTML
2. HTML, HTML5. URL: http://html5book.ru/html-html5/

3. HTML5 Семантичні елементи. URL: https://html5css.ru/html/html5_semantic_elements.php
4. Блокові і рядкові елементи. URL: https://html5book.ru/block-inline-elements/
5. Абсолютні і відносні посилання. URL: https://htmlacademy.ru/blog/boost/frontend/links
6. Зображення в HTML. URL: https://developer.mozilla.org/ru/docs/Learn/HTML/Multimedia_and_embedding/Images_in_HTML
7. Все про HTML списки. URL: https://guruweba.com/html/sozdanie-spiskov-vse-o-html-spiskakh/

ОСНОВИ CSS

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

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

Рис. 5. Структура оголошення стилю елемента в CSS

Додавання CSS до веб-сторінки

Вбудовані таблиці стилів

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

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

Внутрішні стилі елементів

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

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

Зовнішні таблиці стилів

Зовнішня таблиця стилів представляє текстовий файл з розширенням .css, в якому знаходиться весь набір стилів CSS. Задані в

файлі стилі будуть працювати для всіх сторінок веб-сайту. Під'єднати зовнішній файл зі стилями можна в два способи:

Прикріплення до веб-сторінки за допомогою тега <link>, вкладеного в тег <head>:

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

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

Прикріплення до веб-сторінки за допомогою правила @import

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

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

Селектори

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

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

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

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

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

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

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

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

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

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

ul li {text-transform: uppercase;}

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

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

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

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

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

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

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

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

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

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

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

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

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

Псевдокласи

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

a: link – описує стиль невідвідуваного посилання.

a: visited - описує стиль вже відвіданого посилання.

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

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

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

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

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

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

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

Структурні псевдокласи типу. Вказують на конкретний тип дочірнього тегу:

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

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

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

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

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

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

:before – додає певний вміст перед елементом.

:after - додає певний вміст після елемента.

Комбінації селекторів

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

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

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

Угруповання селекторів

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

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

Принцип каскадування і специфічність правила

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

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

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

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

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

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

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

Більше інформації по правилах CSS можна дізнатися з специфікації мови та довідників.

Контрольні питання

1. Для чого застосовують каскадні таблиці стилів?
2. З чого складається правило створення стилю
3. Яким чином можна додати стилі до html-сторінки?
4. Перелічити різні типи селекторів.
5. Які комбінації селекторів можна застосувати?
6. Яким чином визначають пріоритет стилю?

Використані джерела

1. Сучасний підручник CSS. URL: https://idg.net.ua/blog/uchebnik-css
2. Селектори CSS. Види, групування і специфічність. URL: https://itchief.ru/html-and-css/selectors
3. Псевдо-класи і псевдо-елементи в CSS. URL: https://coderoad.ru/8069973/В-чем-разница-между-псевдоклассом-и-псевдо-элементом-в-CSS
4. CSS по методології БЕМ. URL: https://ru.bem.info/methodology/css/
5. Специфічність CSS. URL: https://developer.mozilla.org/ru/docs/Web/CSS/Specificity
6. Каскадність CSS. Пріоритети стилів. URL: https://idg.net.ua/blog/uchebnik-css/azy-css/kaskadnost

Навчальне видання

ВЕБТЕХНОЛОГІЇ ТА ВЕБДИЗАЙН

Конспект лекцій

Укладачі:

Пархоменко Олександр Юрійович
Тищенко Світлана Іванівна

Формат 60х84 1/16. Ум. друк. арк. 6,25.
Наклад 50 прим. Зам. № _____

Надруковано у видавничому відділі
Миколаївського національного аграрного університету
54020, м. Миколаїв, вул. Георгія Гонгадзе, 9

Свідоцтво суб'єкта видавничої справи ДК № 4490 від 20.02.2013 р.