# 3.4. Верстання сторінки (кодинг)

По закінченні роботи зі створення графічного макету дизайну і схвалення його іншими учасниками проекту чи замовником, приступають до створення HTML-шаблону сторінки.

Верстка - це процес перетворення дизайну сайту (картинки) у веб-сторінку. Макет дизайну це макети у середовищі Figma або зображення в форматі PSD (Photoshop), а шаблон веб-сторінки - це інтерактивне відображення сторінки, де можна виділити текст, перейти за посиланням, заповнити форму, додати товар в корзину і так далі.

Верстання сторінок – це процес написання HTML коду сторінки, за яким сторінка набуває вигляд, подібний до дизайну макету. Безумовно, існують різні технології розмітки тексту, які підтримуються браузерами, проте, найпоширенішим варіантом є верстання сторінок за допомогою мови HTML та стилів CSS.

Верстка сторінки робиться поетапно: спочатку створюється HTML-структура (HTML-код), додаються стилі (CSS-код), а далі за потребою пишуться скрипти (JS), додаються необхідні плагіни і бібліотеки.

Для того, щоб навчитися верстанню, слід спочатку зрозуміти значення операторів HTML та CSS і приступати до вивчення верстки. Для початківців починати слід з верстання простих структур, але з часом ускладнювати завдання.

Наприклад, почати з верхньої частини сторінки: створити контейнер, додати до нього фон, встановити логотип картинкою або текстом. Далі - створення меню по частинах: вивести пункти, вирівняти їх, поставили поля і відступи, зробити hover-ефект. І так поступово, верстається вся сторінку.

Поради для початківців:

1. Вивчати HTML та CSS по свіжих матеріалах від фахівців, що мають авторитет серед спільноти розробників.
2. Намагатися верстати, відтворювати сторінки відеоуроків, вивчати код готових шаблонів, яких є багато у відкритому доступі.
3. Накопичений досвід сприятиме доведення навичок верстальника до автоматизму, і в подальшому верстання сторінок буде відбуватися і швидше і якісніше.

## Редактори коду

На сьогоднішній день найпопулярнішими редакторами коду та середовищами для веб-розробки є:

* Visual Studio Code (https://code.visualstudio.com/) - це безкоштовний і відкритий вихідний код редактора коду, розроблений Microsoft. Він підтримує широкий спектр мов програмування, включаючи HTML, CSS, JavaScript, PHP, Python та Java. Visual Studio Code. Містять багато функцій, які роблять його корисним для веб-розробки: підсвічування синтаксису, автозаповнення, налагодження та інтеграція з Git.
* Sublime Text (https://www.sublimetext.com/) – це платний редактор коду з відкритим вихідним кодом. Він відомий своєю швидкістю та продуктивністю, а також своїм широким набором функцій, включаючи підсвічування синтаксису, автозаповнення, налагодження та інтеграцію з Git.
* Atom (https://atom.io/) – це безкоштовний та відкритий вихідний код редактор коду, розроблений GitHub. Він містить багато функцій, які роблять його корисним для веб-розробки, таких як підсвічування синтаксису, автозаповнення, налагодження та інтеграція з Git.
* WebStorm (https://www.jetbrains.com/webstorm/) – це платна IDE від JetBrains, спеціально створена для веб-розробки. Містить багато функцій, які спрощують і прискорюють процес розробки веб-сайтів і додатків, таких як підсвічування синтаксису, автозаповнення, налагодження, інтеграція з Git і інструменти для тестування.

Інші популярні редактори коду:

* Brackets (https://brackets.io/)
* PhpStorm (https://www.jetbrains.com/phpstorm/)
* IntelliJ IDEA (https://www.jetbrains.com/idea/)
* Komodo IDE (https://www.activestate.com/products/komodo-ide/)
* Notepad++ (https://notepad-plus-plus.org/downloads/)

При виборі редактора коду слід враховувати такі фактори:

* Функціональність. Який набір функцій потрібний? Потрібний простий редактор коду з базовими функціями чи потрібна потужна IDE з багатьма функціями?
* Ціна. Деякі редактори коду безкоштовні, інші платні.
* Операційна система. Редактор коду, який працює на Windows, MacOS або Linux?
* Особисті переваги. Який редактор коду більше подобається? Деякі редактори коду мають простіший інтерфейс, інші складніші.

Для спрощення верстання можна скористатися генераторами структури сторінки, які створюють код загальної структури сторінки з CSS-файлом, в якому прописані загальні елементи та стилі. Як приклад таких генераторів є шаблонізатор Dreamweaver.

## Блокова верстка

Сучасний підхід до HTML верстки передбачає розподілення блоків елементів у різних місцях, і це називається блоковою версткою.

В HTML4 як базовий елемент розмітки використовувався тег **div**, що не має певного семантичного значення та особливих вимог на його вміст. Тег **<div></div>** є контейнером для тексту, зображень та інших елементів сторінки і вони позиціонуються в різних місцях. Теги **div** можна помістити в будь-яке місце веб-сторінки, позиціонувати абсолютно (вказати координати x і y) або відносно (вказати відстань до інших елементів сторінки). На рис.1 показано просту структуру сторінки в блоковій верстці з використанням тегу **div**.



Рис.1. Типова структура сторінки в блоковій верстці

Призначення та поведінку блоку **div** потрібно вказувати у властивостях класу чи ідентифікатору. Це створює зайвий код і не завжди є зрозумілим для пошукових систем. Пошукові системи під час індексації веб-сторінок прагнуть розібратися у призначенні блоку, щоб правильно розподілити інформацію у своїх базах.

Наприклад, якщо назву класу вказано .header, то роботи пошукових систем визначать, що інформація в блоці відноситься до верхнього колонтитула. Але розробник може назвати клас за власним уподобанням, припустимо .heder або .shapka. Тоді, для пошукової системи ця назва буде позначати лише один з блоків. Ця проблема була вирішена в подальшому вдосконаленні мови HTML5.

## Семантична верстка

В HTML5 з'явилися нові елементи, які служать заміною для звичайних блочних елементів. На тепер можна використати семантично змістовні блочні елементи, що за призначенням аналогічні до елементу div.

Семантична верстка - підхід до розмітки, який спирається не на зміст сайту, а на значення кожного блоку і логічну структуру документа.



Рис.2. Типова структура сторінки з семантичною версткою

### Основні семантичні елементи HTML

* <article> Незалежна, змістовна одиниця, що відокремлюється, наприклад, стаття, пост.
* <section> Змістовний розділ сторінки, може міститися в <article>.
* <aside> Бічна колонка, що містить додатковий контент для сторінки.
* <nav> Навігаційний розділ із посиланнями на інші сторінки.
* <header> Верхній колонтитул сторінки. Може містити логотип, навігацію, назву чи слоган сайту.
* <main> Основний вміст сторінки, що знаходится між верхнім та нижнім колонтитулами
* <footer> Нижній колонтитул сторінки, що зазвичай містить коротку інформацію про сайт, контактну інформацію, додаткове меню, копірайт.

### Розмітка сторінки з точки зору семантики

Процес розмітки можна розділити на кілька кроків із різним ступенем деталізації.

* Великі змістовні блоки на кожній сторінці сайту. Елементи: <header>, <main>, <footer>.
* Великі змістовні розділи у блоках. Елементи: <nav>, <section>, <article>, <aside>.
* Заголовок всього документа та заголовки змістовних розділів. Елементи: <h1>-<h6>.
* Блокові елементи у змістовних розділах. Списки <ul> <ol>, таблиці <table>, демо-матеріали <iframe>, параграфи <p> та переноси <br>, форми <form>, цитати <cite>, прогрес <progress>.
* Рядкові елементи. Зображення <img>, посилання <a>, кнопки <button>, відео <video>, час <data> та дрібні текстові елементи <strong> <em> <span>.

Існують прості правила для вибору потрібних елементів.

* Вийшло знайти найкращий змістовний елемент — використовувати його.
* Для потокових контейнерів - <div>.
* Для дрібних рядкових елементів (слово чи фраза) – <span>.

#### Особливості сторінки з семантичною версткою:

* Легкий і зрозумілий код сторінки. Код сторінки компактній і не захаращений зайвими класами.
* За рахунок зменшення та структуризації коду полегшується завантаження сторінки.
* Код сторінки є більш зрозумілим і зручним як для людей так і для пошукових роботів.

## Перевірка HTML і CSS коду

Перевіряти код сторінки сайту необхідно постійно, особливо коли здійснюються зміни в структурі сайту (створення, видалення, перенесення блоків). Акуратний верстальник прагне це робити якнайчастіше. Перевірку HTML і CSS коду краще робити з використанням програм-валідаторів чи Інтернет-сервісів, які відображають знайдені помилки відповідно до різних стандартів. Кращим валідатором за допомогою якого можна перевірити любу сторінку, що розташована в Інтернеті або на локальному комп'ютері є валідатор міжнародного Консорціуму W3C (WorldWideWebConsortium) <http://validator.w3.org/>

Сучасні браузери підтримують стандарти W3C значно краще, ніж їх попередні версії. Якщо правильний код відповідає певним формальним правилам, його легше інтерпретувати й обробляти, він швидше аналізується і відображається в браузері, з ним легше працювати пошуковим системам.

Відповідність коду сторінки до стандартів W3C не є гарантією якості та ефективності. Грамотний код - це важливий, але далеко не єдиний показник для оцінювання сторінки.

### Поширені вимоги до HTML та CSS-коду

#### 1. Кросбраузерність

Сайт повинен нормально працювати як в останніх так і старших версіях популярних браузерів, зокрема Mozilla FireFox, Opera, Chrome, Edge, Safari.

#### 2. Застосування коментарів

Великі HTML блоки слід коментувати:

<!-- BEGIN FOOTER -->

<!-- END FOOTER -->

Або

<!--ABOUT-->

<!--./ABOUT-->

#### 3. Впорядкування в таблиці стілів

CSS файл повинен бути розділений за допомогою коментарів на блоки за функціональним або структурним призначенням, наприклад:

 /\* 1. Скидання CSS \*/

 /\* 2. Типові елементи \*/

 /\* 2.1. Заголовки \*/

 /\* 2.2. Посилання \*/

 /\* 2.3. Елементи форм \*/

 /\* 3. HEADER \*/

 /\* 4. FOOTER \*/

 /\* 5. SIDEBAR \*/

Якщо використовуються CSS префікси чи хакі, також потрібні коментарі, що це і для якого браузера. Коментарі допомагають орієнтуватися в коді не лише верстальнику, але і решта учасників проекту: програмісту, контент-менеджеру, оптимізатору.

#### 4. Змістовні назви

Назви класів і id повинні за змістом відповідати застосуванню, наприклад, *text-center, main-menu, gallery, news.* Такі назви будуть зрозумілими для розробника та інших фахівців, що будуть в подальшому працювати з кодом. Для пошукових систем – це також є додатковою інформацією.

#### 5. Вживання Javascript

* Все що можна зробити без скриптів, робити без них засобами CSS. Якщо Javascript коду багато - потрібно його виносити в окремий файл. Обробники подій теж краще відокремити і оголошувати в окремому файлі.
* Заздалегідь узгодити JavaScript-бібліотеки, наприклад, jQuery чи PrototypeJS.
* Якщо в макеті присутній JavaScript, що змінює DOM – потрібно прослідкувати його поведінку в різних браузерах.

#### 6. Ширина сторінки

Для еластичних макетів (на всю ширину вікна браузера) обов'язково повинна бути задана мінімальна і максимальна ширина. Для фіксованого дизайну ширина має адаптуватися під розмір екрану пристрою. В іншому випадку може з’явитися горизонтальний скролінг, що свідчить про низький професійний рівень розробника.

#### 7. Особливості відображення

Обов’язково вказувати колір для фону сторінки (body), навіть якщо він білий. Певна частка відвідувачів встановлює за замовченням фоновий колір браузера, відмінний від білого, що призводить до відображення сторінки не в тому вигляді як задумав дизайнер.

#### 8. Порядок в файлах

Файлова структура повинна бути стрункою і не містити файлів, які не використовуються в сайті (HTML-файлів, зображення, стилі, скрипти).

#### 9. Розташування футера у нижньому боці браузера

В макетах, де висота сторінки залежить від контенту (а таких, як правило, більшість), футер має триматися низу браузера за відсутності або малій кількості основного контенту.

#### 10. Кодування тексту

На сьогоднішній день для тексту стандартним є кодування UTF-8. Це кодування застосовують для HTML, CSS та JS коду.

#### 11. Використання нестандартних шрифтів

Якщо в макеті використовуються нестандартні шрифти, слід з’ясувати за якою технологією їх буде втілено: @ font-face, Google Font API чи інше.

#### 13. Звітність про виправлення

Якщо передача зверстаної сторінки до інших фахівців проводиться більш ніж одним етапом (наприклад, верстальник відправляє сторінки по одній, або якщо йому повертаються на доопрацювання вже зверстані сторінки), а система контролю версій для верстки не використовується, тоді виконавець повинен в обов'язковому порядку прикріпити файл з описом змін в макеті приблизно такого змісту:

* Додано нові картинки в папку img,
* Картинки btnHome.jpg і btnFeedback.jpg вже не потрібні, можна видаляти.
* Змінено html-код в секції файлу anketa.html.
* Додано в кінець файлу main.css нові стилі (починаючи з .form\_row і нижче).

## Контрольні питання

1. Що собою представляє блокова верстка?
2. Які особливості притаманні до семантичної верстки?
3. Перелічити поширені семантичні HTML елементи
4. Як можна здійснити розмітку сторінки з точки зору семантики
5. Які інструменти можна застосувати для перевірки валідності HTML і CSS коду?
6. Перелічити вимоги, що висуваються до правильного HTML і CSS коду

## Використані джерела

1. Блокова верстка, основи анатомії скелету сайтів - <https://www.internet-technologies.ru/articles/blochnaya-verstka-ili-osnovy-anatomii-skeleta-saytov.html>
2. Що таке семантична верстка - <https://htmlacademy.ru/blog/boost/frontend/semantics>
3. HTML5 Семантика - <https://schoolsw3.com/html/html5_semantic_elements.php>
4. Перевірка валідації коду: як знайти помилки в HTML і CSS - <https://pr-cy.ru/news/p/7637-proverka-validatsii-koda-kak-nayti-oshibki-v-html-i-css>
5. Вимоги до HTML та CSS-коду - <https://unibix.ru/prosto_o_slozhnom/dlya-veb-programmistov/tehnicheskie-trebovaniya-k-verstke-saytov-html--css>