# 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 зображення в форматі PSD (Photoshop) чи макети у середовищі Figma, 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. Питання як зверстати це, як зверстати то - відпадуть з накопиченням досвіду.

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

Поради для початківців:

1. Вивчати HTML та CSS по свіжих матеріалах і 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

З найбільш популярних редакторів коду на сьогодні можна виділити три, які схожі за принципом роботи, до них додатково можна під’єднати необхідні модулі та плагіни:

* [Sublime Text](https://www.youtube.com/watch?v=uZm-Z08jPtI) (http://www.sublimetext.com/
* [Atom](https://www.youtube.com/playlist?list=PLY4rE9dstrJzpFizUPSJkSZ9fgdyB4hRL) (https://atom.io/)
* [Brackets](https://www.youtube.com/watch?v=rvo3Mv1Z4qU) (https://brackets.io/)

Існують більш просунуті комплекси IDE (Integrated Development Environment - Інтегроване середовище розробки) такі, як

* Adobe DreaWeaver (https://www.adobe.com/ua/products/dreamweaver.html)
* Visual Studio Code (https://code.visualstudio.com/)
* Web Storm (https://www.jetbrains.com/ru-ru/webstorm/)
* PHP Storm (https://www.jetbrains.com/ru-ru/phpstorm/)

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, онлайн генератори CSStemplater.com і CSScreator.com.

## Блокова верстка

Сучасний підхід до HTML верстки передбачає розподілення елементів сегментів у різних місцях, і це називається блоковою версткою. Блокова верстка базується на принципах розташування і взаємодії за допомогою блокових елементів.

В HTML4 як базовий елемент розмітки використовувався тег div, що не має певного семантичного значення та особливих вимог на його вміст. Тег **<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.



Рис.1. Типова структура сторінки в блоковій верстці

## 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.



Рис.2. Типова структура сторінки з семантичною версткою

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

### Основні семантичні елементи HTML

#### <article>

* Значення: незалежна, змістовна одиниця, що відокремлюється, наприклад, стаття, пост.
* Особливості: бажаний заголовок всередині.
* Типові помилки: плутають із елементами <section> та <div>.

#### <section>

* Значення: змістовний розділ документа. Невідокремлюваний, на відміну від <article>.
* Особливості: бажаний заголовок усередині.
* Типові помилки: плутають із елементами <article> та <div>.

#### <aside>

* Значення: побічний, непрямий для сторінки контент.
* Особливості: може мати власний заголовок. Може зустрічатись кілька разів на сторінці.
* Типові помилки: вважати <aside> елементом для «бічної панелі» і розмічати цим елементом основний контент, пов'язаний з його елементами.

#### <nav>

* Значення: навігаційний розділ із посиланнями на інші сторінки.
* Особливості: використовується для основної навігації, а не всіх груп посилань. Наприклад, меню у підвалі сайту можна не обертати в <nav>. У підвалі зазвичай з'являється короткий список посилань (наприклад, посилання на головну, копірайт та умови) - це не є основною навігацією.
* Типові помилки: багато хто вважає, що у <nav> може бути лише список навігаційних посилань, але згідно зі специфікацією там може бути навігація у будь-якій формі.

#### <header>

* Значення: вступна частина смислового розділу або всього сайту, зазвичай містить підказки та навігацію. Найчастіше повторюється на всіх сторінках сайту.
* Особливості цих елементів може бути кілька на сторінці.
* Типові помилки: використовувати лише як шапку сайту.

#### <main>

* Значення: основне зміст сторінки, що не повторюється на інших сторінках.
* Особливості: має бути один на сторінці, виходячи з визначення.
* Типові помилки: включати тег те, що повторюється на інших сторінках (навігацію, копірайти і так далі).

#### <footer>

* Значення: заключна частина смислового розділу або всього сайту зазвичай містить інформацію про авторів, список літератури, копірайт і так далі. Найчастіше повторюється на всіх сторінках сайту.
* Особливості цих елементів може бути кілька на сторінці. Тег <footer> не повинен перебувати в кінці розділу.
* Типові помилки: використовувати лише як підвал сайту.

### 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>.
* Дрібні елементи у змістовних розділах. Списки, таблиці, демо-матеріали, параграфи та переноси, форми, цитати, контактна інформація та прогрес.
* Фразові елементи. Зображення, посилання, кнопки, відео, час та дрібні текстові елементи.

Існують прості правила для вибору потрібних елементів.

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

Правило для визначення <article>, <section> та <div>:

* Чи можна дати назву розділу і винести цей розділ на інший сайт? - <article>
* Чи можна дати назву розділу, але винести на інший сайт не можна? - <section>
* Не можна дати назву блоку (наприклад «новини» чи «фотогалерея»)? - <div>

#### 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, коли здійснюються кардинальні зміни в структурі сайту. Акуратний верстальник прагне це робити якнайчастіше. Перевірку HTML і CSS коду краще робити з використанням програм-валідаторів чи Інтернет-сервісів, які відображають знайдені помилки відповідно до різних стандартів. Кращим валідатором за допомогою якого можна перевірити 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 Internet Explorer, Mozilla FireFox, Opera, Chrome, Edge.

#### 2. Застосування коментарів

Основні HTML блоки коментарями коментуються в такий спосіб:

<!-- BEGIN FOOTER -->

<!-- END FOOTER -->

CSS блоки в такий:

/\* FOOTER \*/

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

#### 3. Впорядкування в таблиці стілів

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

 /\* 1. Скидання CSS \*/

 /\* 2. Типові елементи \*/

 /\* 2.1. Заголовки \*/

 /\* 2.2. Посилання \*/

 /\* 2.3. Елементи форм \*/

 /\* 3. HEADER (Шапка сайту) \*/

 /\* 4. FOOTER (підвал) \*/

 /\* 5. SIDEBAR (Праворуч) \*/

#### 4. Змістовні назви

Назви класів і id повинні за змістом відповідати застосуванню, наприклад, *text-center, main-menu, gallery, news.*

#### 5. Вживання Javascript

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

#### 6. Ширина сторінки

Для еластичних макетів обов'язково повинна бути задана мінімальна і максимальна ширина. Для фіксованих дизайнів ширина має 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 при відсутностості або малій кількості контенту.

#### 10. Кодування тексту

На сьогоднішній день для тексту стандартним є кодування UTF-8. Це кодування застосовують для HTML, CSS та JS коду, інакше ймовірність довгого виправлення багів критично зростає.

#### 11. Використання нестандартних шрифтів

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

#### 12. Використання невалідних фрагментів

Якщо макет не проходить 100% HTML-валідацію, варто вживати лише виправданий невалідний код.

#### 13. Звітність про виправлення

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

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

## 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>