***Lesson 1. HTML***

**Що таке WEB ?**

WEB — система [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), створена [Дональдом Кнутом](https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%B4_%D0%9A%D0%BD%D1%83%D1%82) як перша реалізація його концепції [«грамотного програмування»](https://uk.wikipedia.org/w/index.php?title=%D0%93%D1%80%D0%B0%D0%BC%D0%BE%D1%82%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F&action=edit&redlink=1): ідеї, що [програмне забезпечення](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) можна створювати як [літературний твір](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D1%82%D0%B5%D1%80%D0%B0%D1%82%D1%83%D1%80%D0%BD%D0%B8%D0%B9_%D1%82%D0%B2%D1%96%D1%80). Такий твір складається з [фрагментів](https://uk.wikipedia.org/wiki/%D0%A1%D0%BD%D1%96%D0%BF%D0%B5%D1%82) [початкового коду](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D0%B0%D1%82%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4), вбудований у текст, що описує програму [природною мовою](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%80%D0%BE%D0%B4%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) людини. Це пряма протилежність більшості [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), де текст для читання людиною (документація) вбудовується у тіло програми.

WEB складається з двох програм: TANGLE, що генерує текст програми на [Паскалі](https://uk.wikipedia.org/wiki/Pascal), і WEAVE, що продукує відформатовану [TeX](https://uk.wikipedia.org/wiki/TeX)-документацію

Існує версія системи WEB для мови [C](https://uk.wikipedia.org/wiki/C_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), CWEB (наприклад, програма Web2C виконує ту ж роль, що і TANGLE для генерації Паскаль-тексту). З інших програм можна відзначити noweb, що не залежить від кінцевої мови програмування.

Найбільш відомі і важливі програми, написані на WEB — це [TeX](https://uk.wikipedia.org/wiki/TeX) і [Metafont](https://uk.wikipedia.org/wiki/Metafont).

**І все? А з чого складається?**

**Вебсторінка** Інформаційний ресурс, доступний в мережі World Wide Web, який можна переглянути у веб браузері.

**Браузер** Це програма, що дозволяє відображати сторінки мережі Інтернет на екрані комп’ютера, мобільного телефона, планшета, розумного годинника, передавати і отримувати дані з «всесвітньої павутини».

**WEB-розробка** Це процес створення WEB-сайтів або WEB-додатків.

Основними етапами процесу є:

* вебдизайн,
* верстка сторінок,
* програмування для веб на стороні клієнта і сервера,
* конфігурування вебсервера.

**Front end та back end**

Front end — це  [інтерфейс](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) для взаємодії між користувачем і back end. Front end та back end можуть бути розподілені між однією або кількома системами.

В [програмній архітектурі](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) може бути багато рівнів між [апаратним забезпеченням](https://uk.wikipedia.org/wiki/Hardware) та [кінцевим користувачем](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87). Кожен з цих рівнів може мати як front end, так і back end. Front — це абстракція, спрощення базового компоненту через надання користувачу зручного ([user-friendly](https://uk.wikipedia.org/wiki/User-friendly)) [інтерфейсу](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0).

**Front - End**

Це зовнішній вигляд сайту, його структура

**Back - End**

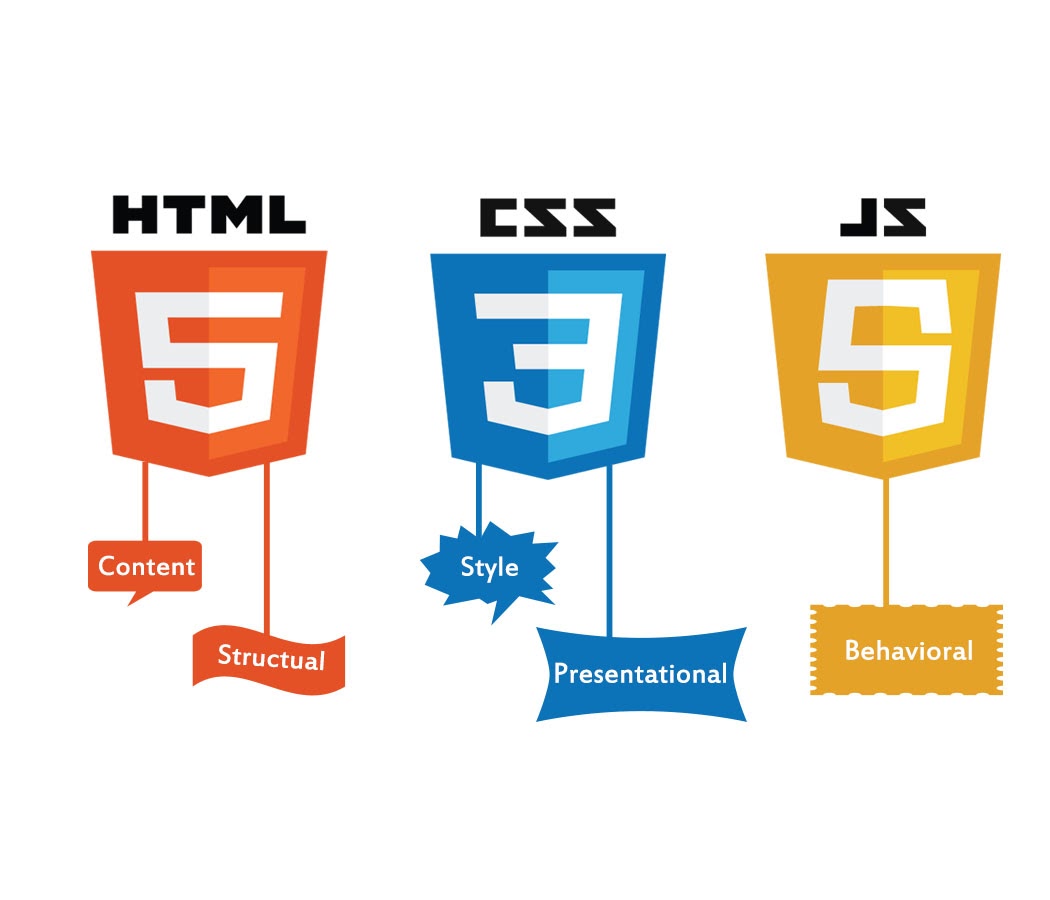
Це вся механіка сайту, його «невидимий двигун»

**Front - End (Фронтенд)**

Все, що браузер може читати, виводити на екран і / або запускати.

Щоб відобразити контент сайту найкращим чином та забезпечити оптимальну взаємодію користувача з сайтом, фронтенд-розробники використовують багато корисних інструментів, найважливіші з яких це

**HTML, CSS та JavaScript.**



**HyperText Markup Language** **— мова розмітки гіпертексту** це код, який використовується для структурування і відображення веб-сторінки та її контенту.

**HTML** це не мова програмування, це мова розмітки, яка каже вашому браузеру, як відображати вміст веб-сторінки, яку ви переглядаєте.

**HTML - документ має свою структуру**

**<!DOCTYPEhtml>**

**<html>**

**<head>**

**<title>WEB programming</title>**

**<script></script>**

**</head>**

**<body>**

**<!–Початок коду-->**

**<div>**

**<!–Контент 1-->**

**</div>**

**<div>**

**<!–Контент 2-->**

**</div>**

**<!–Кінець коду-->**

**</body>**

**</html>**

**Тег** - елемент розмітки гіпертексту. Мітка, яку використовує розробник для вказівки браузеру, як він повинен показувати WEB-сайт. Тег також має свою структуру

***Всі HTML-елементи діляться на п'ять типів:***

* **порожні елементи** - <area>, <base>, <br>, <col>, <embed>, <hr>, <img>, <input>, <link>, <menuitem>, <meta>, <param>, <source>, <track>, <wbr>;
* **елементи з неформатований текстом** - <script>, <style>;
* **елементи, які виведуть звичайний текст** - <textarea>, <title>;
* **елементи з іншого простору імен** - MathML і SVG;
* **звичайні елементи** - всі інші елементи.

# *Як написати і запустити HTML на комп'ютері?*

# *Крок 1. Качаємо текстовий редактор*

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

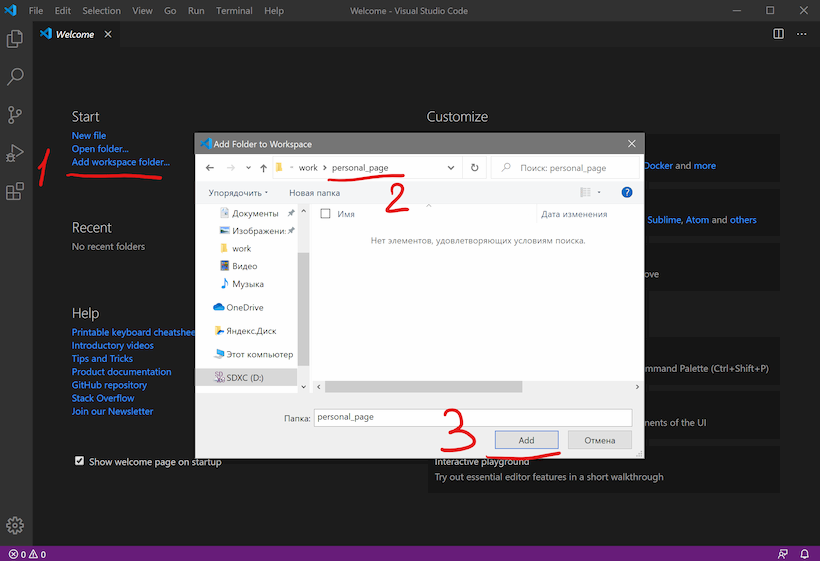
Зайдіть на  <https://code.visualstudio.com/> і скачайте редактор. Якщо у вас Windows, то натисніть на будь-яку з синіх кнопок. Якщо OS X або Linux - натисніть Other platforms.

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

***Крок 2. Запускаємо редактор і оглядаємося***

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

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



По кроках на скріншоті:

1. Add workspace folder - відкриває меню вибору папки.
2. Створимо нову папку personal\_page в будь-якому зручному місці і зайдемо в неї.
3. Натиснемо Add.

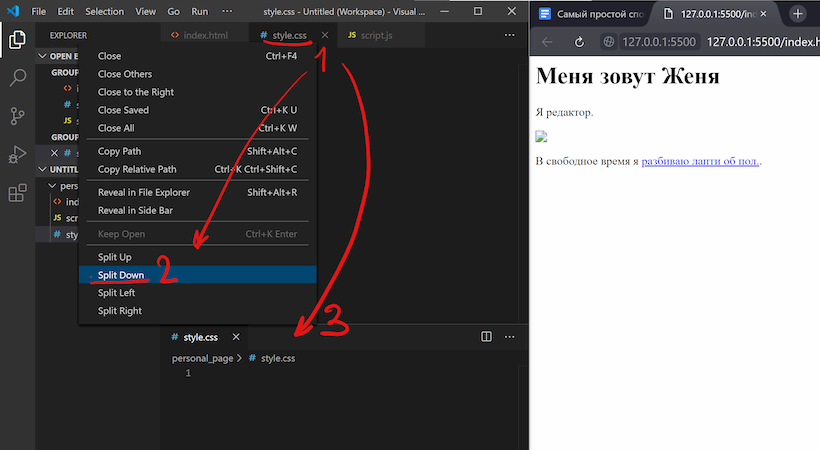
Після цього зліва з'явиться панель Explorer з порожнім робочим простором Untitled (Workspace). Ми створили папку, давайте її наповнимо.

***Крок 3. Додаємо файли***

Після створення папка порожня. Клацнемо правою кнопкою по заголовку personal\_page і додамо три файли, які знадобляться в роботі - index.html, style.css і script.js. Для початку цього вистачить.

***Крок 4. Робимо роботу зручніше***

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



***Крок 5. Додаємо код***

Поки відредагуємо тільки index.html (файл з розміткою) і style.css (файл зі стилями), а script.js залишимо на майбутнє. Якщо у вас вже є який-небудь код, напишіть його, або використовуйте готовий - ми, наприклад, візьмемо код з інтерактивних курсів.

*index.html*

<!DOCTYPE html>

<html lang="ru">

<head>

<title>Сайт начинающего верстальщика</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<header>

<nav>

На главную

</nav>

</header>

<main>

<article>

День первый. Как я забыл покормить кота

Кто бы мог подумать, что семантика это так важно, мне срочно нужно было об этом поговорить.

Взгляд упал на кота. Кот издал настойчивое «Мяу». И я понял — пришло время для первой записи в блог. И покормить кота.

</article>

<aside>

Здесь могла быть ваша реклама.

</aside>

</main>

<footer>

Подвал сайта

</footer>

</body>

</html>

*style.css*

Скопіюємо код зі стилями з файлу <https://htmlacademy.ru/assets/courses/299/outlines.css>  - відкрийте його в браузері, скопіюйте всі рядки і вставте в файл style.css в редакторі.

***Крок 6. Запускаємо код і дивимося на результат***

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

**Базові теги HTML**

У загальному випадку тег записується наступним чином:

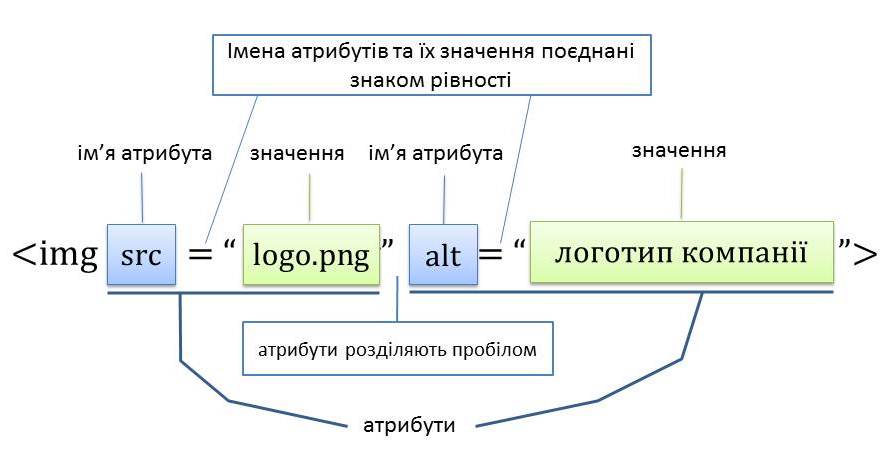
<ім'я тега атрибут1="значення1" атрибут2="значення2" ...>

Ось декілька прикладів тегів з атрибутами:

<p class="important"> ... </p>

<a class="external" href="http://cpto.dp.ua"> ... </a>

<img class = "logo" src = "logo.png">



**Оформлення смислових частин сторінки абзацами:**

* <p>…</p> − розбиває текст на параграфи (абзаци);
* <br> − обриває рядок і починає новий;
* <nobr>…</nobr> − забороняє перенос тексту на інший рядок;
* <wbr>…</wbr> − дозволяє розривати довгі слова (наприклад, при виведенні довгих медичних термінів);
* <!- - … - -> − тег коментарів; розміщений в ньому текст не відображається на web-сторінці.

**Основні теги форматування тексту та елементів дизайну**

* <hn>…</hn> − керує розміром символів в заголовках (1≤n≤6); зі збільшенням n розмір шрифту зменшується.
* <big>...</big> − збільшує розмір шрифту на 10% у порівнянні з базовим.
* <small>…</small> − зменшує розмір шрифту на 10% у порівнянні з базовим.
* <pre>…</pre> − позначає фрагмент тексту, форматування якого здійснено наперед; зручно використовувати при виведенні на екран текстів програм на мовах програмування, або при виведенні тексту, який містить спеціальні символи; в деяких випадках замінює тег <nobr>…</nobr>.
* <code>…</code> − позначає фрагмент тексту усередині рядка, який вдповідає елементам коду.
* <hr> − непарний тег горизонтальної лінії; може бути засобом організації тексту і дизайну.

**Атрибути тега <hr>:**

* + align=left (right, center) − вирівнювання лінії на сторінці;
  + sіze=n − товщина лінії в пікселях (1≤n≤175); за замовчуванням 2 пікселі;
  + color="колір" − колір лінії;
  + width=n% − ширина лінії у відсотках до ширини екрану;
  + noshade − атрибут без параметра; показує об'ємні лінії, встановлюється за замовчуванням;
  + shade − атрибут без параметра; показує контур лінії; якщо заданий колір, то атрибут не працює.
* <left> ... </left> − вирівнювання тексту по лівому краю сторінки.
* <right> ... </right> − вирівнювання тексту по правому краю сторінки.
* <center> ... <center> − вирівнювання тексту по середині сторінки.

Наведені вище теги вирівнювання дещо застаріли. Для цього краще використовувати відповідні властивості CSS.

* <b> … </b> − виділення напівжирним шрифтом.
* <i> … </i> − виділення курсивом.

**HTML-списки** використовуються для групування пов'язаних між собою фрагментів інформації. Існує три види списків:

* **маркований список** - <ul>- кожен елемент списку <li>зазначається маркером,
* **нумерований список** - <ol>- кожен елемент списку <li>зазначається цифрою,
* **список** Кожен елемент списку має додатковий блок, розташований збоку, який не бере участі в компонуванні.**визначень** - <dl>- складається з пар термін <dt>- <dd>визначення.

Кожен список являє собою контейнер, усередині якого розташовуються елементи списку або пари термін-визначення.

Елементи списку поводяться як блокові елементи, розташовуючись один під одним і займаючи всю ширину блоку-контейнера.

1. **Маркований список**

**Маркований список** являє собою невпорядкований список *(від англ. Unordered List)* . Створюється за допомогою елемента <ul>. В якості маркера елемента списку виступає позначка, наприклад, зафарбований кружок.

Браузери за замовчуванням додають наступне форматування блоку списку:

ul {

padding-left: 40px;

margin-top: 1em;

margin-bottom: 1em;

}

CSS

Кожен елемент списку створюється за допомогою елемента <li> *(від англ. List Item)* .

Для елемента <ul>доступні [глобальні атрибути](https://html5book.ru/html-attributes/) .

<ul>

<li>Microsoft</li>

<li>Google</li>

<li>Apple</li>

<li>IBM</li>

</ul>

HTML

* Microsoft
* Google
* Apple
* IBM

ФІГУРА. 1. МАРКОВАНИЙ СПИСОК

**2. Нумерований список**

**Нумерований список** створюється за допомогою елемента <ol>. Кожен пункт списку також створюється за допомогою елемента <li>. Браузер нумерує елементи по порядку автоматично і якщо видалити один або кілька елементів такого списку, то інші номери будуть автоматично перераховані.

Блок списку також має стилі браузера за замовчуванням:

ol {

padding-left: 40px;

margin-top: 1em;

margin-bottom: 1em;

}

CSS

Для елемента <li>доступний атрибут value, який дозволяє змінити номер за умовчанням для вибраного елементу списку. Наприклад, якщо для першого пункту списку задати <li value="10">, то інша нумерація буде перерахована щодо нового значення.

Для елемента <ol>доступні наступні атрибути:

|  |  |
| --- | --- |
| ТАБЛИЦЯ 1. АТРИБУТИ ЕЛЕМЕНТА <OL> | |
| **Атрибут** | **Опис, прийняте значення** |
| reversed | Атрибут reversedзадає відображення списку в зворотному порядку (наприклад, 9, 8, 7 ...). |
| start | Атрибут startзадає початкове значення, від якого піде відлік нумерації, наприклад, конструкція <ol start="10">першого пункту присвоїть порядковий номер «10». Також можна одночасно ставити тип нумерації, наприклад, <ol type="I" start="10">. |
| type | Атрибут typeзадає вид маркера 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>

<li>Microsoft</li>

<li>Google</li>

<li>Apple</li>

<li>IBM</li>

</ol>

HTML

1. Microsoft
2. Google
3. Apple
4. IBM

ФІГУРА. 2. НУМЕРОВАНИЙ СПИСОК

**3. Список визначень**

**Списки визначень** створюються за допомогою елемента <dl>. Для додавання терміна застосовується елемент <dt>, а для вставки визначення - елемент <dd>.

Блок списку визначень має наступні стилі браузера за замовчуванням:

dl {

margin-top: 1em;

margin-bottom: 1em;

}

CSS

Для елементів <dl>, <dt>і <dd>доступні [глобальні атрибути](https://html5book.ru/html-attributes/) .

<dl>

<dt>Режиссер:</dt>

<dd>Петр Точилин</dd>

<dt>В ролях:</dt>

<dd>Андрей Гайдулян</dd>

<dd>Алексей Гаврилов</dd>

<dd>Виталий Гогунский</dd>

<dd>Мария Кожевникова</dd>

</dl>

HTML

режисер:

Петро Точилін

У ролях:

Андрій Гайдулян

Олексій Гаврилов

Віталій Гогунський

Марія Кожевнікова

ФІГУРА. 3. СПИСОК ВИЗНАЧЕНЬ

**4. Як створити вкладений список**

Найчастіше можливостей простих списків не вистачає, наприклад, при створенні змісту ніяк не обійтися без **вкладених пунктів** . Розмітка для вкладеного списку буде наступною:

<ul>

<li>Пункт 1.</li>

<li>Пункт 2.

<ul>

<li>Подпункт 2.1.</li>

<li>Подпункт 2.2.

<ul>

<li>Подпункт 2.2.1.</li>

<li>Подпункт 2.2.2.</li>

</ul>

</li>

<li>Подпункт 2.3.</li>

</ul>

</li>

<li>Пункт 3.</li>

</ul>

HTML

* Пункт 1.
* Пункт 2.
  + Підпункт 2.1.
  + Підпункт 2.2.
    - Підпункт 2.2.1.
    - Підпункт 2.2.2.
  + Підпункт 2.3.
* Пункт 3.

ФІГУРА. 4. ВКЛАДЕНИЙ СПИСОК

**5. Багаторівневий нумерований список**

Багаторівневий список використовується для відображення елементів списку на різних рівнях з різними відступами. Розмітка для багаторівневого нумерованого списку буде наступною:

<ol>

<li>пункт</li>

<li>пункт

<ol>

<li>пункт</li>

<li>пункт</li>

<li>пункт

<ol>

<li>пункт</li>

<li>пункт</li>

<li>пункт</li>

</ol>

</li>

<li>пункт</li>

</ol>

</li>

<li>пункт</li>

<li>пункт</li>

</ol>

HTML

Така розмітка за замовчуванням створить для кожного вкладеного списку нову нумерацію, яка починається з одиниці. Щоб зробити вкладену нумерацію, потрібно використовувати такі властивості:

* counter-reset скидає один або кілька лічильників, задаючи значення для скидання;
* counter-incrementзадає значення приросту лічильника, тобто з яким кроком нумеруватиметься кожний наступний пункт;
* content - генерується вміст, в даному випадку відповідає за виведення номера перед кожним пунктом списку.

ol {

/\* убираем стандартную нумерацию \*/

list-style: none;

/\* Идентифицируем счетчик и даем ему имя li. Значение счетчика не указано - по умолчанию оно равно 0 \*/

counter-reset: li;

}

li:before {

/\* Определяем элемент, который будет нумероваться — li. Псевдоэлемент before указывает, что содержимое, вставляемое при помощи свойства content, будет располагаться перед пунктами списка. Здесь же устанавливается значение приращения счетчика (по умолчанию равно 1). \*/

counter-increment: li;

/\* С помощью свойства content выводится номер пункта списка. counters() означает, что генерируемый текст представляет собой значения всех счетчиков с таким именем. Точка в кавычках добавляет разделяющую точку между цифрами, а точка с пробелом добавляется перед содержимым каждого пункта списка \*/

content: counters(li, ".") ". ";

}

CSS

1. пункт
2. пункт
   1. пункт
   2. пункт
   3. пункт
      1. пункт
      2. пункт
      3. пункт
   4. пункт
3. пункт
4. пункт

***Lesson 2. CSS***

***Структура WEB проекту***

* **Project (Root) Folder:** основна папка, містить всі інші папки та файли
* **CSS Folder:** усі файли, що відносяться до стилів (.css файли)
* **Images Folder:** усі зображення (.jpg, .png, .gif, etc)
* **JavaScript Folder:** усі JavaScript файли (.js)

***Термінал на Linux***

**ctrl + alt + T - launch terminal**

**Основні команди**

clear - Очистити консоль

### **Навігація**

pwd- Показати поточний каталог  
ls - Показати файли в цій папці, крім прихованих  
ls -f- Показати файли в цій папці, включаючи і приховані  
cd c:/- Перейти в конкретний каталог  
cd -- Повернутися назад  
cd ..- Вийти на 1 рівень вгору  
cd ../..- Вийти на 2 рівня вгору

### **створення каталогів**

mkdir- Створити папку  
cd !$- Перейти в тільки що створену папку  
mkdir -p {app1,app2}- Створити відразу кілька папок  
mkdir -p app/{css,js}- Створити відразу кілька вкладених папок

### **створення файлів**

touch index.html- Створити файл index.html  
touch app/{css/main.css,js/main.js,index.html}- Створити відразу кілька файлів, ніяких зайвих прогалин бути не повинно

### **видалення файлів**

touch- дозволяє створювати файли  
rm test- Видалити порожню папку test  
rm -r test- Видалити папку test з файлами всередині неї

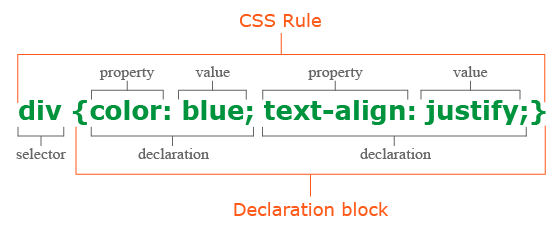
### **переміщення файлів**

mv app1/\*.\* app2 - Перемістити всі файли з папки app1 в папку app2

***Cascading Stylesheets***

***(CSS, каскадні таблиці стилів)***

це код, що використовується для стилізації сайту. Це мова таблиць стилів, яка дозволяє задавати стилі обраним елементам у HTML документах



Вся структура називається "**rule set" — "набір правил"** (частіше, скорочено, "правило"). Його частини також мають свої назви:

* **Selector — Селектор** Назва елементу HTML на початку правила. Селектор вибирає елемент чи елементи, які будуть стилізовані (у нашому випадку, елементи p). Для стилізації інших елементів просто змініть селектор.
* **Declaration — Визначення (декларація)** Одне правило на зразок color: red; вказує, яку властивість елементу ви бажаєте стилізувати.
* **Properties — Властивості** Шляхи, якими ви можете стилізувати даний HTML елемент. (У цьому випадку, color — це властивість елементів p). У CSS ви обираєте властивості, які хочете змінити у вашому правилі.
* **Property value — Значення властивості** Праворуч від властивості, після двокрапки, ми вказуємо **значення властивості,** обираючи з багатьох можливих значень для наданої властивості (є багато інших значень властивості color окрім red).



Необхідно приєднати **CSS** до вашого **HTML** документу, інакше стилі **CSS** не вплинуть на те, як браузер буде відображати **HTML** документ.

<link rel="stylesheet" href="style.css">

***Деякі CSS правила***

color - колір тексту

background-color - фон

font-size - розмір шрифта

width - ширина

height – висота

***Lesson 3. Tables***

**Елемент <img>** повинен містити файли у графічному форматі GIF, JPEG, SVG або PNG. Адреса файлу з картинкою задається через атрибут src.

**GIF** (Graphics Interchange Format — «формат обміну зображеннями»). Одними із головних особливостей формату є підтримка анімації та прозорості.

**PNG** (Portable Network Graphics) — збереження графічної інформації, що використовує стиснення без втрат. (Має прозорий фон).

**JPEG** (Joint Photographic Experts Group) – формат для зберігання фотозображень і подібних до них зображень.

**SVG** (Scalable Vector Graphics  масштабована векторна графіка) — специфікація мови розмітки, що базується на XML, та формат файлів для двовимірної векторної графіки, як статичної, так і анімованої та інтерактивної.

### Опис

Елемент **<table>** служить контейнером для елементів, що визначають вміст таблиці. Будь-яка таблиця складається з рядків і осередків, які задаються за допомогою тегів **<tr>** і **<td>** . Усередині **<table>** допустимо використовувати наступні елементи: **<caption>** , **<col>** , **<colgroup>** , **<tbody>** , **<td>** , **<tfoot>** , **<th>** , **<thead>** і **<tr>** .

Таблиці з невидимою кордоном довгий час використовувалися для верстки веб-сторінок, дозволяючи розділяти документ на модульні блоки. Подібний спосіб застосування таблиць знайшов втілення на багатьох сайтах, поки йому на зміну не прийшов більш сучасний спосіб верстки за допомогою шарів.

### Синтаксис

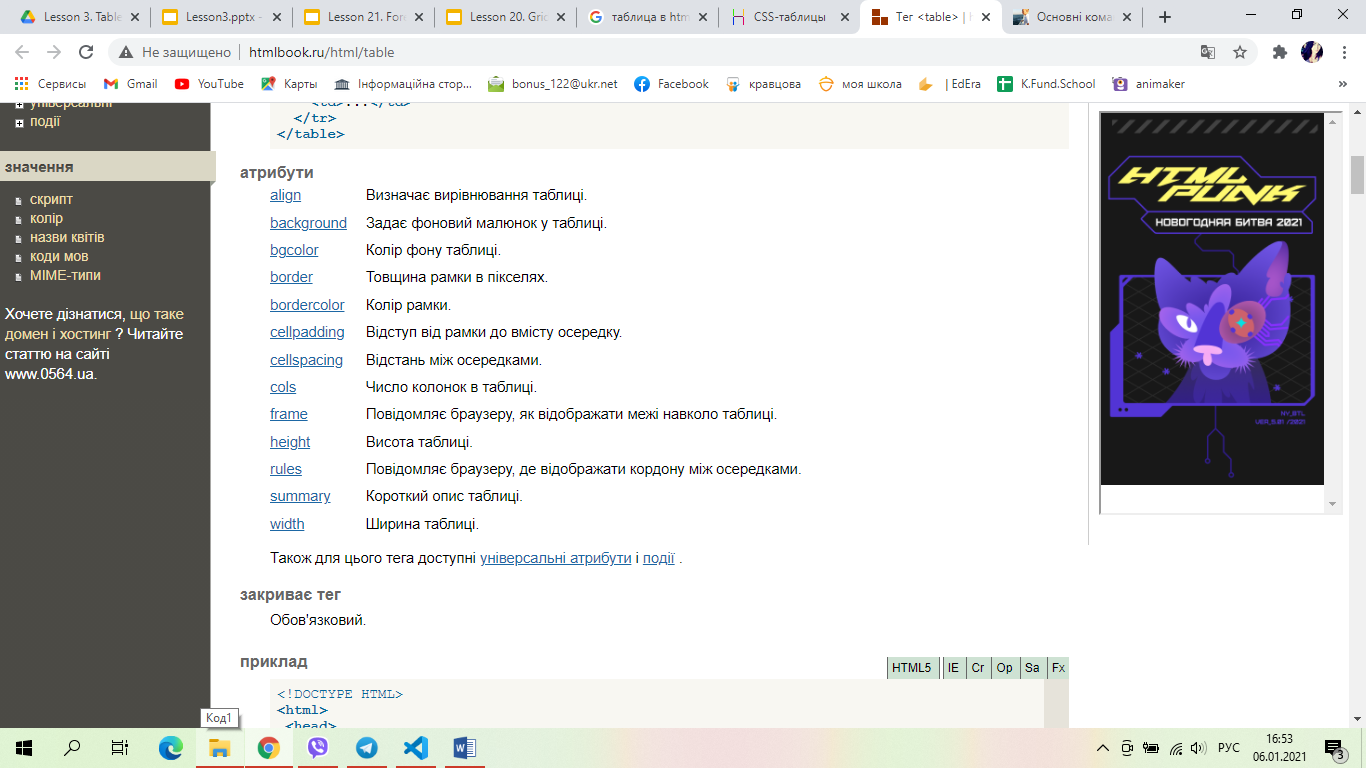
**<table>**

**<tr>**

**<td>**...**</td>**

**</tr>**

**</table>**



***Таблиця має свою структуру***

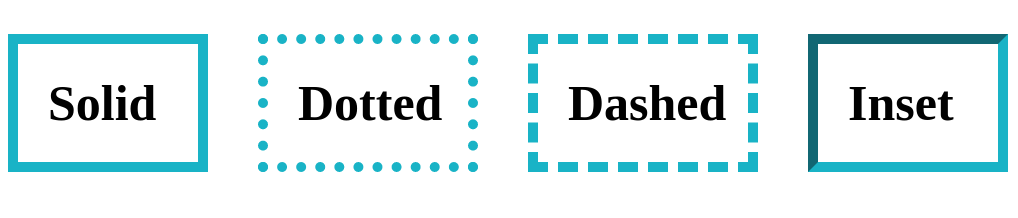


**border: 2px solid green;**

**2px - border width**

**solid - border type**

**green - border color**



***Lesson 4. Класи***

**Як зробити однакові елементи різними?**

Атрибут **class** задає стильовий клас, який дозволяє зв'язати певний тег зі стильовим оформленням.

**HTML**

<element class="classname">

**CSS**

.classname { color: red; }

**ID** - унікальне ім'я елемента, яке використовується для зміни його стилю і звернення до нього через скрипти.

**HTML**

<element id="idname">

**CSS**

#idname { color: red; }

***Lesson 5. GitHub***

***Як декілька людей розробляють один проект?***

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

Механизм, которым пользуется Git при вычислении хеш-сумм, называется SHA-1 хеш. Это строка длинной в 40 шестнадцатеричных символов (0-9 и a-f), она вычисляется на основе содержимого файла или структуры каталога.

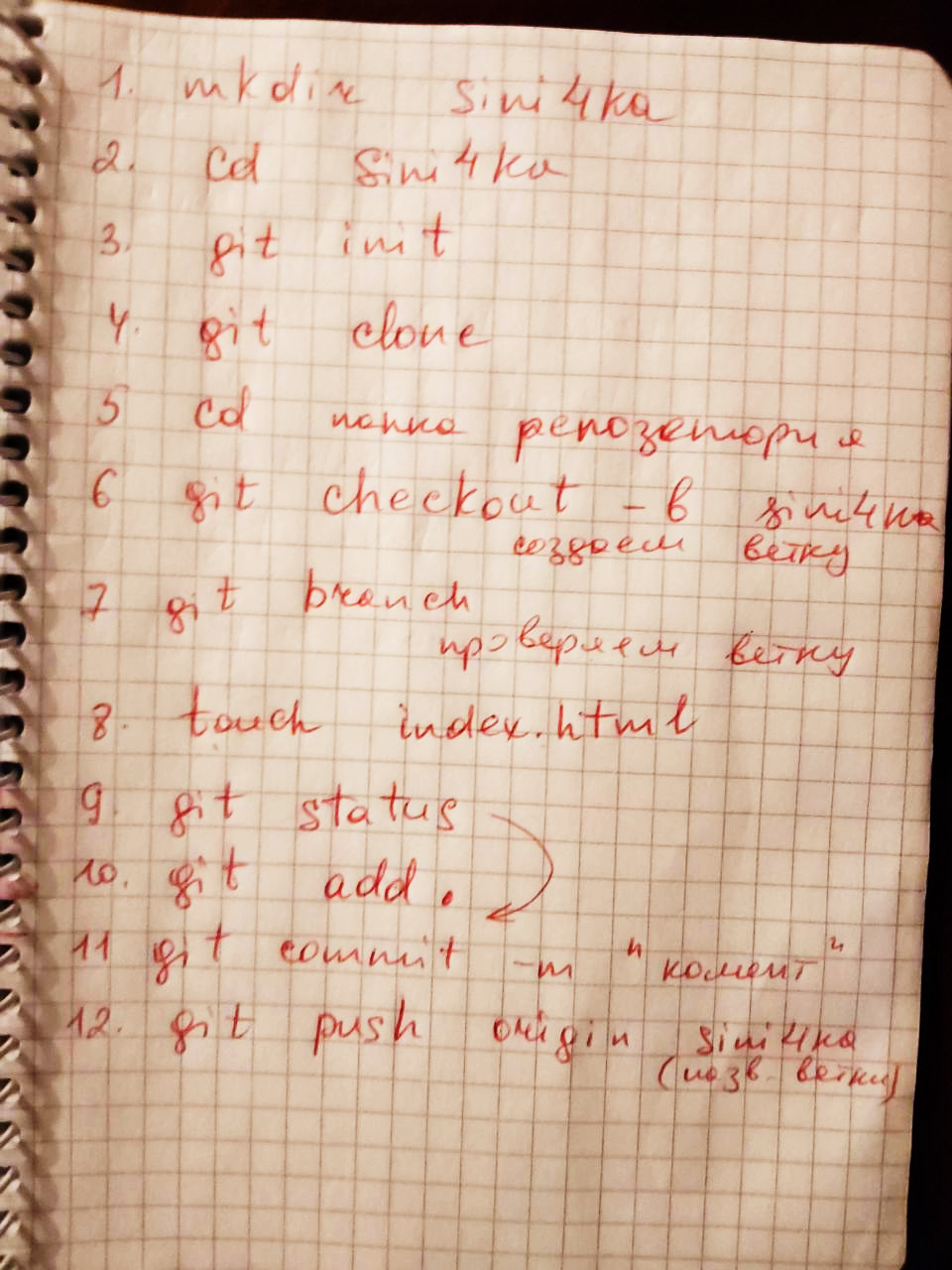
Git-директория — это то место, где Git хранит метаданные и базу объектов вашего проекта. Это самая важная часть Git, и это та часть, которая копируется при клонировании репозитория с другого компьютера.

Рабочая директория является снимком версии проекта. Файлы распаковываются из сжатой базы данных в Git-директории и располагаются на диске, для того чтобы их можно было изменять и использовать.

Область подготовленных файлов — это файл, обычно располагающийся в вашей Git-директории, в нём содержится информация о том, какие изменения попадут в следующий коммит. Эту область ещё называют “индекс”, однако называть её stage-область также общепринято.

***Власний акаунт GITHub***

* 1. Go to <https://github.com/>
  2. Click Sign Up
  3. Verify your account using e-mail
  4. Welcome to the **GIT family**



***Lesson 6. Блокова верстка сайту***

**Веб-сторінка** складається з безлічі різних елементів, які можуть мати складну структуру. Тому при створенні веб-сторінки виникає необхідність потрібним чином позиціонувати ці елементи, стилізувати їх так, щоб вони розташовувалися на сторінці потрібним чином. Тобто виникає питання створення макета сторінки, її верстки. **Блокова верстка**

**це підхід, при якому сайт будують на основі блоків, як блоки виступають теги div.**

Тег **<div>** - це так званий контейнер (блок), який може містити форматований текст, зображення та ін. Важливою особливістю блоків є їх здатність накладатися один на одного при верстці. Ця особливість надає блокам набагато більше можливостей по верстці сайту, ніж, приміром, таблиці.

**<div> - блоковий елемент, тому, якщо не задана ширина, розтягується на всю ширину вікна браузера**

**DEFAULT  
div {**

**width: 100%;**

**}**

**CUSTOM  
div {**

**width: 300px;**

**}**

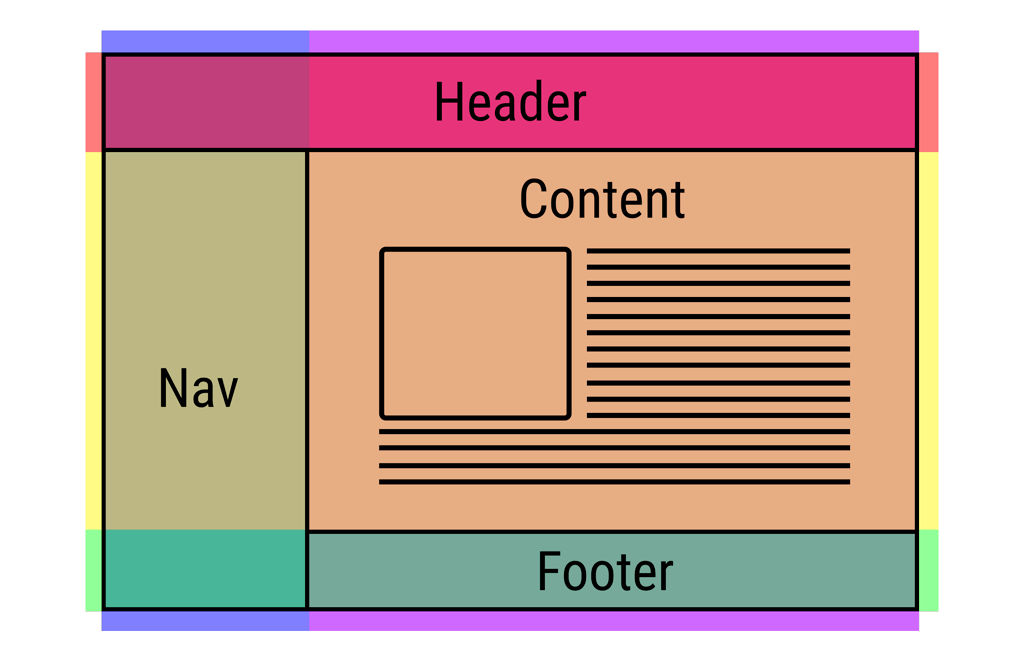
**<div> - не має оформлення. Щоб його побачити потрібно задати йому стилі в CSS.**

**Висота блоку, коли її не вказано, дорівнює вмісту. Порожній блок div має висоту - 0 px, тому не відображається на сторінці**

**div {**

**height: 300px;**

**}**

******

***Як же розташувати усі ці блоки?***

Відображення елементів на сторінці браузера здійснюється в тому порядку, в якому вони слідують у HTML коді.

Але, це можна змінити за допомогою CSS. Стилі дають нам можливість позиціонувати вміст і елементи на сторінці. Існує декілька різних типів позиціонування в CSS

***Властивість ‘display’***

Елементи, що мають блокове відображення (display: block) відображаються в потоці як прямокутні області, кожна з них на новій лінії одина під одній зверху вниз.

Елементи зі рядковим відображенням (display: inline) слідують один за одним зліва направо. Якщо простір праворуч закінчився, то вони переносяться на наступний рядок

***Властивість ‘float’***

Визначає, з якого боку буде вирівнюватися елемент, при цьому інші елементи будуть обтікати його з інших сторін.

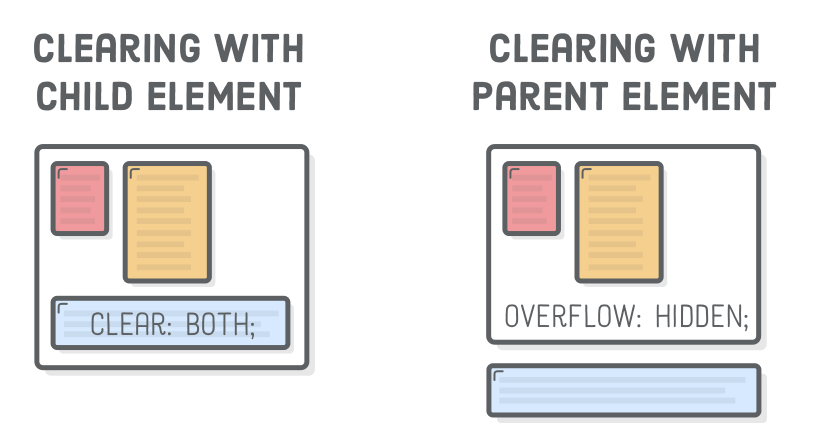
float: left | right | none | inherit

Властивість float має один побічний ефект. Її використання може впливати на батьківський елемент. Якщо такий елемент містить тільки float-елементи, то він нібито стискається, тобто його висота дорівнює нулю.

***Як це виправити?***

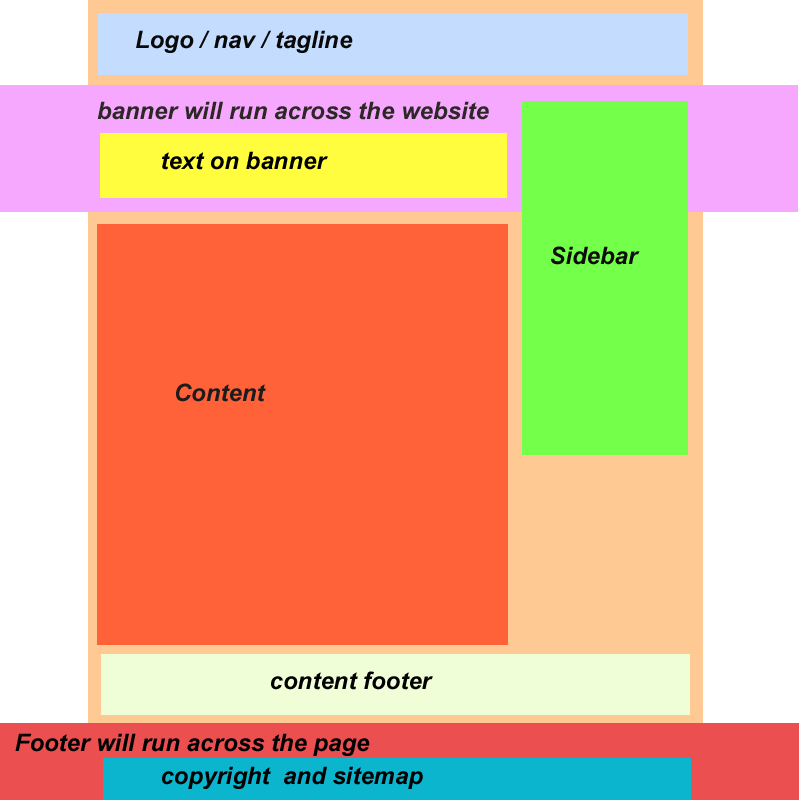
Метод overflow. Заснований на тому, що батьківського елементу необхідно встановити властивість overflow. Якщо значення цієї властивості встановлено в auto або hidden, то батьківський елемент збільшиться, щоб вмістити в себе всі float-елементи.

Властивість clear (left,right,both). Відміняє дію float. Будь-який елемент, у якого встановлено властивість clear, що не буде піднятий вгору, а відобразиться нижче, після float-елементів

***Lesson 7. Position***

***А якщо потрібно розташувати блок******“незвично”?***

CSS властивість **position** - це одна з властивостей за допомогою якої можна змінити базову поведінку елементів в потоці. Іншими словами, ця властивість дозволяє «висмикнути» будь-який елемент з потоку документа і розмістити його в іншому місці щодо вікна браузера або інших елементів на веб-сторінці.



***Яке буває позиціонування?***

Властивість position має 5 значень:

* **static** (статичне позиціонування);
* **relative** (відносне);
* **absolute** (абсолютне);
* **fixed** (фіксоване);
* **sticky** (липке).

Воно застосовується в поєднанні з властивостями зміщення блоку, які точно визначають, де елемент буде розташований шляхом переміщення елемента в різних напрямках

**Top (верх)**

**right (право)**

**bottom (низ)**

**left (ліво)**

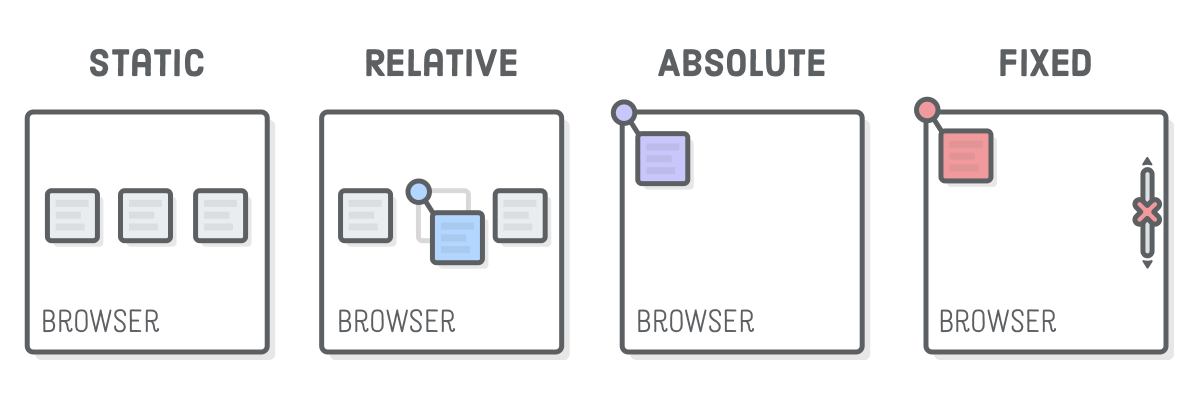
**.offset {**

**left: 20px;**

**position: relative;**

**top: 20px;**

**}**

***Lesson 8. FlexBox***

***Як вирівнювати всі ці блоки?***

**CSS Flexible Box Layout Module**, коротко flexbox (Флексбокс), створений щоб прибрати недоліки різних HTML конструкцій, в тому числі адаптованих під різну ширину і висоту,

**Flexbox** дозволяє контролювати найрізноманітніші параметри елементів контейнера: напрямок, порядок, ширину, висоту, вирівнювання вздовж і поперек, розподіл вільного місця, розтягнення і стиснення елементів.

**Flexbox** визначає набір CSS властивостей для контейнера (flex-контейнер) і його дочірніх елементів (flex-блоків). Перше, що потрібно зробити - це вказати контейнеру **display: flex**

Одним з основних понять в **fleхbox** є осі.

**Головною віссю** flex-контейнера є напрям, відповідно до якого розташовуються всі його дочірні елементи.

**Поперечною віссю** називається напрямок, перпендикулярний головної осі.

Головна вісь за замовчуванням розташовується зліва направо. Поперечна - зверху вниз. Напрямок головної осі flex-контейнера можна задавати, використовуючи базове css властивість **flex-direction**.

Властивість **justify-content** визначає те, як будуть вирівняні елементи вздовж головної осі. Доступні значення **justify-content**:

* **flex-start**
* **flex-end**
* **center**
* **space-between**
* **space-around**
* **space-evenly**

Якщо justify-content працює з головною віссю, то **align-items** працює з віссю, перпендикулярної головній осі.

* **flex-start**
* **flex-end**
* **center**
* **stretch**
* **baseline**

**flex-wrap**: керує перенесенням елементів, що не поміщуються у контейнер .

* **nowrap (default)** - вкладені елементи розташовуються в один ряд/колонку незалежно від того поміщаються вони в контейнер чи ні.
* **wrap** - включає перенесення елементів на наступний ряд, якщо вони не поміщаються в контейнер.
* **wrap-reverse** - теж що wrap тільки перенесення буде в зворотному напрямку).

<https://codepen.io/enxaneta/full/adLPwv> - на допомогу кожному

***Lesson 9. Forms. Beginning***

**Веб-форма** на сайті - це аналог паперової форми, анкети, бланка і опитувального листа. У формах є поля, призначені для заповнення, а також списки і перемикачі, які дозволяють користувачу вибрати один або кілька елементів.

Форму можна використовувати для того щоб отримувати відомості від відвідувачів сайту. Наприклад, за допомогою web-форми можна здійснювати збір персональних даних, відомості про замовлення, відомості, які необхідні для виставлення рахунку, способи доставки і так далі.

Основу будь-якої форми становить елемент **<form> ... </ form>**. Він не передбачає введення даних, так як є контейнером, утримуючи разом всі елементи управління форми - поля.

Елемент **<input>** створює більшість полів форми. Атрибути елемента відрізняються в залежності від типу поля, для створення якого використовується цей елемент.

Основний атрибут тега <input>, що визначає вид елемента - **type**.

**<input type="text"> - Displays a single-line text input field**

**<input type="radio"> - Displays a radio button (for selecting one of many choices)**

**<input type="checkbox"> - Displays a checkbox (for selecting zero or more of many choices)**

**<input type="submit"> - Displays a submit button (for submitting the form)**

**<input type="button"> - Displays a clickable button**

Елемент **<textarea> ... </ textarea>** використовується замість елемента

<input type = "text">, коли потрібно створити великі текстові поля. Розміри поля встановлюються за допомогою атрибутів **cols** - розміри по горизонталі, **rows** - розміри по вертикалі.

Списки дають можливість розташувати велику кількість пунктів компактно. Розкриваються списки створюються за допомогою елемента **<select> ... </ select>**. Вони дозволяють вибрати одне або кілька значень із запропонованого переліку.

Щоб додати до списку пунктів використовуються елементи

**<option> ... </ option>**, які розташовуються всередині **<select>**.

***Як зрозуміти де який******input?***

Написи до елементів форми створюються за допомогою елемента

**<label> ... </ label>**.

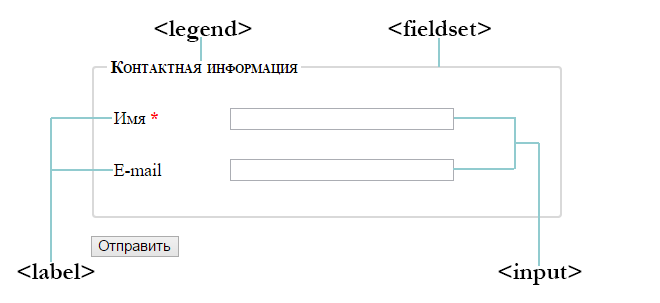
Існує два засоби поєднання напису і поля: з атрибутом ‘for’ та без нього.

Якщо поле знаходиться всередині елемента <label>, то атрибут for вказувати не потрібно.

***Групування елементів форми***

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

Кожній групі елементів можна привласнити назву за допомогою елемента **<legend>**, який йде відразу за тегом **<fieldset>**.

<form>

**<fieldset>**

**<legend>**Контактная информация**</legend>**

     <p><label for="name">...</p>

     <p><label for="email">...</p>

**</fieldset>**

<p><input type="submit" value="Отправить"></p>

</form>

***Кнопки Submit***

Елемент **<button> ... </ button>** створює клікабельні кнопки. На відміну від кнопок, створених <input>, всередину елемента <button> можна помістити контент - текст або зображення.

Для коректного відображення елемента <button> різними браузерами потрібно вказувати атрибут type, наприклад, <button type = "submit"> </ button>.

Додавання атрибута **required** в будь-який елемент області введення тексту скаже браузеру, що перед відправкою форми потрібно ввести значення. Таким чином, форму не можна буде відправити, якщо потрібне поле залишилося незаповненим.

**<input type="text" name="name" required />**

***Lesson 10.***

**:link - unvisited link**

**:hover - hover an element**

**:visited - visited link**

**:first-child - first element**

**:active - active elements**

**Селектор :link застосовується до посилань, які ще не відвідувалися користувачем та задає їм стильове оформлення.**

**a:link** {

  color: #FF0000;

}

**Селектор :visited використовується для вибору вже відвіданих посилань.**

**a:visited** {

  color: #FF0000;

}

**Селектор :hover використовується для вибірки та задання стилю посиланням, на які наведено курсор.**

**a:hover** {

  color: #FF0000;

}

**Селектор :focus застосовується, коли елемент отримує фокус (за кліком або клавішею ‘tab’)**

**a:focus**{

  border-color: #FF0000;

}

**Селектор :first-child використовується для обрання визначених елементів, які є першим нащадком свого батька.**

**li:first-child** {

  color: #FF0000;

}

***Додай HTML без HTML***

**Псевдоелемент:** використовується для створення нових елементів, які не існують в документі, він набуває тиж самих властивостей, що і звичайний селектор.

Тобто використання псевдоедементів дозволяє фактично додати HTML-елементи за допомогою CSS-коду

**:: before** - дозволяє вивести дані перед вмістом елемента

**:: after** - дозволяє вивести необхідні дані після вмісту елемента

Вміст псевдоелемента задається за допомогою CSS властивості **content**. Якщо псевдоелемент буде без вмісту, все одно необхідно вказувати цю властивість к

content: "". Якщо цього не зробити - псевдоелемент відображатися не буде.

***Lesson 11.***

***Чому зростає розмір блоку?***

Властивість **box-sizing** дозволяє змінити алгоритм розрахунку розмірів блоку, щоб властивості **width** і **height** задавали розміри не контенту, а саме розміри блоку.

CSS властивість **position** - це одна з властивостей за допомогою якої можна змінити базову поведінку елементів в потоці.

Властивість **border-radius** - універсальна властивість і дозволяє одночасно задати значення округлень всіх кутів

Властивість **transform** дозволяє застосовувати 2D або 3D трансформацію до елементу.

Ця властивість дозволяє обертати, масштабувати, переміщувати, нахиляти елемент, тощо.

Властивість **transform** отримує в якості значень "трансформуючі функції".

Ці функції виглядають як:

* **scale() -** масштабування элемента по горизонталі та вертикалі. Значення меньше 1 - зменшує елемент, більше - збільшує
* **translateX() -** зсув по горизонталі і вертикалі
* **rotate() -** обертає елемент, на заданий кут, в двомірному просторі.

***“Правила” малювання***

Кожен проект на чистому **CSS** повинен складатися з наступних компонентів:

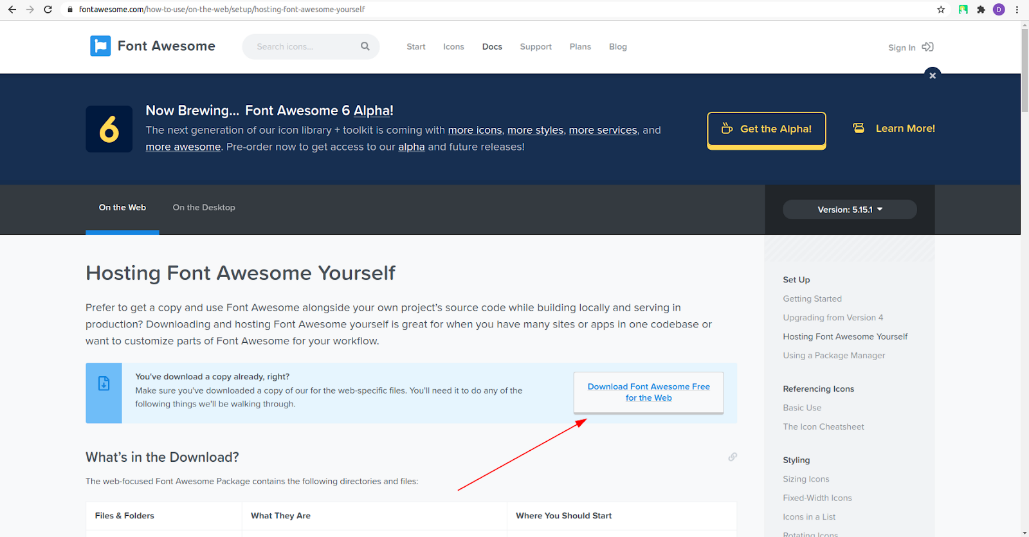
* тег **div** для кожної фігури;
* певний **CSS-клас**, для кожного **div**;

***Lesson 12. Font Awesome.***

**Font Awesome** - це шрифт, створений з іконок.

**Векторні іконки** - вони добре виглядають незалежно від розміру відображення.

**Шрифт**, що легко налаштовується - можна керувати, як будь-якими іншими шрифтами(колір, анімація тощо).

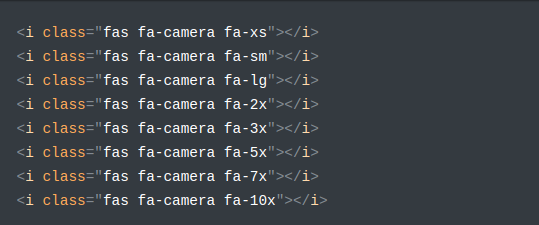
**Кросбраузерні** - шрифтові іконки відмінно працюють з будь-яким браузером.

[**https://fontawesome.com/**](https://fontawesome.com/)

Для роботи з іконками використовуйте тег

**<i class=“fa fa-name”></i>**

**Розмір іконок = спеціальний клас до елемента з потрібною іконкою.**



Класи **fa-rotate** та **fa-flip** використовуються для створення ефекту обертання іконок.

**fa-rotate –** здійснює обертання іконки на заданий кут.

**fa-flip –** віддзеркалює задану іконку у заданому напрямку.

Класи fa-spin та fa-pulse використовуються для створення простих анімацій обертання за допомогою іконок.

**fa-spin –** здійснює обертання іконки за годинниковою стрілкою рівномірно.

**fa-pulse –** здійснює обертання іконки за годинниковою стрілкою пульсуючим способом(8 кроків).

***Lesson 13. Media queries***

**Адаптивне верстання** – підхід, що припускає зміну дизайну залежно від поведінки користувача, розміру екрана, платформи і орієнтації девайса.

Медіазапроси використовуються в тих випадках, коли потрібно застосувати різні CSS-стилі, для різних пристроїв

**Використання медіазапросу може залежати від:**

* типу відображення (для принтера, монітора або смартфона)
* конкретних характеристик пристрою (ширини вікна перегляду браузера)
* зовнішнього середовища

(зовнішнє освітлення)

Почнемо визначення медіа-запитів з правила **@media**, а потім включимо правила CSS в круглі дужки. Воно використовується для вказівки типів цільових пристроїв

**@media** () {

//Правила CSS

}

**Усередині дужки ми встановлюємо умову.** Наприклад, нам потрібно збільшити розмір шрифту для мобільних пристроїв. Для цього встановлюємо максимальну ширину, яка перевіряє ширину пристрою

**.text** {

  font-size: 14px;

}

**@media** (**max-width: 480px**) {

**.text** {

    font-size: 16px;

  }

}



Існує досить багато параметрів, які можуть бути використані у якості умови для медіазапросу. Ми зупинимось на основних з них.

***Типи пристроїв***

* **all** - для всіх типів пристроїв;
* **print** - для принтерів;
* **screen** - для екранів комп'ютерів,

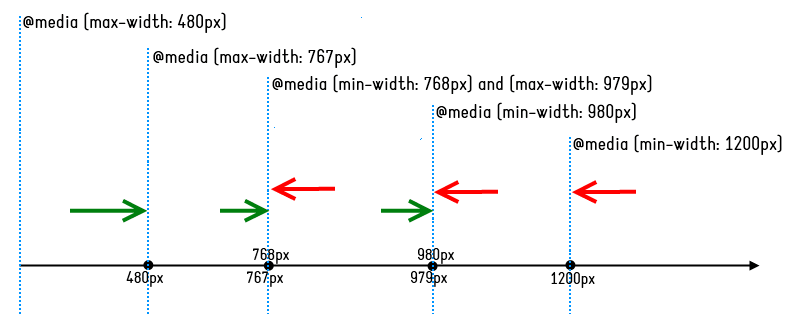
планшетів і смартфонів;

* **speech** - для скрінрідерів, які «читають» сторінку вголос.

***Деякі медіа функції***

* **max-width** - вказує на те, якою має бути максимальна робоча область пристрою (браузера).
* **min-width -** задає мінімальну ширину області viewport в px, em або інших одиницях
* **orientation -** функція, яка перевіряє те, в якому режимі (portrait або landscape) відображається сторінка

**Breakpoints** (переломні / контрольні точки) - зміна умов з однієї на іншу, при якій зовнішній вигляд сайту (розкладка) змінюється.



Іноді треба використати декілька параметрів пристрою, додати специфічні умови. У такому разі використовуються логічні оператори

**Логічні оператори**

* **and -** вимагає обов'язкового виконання всіх зазначених умов

**@media** screen **and** (min-width: 1200px) and (orientation: landscape) { /\* Стили CSS ... \*/ }

* **, (кома) -** вимагає обов'язкового виконання хоча б однієї із зазначених умов в медіа запиті.

**@media** (min-width: 544px)**,** (orientation: landscape) { /\* Стили CSS ... \*/ }

* **not -** призначений для заперечення зазначеної умови

**@media not** screen and (orientation: portrait) {/ \* Стилі CSS ... \* /}

***Lesson 14. Drop-down menu***

***Як знайти щось потрібне на сайті?***

**Drop-down menu** – це простий варіант відображення ряду/списку варіантів. Доступні варіанти, зазвичай, з’являються тільки після наведення або кліку на основний елемент

***HTML - каркас***

Для створення такого меню використовується тег **<ul>**. Скільки рівнів у меню стільки списків повинно бути у структурі HTML - документу

Маркований (невпорядкований) список в HTML визначається тегом **<ul>** (Unordered List Element).

Кожен елемент списку повинен починатися з тега **<li>** (скорочене від англійського list item - елемент списку).

Між відкритим і закритим тегами ul повинен бути один або кілька елементів li.

***<ul>***

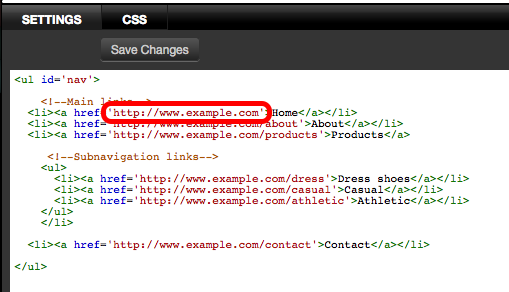
***<li>Element1</li>***

***<li>Element2</li>***

***<li>Element3</li>***

***</ul>***

***Структура drop-down menu може виглядати так:***



Існує декілька засобів, як зробити елемент невидимим на сторінці. Це можна зробити за допомогою властивостей: **display, opacity, visibility**

Властивість **display: none** приховує елемент на сторінці. Макет формується, немов елемента і не було.

Використання **visibility: hidden** трохи відрізняється. Даний спосіб прибирає елемент з поля зору, при цьому не зачіпаючи потік html-документа, залишаючи порожній простір в тому місці, де даний елемент повинен був знаходитися.

За допомогою властивості **opacity** ми можемо змінювати прозорість елементу. Значення **opacity: 0;** робить прозорим весь елемент цілком, включаючи його вміст, а не тільки фон елемента

Селектор **:hover** використовується для вибірки та задання стилю елементам, на які наведено курсор.

Розмітка горизонтального багаторівневого меню базується на **css-позиціонуванні**. Всім елементам списку (**li**) задається відносне позиціонування, а для меню, що випадає (**ul**) будь-якого рівня - абсолютне позиціонування.

***Lesson 15. Bootstrap. Intro. Pro***

**Bootstrap** є найбільш популярним HTML, CSS, та JS **фреймворком** для зовнішнього вигляду та інтерактивності веб‑сторінок.

**Bootstrap** допомагає швидше та простіше розробляти **зовнішній вигляд веб-сторінок**. Він підходить для людей з будь-яким рівнем досвіду, для пристроїв будь-яких форматів, та проектів будь-якого розміру.

**Ряд** - це елемент бутстрап сітки, який виступає в ролі безпосереднього контейнера для адаптивних блоків (інших елементів сітки, які ми розглянемо нижче).

**Ряд** - це просто батьківський елемент для адаптивних блоків. Тобто адаптивні блоки завжди повинні бути розміщені безпосередньо в ньому.

**<div** class="row"**>**...**</div>**

**Адаптивні блоки** - це основні будівельні елементи сітки. Саме від них буде залежати, як макет веб-сторінки буде відображатися на різних контрольних точках (без позначення, sm, md, lg і xl).

**Адаптивні блоки** - це блоки, ширина яких (у відсотковому відношенні від батька) на різних breakpoint може бути різною. Блок на sm може мати ширину, що дорівнює 50% від батьківської, а на md - 25%.

***З чого складається сітка?***

* обгорткових контейнерів (елементів з класом container і container-fluid);
* рядів (елементів з класом row):
* адаптивних блоків (елементів, що мають один або кілька класів, рівних col або починаються з col).

Основний принцип верстки макету веб-сторінки на сітці **Bootstrap 4** полягає у вкладанні одних адаптивних блоків в інші.

При цьому ширина адаптивних блоків це завжди відносна величина, яка задається в колонках Bootstrap і залежить тільки від **ширини батьківського елементу**, тобто ряду.

**CSS Flexible Box Layout Module**, коротко flexbox (Флексбокс), створений щоб прибрати недоліки різних HTML конструкцій, в тому числі адаптованих під різну ширину і висоту,

**Flexbox** дозволяє контролювати найрізноманітніші параметри елементів контейнера: напрямок, порядок, ширину, висоту, вирівнювання вздовж і поперек, розподіл вільного місця, розтягнення і стиснення елементів.

***Lesson 16. Advanced form. Simple animation***

Основу будь-якої форми становить елемент **<form> ... </ form>**. Він не передбачає введення даних, так як є контейнером, утримуючи разом всі елементи управління форми - поля.

Елемент **<input>** створює більшість полів форми. Атрибути елемента відрізняються в залежності від типу поля, для створення якого використовується цей елемент.

Основний атрибут тега <input>, що визначає вид елемента - **type**.

Написи до елементів форми створюються за допомогою елемента

**<label> ... </ label>**.

Існує два засоби поєднання напису і поля: з атрибутом ‘for’ та без нього.

Якщо поле знаходиться всередині елемента <label>, то атрибут for вказувати не потрібно.

Зовнішній вигляд елементів <input> з типом **‘checkbox’** та **‘radio’** залежить від стилів браузера. Але, все ж таки їх можна змінити

Цей процес здійснюється за допомогою скриття стандартного елемента та створення за допомогою CSS іншого «несправжнього», такого як ми хочемо

Це можна виконати завдяки тому, що в HTML переключити стан перевірки можна не тільки за допомогою самого елемента **input**, але і за допомогою пов'язаного з ним **label**.

**Це можна зробити двома засобами**

* Помістивши **input** в **label**

**<label>**

**<input type="checkbox" name="happy">Happy**

**</label>**

* Задати елементу **input** атрибут **id,** а для **label** – **for** з таким же значенн

**<input type="checkbox" id="happy** ям як у **id " name="happy">**

**<label for="happy">Happy</label>**

***Тепер необхідно “сховати” стандартний елемент***

Нам потрібно залишити функціональність цього елементу, тому ми не можемо використовувати **display:none**, бо тоді елемент перестане існувати. Тож, який засіб ми можемо обрати?

***Необхідні CSS правила***

**position:absolute - прибрати елемент з потоку**

**z-index: -1 - помістити елемент нижче за інші**

**opacity: 0 - зробити повністю прозорим**

***Створення чекбоксу-підробки***

Створення такого елементу можна реалізувати за допомогою псевдоелементу **::before**

Вміст псевдоелемента задається за допомогою CSS властивості **content**. Якщо псевдоелемент буде без вмісту, все одно необхідно вказувати цю властивість к content: "". Якщо цього не зробити - псевдоелемент відображатися не буде.

**input[type="checkbox"] + label::before {**

**content: "";**

**display: inline-block;**

**}**

Псевдоклас **:checked** застосовується до елементів інтерфейсу, таким як перемикачі (checkbox) і прапорці (radio), коли вони знаходяться у положенні «включено». Тож ми можемо змінювати стилі для різних станів: “опцію обрано” та “опцію не обрано”.

***Як же налаштувати анімацію?***

За допомогою наступних властивостей можна контролювати transition-анимацию:

* transition-property;
* transition-duration;
* transition-timing-function;
* transition-delay;

**transition-property** - вказує список властивостей, які будуть анімовані; властивості, які тут не вказані, будуть змінюватися звичайним чином. Можна анімувати всі властивості для конкретного елементу, вказавши значення all.

**transition-property: width;**

**transition-duration** - задає значення тривалості анімації, час можна вказувати в секундах або мілісекундах.

**transition-duration: 1s;**

**transition-timing-function** - часова функція, вказує точки прискорення і уповільнення за певний період часу для контролю зміни швидкості анімації. Наприклад, ми можемо прискорити анімацію на початку і уповільнити в кінці, або навпаки.

**transition-timing-function: cubic-bezier(0, 0, 1, 1);**

**transition-delay** - задає затримку часу до початку анімації, можна вказувати в секундах або мілісекундах.

**transition-delay: 500ms;**

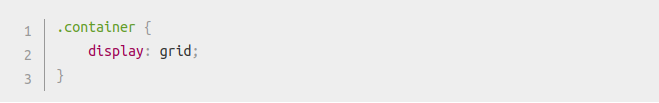
***Lesson 17. Grids***

**Grid** - це сукупність горизонтальних та вертикальних ліній, що створюють візерунок, на якому ми можемо побудувати наші елементи дизайну. Вони допомагають нам створювати конструкції, при яких елементи не перескакують і не змінюють ширину під час переходу від сторінки до сторінки, забезпечуючи більшу узгодженість на наших веб-сайтах.

**Grid**, як правило, матиме **стовпці**, **рядки**, а потім проміжки між кожним рядком і стовпцем - зазвичай їх називають жолобами.

**Як розпочати?** **display: grid**

Для визначення сітки ми використовуємо значення сітки властивості **display**. Як і у **Flexbox**, це вмикає макет сітки, і всі прямі дочірні елементи контейнера стають елементами сітки. Додайте це до CSS у вашому файлі:

**Line-based placement**

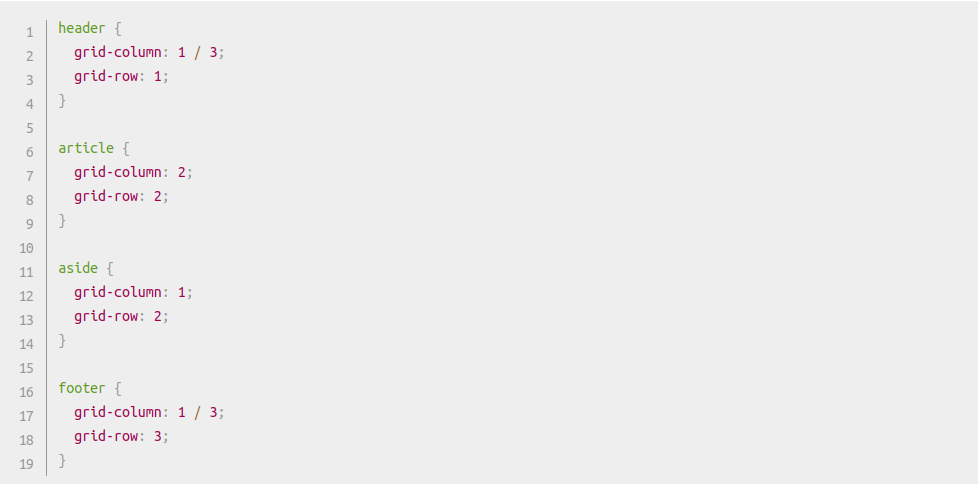
Тепер ми переходимо від створення сітки до розміщення речей у сітці. У нашій сітці завжди є рядки, які починаються з 1 і відносяться до режиму написання документа.

Тому англійською мовою рядок стовпця 1 знаходиться ліворуч від сітки, а рядок рядка 1 зверху. В арабському рядку стовпця 1 буде праворуч, оскільки арабська пишеться справа наліво.

Ми можемо розмістити речі відповідно до цих рядків, вказавши **початковий і кінцевий рядки**. Ми робимо це, використовуючи такі властивості:

* **grid-column-start**
* **grid-column-end**
* **grid-row-start**
* **grid-row-end**

Натомість ми розмістимо всі елементи нашого сайту на сітці, використовуючи лінії сітки. Додайте наступні правила внизу вашого **CSS**:

 **Функція** **minmax ()** дозволяє встановити мінімальний та максимальний розмір доріжки, наприклад **minmax** (100px, авто). Мінімальний розмір - 100 пікселів, але максимальний - автоматичний, який розшириться відповідно до вмісту.

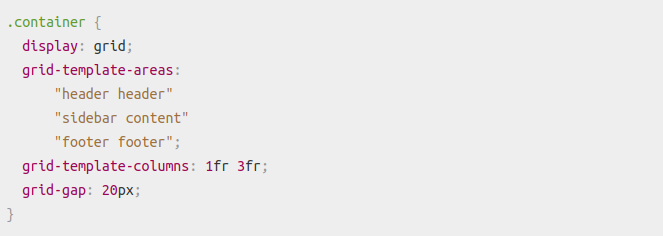


*Якщо ви додасте додатковий вміст, ви побачите, що доріжка розширюється, щоб дозволити їй відповідати. Зверніть увагу, що розширення відбувається прямо вздовж рядка.*

**grid-template-areas**

Альтернативний спосіб розміщення елементів у вашій сітці - використання властивості grid-template-areas та надання імені різним елементам вашого дизайну.

Видаліть позиціонування на основі рядків з останнього прикладу (або повторно завантажте файл, щоб отримати нову початкову точку), і додайте наступний CSS.



**implicit and explicit grid**

Наразі ми вказали лише доріжки стовпців, і все ж створюються рядки для зберігання нашого вмісту. Це приклад явної та неявної сітки. Явна сітка - це та, яку ви створюєте, використовуючи **grid-template-columns** або **grid-template-rows**. Явна сітка створюється, коли вміст розміщується за межами цієї сітки - наприклад, у наших рядках.

Якщо ви хочете надати неявні розміри доріжок сітки, ви можете використовувати властивості **grid-auto-row** та **grid-auto-columns**.

**implicit and explicit grid**

За замовчуванням доріжки, створені в неявній сітці, мають автоматичний розмір, що загалом означає, що вони достатньо великі, щоб відповідати їх вмісту. Якщо ви додасте **grid-auto-rows** зі значенням *100px* до вашого **CSS**, ви побачите, що ці створені рядки тепер мають 1*00 пікселів* заввишки.



**<div** class="container"**>**

**<div>**One**</div>**

**<div>**Two**</div>**

**<div>**Three**</div>**

**<div>**Four**</div>**

**<div>**Five**</div>**

**<div>**Six**</div>**

**<div>**Seven**</div>**

**</div>**

body {

  width: 90%;

  max-width: 900px;

  margin: 2em auto;

  font: .9em/1.2 Arial, Helvetica, sans-serif;

}

.container > div {

  border-radius: 5px;

  padding: 10px;

  background-color: rgb(207,232,220);

  border: 2px solid rgb(79,185,227);

}                  .container {

  display: grid;

  grid-template-columns: repeat(3, 1fr);

  grid-auto-rows: 100px;

  grid-gap: 20px;

}

**Grid generator**

Модуль **CSS Grid** являє собою простий і в той же час потужний інструмент для створення сайту.

У модуля також з'явилася вбудована підтримка популярних браузерів (**Safari, Chrome, Firefox, Edge**), тому будь-який веб-розробник в будь-якому випадку зіткнеться з цією технологій:

***Додаємо трохи тіней***

Властивість **box-shadow** додає тінь до елементу. Допускається використовувати кілька тіней, вказуючи їх параметри через кому, при такому записі тіней, вони будуть накладатися одна на одну, перша тінь зі списку буде зверху, остання в самому низу. Ти можеш встановити колір тіні, а також задати її зміщення (як далеко вона буде від елементу)

***Переходимо до сайту та трохи пограємось з тінями***

***https://www.cssmatic.com/box-shadow***