**HTML\_CSS**

<!DOCTYPE html> // версія html 5

<html lang="en"> // Мова яка буде використовуватись на сторінці

<head> // шапка сайта котра використовується для налаштувань сторінки (не видима)

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge"> // кодування сторінки, застарілий тег

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title> // назва сторінки

<link rel="icon" href="icon.png"> - іконка для сторінки

</head>

<body> // тіло документа (інформація котру видно на сторінці)

    <script src="index.js"></script>

</body>

</html>

**Веб сторінка - программа написана на html яку можна запустити в браузері**

Сайт - декілька веб сторінок розміщених на сервері

Сервер - компютер на якому запущена программа для отримання запросів з інтернету

тег **<meta charset="UTF-8">** - встановлює кодування символів (вся інформація в форматі нулів і одиничок)

Підсказування тегів - еммет абревіатура

**Теги:**

* **Інлайнові -** займають рівно стільки місця скільки потрібно
* Блочні - займають цілу строку

**Парні теги:**

* **<div></div>** - групування елементів на сторінці, поділ сторінки на блоки блочний
* **<span></span>** - інлайновий елемент, область тількиина ширину і висоту контенту
* **<p></p>** - параграф, обгортання тексту
* **<ul></ul>** -unordered list ненумерований список повинен містити теги li
* **<ol></ol>** - нумерований список
* **<li></li>** - list item, елемент списку
* **<b></b>** - жирний текст
* **<strong></strong>** - жирний текст (підчас голосового прочитаття сайту наголос буде на інформацію розміщену в тегові)
* **<em></em>** - курсив
* **<pre></pre>** - в точності відображає зміст, з пробілами і відступами
* **<a href=""></a>** - ссилка - атрибут href адреса за якою ми пререйдемо на іншу сторінку атрибут **target="\_blank"** - при нажатті на ссилку відкриває її в новому вікні

**Одинарні теги:**

* <br> - перенос строки
* <hr> - горизонтальна лінія
* <input> - вікно для введення інформації від користувача
* <img src=""> - вставка зображення, атрибут **src** описує місце знаходження файлу на компютері або з інтернет ресурсу, alt показує текст опис зображення якшо картинка не загрузилась

**Атрибути(або властивості тега)** - надають тегові шлях для відображення інформації. Знаходяться в відкриваючому тегові відділені між собою пробілом, значення атрибути розміщується в кавичках після = в відкриваючому тегові.

**<img src=' lesson1/nature.jpg'. Якщо файл розміщено в папці вище ніж файл, ми можемо дістатись до нього за допомогою .. /**

**Xml - формат розміщення інформації, відкриваючий тег інформація закриваючий тег**

**Link: (атрибути)**

* **target (якшо атрибуту присвоїти значення \_blank ссилка відкриється в сусідньому вікні)**
* **href (=ссилка сторінку або файл)**

**<Img>:атрибути**

* **src=" " -** джерело інформації
* width =" " - ширина
* heigth=" " - висота
* alt = " " - альтернативний текст в випадку якшо по якимось причинам фото не показує

**<Input>: атрибути**

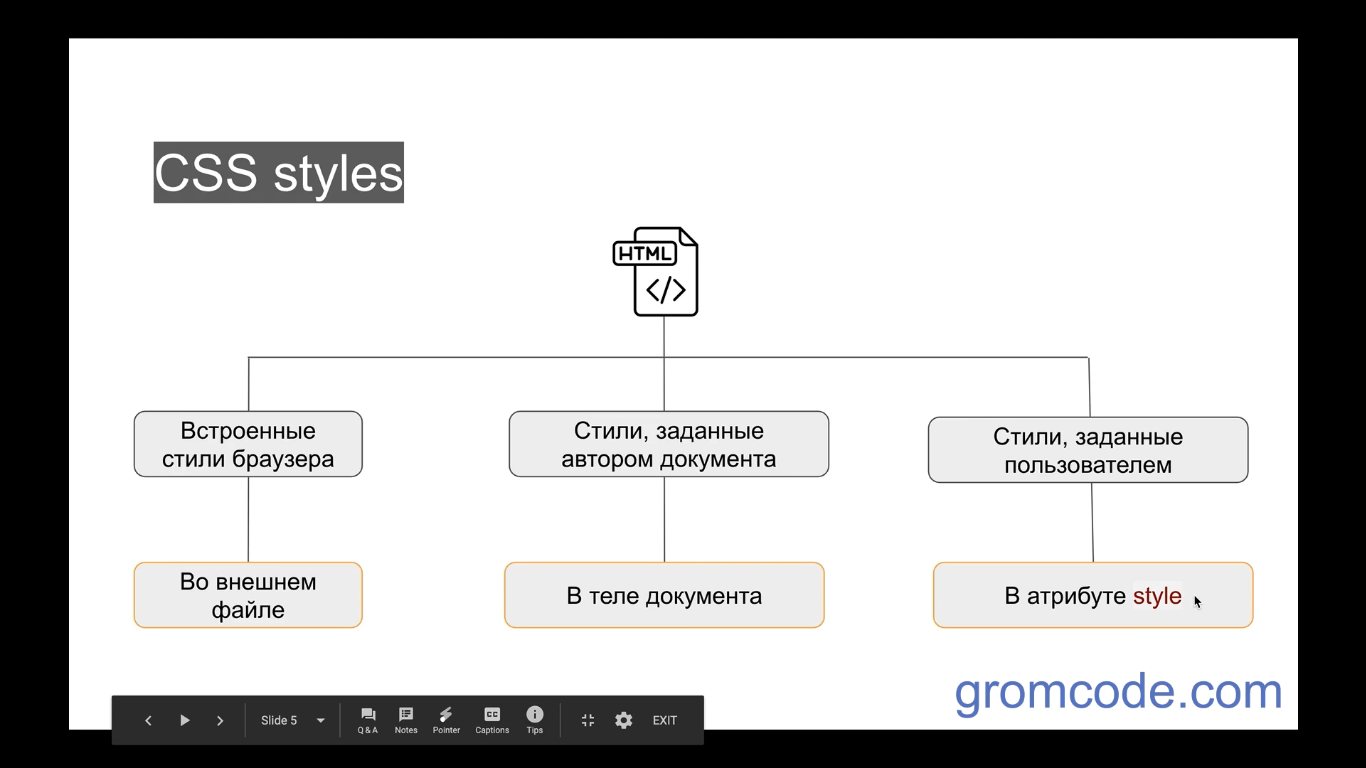
* **type**=" " - тип
* **"text"**
* **"checkbox"**
* **"password"**
* value= " " - значення

**Семантика - положення тексту на сторінці. В кожного тегу є своє призначення.**

**Семантичні теги:**

* **header** - заголовок, логотип
* **nav** - навігаційна панель
* **main** - основний контент
* **article** — незалежна або самостійна контентна композиція багаторазового використання, може бути, наприклад, постом на форумі, записом в блозі, газетній статті, відкликанням користувача або окремим віджетом.
* **footer** - підвал (додаткові матеріали, ссилки)
* **aside** - можна використовувати як виносну цитату, бічні панелі для реклами фільтрів і сортувань товарів або організації навігації
* **details** — меню, що випадає, або додаткові деталі, які користувач може приховати або показати.
* **summary** - використовується як видимий заголовок елемента details
* **figure and figcaption** - використовується для підпису картинок, діаграм, таблиць
* **mark** - використовується для виділення важливого тексту

**CSS - Стилі**

****

**CSS - cascade style sheet (каскадні таблиці стилів)**

**Способи задавання стилів:**

1. В атрибутах **<p style="color:green"></p> - інлайнові стилі**
2. В шапці сторінки(<head>) в тег **<style> div { color: red} </style>**
3. В окремий документ CSS **(styles.css > div {color: red})** в самому документі прописати посилання на файл: **<link rel=" stylesheet" href=" styles.css">**

Перед тим як прописувати стилі в css можна подивитись як виглядатиме ця сторінка в **developer tool.**

Перед задаванням своїх стилів потрібно нормалізувати .css за допомогою**:**

**https://necolas.github.io/normalize.css/**

**<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/normalize.min.css">** або

**або обнулити**

**https://meyerweb.com/eric/tools/css/reset/**

при задаванні стилів для всіх елементів \* {

* **all: initial** -як властивість задана елементу, обнуляє всі стилі
* **inherit** - Ключевое слово inherit сообщает браузеру, что значение свойства нужно найти у ближайшего родительского элемента и унаследовать его для текущего элемента. Если у ближайшего родителя также задано значение inherit, то браузер продолжит подниматься по DOM, пока не найдет какое-нибудь значение
* **all:unset** - якшо властивість наслідувана то **unset** працює як **inherit**, якщо не наслідувана то **unset** працюватиме як **initial**
* **display:revert** - значення не за замовчуванням для властивості, а за замовчуванням для браузера

**Стилі:**

* **width** - ширина елемента
* **heigth** - висота елемента
* **display** - визначає як елемент буде показаний в тексті
* **color** - колір тексту (#845) -hexadesimal
* **background-color** - колір фону
* **font-size** - розмір шрифту
* **margine** - зовнішні відступи блока
* **border** - рамка
* **padding** - внутрішній відступ
* **line height** - висота строки
* lorem20 - команда збільшення розміру і кількості тексту з 20 слів

**Вирівнювання тексту**

Вирівнювання тексту по вертикалі: зробити висоту строки **line-height** такою ж як висота **div.**

Вирівняти текст по горизонталі вправо, центр, ліво допоможе функція **text-align: center**

**Способи задавання кольорів в CSS:**

* **rgb (red, green,blue)** - колір в залежності від комбінації червоного зеленого синього задається в відсотках (255%, 40%, 20%)
* **#000000** - скорочення коду rgb де на кожен колір виділяється 2 цифри - 16 рівнева система зчислення від 0 до ff=16, #ff0000 = #f00 - скорочення коду
* **background-color: black** - через назву кольору
* rgba (50, 70,25,**0-1)** - остання цифра вказує на прозорість кольору

**Шрифти:**

* в css задається декілька шрифтів або сімейство, на випадок якшо якийсь компютер не підтримуватиме вибраний шрифт
* @ font-face - для кожного шрифта прописується директива (авторські шрифти) і ссилка на файл
* @font-face {
* font-family: Roboto;
* src: url(fonts/Roboto-regular.ttf);
* }
* @font-face {
* font-family: Roboto;
* font-weight: bold;;
* src: url(fonts/Roboto-bold.ttf);
* }
* @font-face {
* font-family: Roboto;
* font-weight: bold;
* font-style: italic;
* src: url(fonts/Roboto-italic.ttf);
* }
* body {
* font-family: Roboto,Arial, Helvetica, sans-serif
* }
* шрифти з інтернету (google fonts) котрі ми підключаємо шапку head за допомогою тегу

<link rel="preconnect" href="https://fonts.googleapis.com">  
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>  
<link href="https://fonts.googleapis.com/css2?family=**Montserrat:wght@500**&display=swap" rel="stylesheet">

* **пропорції на сторінці** - при планування розмірів елементів величини можна подавати не в **px,** а в **em** - відношення до батьківського елементу. При подаванні величин слід памятати що **em** означає відношення до величини батьківського елементу тільки для **font size**, а для інших властивостей таких як **margine**, **em** це розмір шрифту цього ж елементу.
* **rem** - root em - корінний елемент на сторінці (html). Тому шрифти краще передавати в стилях в елементі html ---- **рідко використовуються**
* html {
* font-family: Arial, Helvetica, sans-serif;
* font-size: 12px;
* }
* .title {
* font-size: 2rem;
* margin: 0 0.5rem; }

**Розміри елементів:**

Кожен елемент на сторінці або інлайновий або блочний:

**Інлайновий** - займає місця рівно на розмір контенту, висота і ширина не задається

**Блочний** - займає цілу строку, тому ми не можемо розмістити блоки в рядок

Функція **inline**-**block** -- дозволяє розмістити блочні елементи в 1 строку якщо вони поміщаються по ширині, але зручніше буде присвоїти контейнеру властивість display: flex

**Способи** **задавання** **розмірів**:

* в % - від батьківського елементу для margin від ширини
* в px - пікселях
* в vh i vw - viewport height i viewportwidth - видима частина висоти і ширини
* min-width i max-width - задати мінімальну та максимальну вистоту
* heigth calc(100vh -.....) - вирахувати за допомогою формули
* custom properties: body { --header-height: 50px}

header{ heigth:var(--header-heigth)} - але працює такий спосіб тільки тоді коли в елементів спільний батьківський елемент (body або html)

**Правило**: для зовнішнього блока контейнера ми задаємо padding, а шоб задати відстань від одного елементу до іншого всередині контейнера ми використовуємо margine по одній зі сторін

**Правило**: за замовчування батьківський елемент розтягується на мінімально необхідну висоту для того щоб вмістити зміст внутрішнього елементу

Для того щоб керувати розмірами внутрішніх елементів контейнера ширину та висоту можна писати в %, але працює вона тільки тоді коли задано батьківському елементу задані параметри висоти та ширини

**Margin** не впливає на розмір елементу і для інлайнових елементів працює тільки по ширині.

**Line** **height** - для інлайнових елементів розтягує висоту строки але не сам елемент

Вертикальні падінги для інлайнових елементів не розсуває сусідні строки але розсувають по ширині.

Якщо поставити 2 інлайнових елементи поряд (в html) то між ними буде пусте місце, бо перенос строки і 4 пробіли відображається в html як 1 пробіл.

Всередині інлайнового елементу не може знаходитись блочний

**vertical**-**align**: **top** - функція котра вирівнює текст по верху

**Фон елементу розповсюджується на його content box, на pading і border але не на margin**

**Випадіння** **margin** - це коли верхній margin верхнього елементу або нижній margin нижнього елементу випадає за мері батьківського елементу, щоб цього уникнути треба контейнеру задати рамку, або мінімальні padding.

**Margin** **collapse** -2 вертикальних margin сусідніх елементів зверху зустрілись і наклались 1 на одного, в результаті відступ буде дорівнювати не їх сумі а максимальному з них, щоб цього уникнути треба між елементами в html вставити **div** з мінімальною висотою, або **padding** або **border**

Приклеїти **footer** до низу сторінки за допомогою **min-height: calc (100vh - height of header - height of footer)** але якщо треба буде змінити висоту хедера або футера то вся конструкція зломається, тому краще використовувати кастомні властивості (css змінні), але вони повинні бути прописані в батьківському елементі обох контейнерів:

body {

  --header-height: 50px;

  --footer-height: 20px;

}

header {

  height: var(--header-height)

}

 main {

   min-height: calc(100vh - var(--header-height)- var(--footer-height))

 }

 footer {

   height: var(--footer-height);

 }

**Селектори**

**Базові селектори:**

* **\*** - універсальний селектор (стилі використовуються для всіх тегів) - використовується для

\* {

  all:unset;

  display: revert;

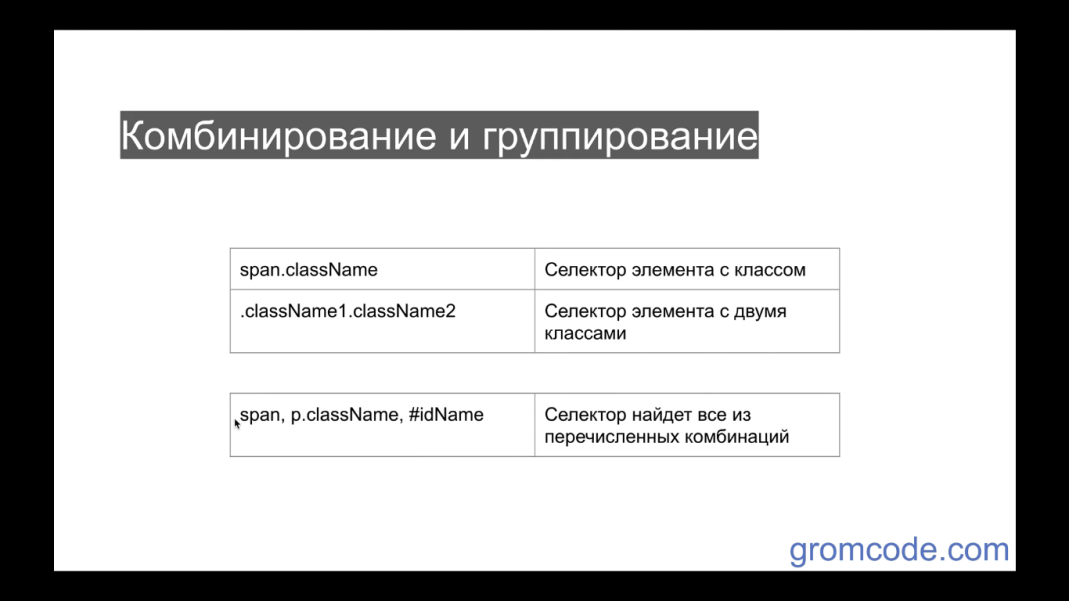
  box-sizing: border-box;

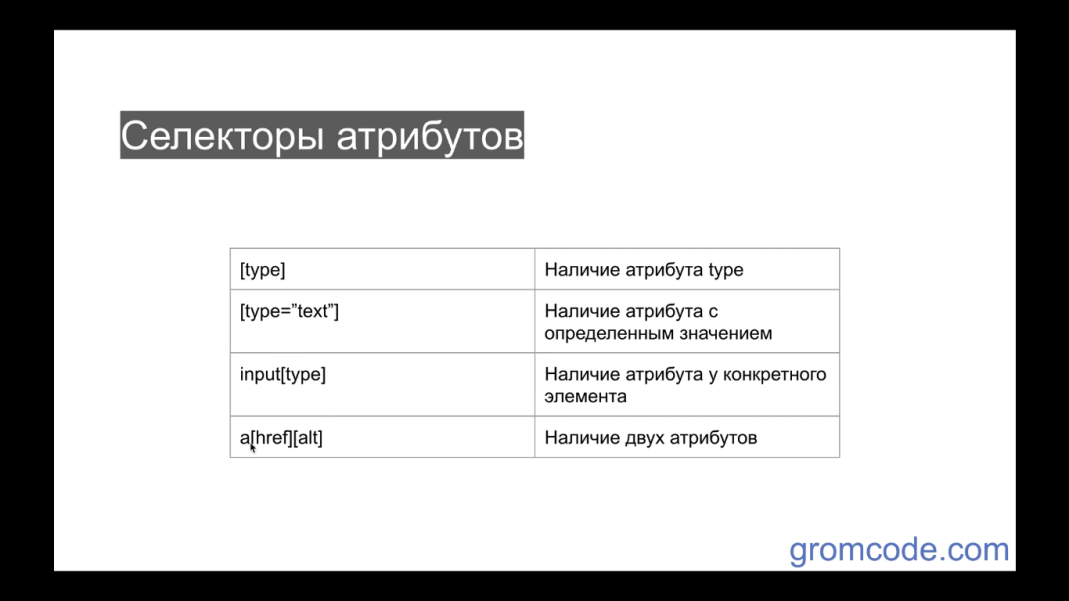
}

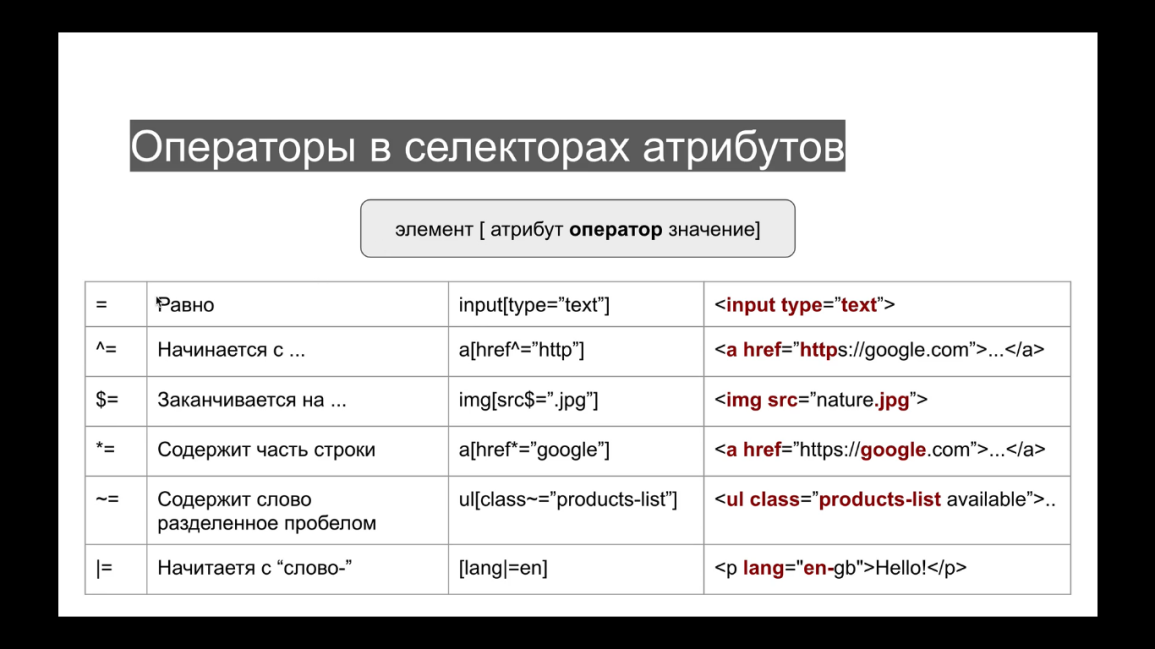
* **div**- селектор елемента (стилі для тегів div і їх дочірнім елементам)
* **#myId** - селектор ідентифікатора (повинен бути 1 на сторінці і повинен бути унікальним <li id="myId"></li>
* елемент форми звязується зі своїм підписом
* навігація по сторінці з використанням якорів
* використовується в java script для задання поведінки
* **.className** - селектор класу (для того щоб різним тегам присвоїти ті самі стилі елементам присвоюють клас <h1 class="page-header"> </h1>
* **.hightlighted .important** (класи перечислені через пробіл застосовуються для потомків) (будуть застосовуватись тільки для тих елементів з класом important, які знаходяться в елементі з класом hightlighted)
* **.hightlighted, .important -(через кому)** стилі застосовуються для обох класів
* **.hightlighted >.important -** для потомків першого рівня
* **.hightlighted + .important -** стилі застосовуються тільки для наступного елементу після hightlighted з класом important
* **.hightlighted ~ .important -** для всіх **important** після hightlighted на тому ж рівні
* **span.user-data** (якщо 2 елементи містять однакові атрибути, то пишемо для якого елементу і якого атрибуту через крапку)
* .**company-data, li** (кома відіграє роль "АБО" стилі будуть застосовані для елементів з класом company-data або елементів списку
* **.hightlighted.important -** класи перечисленні без пробілів тільки для елентів в яким присвоєно 2 класи
* **.hightlighted > \* -** для всіх потомків 1 рівня контейнера з класом **.hightlighted**
* **.hightlighted \* -** всі елементи всередині контейнера
* **.hightlighted + \* - будь який перший елемент котрий стоїть одразу після контейнера**
* **.hightlighted ~ \* - всі елементи котрі стоять після контейнера**
* **.hightlighted >\*>\*внуки секції**
* **.hightlighted >\*>div.block - внуки hightlighted з div класом block**

**Селектори атрибутів:**

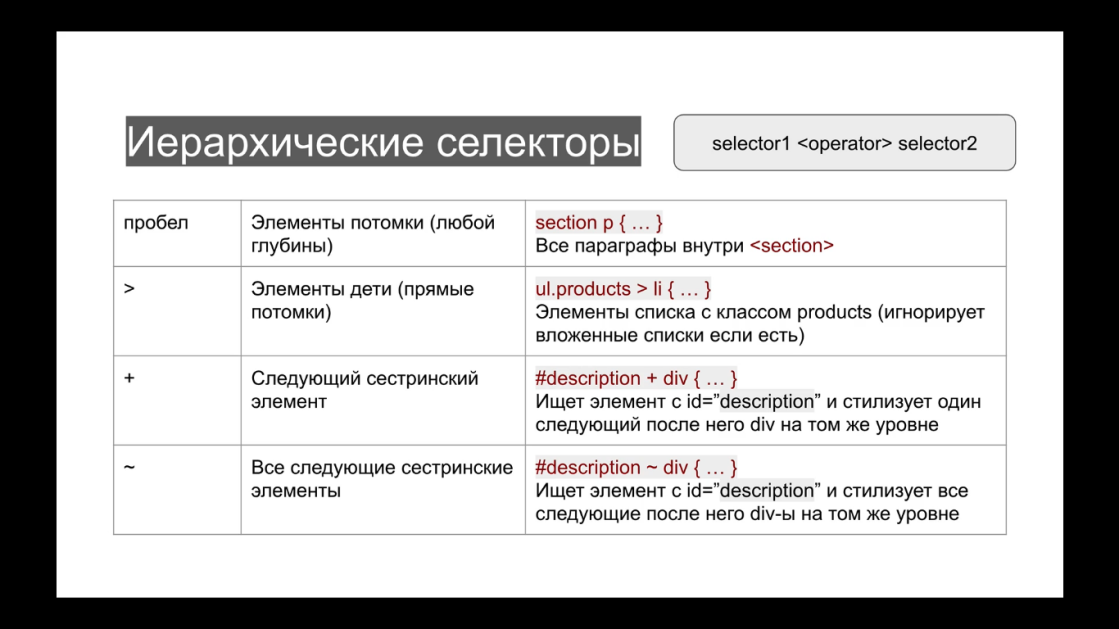
* [ назва атрибуту] {стилі }
* [ ] [ ] { } - для елементів які мають 2 атрибути
* [ data-description="archors"] { } - для елементів атрибути яких = archors





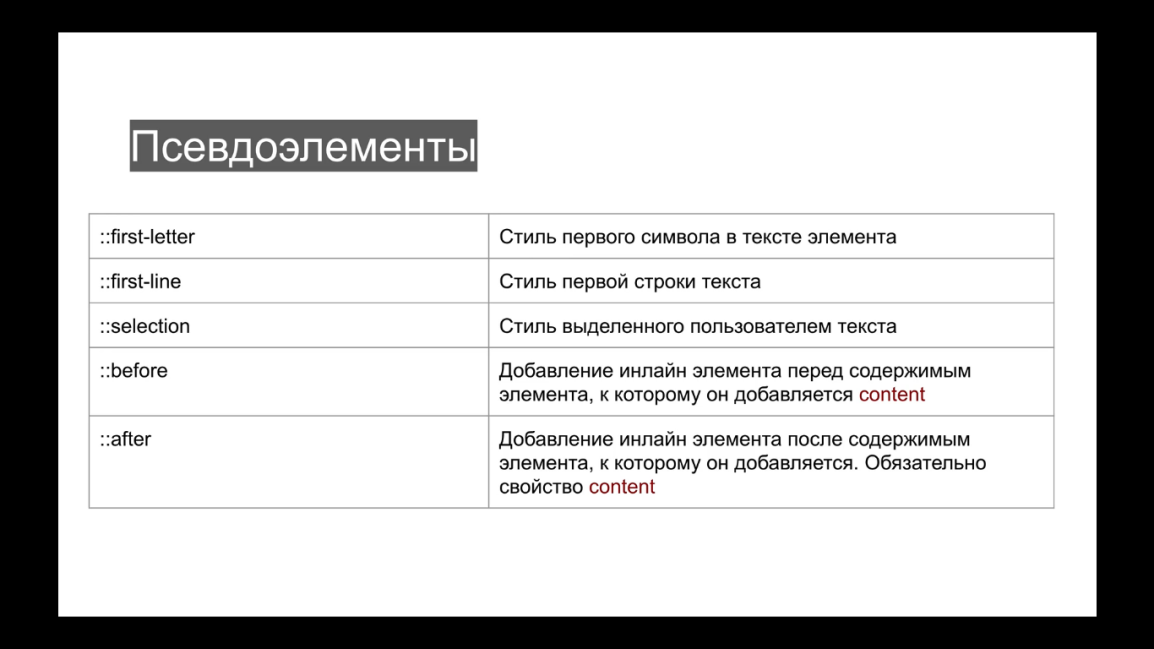


Стилі завжди застосовуються до елемента який знаходиться найправіше в селекторі



**Псевдоклас** - відповідний стан елементу, оснований на користувацьких діях.

**Псевдоелемент** - слово котре додається до селектора і допомагає стилізувати вибрану частину елементу



**Псевдоелементи:**

* :: first-letter - перша літера елементу
* :: first-line - перша строка елементу
* :: selection - виділений текст
* :: before {content: '<'} - вставка елемента перед текстом
* :: after {content: '>'} - вставка елемента в кінці тексту



**:nth-child(n)-**всі елементи діти

**:nth-child(n +3)** - всі елементи починаючи з 3

**:nth-child(2n)** - всі парні числа

**:nth-child(2n+3)** - всі непарні починаючи з 3

**.container h2:first-of-type** - h2 котрий всередині свого батька є першим по рахунку елементом данного типу (тип мається на увазі назва тегу)

**.container p:not (.info)** - знайде всі параграфи в котрих немає класу info

**.container:not(p).info** - всі елементи котрі не є параграфами але мають клас info

**.container::before** { - додасть слово before перед conteiner без змісту

content: 'Before'

background color: red }

**.container::after - додасть елемент після container**

.container::selection { - підчас виділення тексту фон буде red а текст yellow

background color: red;

color: yellow; }

В html пишемо: ul>li\*10{$$$} і отримуємо

<ul>

  <li>001</li>

  <li>002</li>

  <li>003</li>

  <li>004</li>

  <li>005</li>

  <li>006</li>

  <li>007</li>

  <li>008</li>

  <li>009</li>

  <li>010</li>

</ul>

**Ссилки**

**../abc** - папка вище

Використання якорів для швидкого переміщення по сайту:

  <body id="page-top">

    <nav>

      <a href="#section-1">Section1</a>

      <a href="#section-2">Section2</a>

      <a href="#section-3">Section3</a>

      <a href="#section-4">Section4</a>

    </nav>

    <section id="section-1">

      <h2>Section 1</h2>

      <a href="#page-top">Go to top</a>

    </section>

    <section id="section-2">

      <h2>Section 2</h2>

      <a href="#page-top">Go to top</a>

    </section>

    <section id="section-3">

      <h2>Section-3</h2>

      <a href="#page-top">Go to top</a>

    </section>

    <section id="section-4">

      <h2>Section-4</h2>

      <a href="#page-top">Go to top</a>

    </section>

  </body>

При нажиманні на section сторінка прокрутиться на елемент при нажиманні на ссилку go to top поверне на початок сторінки

В html пишемо **scroll-behavior: smooth** - плавна прокрутка сторінки

**Стилізація ссилки:**

section:target (вибрана) - { background color: yellowgreen} - при нажиманні ссилки колір секції зміниться

Порядок в якому треба стилізувати ссилки: "LoVe HAte"

**Стан ссилки:**

* **(Lo) link** - нормальний стан
* **(Ve) visited** - відвідана ссилка, після нажаття
* **(H) hover** - наведення курсору
* **(Ate) active** - зажата кнопка мишки на ссилці

**Стан кнопки:**

* active: - зажата клавіша миші
* disabled: - виключений стан
* hover: - наведений курсор мишки
* enabled: - включена кнопка

**Стан input:**

* :focus - клікнувши інпут
* :checked - поставивши галочку

**Специфічність**

Специфічність допомагає браузеру зрозуміти яке зі значень потрібно використовувати якщо в 2 і більше правилах для одної і тієї ж властивості задаємо різні значення. Буде примінено значення, з того правила специфічність селектора котрого найвища. Якщо специфічність однакова, то буде примінено то правило яке стоїть останнє

* \*= 0
* назва елемента - 1
* назва класу, псевдо елемент, псевдо клас, атрибут - 2
* #id - 3
* :not() - не додає специфічності
* стилі котрі ми застосовуємо інлайново всередині тегу мають більшу специфічність ніж стилі прописані в css і в head
* можливість перебити всі правила можна за допомогою !important - але його краще ніколи не використовувати при власному написанні коду - можуть звільнити з роботи)))
* Стилізувати елемент в якомусь стані краще після його стилізації в звичайному стані
* не варто ніколи використовувати id для стилізації краще класи

**Box Model**

* **width,heigth** - розмір контенту
* **padding** - внутрішні відступи
* **border** - рамка
* **margine** - зовнішній відступ
* **border-radius** - заокруглення країв
* **box-sizing: border box**- розмір елементу включаючи рамку, внутрішню відступи і контент

margine : 20px відступ застосовується до 4 сторін

margine : 20px 50px відступи зверху і знизу 20 справа і зліва 50

margine-left : 20px відступ тільки зліва

Border : 10px solid red

Border - width: 10px

Border-style: solid

Border-color: red

Margin collapse - накладування відступів один на одного

margin - мінімально гарантована відстань до перешкоди

Форма блочного елемента задається шириною і висотою:

**Круг:** square

* Width: 50px
* Heigth: 50px
* Border - radius: 50% - заокруглення країв

**Овал:** oval

* width: 100px
* heigth: 50px
* border radius: 50%

**Прямокутник з заокругленними краями:**  rectangle

* width: 100px
* heigth: 50px
* border radius: 10px

**Трикутник:** triangle

* width: 50px
* heigth: 50px
* margine: 50px
* box-sizing: border-box
* border: 25px solid transparent (прозорий)
* border-top-color: red

**Властивості рамки (лінії довкола):**

* Solid - суцільна
* dashed - штрихпунктирна
* dotted - крапкова

**Центрування блочних елементів**

* по горизонталі - встановити горизонтальні зовнішні відступи **margin: 0, auto**

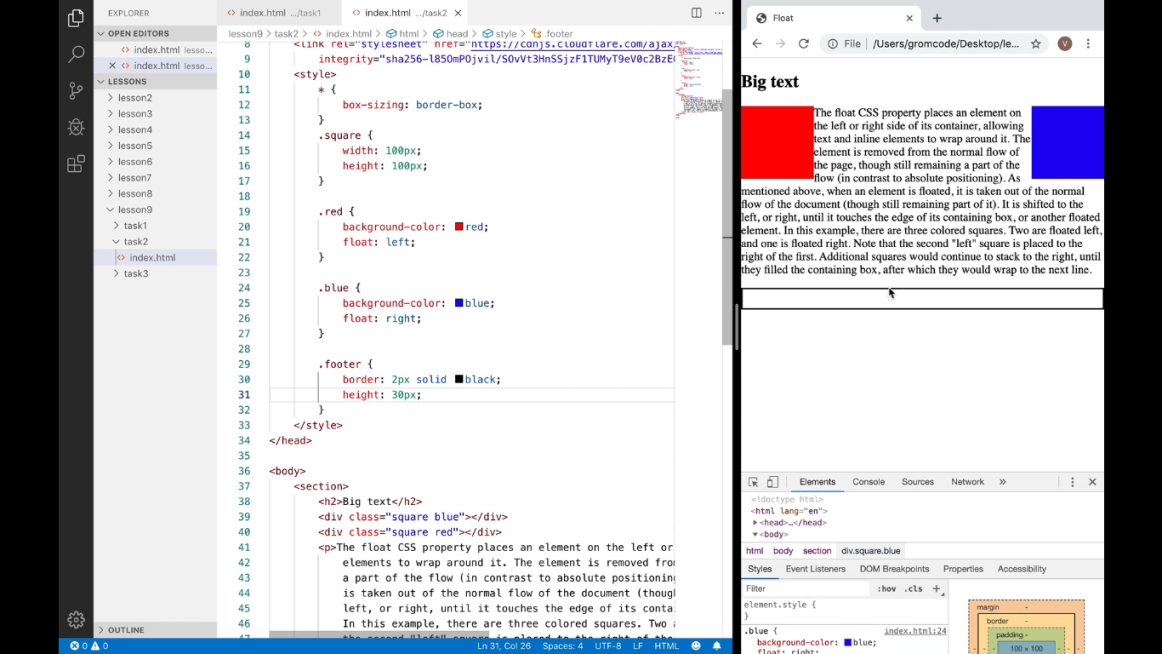
**Positioning element**

Правило креслення - всі елементи на сторінці розміщуються зліва на право і зверху вниз.

Position:

* static - властивість за замовчуванням
* absolute - розмір на ширину контенту. Позиція відраховується відносно найближчого елементу relative, якщо такого не має то відносно body (top,left,rigth,bottom)
* relative - позиція відносно батьківського елементу (top, bottom,left,right)
* fixed - фіксована позиція котра не міняється навіть при прогортуванні контенту

float - властивість котра дозволяє обтікання контенту, наприклад стаття з фото.



При використанні **float** властивості трапляється явище схлопування елементів, для того щоб такого не траплялось потрібно чистити потік властивістю **clear:both**

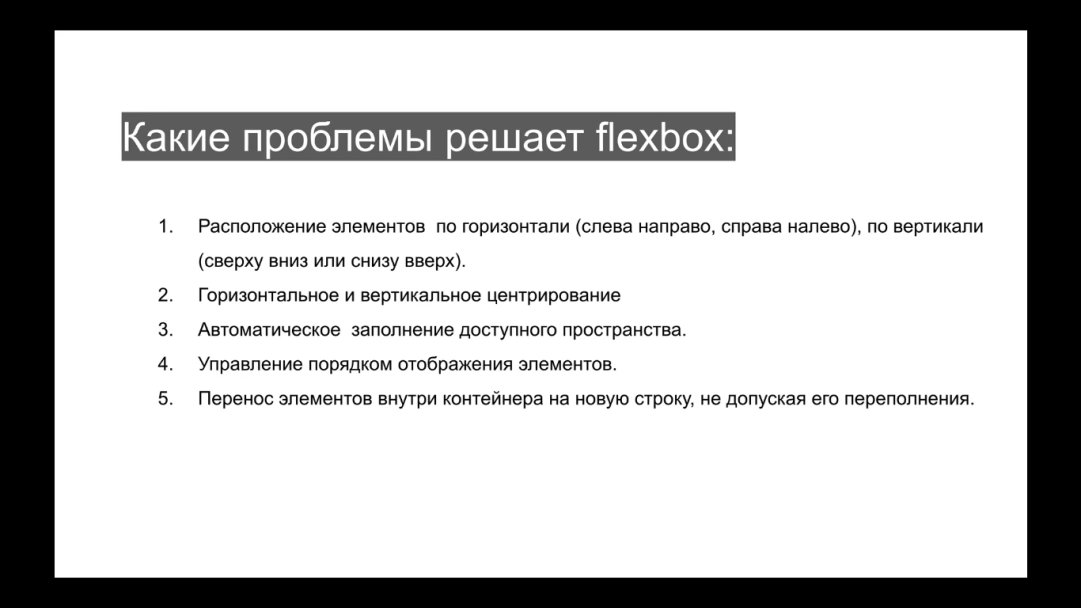
Також **float** можна використовувати для розміщення блочних елементів в 1 строку

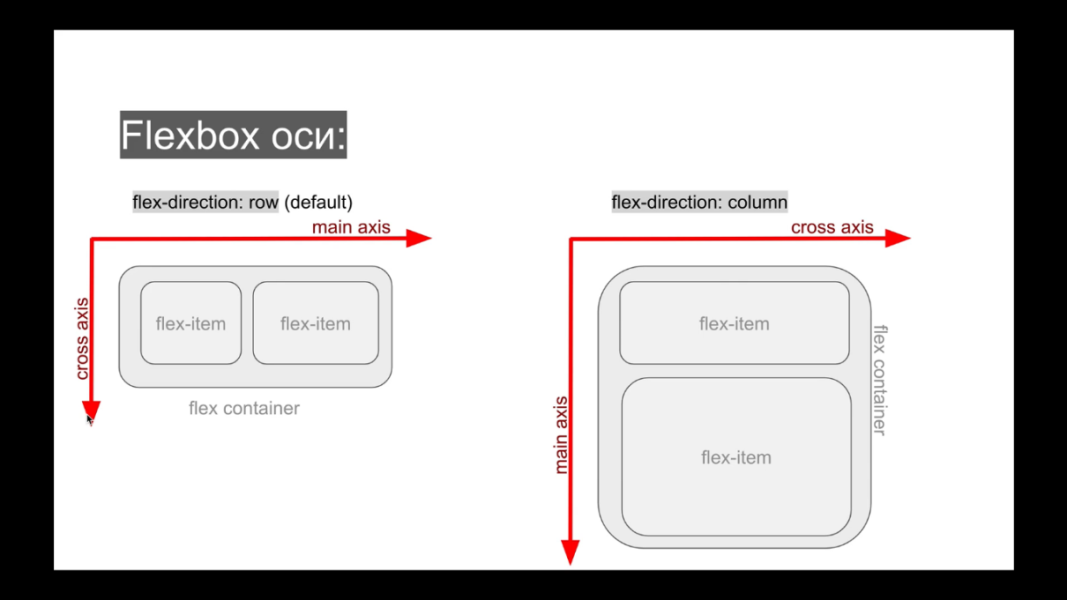
Якщо 2 елемента розміщується в 1 точці то видимий буде той котрий в розмітці нижче, або можна надати потрібному елементові **z-index:** ... щоб висунути його наперед.

**line-heigth** - регулювання положення строки по висоті

**list-style-type:none** - забирає маркери в list item

**FlexBox**





**display:flex** - відкриває властивості гнучкої коробки

* **align-items: center** - вирівнювання тексту по вторинній осі (вертикальній)
* **justify-content:**
* **center** - вирівнювання коробки по головній осі (горизонтальній)
* **space-around** - рівномірне розміщення по площині
* **space-between** - рівномірні відступи між елементами, крайні ел. прижаті до країв
* **flex-direction** - змінює напрямок головної осі
* **column** - вертикальний
* row - горизонтальний
* **flex-shrink** - описує як буде зжиматись елемент 0-1
* **flex-grow** - описує як буде рости 0-...
* **flex-wrap** (для контейнерів) - регулює положення вторинній осі
* **nowrap** - не переносити на іншу строку
* **wrap** - перенести на іншу строку
* **heigth: 100vh** - 100% view port heigth (100 одиниць видимої частини екрану)



**Display Visibility**

Елементи на сторінці можна сховати, їх не буде видно, але вони там будуть:

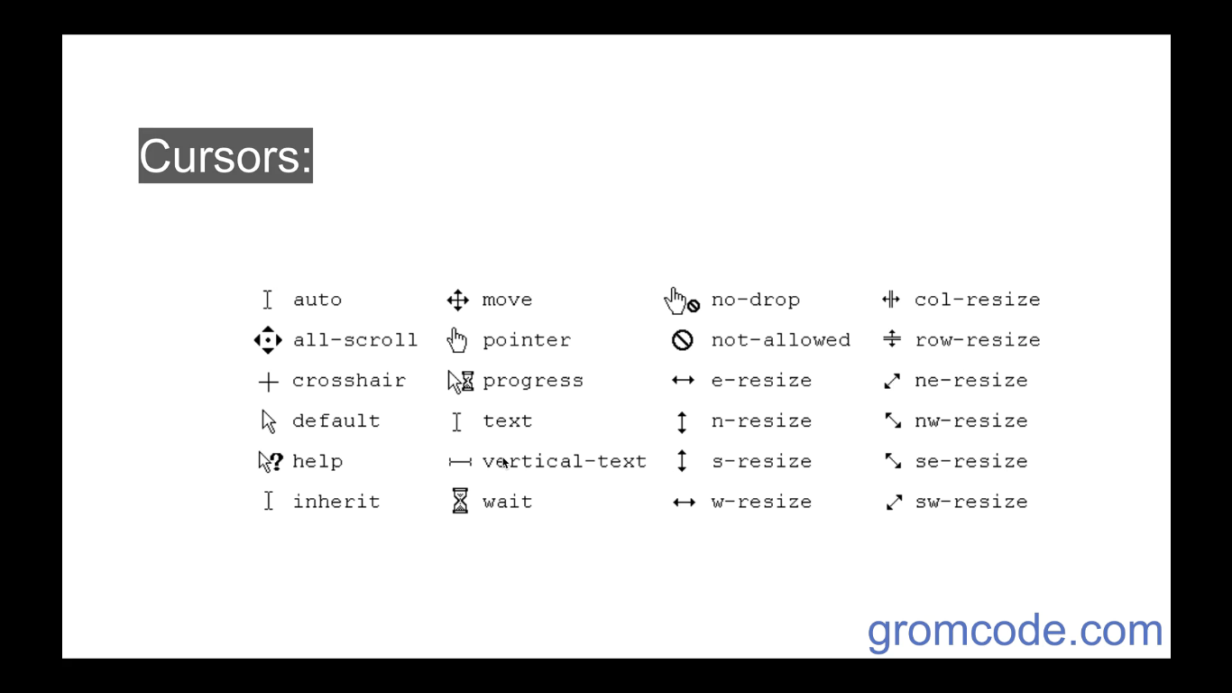
* **display:none**
* **visibility: hidden**

Якшо величина контейнера менша за розміщений в нього контент то це переповнення , властивість overflow описує що буде з контентом котрий не вміщається

**Overflow:**

* **hidden** - сховає контент який виходить за рамки контейнера
* **scroll** - наводячи курсор на контейнер доступний скрол котрий дозволяє скролити контент
* **overflow-x або y - auto** - якшо вміст контенту перевищує по осі -у появиться вертикальний скрол, якшо по осі -х то горизонтальний скрол

Cursor: при наведенні курсора на елемент курсор мінятиметься



**Styling text**

Ресурс шрифтів - google fonts

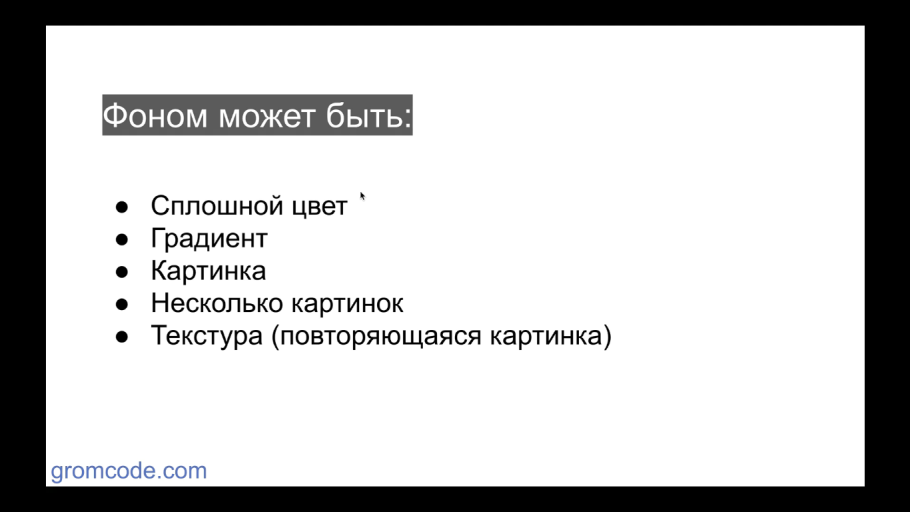
Копіюємо тег лінк і вставляємо в шапку профілю

В стилях прописуємо шрифти для цілого документу

* **html { font-family: назва загруженого шрифту, geneva, tahoma}**
* **word-spacing:5px - відстань між словами**
* **letter-spacing: 2px - відстань між буквами**
* **text-transform: uppercase - всі слова з великої літери**
* **line-heigth - висота лінії, вирівнювання тексту по вертикалі, щоб вирівняти текст по вертикалі line-heigth повинно бути таке як висота блока**
* **font-size: 2em - розмір шрифту в 2 рази більший ніж батьківського елементу**
* **font-weight - насиченість шрифту**
* **text-align: center - розміщення тексту по центрі**
* **white-space: nowrap - не переносити текст на наступну строку**
* **text-overflow: elipsis - прихований текст ...**
* **<b></b> - тег жирного тексту**
* **<em></em> - тег похиленого тексту italic**
* **<a href=" \_"></a> - тег ссилки**
* **text-decoration:underline - нижнє підкреслювання**
* **list-style-type:none - забрати маркери списків**
* **text-decoration-style: wavy - нижню підкреслювання в вигляді хвильки**
* ****
* **<pre>\_</pre> - всередині тегу pre текст відображається так як в html використовується найчастіше для кусочків коду всередині документу**
* **для вставки html коду в документ використовують онлайн сервіси для екранування тексту**

**Фон елементу**

****

****

**background-image: url (nature.jpg)** - картинка як фон

**background-position: center** - якшо картинка велика то ми можемо регулювати яку частину картинки показувати

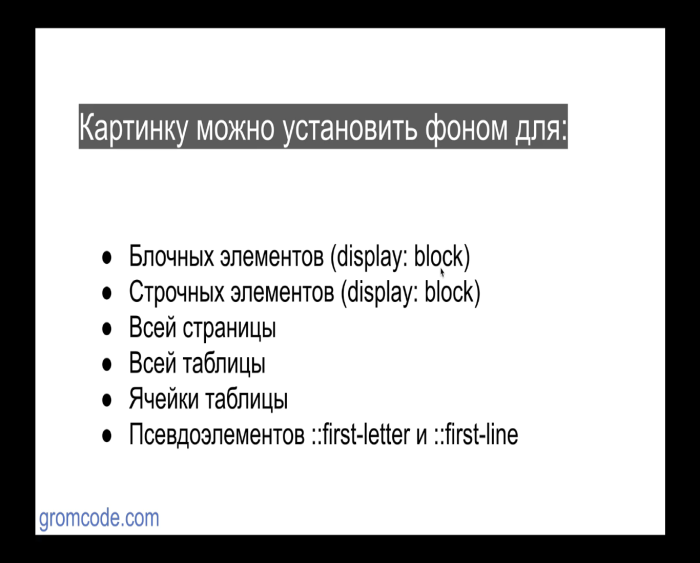
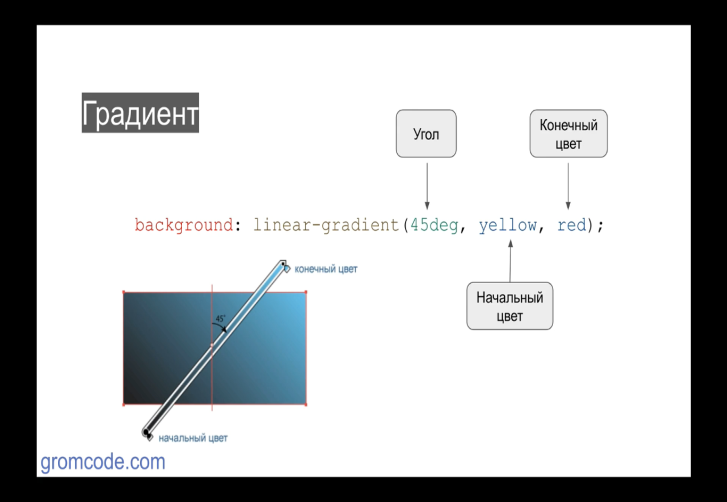
**background-size: cover** - по центру на всю висоту

**background-contain: contain** - картинка вміщається цілком в заданий маштаб і може повторюватись

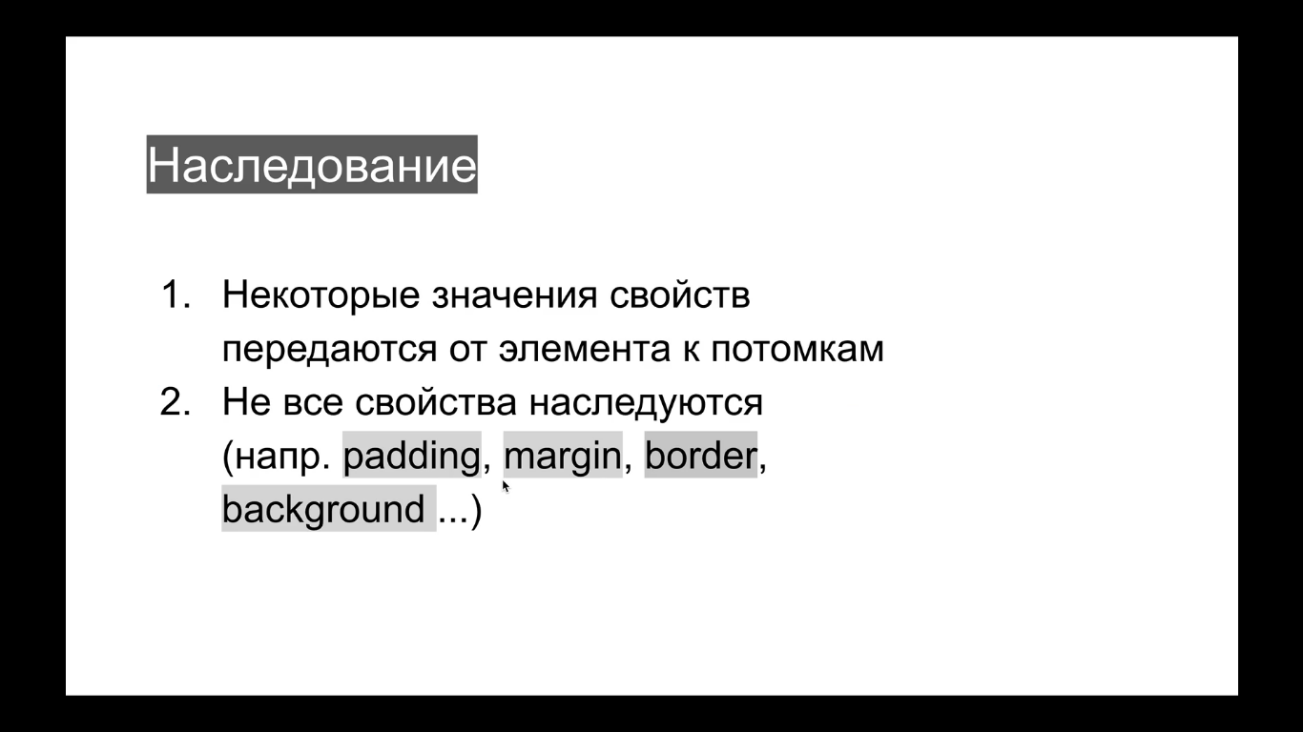
**background-repeat: norepeat** - не повторювати картинку в тій самій області

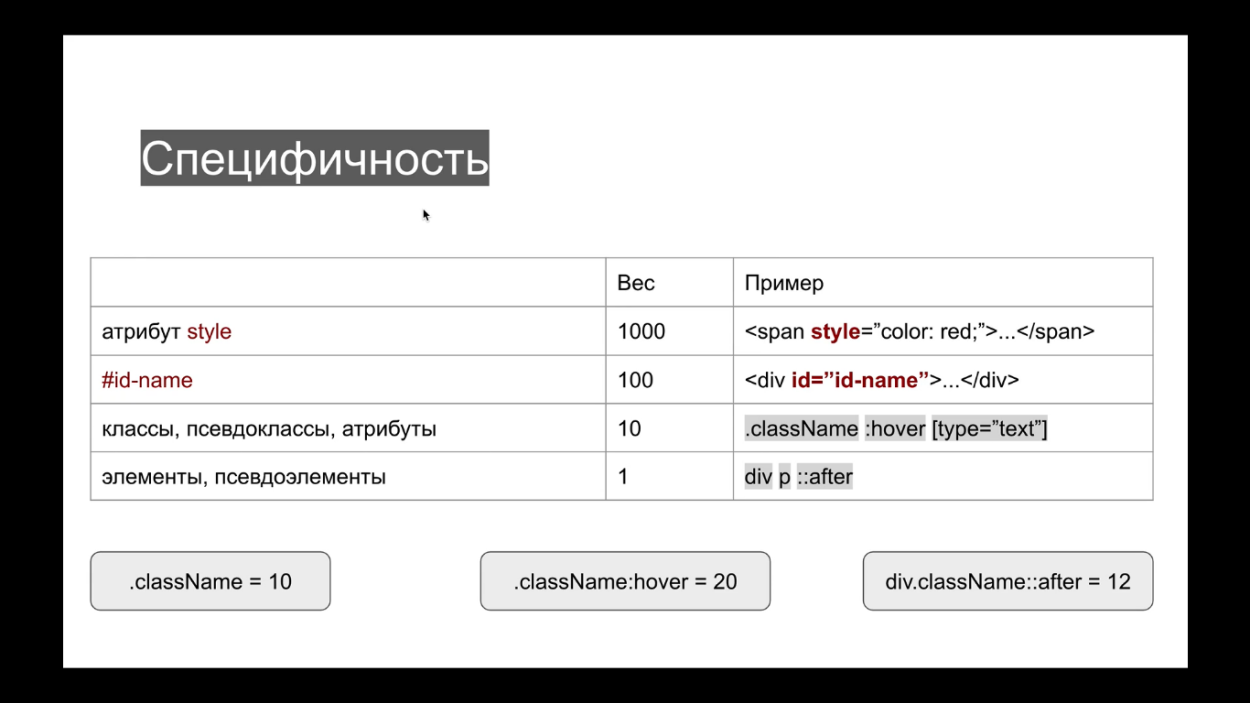
**background-image: linear-gradient** (45deg,#f00, #ffff00)

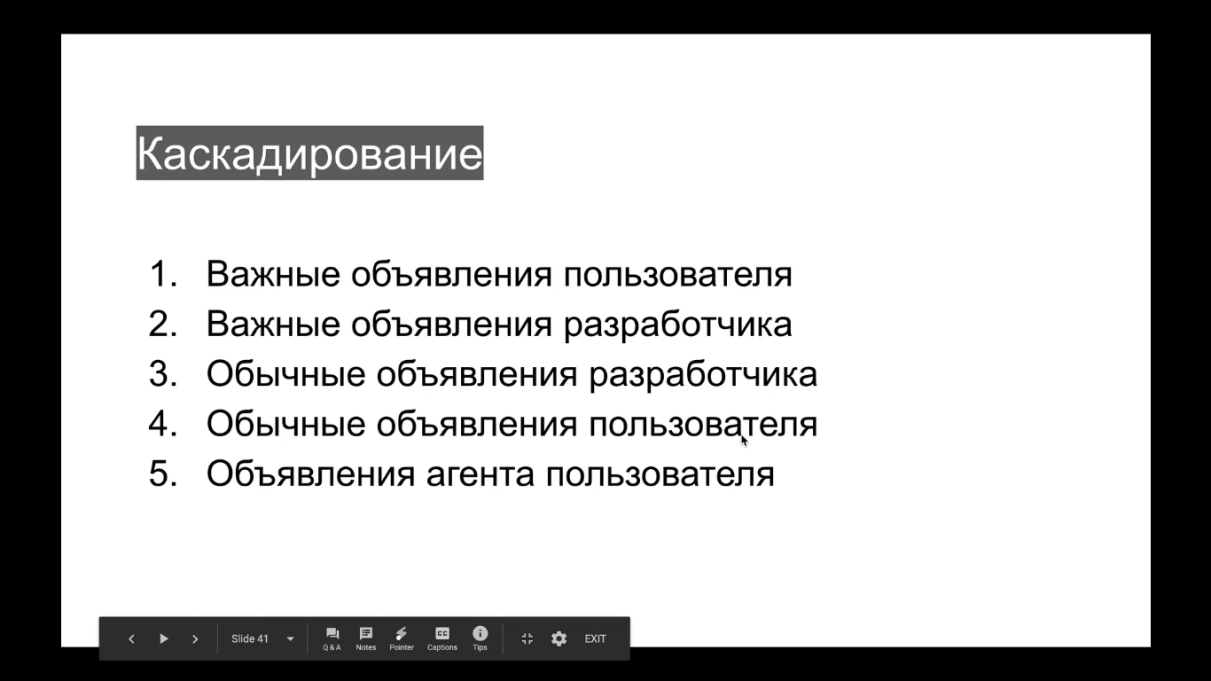
Не всі браузери підтримують деякі функції тому при стилізації потрібно додавати вендерний префікс, в кожного браузера він свій

**Наслідування і каскадність**

****





Multimedia and Embedding

**Графіка:**

* **вектрова -** зберігаються не точки а формули котрі оприділяють лінії, якість зображення не втрачається і може бути записане прямо в вигляді тегів в html, менша вага файлів **svg - scalable vector graffic**
* **растрова -** зображення зберігається по точкам, в кожній точці зберігається її колір (jpeg, png....)

Якщо зображенню змінити пропорції ширини і висоти, то воно здеформується, керувати цим станом можна за допомогою властивості

* **object-fit: fill** - картинка підлаштовується
* **object-fit: none** - картинка залише свій розмір а контейнер обрізає краї
* **object-fit: contain** - картинка всередині без зміни пропорцій
* **object-fit: cover** - картинка закриє всю площу обрізаючи краї
* **object-position: top,left**.... - керує положенням картинки

**Вставка зображення:**

* **<img src=" nature.jpeg" alt="nature"> -** картинка являється контентом і додатковий контент зсуває картинку, має свої розміри і не повторюється
* **background-image: url (https:// .....) -** не являється контентом не зсуває інший контент і використовується просто для стилізації поверх зображення може бути текст, повторюється щоб заповнити весь простір, але якшо ми цього не хочемо ми можемо застосувати

**background-repeat: no repeat**

**background-repeat: repeat -x - або -y** картинка повторюватиметься по горизонталі

**background-size: contain / 400px / cover**

**background-position: 0,0 - позиція по х і у, center**

**Розміщення картинки в тексті: (для картинки застосувати властивість)**

* **float:left/right -** текст обтікає по правій або лівій сторні
* **clear:left/right/both -** якшо я не хочу шоб текст обтікав по якійсь сторні

**alt** - в випадку коли картинка не відображатиметься атрибут alt інформує що то за картинка

**max-width** - адаптація зображення до ширини екрану

**<figure> </figure>** - тег для роботи з зображеннями

**<figcaption> </figcaption>** - опис картинки, підпис до фото

**box-shadow: 0px 5px 30px -9px rgba(0,0,0,0.75)**  - тінь елемента

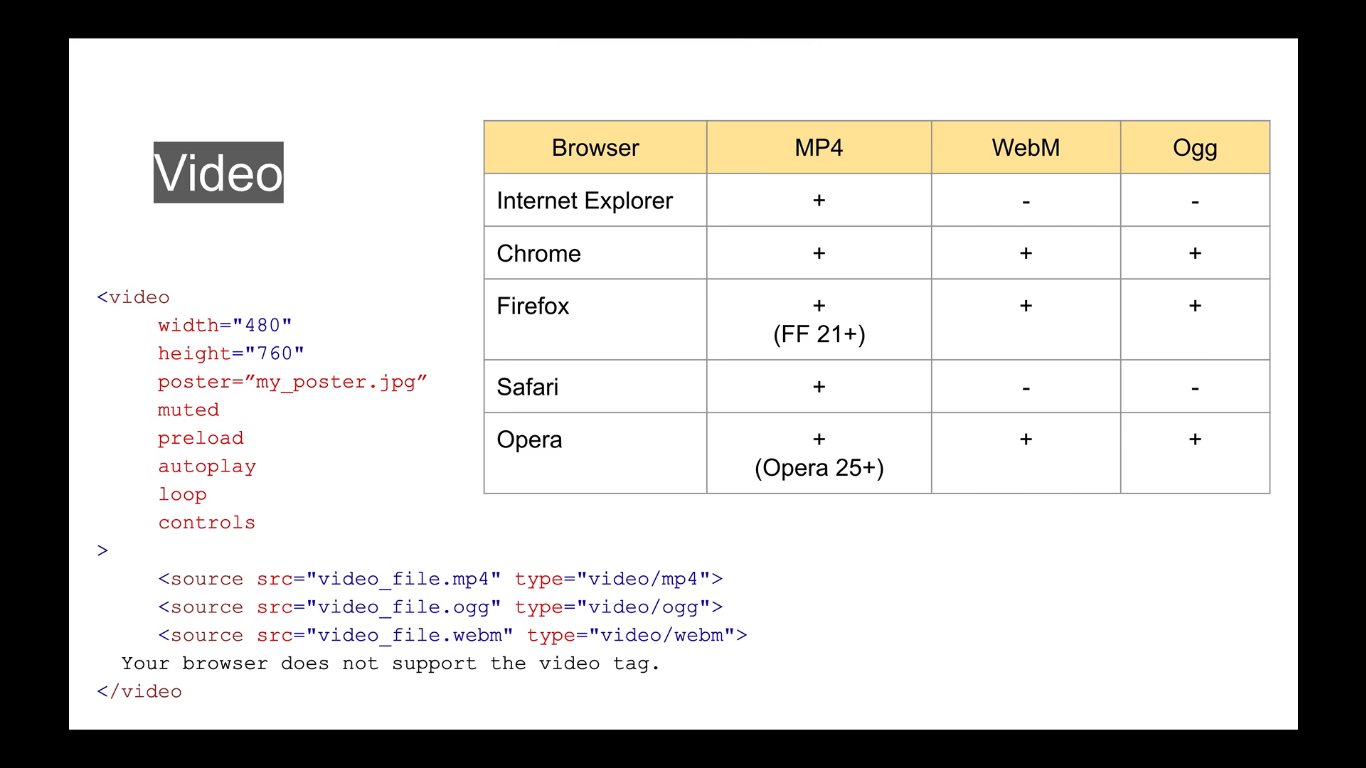
**box shadow css generator** - джерело для підбору стилю тіні

**filter:**

* **blur(4px)** - властивість котра регулює розмитість елементу
* **opacity(50%)** - властивість котра регулює прозорість елементу
* **invert (100%)** - властивість котра інвертує кольори (міняє на протилежні)

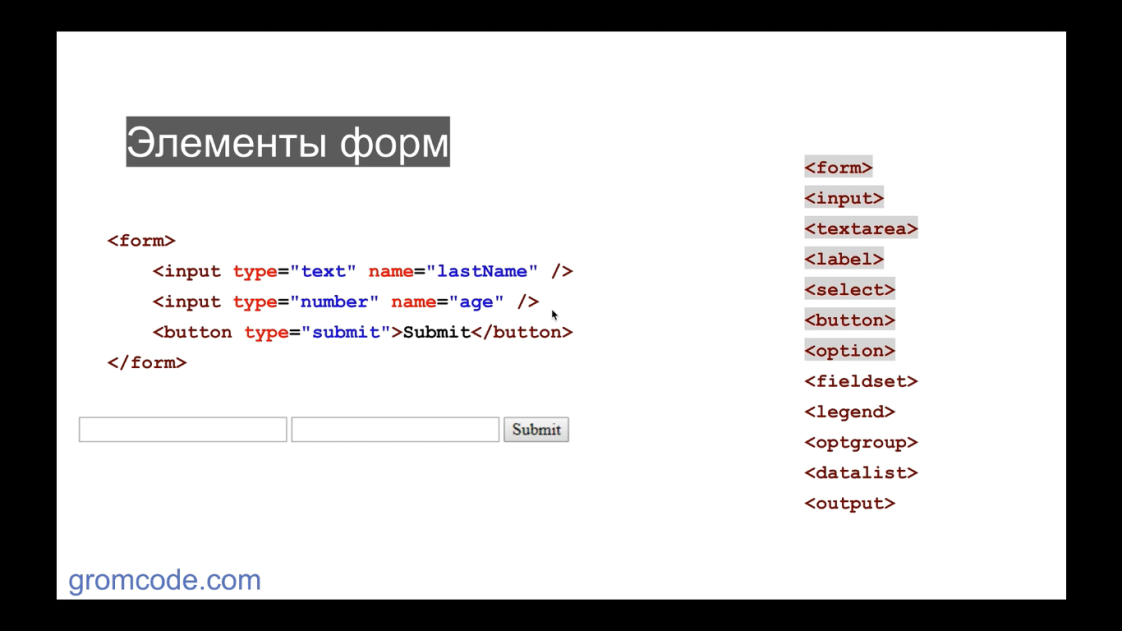
**<video src="дорога до файлу" >** :

* **controls** - атрибут controls дозволяє взаємодіяти з відеофайлом, без нього в документі видно тільки кадр з відео
* **poster** - картинка перед початком відео

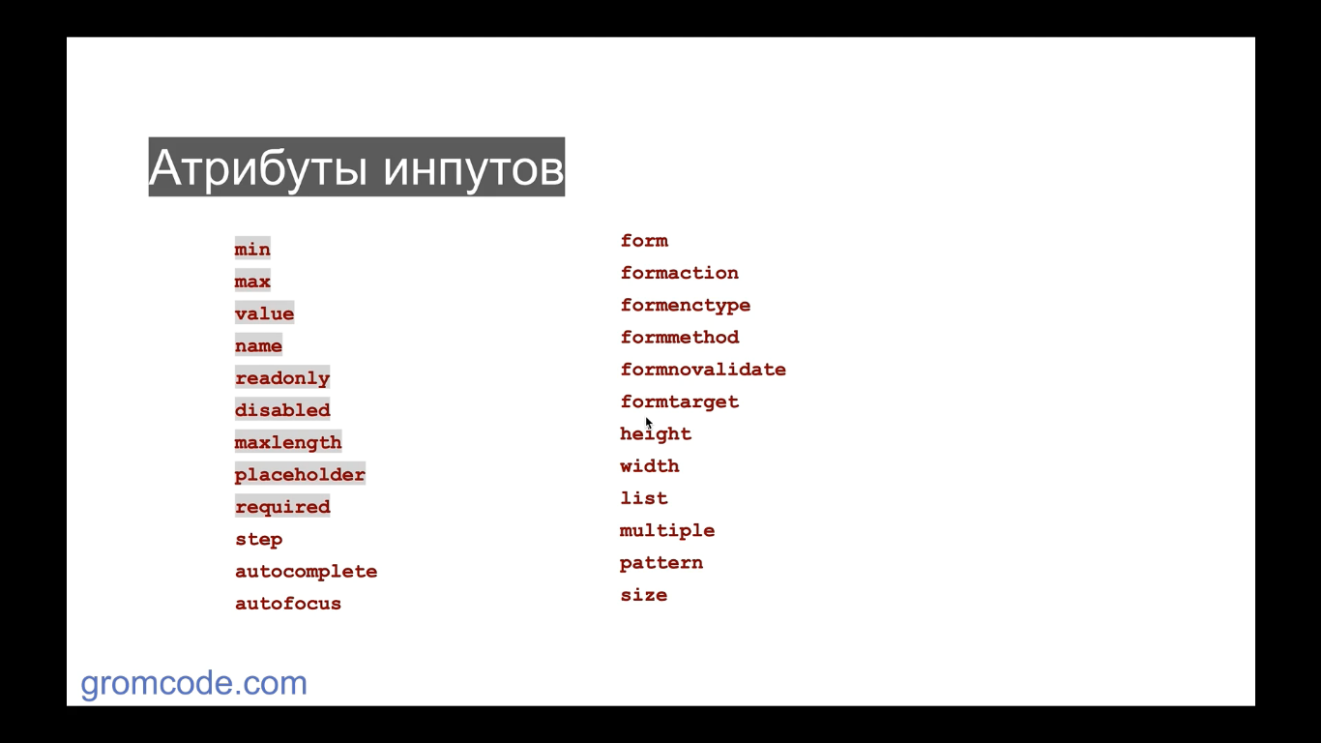


**<iframe width="560" height="315" src="https://www.youtube.com/embed/bK29EBx7AH8" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>** - за допомогою тегу iframe можна ділитись відео з різних джерел

**Форми**







**Tables**

**<tbody> </tbody> -** таблиця повинна бути завернута в тег tbody

**<tables> </tables>** - таблиця

**<tr> <tr/>** - строки

**<td> </td>** - клітинки які формуюють колонки

**<thead> </thead>** - вершок таблиці

**<th> </th>** - клітинки вершка

**<caption> </caption>** - підпис до таблиці

**<tfoot> </tfoot>** - підошва таблиці

border-collapse: collapse - схлопування відступів клітинок

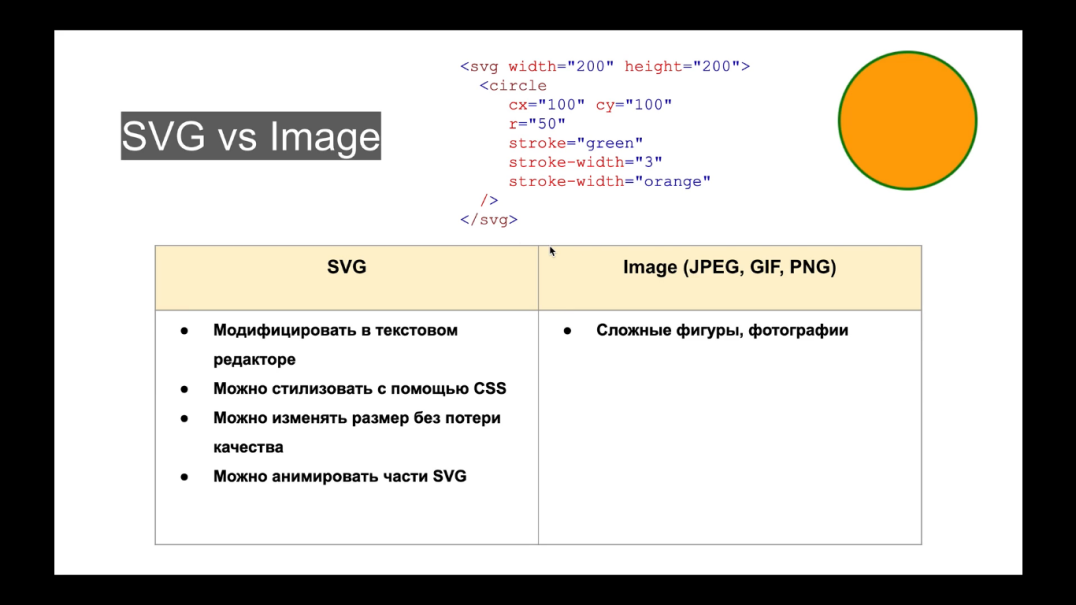
<td colspan="3"> - атрибут colspan описує скільки клітинок займатиме 1

<td rowspan="2"> - атрибут котрий обєднує клітинки по вертикалі



**Icons\_SVG**

SVG - sceable vector graffic



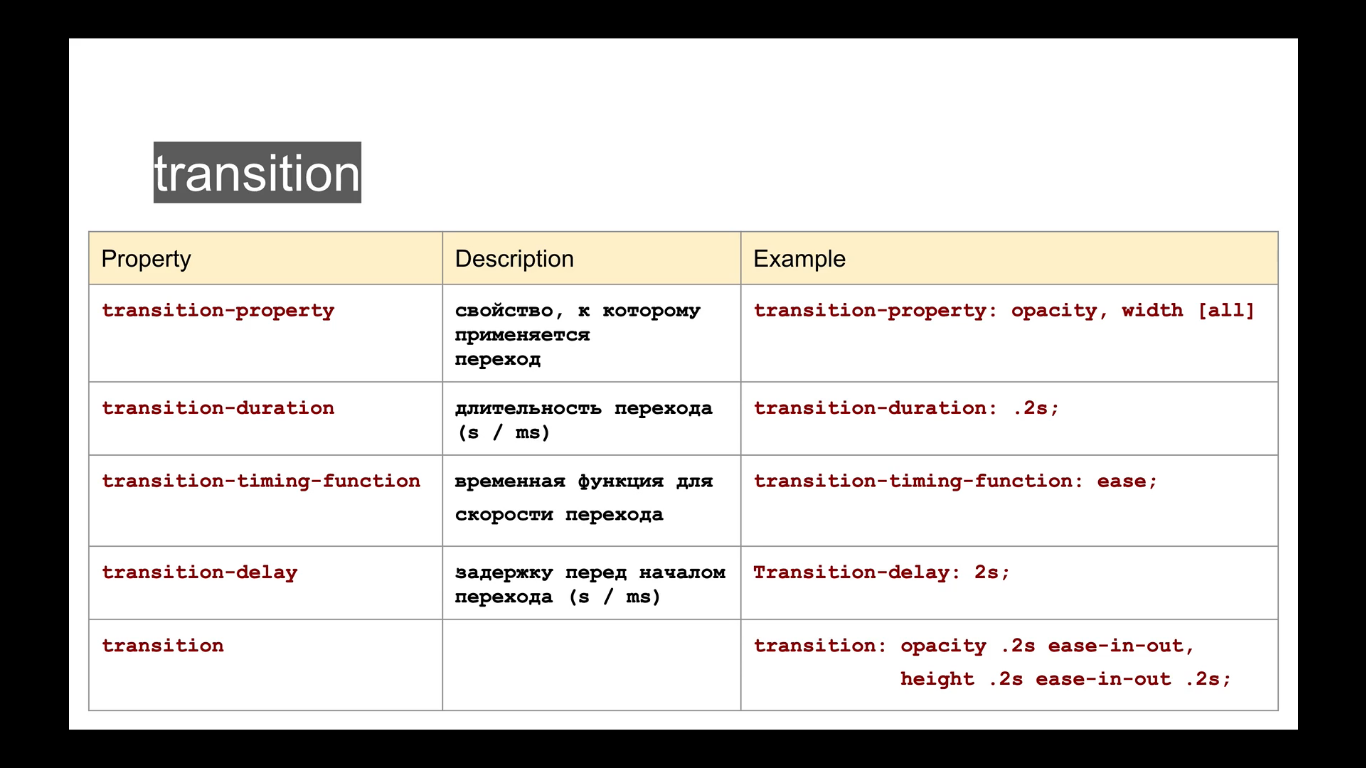
**SVG** - якість картинки не залежить від розмірів і залишається такою ж, прописується в html векторами

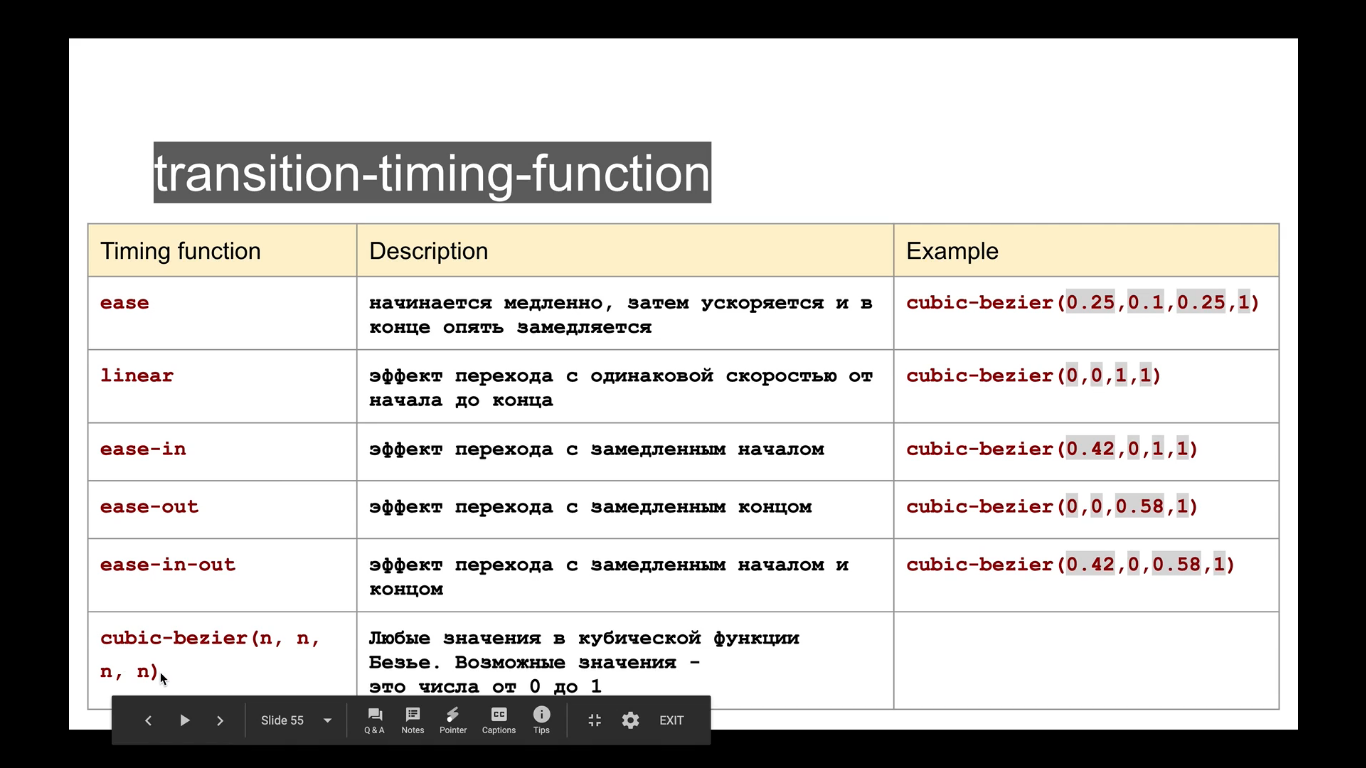
**Джерело готових іконок** : font aweasome

**Для підключення**: font aweasome CDN



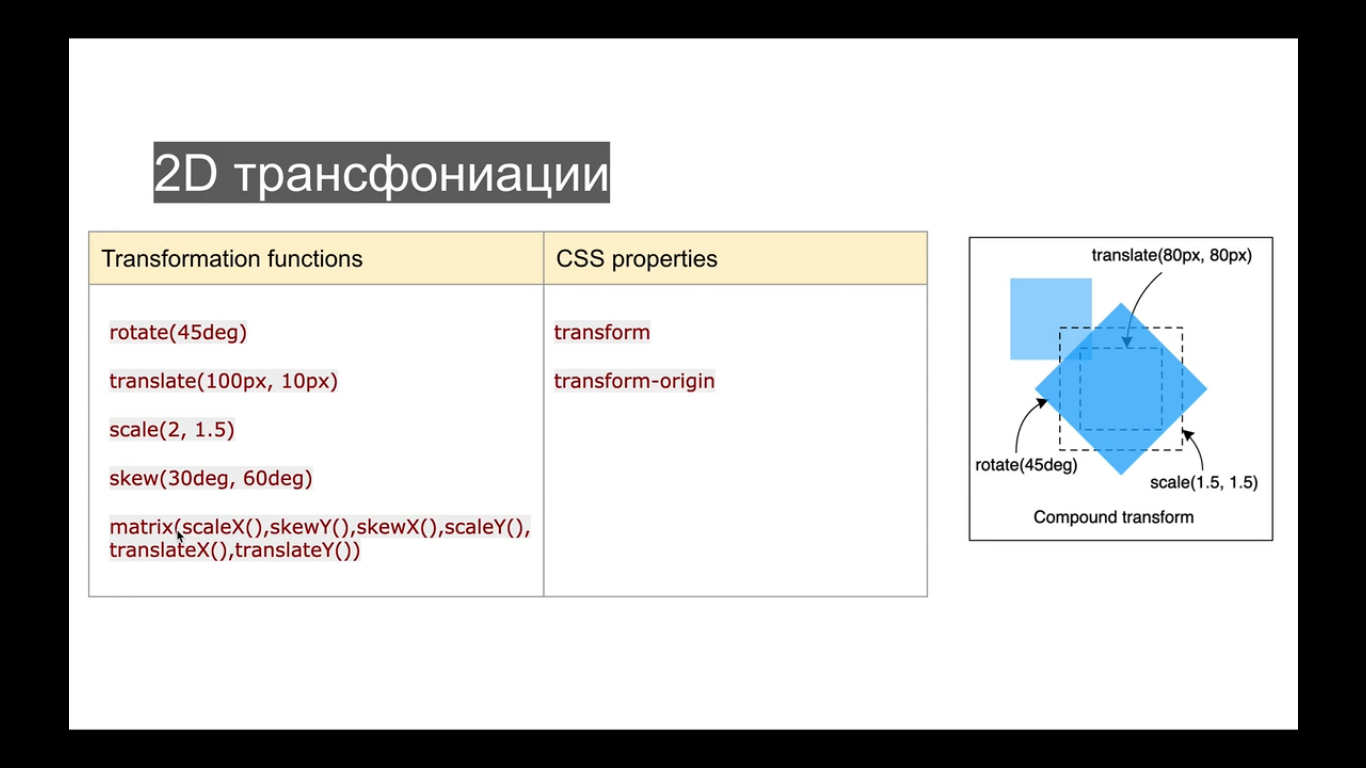
**Animation**

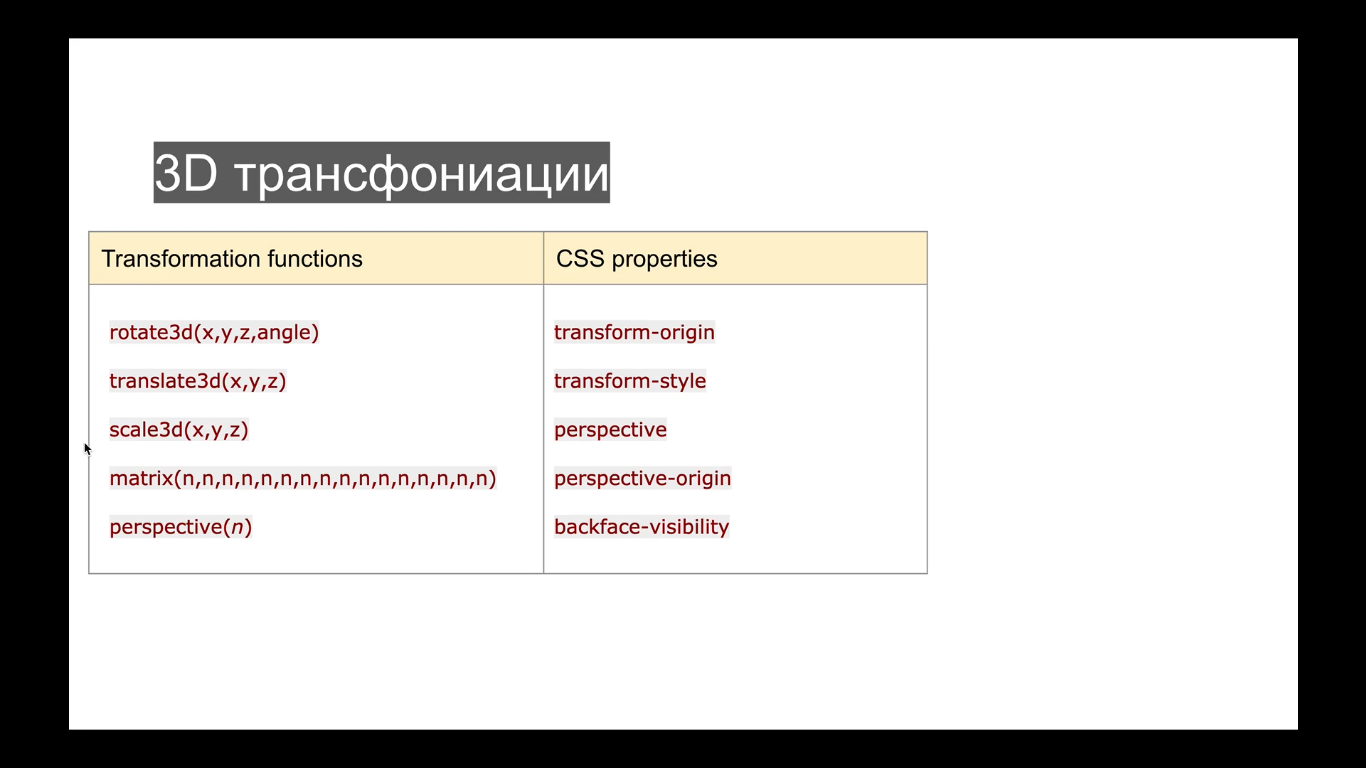






**Transformation**

****

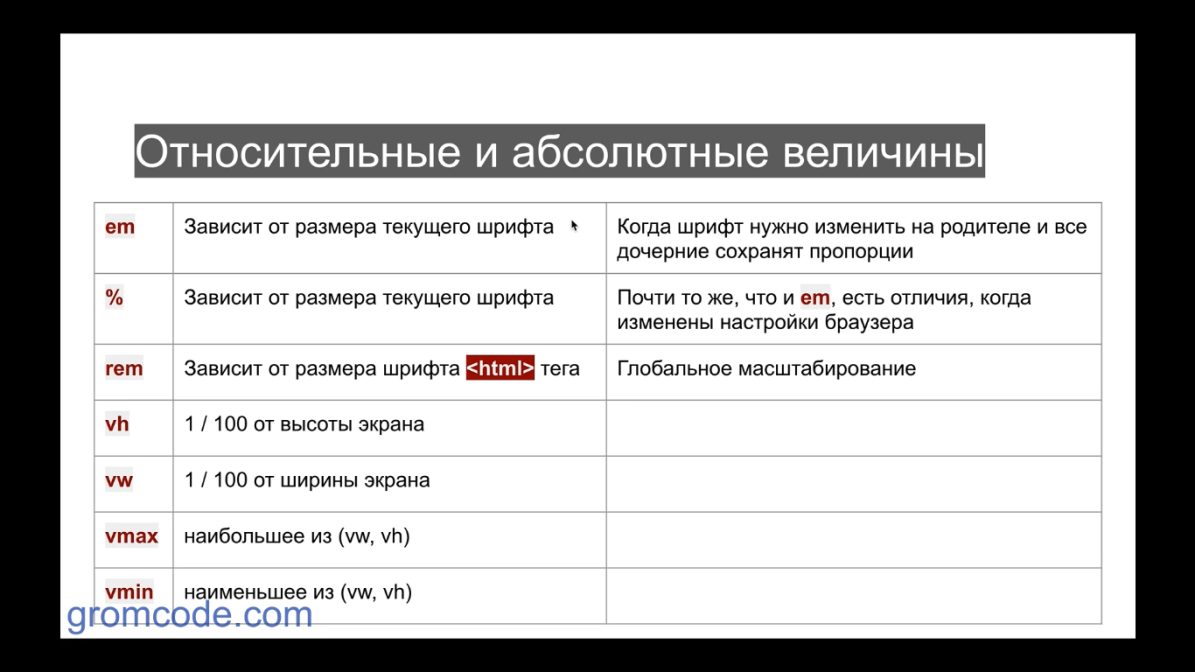


Responsive layout

Viewport - метатег в шапці профілю, котрий вказує як будуть відображатись елементи сторінки на різних девайсах.

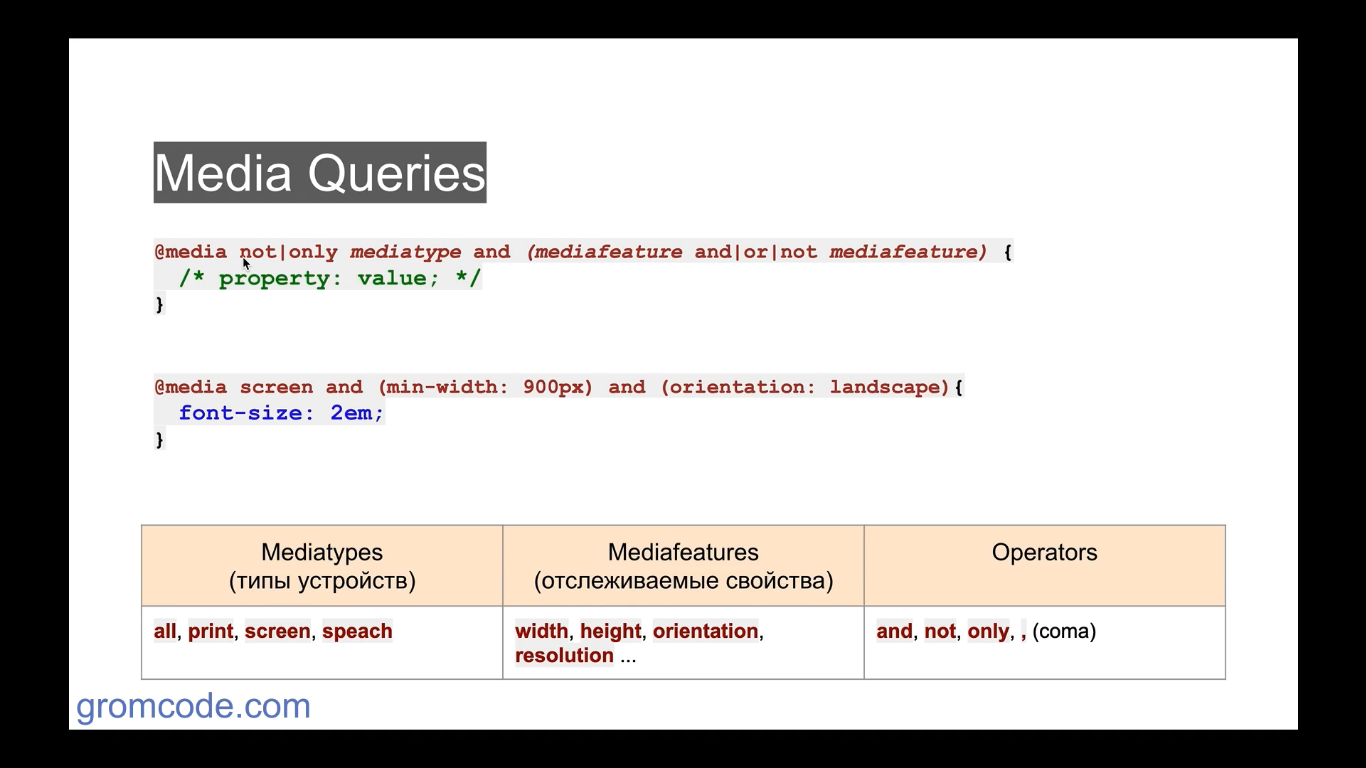




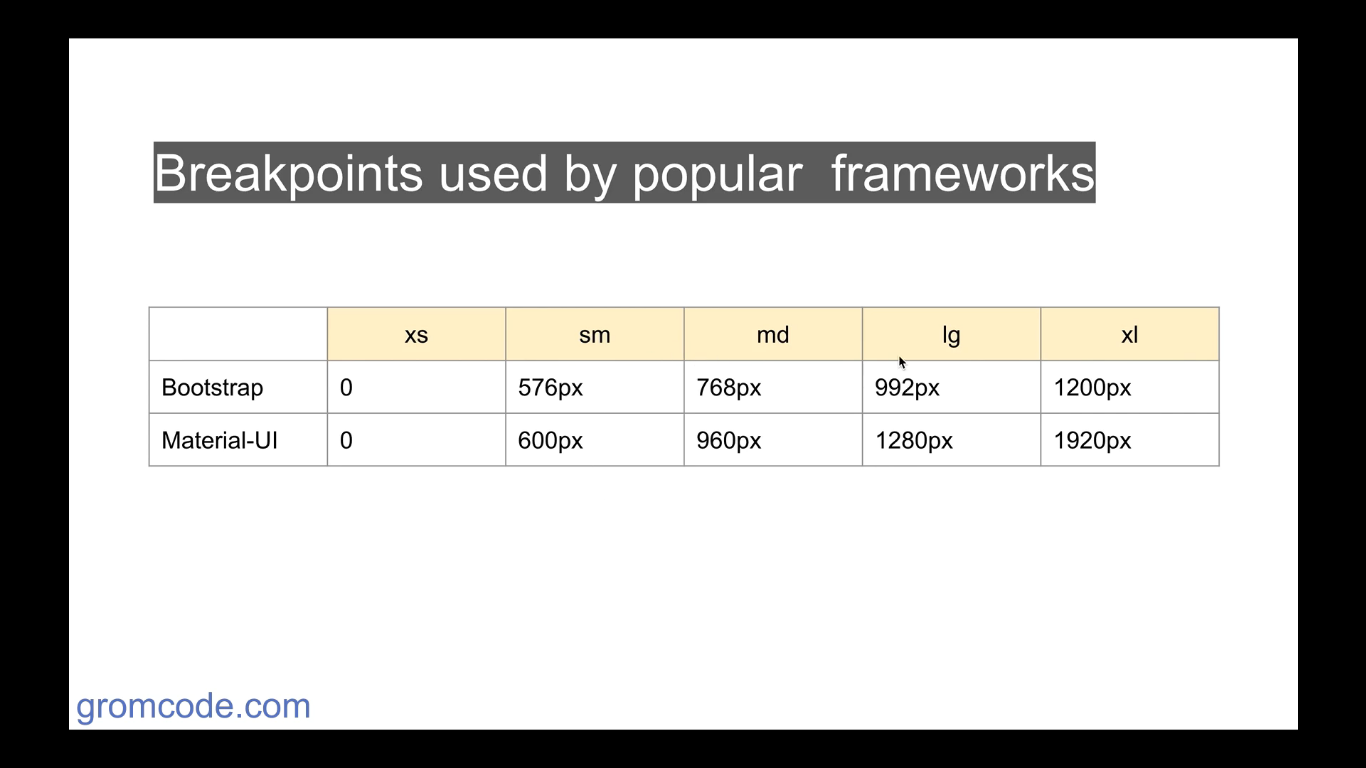


**Адаптивні веб сайти**

**Adaptive layout**



BREAKpoint - мінімальна ширина перестройки сайту



Препроцесори

