**Основи CSS**

Cascading Style Sheets (каскадні таблиці стилів) - технологія опису зовнішнього вигляду документа, написаного мовою розмітки. CSS використовується переважно для оформлення HTML- і XHTML-документів, але іноді і для інших XML-структурованих документів (наприклад, в браузері Mozilla для оформлення елементів графічного інтерфейсу, XUL).

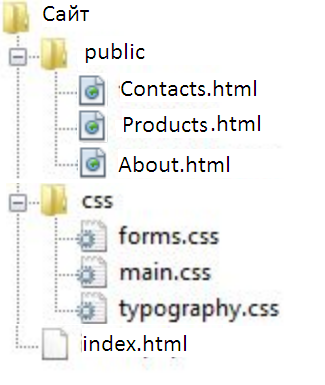
CSS використовується авторами веб-сторінок для завдання кольорів, шрифтів, розташування і інших аспектів представлення документа. Основною метою розробки CSS було розділення вмісту (написаного на HTML або іншій мові розмітки) і представлення документа (написаного на CSS). Це розділення може збільшити доступність документа, надати велику гнучкість і можливість управління його представлення, а також зменшити складність і повторюваність в структурному вмісті. Крім того, CSS дозволяє представляти один і той же документ в різних стилях або методах виводу в залежності від пристрою виводу.

Таблиця стилів складається з набору правил. Стандарт CSS визначає пріоритети, у порядку яких застосовуються правила стилів, якщо для якогось елементу підходять деякі правила одночасно. Це називається "каскадом", в якому для правил розраховуються пріоритети або "ваги", що робить результати передбаченими.

До появи CSS оформлення веб-сторінок здійснювалося безпосередньо усередині вмісту документа.

<p style=”color:red”> Важлива інформація </p>

Проте з появою CSS стало можливим принципове розділення змісту (HTML-розмітка) і представлення (форматування) документа (CSS).



 За рахунок цього нововведення стало можливим легке застосування єдиного стилю оформлення для маси схожих документів, а також швидка зміна цього оформлення.

Переваги CSS розмітки:

* Декілька дизайнів сторінки для різних пристроїв перегляду. Наприклад, на екрані дизайн буде розрахований на велику ширину, під час друку меню не виводитиметься, а на КПК і стільниковому телефоні меню буде розташоване не збоку а вслід за вмістом.
* Зменшення часу завантаження сторінок сайту за рахунок перенесення правил представлення даних в окремий CSS-файл. В цьому випадку браузер завантажує тільки структуру документа і дані, що зберігаються на сторінці, а представлення цих даних завантажується браузером тільки один раз і кешується.
* Простота подальшої зміни дизайну. Не потрібно правити кожну сторінку, а лише змінити CSS.
* Додаткові можливості оформлення. Наприклад, за допомогою CSS-розмітки можна зробити блок тексту, який решта тексту обтікатиме (наприклад для меню) або зробить так, щоб меню було завжди видно при скролінгу сторінки.

***Способи задання стилів***

**Зовнішні таблиці стилів**

|  |  |
| --- | --- |
| Загальний вигляд | **<link rel="stylesheet" type="text/css" href="шлях до  файлу">**  у HTML5 атрибут type можна не вказувати  **<link rel="stylesheet" href="шлях до файлу">** |
| Приклад | <head>    <link rel="stylesheet" href="css/style.css">    <link rel="stylesheet" href="css/assets.css" media="all">  </head> |

Розглянемо деякі атрибути тегу  <link>

|  |  |
| --- | --- |
| Атрибут | Опис атрибута |
| [href](https://css.in.ua/html/tag/link/href) | Повний або відносний шлях до об'єкту що підключається. |
| [media](https://css.in.ua/html/tag/link/media) | Визначає, які стилі застосовувати для конкретного пристрою |
| [rel](https://css.in.ua/html/tag/link/rel) | Визначає тип взаємозв'язку між поточним html-документом і пов'язаним |
| [type](https://css.in.ua/html/tag/link/type) | MIME-тип даних зв'язуваного файлу. У HTML5 за замовчуванням має значення “type/css”, тому при підключенні зовншінх таблиць стилів може бути не вказаним. |

**Внутрішні (глобальні) стилі**

|  |  |
| --- | --- |
| Загальний вигляд | <head>    <style type="text/css">      опис стилю згідно синтаксису CSS    </style>  </head>  у HTML5 атрибут type можна не вказувати |
| Приклад | <head>    <style>      h2 {           color: red;           font-family: "Times New Roman",                        Georgia, Serif;           line-height: 1.3em;          }          ...     </style>     ...  </head>  <body>  ...  </body> |

**Вбудовані стилі**

|  |  |
| --- | --- |
| Загальний вигляд | < тег  **style =** “властивість1:значення1; властивість2:значення2;…” > |
| Приклад | <p style="font-weight: bold; color: red;"> текст</p>  <p style="font-size: 120%; font-family: monospace; color:#cd66cc">приклад</p> |

**Правило @import**

<style>

**@import url("styles.css");**

  ... інші оператори імпорту або стилі CSS...

</style>

@іmport може бути використаним для підключення шрифтів:

@import url(<https://fonts.googleapis.com/css>? family=Open+Sans &subset=latin,cyrillic);

@іmport також використовують для підлючення (вставки) одних таблиць стилів в інші.

*Cинтаксис CSS*

|  |  |
| --- | --- |
| Загальний вигляд | cелектор{            властивість1:значення1;            властивість2:значення2;             …            властивістьN:значення;  } |
| Приклад | body {      color: red;      font-weight: normal;  }  h1, h2 {      color: #00ff00;  }  p {      color: rgb(0,0,255);  } |

h1 { color: #a6780a; font-weight: normal; border-bottom: 2px solid black}

H1 {

border-bottom: 2px solid black

color: #a6780a;

font-weight: normal;

}

* Для селектора допускається додавати кожне правило окремо:

td {background: olive;}

td {color: white;}

td {border: 1px solid black;}

Однак такий запис не дуже зручний. Доводиться повторювати кілька разів один і той же селектор, та й легко заплутатися в їх кількості. Тому пишіть все властивості для кожного селектора разом. Компактна форма запису

td {

   background: olive;

   color: white;

   border: 1px solid black;

}

* Якщо для селектора спочатку задається властивість з одним значенням, а потім те ж властивість, але вже з іншим значенням, то застосовуватися буде те значення, яке в коді встановлено останнім

p {color: green;}

p {color: red;}

**Коментарі**

Щоб позначити, що текст є коментарем, застосовують наступну конструкцію / \* …коментар... \* /

|  |  |
| --- | --- |
| Загальний вигляд | / \* …коментар... \* / |
| Приклад | **/\***  **\*\*\*\*\*\*\*\*\*\*\*\*Мій стиль\*\*\*\*\*\*\*\*\*\*\*\*\***  **Зроблено для ознайомлення**  **\*/**  body {      color: red;  }  p {      color: rgb(0,0,255);  } |

*Типи даних*

У якості значення для властивостей можуть бути вказані величини таких типів даних:

* рядки;
* числа;
* відносні або абсолютні одиниці виміру;
* адреси;
* кольори.

**Рядки**

Будь-які рядки необхідно брати в подвійні лапки. Якщо всередині рядка потрібно використати лапки чи слеш, то необхідно додати перед ними додатковий слеш (екранувати символ).

Приклад.

"Готель **\"** Турист **\"** "

**Числа**

Значенням може виступати ціле число, що містить цифри від 0 до 9 і десятковий дріб, в якій ціла і десяткова частина розділяються крапкою. Приклад.

 500

 3.14

**Розміри**

Для завдання розмірів різних елементів, в CSS використовуються *абсолютні* і *відносні* одиниці виміру. Абсолютні одиниці не залежать від пристрою виводу, а відносні одиниці визначають розмір елементу щодо значення іншого розміру.

***Відносні одиниці*** зазвичай використовують для роботи з текстом, або коли треба обчислити  співвідношення між елементами.

|  |  |
| --- | --- |
| em | Розмір шрифту поточного елементу |
| ex | Висота символу x |
| px | Піксель |
| % | Відсоток |

Одиниця **em** це змінюване значення, яке залежить від розміру шрифту поточного елементу (розмір встановлюється через стильову властивість font-size). В кожному браузері закладений розмір тексту, застосовуваний в тому випадку, коли цей розмір явно не заданий. Тому спочатку 1em дорівнює розміру шрифту, заданого в браузері за замовчуванням або розміром шрифту батьківського елементу. Відсотковий запис ідентичний до em, в тому сенсі, що значення **1em** і **100%** рівні.

Одиниця **ex** визначається як висота символу «x» в нижньому регістрі. На ex розповсюджуються ті ж правила, що і для em, а саме, він прив'язаний до розміру шрифту, заданого в браузері за замовчуванням, або до розміру шрифту батьківського елементу.

Піксель це точка певного кольору, що відображається монітором або іншим подібним пристроєм, наприклад, смартфоном. Розмір пікселя залежить від розшення пристрою і його розмірів.

Відсотковий запис зазвичай застосовується в тих випадках, коли треба змінити значення щодо батьківського елементу або коли розміри залежать від зовнішніх умов. Так, ширина таблиці 90% значить, що вона буде підлаштовуватися під розміри вікна і змінюватися разом з шириною вікна.

***Абсолютні одиниці*** застосовуються рідше, ніж відносні і зазвичай при роботі з текстом.

|  |  |
| --- | --- |
| in | Дюйм (1 дюйм дорівнює 2,54 см) |
| cm | Сантиметр |
| mm | Міліметр |
| pt | Пункт (1 пункт дорівнює 1/72 дюйма) |
| pc | Піка (1 піка дорівнює 12 пунктам) |

Найпоширенішою одиницею є пункт, який використовується для вказівки розміру шрифту. Це єдина величина з неметричної системи вимірювання, яка використовується дуже широко. І все завдяки текстовим редакторам і видавничим системам.

**Адреси**

(URI, Uniform Resource Identifiers, уніфікований ідентифікатор ресурсів) застосовуються для вказівки шляху до файлу, наприклад, для установки фонової картинки на сторінці. Для цього застосовується ключове слово url (), всередині дужок пишеться відносний або абсолютний адресу файлу. При цьому адреса можна задавати в необов'язкових подвійних лапках.

**Колір**

У стилях можна задавати трьома способами:

* за значення у шістнадцятковій системі числення (3 набори з двоцифрових чисел, кожне число (від 00 до ff) – інтенсивність відповдіно червогоно, зеленого і синього ). Починається значення з символу «#». Наприклад: [#3CB371](https://www.w3schools.com/colors/color_tryit.asp?hex=3CB371);
* за назвою (наприклад: white, silver, gray, black, red,orange, yellow, green, blue, navy, purple);
* у форматі RGB (наприклад, rgb(141, 176, 145)).

Назви кольорів та їх шістнадцяткові значення можна, наприклад, знайти за адресою https://www.w3schools.com/colors/);

Значеня кольору у форматі RGB задається шляхом задання інтенсивності кожної із складових, «змішуванням» яких одержуємо потрібний колір (червоного, зеленого, синього). Значення кожного з кольорів може змінватися від 0 до 255. Також можна задавати колір у відсотковому відношенні. Спочатку вказується ключове слово rgb, а потім в дужках, вказуються компоненти кольору, наприклад rgb (255, 51, 51) або rgb (100%, 20%, 20%).

Приклад

body {

    color: red;

}

h1 {

    color: #00ff00;

}

p {

    color: rgb(0,0,255);

}

*Види селекторів*

**Універсальний селектор**

|  |  |
| --- | --- |
| Загальний вигляд | \* {            властивість1:значення1;             …            властивістьN:значення;  } |
| Приклад | \*{        border: 1px solid #000000;  } |

**Селектор елемента**

|  |  |
| --- | --- |
| Загальний вигляд | **ім’я тега** {            властивість1:значення1;             …            властивістьN:значенняN;  } |
| Приклад | p {       text-align: justify;       color: green;  } |

**Селектор класу**

|  |  |
| --- | --- |
| Загальний вигляд | <тег **class** = "Ім'я\_класу"> |
| Приклад | <h1 **class="headline"**> Зміст </h1> |

|  |  |
| --- | --- |
| Загальний вигляд | **.** Ім'я\_класу{            властивість1:значення1;             …            властивістьN:значення;  } |
| Приклад | ------ У тексті HTML-сторінки -----  <h1 **class="headline"**> Зміст </h1>  <p  **class="headline"**> Важливий текст </p>  ------ У таблиці стилів CSS ----  **.headline** {    text-transform: uppercase;    color: lightblue;  } |

|  |  |
| --- | --- |
| Загальний вигляд | **тег.**ім'я\_класу{            властивість1:значення1;             …            властивістьN:значенняN;  } |
| Приклад | ------ У тексті HTML-сторінки -----  <h1 **class="headline"**> Зміст </h1>  ------ У таблиці стилів CSS ----  **h1.headline** {    text-transform: uppercase;    color: lightblue;  } |

**Селектор ідентифікатора**

|  |  |
| --- | --- |
| Загальний вигляд | <тег  **id** = "ідентифікатор"> |
| Приклад | ------ У тексті HTML-сторінки -----  <p **id="first\_line"**> Зміст </p> |

|  |  |
| --- | --- |
| Загальний вигляд | # **id** елемента **{**            властивість1:значення1;             …            властивістьN:значенняN;  **}** |
| Приклад | ------ У тексті HTML-сторінки -----  <p **id="first\_line"**> Зміст </p>  ------ У таблиці стилів CSS ----  **#first\_line** {    text-transform: uppercase;    color: lightblue;  } |

|  |  |
| --- | --- |
| *Ідентифікатори* (id) | *Класи* |
| у коді документа кожен ідентифікатор унікальний і повинен бути включений лише один раз | клас може використовуватися в коді багато разів для різних елементів |
| ім'я ідентифікатора чутливе до регістру | імена класів чутливі до регістра |
| стиль для ідентифікатора має пріоритет вище, ніж у класів |  |
| значення ідентифікатора визначається як одна величина | класи можна комбінувати між собою, додаючи кілька класів до одного тегу розділяючи їх пробілом |

**Селектор нащадків**

|  |  |
| --- | --- |
| Елементи у HTML документі | <тег\_1>    . . . .  **<тег\_2>**       ... вміст тегу ...  **</тег\_2>**    . . .  **<тег\_2>**       ... вміст тегу ...  **</тег\_2>**    . . .  </тег\_1> |
| Селектор вибору | тег\_1 тег\_2 {            властивість1:значення1;             …            властивістьN:значенняN;  } |
| Приклад | ------ У тексті HTML-сторінки -----  **<section>**      <h2> Підрозділ 1 </h2>  **<p>**      . . . Текст підрозділу 1 . . .  **</ p >**      <h2> Підрозділ 2 </h2>  **< p >**      . . . Текст підрозділу 1 . . .  **</ p >**  **</section >**  ------ У таблиці стилів CSS ----  /\*Вибрати усі елементи-нащадки <**p**>, які знаходяться всередині тегів-предків <**section**> \*/  **section p**{    text-transform: uppercase;    color: lightblue;  } |

|  |
| --- |
| ------ У тексті HTML-сторінки -----  **<section>**      <h2> Підрозділ 1 </h2>  **<p>**      . . . Текст підрозділу 1 . . .  **</p>**      <h2> Підрозділ 2 </h2>      <div>         . . .  **<p>**      . . . Текст підрозділу 1 . . .  **</p>**  </div>  **</section >**  ------ У таблиці стилів CSS ----  /\*Вибрати усі елементи-нащадки **<p>**, які знаходяться всередині тегів-предків < **section** > \*/  **section p**{    text-transform: uppercase;    color: lightblue;  } |

|  |  |
| --- | --- |
| Предок є елементом певного класу | 1) Усі теги <p>, які знаходяться всередині елемента з класом “info”  .info p{     color:red  }  2)Усі теги  <p>, які знаходяться всередині тегів  <section> з класом “info”  section.info p{     color:red  }  3) Усі теги <p>, які знаходяться всередині тегів з класом “info”, а вони в свою чергу знаходяться всередині тегу  <section> (між тегом і класом вставлено пропуск)  section .info p{     color:red  }  3) Усі теги з класом “hot”, які знаходяться всередині тегів з класом “news”  .news .hot{     color:red  } |
| Предок є елементом із заданим id | Елементи <p>, які знаходяться всередині тегу із вказаним id=“**intro**”  **#intro** p{     color:red  } |

|  |  |
| --- | --- |
| Правило | Елементи, для яких буде застосовано |
| p \*{    color:red  }  **Усі теги всередині тегів р** | <p>      The **<em>*main* </em>** part   </p>  - - - - - - - - - - - - - - - - - -  - - - - - -  <p>        🡨 відповідає тегу “p”       The  **<em> *main*       🡨** відповідає “\*”  **<strong>*important*</strong>**  **</em>**  part  </p> |
| p \* \*{    color:red  }  **Усі елементи, які знаходяться всередині дякого тегу, який знаходиться всеридині тегу “p”** | <p>The                  🡨 відповідає тегу “p”      <em>        🡨 відповідає першій “\*”  *main*  **<strong>     🡨** відповідає другій “\*”  *important*  **</strong>**  </em>     part  </p> |
| p \* a{    color:red  }  **Усі теги “a”, які знаходяться всередині деякого тегу, який знаходиться  всередині тегу “p”** | <p>The      <em> *main*         <strong>*important*</strong>  *part of*  **<a href= “#”>*information*</a>**  </em>        part         <a href= “#”>*information2*</a>  </p> |

**Дочірній селектор**

|  |  |
| --- | --- |
| Загальний вигляд | предок > нащадок{    . . . правила форматування . . .  } |
| Приклад | ------ У таблиці стилів CSS ----  Усі теги  <p>, для яких тег  <section> є безпосереднім предком  section **>** p{     color:red  }  ------ У тексті HTML-сторінки -----  **<section>**      <h2> Підрозділ 1 </h2>  **<p>   🡨 *Підходить***      . . . Текст підрозділу 1 . . .  **</p>**      <h2> Підрозділ 2 </h2>      <div>         . . .  <p>  🡨 ***Даний*** <p> ***не підходить бо не є***  ***бепосереднім нащадком* <section>**      . . .  </p>  </div>  **</section >** |

**Селектор сусідів**

|  |  |
| --- | --- |
| Є безпосередніми сусідами | **<h1>…</h1>**... допустимий деякий текст без тегів ...**<p>…</p>** |
| Не є безпосередніми сусідами | **<h1>…</h1>**... є <b>принаймні один</b> тег...**<p>…</p>** |

|  |  |
| --- | --- |
|  | <section>   🡨 **спільний предок**  <p>    Текст сусіда, але не безпосереднього  </p>  Деякий довільний текст (можливо і з наявністю тегів)  <h1>Підрозділ 1.1</h1>    Деякий текст БЕЗ ТЕГІВ  **<p>**  **ТЕКСТ БЕЗПОСЕРЕДНЬОГО СУСІДА**  **</p>**    Деякий довільний текст (можливо і з наявністю тегів)    <p>        Текст сусіда, але не безпосереднього    </p>    Деякий довільний текст (можливо і з наявністю тегів)    <p>        Текст сусіда, але не безпосереднього    </p>      <h1>Підрозділ 1.2</h1>  **<p>**  **ТЕКСТ БЕЗПОСЕРЕДНЬОГО СУСІДА**  **</p>**  </section> |
|  | h1+p{     color:red  } |

|  |  |
| --- | --- |
|  | <section>   🡨 **спільний предок**  <p> 🡨 не підходить, бо знаходиться перед базовим    Текст сусіда, але не безпосереднього  </p>  Деякий довільний текст (можливо і з наявністю тегів)  <h1>Підрозділ 1.1</h1>   🡨 базовий    Деякий текст БЕЗ ТЕГІВ  **<p>**  **ТЕКСТ БЕЗПОСЕРЕДНЬОГО СУСІДА**  **</p>**    Деякий довільний текст (можливо і з наявністю тегів)  **<p>**  **Текст сусіда, але не безпосереднього**  **</p>**    Деякий довільний текст (можливо і з наявністю тегів)  **<p>**  **Текст сусіда, але не безпосереднього**  **</p>**      <h1>Підрозділ 1.2</h1>  🡨 базовий  **<p>**  **ТЕКСТ БЕЗПОСЕРЕДНЬОГО СУСІДА**  **</p>**  </section> |
|  | h1~p{     color:red  } |

**Селектор атрибуту**

**Визначено певний атрибут (значення атрибуту не є важливим)**

|  |  |  |
| --- | --- | --- |
|  | Загальний вигляд | Приклад |
| Для будь-яких тегів, які містять атрибут | [назва атрибуту]  {      правила форматування  } | [**id**]{    color:red  }  **Усі теги з атрибутом “id”** |
| Тільки для заданих тегів, які містять вказаний атрибут | назва тегу [назва атрибуту]  {      правила форматування  } | **p**[**id**]{    color:red  }  **Усі теги “p” з атрибутом “id”** |

|  |  |
| --- | --- |
| Загальний вигляд | [назва атрибуту]  {      правила форматування  } |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **[id]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p class= “start”>First </p>  <p **id= “middle”**>Second </p> 🡨Підходить бо має атрибут id  <p> Third </p>  <a **id= “lk”** href=”#”>Link</a> 🡨Підходить бо має атрибут id |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **[class]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **class= “start”**>First </p> 🡨Підходить бо має атрибут class  <p id= “middle”>Second </p>  <p>Third </p>  <p **class= “last”**>End </p> 🡨Підходить бо має атрибут class |

|  |  |
| --- | --- |
| Загальний вигляд | назва тегу [назва атрибуту]  {      правила форматування  } |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **p[id]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p class= “start”>First </p>  <p **id= “middle”**>Second </p> 🡨Підходить, бо це “p” із “id”  <p> Third </p>  <a id= “lk”href=”#”>Link</a> |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **em[class]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p class= “start”>First </p>  <p id= “middle”>Second </p>  <p>Third </p>  <em **class= “last”**>*End* </em> 🡨Підходить, бо це “em” з “class” |

**Значення атрибуту співпадає із заданим**

|  |  |  |
| --- | --- | --- |
|  | Загальний вигляд | Приклад |
| Для будь-яких тегів, які містять атрибут з вказаним значенням | [атрибут=“значення”]  {      правила форматування  } | [**class=“info”**]{    color:red  } |
| Тільки для заданих тегів, які містять вказаний атрибут з зазначеним значенням | тег [атрибут= “значення”]  {      правила форматування  } | **p**[**class=“info”**]{    color:red  } |

|  |  |
| --- | --- |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **[class= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **class= “start”**>First</p>           🡨Підходить  <p class= “start middle”>Second </p>  <p> Third </p>  <a **class = “start”** href=”#”>Link</a>         🡨Підходить |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **p[class= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **class= “start”**>First</p>           🡨Підходить  <p class= “start middle”>Second </p>  <p> Third </p>  <a class = “start”href=”#”>Link</a> |

**Значення атрибуту починається із заданої частини**

|  |  |  |
| --- | --- | --- |
|  | Загальний вигляд | Приклад |
| Для будь-яких тегів, для яких значення атрибуту починається з вказаного фрагменту | [атрибут **^=** “значення”]  {      правила форматування  } | [**class^=“info”**]{    color:red  } |
| Тільки для заданих тегів, для яких значення атрибуту починається з вказаного фрагменту | тег [атрибут **^=** “значення”]  {      правила форматування  } | **p**[**class^=“info”**]{    color:red  } |

|  |  |
| --- | --- |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **[class^= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **class= “start”**>First</p>           🡨Підходить  <p class= “middle\_start\_info”>Second </p>  <p class= “middle\_info\_ start”> Third </p>  <a **class = “startup”** href=”#”>Link</a>         🡨Підходить |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **p[id^= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p class= “start”>First</p>  <p class= “middle\_start\_info”>Second </p>  <p id = “start\_info”> Third </p>      🡨Підходить  <a id = “start-link”href=”#”>Link</a>      🡨Підходить |

**Значення атрибуту закінчується заданим фрагментом**

|  |  |  |
| --- | --- | --- |
|  | Загальний вигляд | Приклад |
| Для будь-яких тегів, для яких значення атрибуту закінчується вказаним фрагментом | [атрибут **$=** “значення”]  {      правила форматування  } | [**class $= “info”**]{    color:red  } |
| Тільки для заданих тегів, для яких значення атрибуту закінчується вказаним фрагментом | тег [атрибут **$=** “значення”]  {      правила форматування  } | **p**[**class $= “info”**]{    color:red  } |

Розглянемо приклади

|  |  |
| --- | --- |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **[class $= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p class= “start”>First</p>  <p **class= “middle\_zonestart”**>Second </p>   🡨Підходить  <p **class= “middle-info-start**”> Third </p>  🡨Підходить  <a class = “startup”href=”#”>Link</a> |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **p[id $= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **id= “part-start”**>First</p>             🡨Підходить  <p class= “middle\_start\_info”>Second </p>  <p id = “start\_info”> Third </p>  <a **id = “my\_first\_start”** href=”#”>Link</a>      🡨Підходить |

**Значення атрибуту містить вказаний фрагмент у якомусь довільному (на початку, всередині, в кінці) місці**

|  |  |  |
| --- | --- | --- |
|  | Загальний вигляд | Приклад |
| Для будь-яких тегів, для яких значення атрибуту містить вказаний фрагмент (будь-де) | [атрибут **\*=** “значення”]  {      правила форматування  } | [**class \*= “info”**]{    color:red  } |
| Тільки для заданих тегів, для яких значення атрибуту містить вказаний фрагмент (будь-де) | тег [атрибут **\*=** “значення”]  {      правила форматування  } | **p**[**class \*= “info”**]{    color:red  } |

Розглянемо приклади

|  |  |
| --- | --- |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **[class \*= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **class= “start”**>First</p>                🡨Підходить  <p **class= “middle\_start\_zone”**>Second </p>  🡨Підходить  <p **class= “middle-info-start**”> Third </p>  🡨Підходить  <a **class = “startup”** href=”#”>Link</a>             🡨Підходить |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **p[id \*= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **id= “part-start”**>First</p>             🡨Підходить  <p class= “middle\_start\_info”>Second </p>  <p **id = “start\_info”**> Third </p>          🡨Підходить  <a id = “my\_start \_first”href=”#”>Link</a> |

**Значення атрибуту містить вказаний фрагмент як ціле слово, можливо відділене від інших пробілом**

|  |  |  |
| --- | --- | --- |
|  | Загальний вигляд | Приклад |
| Для будь-яких тегів, для яких значення атрибуту містить вказане слово (можливо відділене пробілом) | [атрибут **~=** “значення”]  {      правила форматування  } | [**class ~= “info”**]{    color:red  } |
| Тільки для заданих тегів, для яких значення атрибуту містить вказане слово (можливо відділене пробілом) | тег [атрибут **~=** “значення”]  {      правила форматування  } | **p**[**class ~= “info”**]{    color:red  } |

|  |  |
| --- | --- |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **[class ~= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **class= “start”**>First</p>                🡨Підходить  <p class= “middle\_start\_zone”>Second </p>  <p **class= “info start**”> Third </p>         🡨Підходить  <a **class = “base start info”** href=”#”>Link</a>      🡨Підходить |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **p[class ~= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **class= “part start”**>First</p>             🡨Підходить  <p class= “middle\_start\_info”>Second </p>  <p id = “start-info”> Third </p>  <a id = “my start”href=”#”>Link</a> |

**Значення атрибуту співпадає із заданим фрагментом або починається з вказаного фрагменту як ціле слово, відділене від інших дефісом**

|  |  |  |
| --- | --- | --- |
|  | Загальний вигляд | Приклад |
| Для будь-яких тегів, для яких значення атрибуту починається з вказаного фрагменту (можливо відділене дефісом) | [атрибут **|=** “значення”]  {      правила форматування  } | [**class |= “info”**]{    color:red  } |
| Тільки для заданих тегів, для яких значення атрибуту починається з вказаного фрагменту (можливо відділене дефісом) | тег [атрибут **|=** “значення”]  {      правила форматування  } | **p**[**class |= “info”**]{    color:red  } |

|  |  |
| --- | --- |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **[class |= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p **class= “start”**>First</p>                🡨Підходить  <p class= “middle\_start\_zone”>Second </p>  <p class= “start info”> Third </p>  <a **class = “start-base info”** href=”#”>Link</a>      🡨Підходить |
| Приклад | - - - - - - - Правило CSS  - - - - - - - -  **p[id |= “start”]**{     color:red  }  - - - - - - - У файлі HTML  - - - - - - - -  <p id= **“**part start**”**>First</p>  <p class= “middle\_start\_info”>Second </p>  <p **id = “start-info”**> Third </p>          🡨Підходить  <a id = “start-link”href=”#”>Link</a> |

**Селектор псевдокласу**

***Синтаксис псевдокласів***

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| з використанням додаткового селектора | селектор : псевдоклас {     Опис правил стилю  } | A.menu **: hover** {     color: green  } |
| без використання додаткового селектора (правило буде застосовано до усіх елементів) | : псевдоклас {     Опис правил стилю  } | **: hover** {     color: green  } |

***Псевдокласи, які визначають стан елементів***

|  |  |  |
| --- | --- | --- |
|  | **Для чого застосовується** | **Приклад** |
| **:link** | Застосовується до невідвіданих посиланнях, тобто таких посиланнях, на які користувач ще не натискав | a**:link** {   color:brown  } |
| **:visited** | Застосовується до відвіданих посилань. Зазвичай таке посилання змінює свій колір за замовчуванням на фіолетовий, але за допомогою стилів колір і інші параметри можна задати самостійно. | a**:visited** {   color:red  } |

*Стан елементів*

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| :**active** | Відбувається при активації користувачем елементу. Наприклад, посилання стає активним, якщо навести на нього курсор і клацнути мишкою. Незважаючи на те, що активним може стати практично будь-який елемент сторінки, псевдоклас: active використовується переважно для посилань | a: **active**{     color:green  } |
| **: focus** | Застосовується до елемента при отриманні ним фокусу. Наприклад, для текстового поля форми отримання фокусу означає, що курсор встановлений в полі, і за допомогою клавіатури можна вводити в нього текст. Результат буде видно тільки для тих елементів, які можуть отримати фокус. Зокрема, це  <a>, <input>, <select> і <textarea>. | a: **focus**{     color:red  } |
| **:checked** | виділені (вибрані користувачем) елементи форми | **:checked{**  color:blue  **}** |
| **:hover** | активізується тоді, коли курсор миші знаходиться в межах елемента, але клацання по ньому не відбувається. Приклад таблиці, яка виділяє рядок, на який наведено курсор. | tr**:hover**{     background: #fc0; } |

*Доступність для зміни елемента*

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| **:enabled** | всі доступні для зміни (не заблоковані) елементи форми | :enabled{    color:green  } |
| **:disabled** | недоступні для зміни (заблоковані) елементи форми | :disabled{    color:green  } |

*Коректність заповнення полів форми*

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| **:valid** | поля форми, які заповнені коректно (пройшли перевірку у браузері на відповідність визначеним правилам) | **:valid**{    color:green  } |
| **:invalid** | поля форми, значення яких не відповідають визначеним для них правилам | **:invalid**{    color:green  } |
| **:in-range** | поля форми, значення яких знаходяться у заданому діапазоні | **:in-range** {    color:green  } |
| **:out-of-range** | поля форми, значення яких не знаходяться у заданому діапазоні | **: out-of-range** {    color:red  } |

*Особливі властивості елемента у документі*

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| **:target** | Адреса у вигляді - "https://website.ua/forum#**first**" вказує на певний елемент на сторінці з ідентифікатором first (id="first"). Цей елемент називається цільовим.  Селектор **:target** обирає цей цільовий елемент. | **:target**{    color:green  } |
| **:lang** | За допомогою псевдокласів можна змінювати стиль оформлення іноземних текстів, а також деякі налаштування. За допомогою псевдокласу **:lang** можна задавати певні налаштування, характерні для різних мов, наприклад, вид лапок в цитатах. В якості мови можуть виступати наступні значення: en - англійська; de - німецька; ru - російська;  fr - французька; it – італійська та ін. | **:lang(en)** {    color:blue  } |
| **:not(селектор)** | елементи, які не відповідають вказаному селектору | **:not([type="submit"])**{    color:blue  } |

*Селектори структурних псевдоелементів*

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| **:first-child** | вибирає тільки перший дочірній елемент | **:first-child**{     color:blue  } |
| **:last-child** | вибирає тільки останній дочірній елемент | **:last-child**{     color:blue  } |
| **:only-child** | вибирає дочірні елементи тільки у випадку, якщо вони є єдиними у їх батьківському елементі | **:only-child**){     color:blue  } |
| **:nth-child(odd)** | вибирає непарні дочірні елементи | :nth-child(odd){     color:blue  } |
| **:nth-child(even)** | вибирає парні дочірні елементи | :nth-child(even){     color:blue  } |
| **:nth-child(номер)** | вибір дочірнього елемент з вказаним **номером** | **:nth-child(3){**     color:blue  **}**  **третій дочірній** |
| **:nth-child(n+ номер )** | вибір усіх дочірніх елементів починаючи з вказаного **номера** | **:nth-child(n+2)**{     color:blue  }  **кожен починаючи з другого** |
| **:nth-child(число n)** | вибір елементів з номерами кратними вказаному **числу** | :nth-child(**3**n)**{**    color:blue  }  **кожен третій** |
| **:nth-child**  **(число n + номер )** | вибір елементів з номерами кратними вказаному **числу** починаючи з вказаного початкового **номера** | :nth-child(**3**n+**2**){    color:blue  }  **кожен третій починаючи з другого** |
| **:nth-last-child (число n + номер )** | аналогічний до **:nth-child**, але відлік починається з кінця | **p:nth-last-child(2)** {     color:blue  }  **другий дочірній починаючи відлік з кінця** |
| **:empty** | вибирає елементи, у яких немає дочірніх | **:empty{**  color:blue  } |
| **:root** | вибирає елемент, який є кореневим у документі – html | **:root{**    color:blue  } |

*Селектор струткурних псевдокласів типу*

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| **:nth-of-type()** | вибирає елементи по аналогії з  :nth-child(), але при цьому враховується тип дочірнього елемента | p:nth-of-type(2) **{**    color:blue  }  **Вибирає кожен елемент <p>, що є другим дочірнім <p> елемент свого батька** |
| **:nth-last-of-type()** | аналогічно до :nth-of-type() але відлік починається з кінця | p:nth-last-of-type(2) **{**    color:blue  }  **Вибирає кожен елемент <p>, що є другим дочірнім <p> елемент свого батька з кінця** |
| **:first-of-type** | вибирає перший дочірній елемент вказаного типу | p:first-of-type**{**    color:blue  } |
| **:last-of-type** | вибирає останній дочірній елемент вказаного типу | p:last-of-type**{**    color:blue  } |
| **:only-of-type** | вибирає дочірній елемент вказаного типу тільки у випадку, якщо він єдиним дочірнім елементом вказаного типу для свого предка | p:only-of-type**{**    color:blue  } |

***Селектори псевдоелеметі***

*Вибірка умовних елементів*

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| **::first-letter** | Визначає стиль першого символу в тексті **блокового** елемента, до якого додається | p::first-letter  **Обирає першу літеру кожного елементу <p>** |
| **::first-line** | Визначає стиль першого рядка **блокового** елемента | p::first-line |
| **::selection** | Селектор ::selection обирає текст, який користувач виділив.  Тільки кілька CSS властивостей можуть бути застосовані до селектору ::selection     * color (колір тексту) * background (фон тексту) * cursor (вигляд курсору) * outline (контур навколо тексту) | ::selection {   color: red;   background: yellow;  } |
| **::placeholder** | Селектор ::placeholder дозволяє змінювати стильове оформлення тексту підказки. Підсказка задається атрибутом placeholder | input[type="text"]::placeholder {     color:blue  } |

*Вставка контенту*

Для вставки контенту використовується спеціальна властивість content

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| ::before | додає вказаний контент всередину елемента (перед вмістом цього елемента) | p::before {    content:"Read this: ";  }  -----------------------  li::before {       content: "¶ ";  }  li {     list-style: none;  }  **відміняє станадртні маркери** |
| ::after | додає вказаний контент всередину елемента (після вмісту цього елемента) | p::after {    content:"Read this: ";  }  q::before {    content: "«";    color: blue;  }  q::after {    content: "»";    color: red;  } |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |

|  |  |  |
| --- | --- | --- |
| **Селектор** | **Приклад** | **Пояснення до прикладу** |
| [.class](https://css.in.ua/css/selector/class) | .intro | Вибірка всіх елементів з класом "intro". |
| [#id](https://css.in.ua/css/selector/id) | #firstname | Вибирає елемент з ідентифікатором id="firstname" |
| [\*](https://css.in.ua/css/selector/all) | \* | Обирає всі елементи |
| [element](https://css.in.ua/css/selector/element) | p | Обирає всі елементи <p>. |
| [elemen1t,element2](https://css.in.ua/css/selector/multielement) | div, p | Обирає всі елементи <div> і всі елементи <p>. |
| [element1 element2](https://css.in.ua/css/selector/inside-element) | div p | Обирає всі елементи <p> в елементі <div> |
| [element1>element2](https://css.in.ua/css/selector/element-child) | div > p | Обирає всі елементи <p> для яких батьківським елементом є елемент <div>. |
| [element1+element2](https://css.in.ua/css/selector/adjacent) | div + p | Обирає елемент <p> який розташовується одразу за елементам <div>. |
| [element1~element2](https://css.in.ua/css/selector/sibling) | p ~ ul | Обирає всі елементи <ul> які розташовані перед елементом <p> |
| [[attribute]](https://css.in.ua/css/selector/attribute) | [target] | Обирає всі елементи, у котрих вказано атрибут target. |
| [[attribute=value]](https://css.in.ua/css/selector/attribute-value) | target="\_blank" | Обирає всі елементи з target="\_blank" |
| [[attribute~=value]](https://css.in.ua/css/selector/attribute-value-contains) | [title~=flower] | Обирає всі елементи з атрибутом title в яких міститься слово flower |
| [[attribute|=value]](https://css.in.ua/css/selector/attribute-value-lang) | [lang|=en] | Обирає всі елементи з атрибутом lang значенням якого починається з "en". |
| [[attribute^=value]](https://css.in.ua/css/selector/attribute-value-begin) | a[href^="https"] | Обирає всі елементи <a> в яких значення атрибута href починається з "https". |
| [[attribute$=value]](https://css.in.ua/css/selector/attribute-value-end) | a[href$=".pdf"] | Обирає всі елементи <a> в яких значення атрибута href закінчується рядком ".pdf" |
| [[attribute\*=value]](https://css.in.ua/css/selector/attribute-value-contain) | a[href\*="w3schools"] | Обирає всі посилання в яких атрибут href містить слово "w3schools". |
| [:active](https://css.in.ua/css/selector/active) | a:active | Вибірка активного посилання. |
| [::after](https://css.in.ua/css/selector/after) | p::after | Додає що-небудь після контенту елемента <p> |
| [::before](https://css.in.ua/css/selector/before) | p::before | Додає що-небудь перед контентом елемента <p> |
| [:checked](https://css.in.ua/css/selector/checked) | input:checked | Обирає кожний елемент <input> який має значенняі checked. |
| [:disabled](https://css.in.ua/css/selector/disabled) | input:disabled | Обирає всі вимкнені елементи введення. |
| [:empty](https://css.in.ua/css/selector/empty) | p:empty | Обирає кожен елемент <p> без тексту та інших HTML елементів. |
| [:enabled](https://css.in.ua/css/selector/enabled) | input:enabled | Обирає всі увімкнені елементи введення. |
| [:first-child](https://css.in.ua/css/selector/first-child) | p:first-child | Селектор :first-child обирає всі елементи <p>, які є першим нащадком свого батька. |
| [::first-letter](https://css.in.ua/css/selector/first-letter) | p::first-letter | Обирає першу літеру кожного елементу <p>. |
| [::first-line](https://css.in.ua/css/selector/first-line) | p::first-line | Обирає перший рядок кожного елементу <p> |
| [:first-of-type](https://css.in.ua/css/selector/first-of-type) | p:first-of-type | Обирає елемент <p>, якщо він перший елемент свого батька. |
| [:focus](https://css.in.ua/css/selector/focus) | input:focus | Обирає елемент <input>, який є у фокусі |
| [:hover](https://css.in.ua/css/selector/hover) | a:hover | Обирає посилання на яке наведено курсор. |
| [:in-range](https://css.in.ua/css/selector/in-range) | input:in-range | Обирає елементи <input> із значенням, що знаходяться в дозволеному діапазоні. |
| [:invalid](https://css.in.ua/css/selector/invalid) | input:invalid | Обирає всі елементи input зі значенням, що не пройшло перевірки. |
| [:lang(language)](https://css.in.ua/css/selector/lang) | p:lang(it) | Обирає кожен елемент <p> з атрибутом lang у значенні "it" (Italian) |
| [:last-child](https://css.in.ua/css/selector/last-child) | p:last-child | Обирає елемент <p>, який є останнім нащадком свого батька. |
| [:last-of-type](https://css.in.ua/css/selector/last-of-type) | p:last-of-type | Обирає останній елемент свого батька. |
| [:link](https://css.in.ua/css/selector/link) | a:link | Обирає всі посилання по яким ще не переходили. |
| [:not(selector)](https://css.in.ua/css/selector/not) | :not(p) | Вибірка елементів, котрі не містять вказаний селектор. |
| [:nth-child(n)](https://css.in.ua/css/selector/nth-child) | p:nth-child(2) | Обирає елемент <p>, якщо він другий елемент свого батька. |
| [:nth-last-child(n)](https://css.in.ua/css/selector/nth-last-child) | p:nth-last-child(2) | Обирає кожен елемент <p>, що є другим нащадком свого батька починаючи відлік з кінця. |
| [:nth-last-of-type(n)](https://css.in.ua/css/selector/nth-last-of-type) | p:nth-last-of-type(2) | Обирає кожен елемент <p>, що є другим елементом <p> свого батька з кінця. |
| [:nth-of-type(n)](https://css.in.ua/css/selector/nth-of-type) | p:nth-of-type(2) | Вибирає кожен елемент <p>, що є другим <p> елемент свого батька. |
| [:only-of-type](https://css.in.ua/css/selector/only-of-type) | p:only-of-type | Обирає кожен елемент <p>, що є єдиним елементом <p> такого типу свого батька. |
| [:only-child](https://css.in.ua/css/selector/only-child) | p:only-child | Вибирає елемент <p>, якщо він є єдиним нащадком свого батька. |
| [:optional](https://css.in.ua/css/selector/optional) | input:optional | Вибірка елементів <input>, що без атрибута "required". |
| [:out-of-range](https://css.in.ua/css/selector/out-of-range) | input:out-of-range | Вибірка всіх <input> елементів у котрих введене значення вийшло за межі дозволеного діапазону. |
| [:read-only](https://css.in.ua/css/selector/read-only) | input:read-only | Обирає <input> елементи, котрі з атрибутом "readonly". |
| [:read-write](https://css.in.ua/css/selector/read-write) | input:read-write | Вибирає <input> елементи у яких не вказаний "readonly" атрибут. |
| [:required](https://css.in.ua/css/selector/required) | input:required | Вибірка елементів <input>, що обов'язкові для заповнення. |
| [:root](https://css.in.ua/css/selector/root) | :root | Вибирає кореневий елемент документа |
| [::selection](https://css.in.ua/css/selector/selection) | ::selection | Вибірка виділеного користувачем тексту. |
| [:target](https://css.in.ua/css/selector/target) | #news:target | Обирає поточний активний елемент, що має значення атрибута id="news" |
| [:valid](https://css.in.ua/css/selector/valid) | input:valid | Обирає всі елементи введення з допустим значенням. |
| [:visited](https://css.in.ua/css/selector/visited) | a:visited | Обирає посилання, по яким вже було здійснено перехід |
| [:default](https://css.in.ua/css/selector/default) | input:default | Обирає елемент введення форми, що без задання. |
| [::placeholder](https://css.in.ua/css/selector/placeholder) | input[type="text"]::placeholder | Змінює стиль підсказки текстового поля <input>. |

**Вирахування специфічності**

Для кожного правила браузер визначає специфічність селектора і у випадку застосування до якогось з елементів декількох правил буде використане те, яке має найбільшу специфічність. Значення специфічності розраховується з урахуванням чотирьох складових (чотирьох чисел), які могли бути зазначені при записі селектора вибору:

* вбудований стиль – 1,0,0,0;
* задання ідентифікатора – 0,1,0,0;
* задання класу, псевдокласу, атрибуту – 0,0,1,0;
* задання тегу елемента чи задання псевдоелемента – 0,0,0,1;
* універсальний селектор не враховується.

Якщо при заданні правила було використано декілька складових, то значення специфічності додаються (як чотирицифрові числа).

Приклад. Розрахунок специфічності

h1 {color: lightblue;} /\* 0, 0, 0, 1\*/

em {color: silver;} /\* 0, 0, 0, 1\*/

h1 em {color: gold;} /\*: 0, 0, 0, 1 + 0, 0, 0, 1 = 0, 0, 0, 2\*/

div#main p.about {color: blue;} /\*: 0, 0, 0, 1 + 0, 1, 0, 0 + 0, 0, 0, 1 + 0, 0, 1, 0 = 0, 1, 1, 2\*/

.sidebar {color: grey;} /\* 0, 0, 1, 0\*/

#sidebar {color: orange;} /\* 0, 1, 0, 0\*/

li#sidebar  {color: aqua;} /\* 0, 0, 0, 1 + 0, 1, 0, 0 = 0,1,0,1\*/

 У випадку задання декількох правил застосовується те, у якого специфічність є більшою (розглядаємо специфічність як чотирицифрове ціле число).

*Форматування тексту*

**inherit**-Успадковує значення батька.

**color:** колір | inherit  - Задає колір.

**background:** [[background-attachment](http://htmlbook.ru/css/background-attachment) || [background-color](http://htmlbook.ru/css/background-color) || [background-image](http://htmlbook.ru/css/background-image) || [background-position](http://htmlbook.ru/css/background-position) || [background-repeat](http://htmlbook.ru/css/background-repeat)] | inherit

**background-attachment**: fixed | scroll | inherit

fixed- Робить фонове зображення елемента нерухомим.

Scroll Дозволяє переміщатися фону разом з вмістом.

**background-color**: колір | transparent | inherit

transparent – прозорий фон

**background-image**: url(файл) | none | inherit – фонове зображення

[**background-position**](http://htmlbook.ru/css/background-position) Задає початкове положення фонового зображення, встановленого за допомогою властивості background-image. Два значення: положення по горизонталі (може бути - left, center, right) і вертикалі (може бути - top, center, bottom). Крім використання ключових слів положення також можна задавати у відсотках, пікселах або інших одиницях. Якщо застосовуються ключові слова, то порядок їх проходження не має значення, при процентній запису спочатку задається положення малюнка по горизонталі, а потім, через пропуск, положення по вертикалі.

[**background-repeat**](http://htmlbook.ru/css/background-repeat) Визначає, як буде повторюватися фонове зображення, встановлене за допомогою властивості background-image. Можна встановити повторення малюнка тільки по горизонталі, по вертикалі або в обидві сторони. Допустимо вказувати два значення, перше ключове слово задає повторення по горизонталі, друге по вертикалі.

no-repeat Встановлює одне фонове зображення в елементі без його повторень, положення якого визначається властивістю background-position Аналогічно no-repeat no-repeat.

repeat Фонове зображення повторюється по горизонталі і вертикалі. Аналогічно repeat repeat.

repeat-x Фоновий малюнок повторюється тільки по горизонталі. Аналогічно repeat no-repeat.

repeat-y Фоновий малюнок повторюється тільки по вертикалі. Аналогічно no-repeat repeat.

**font**: [font-style||font-variant||font-weight] font-size [/line-height] font-family | inherit

**font-style**: normal | italic | oblique | inherit

normal Звичайне написання тексту.

Italic Курсивне зображення.

oblique     Похиле написання. Курсив і похилий шрифт при всій їх схожості не одне і те ж. Курсив це спеціальний шрифт імітує рукописний, похилий ж утворюється шляхом нахилу звичайних знаків вправо.

**font-variant**: normal | small-caps | inherit

small-caps – Задає напис капітелем. ПРИКЛАД ТЕКСТУ

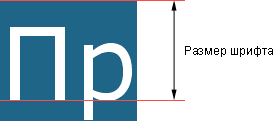
**font-weight**: bold|bolder|lighter|normal|100|200|300|400|500|600|700|800|900

Нормальне зображення шрифту (яке встановлено за умовчанням) еквівалентно 400, стандартний напівжирний текст - значенням 700.

**font-size** Визначає розмір шрифту елемента. Розмір може бути встановлений декількома способами. Набір констант (xx-small, x-small, small, medium, large, x-large, xx-large) задає розмір, який називається абсолютним. По правді кажучи, вони не зовсім абсолютні, оскільки залежать від налаштувань браузера та операційної системи.

Інший набір констант (larger, smaller) встановлює відносні розміри шрифту. Оскільки розмір успадкований від батьківського елементу, ці відносні розміри застосовуються до батьківського елементу, щоб визначити розмір шрифту поточного елементу. Розмір шрифту сильно залежить від значення властивості font-size у батьківського елемента.

Сам розмір шрифту визначається як висота від базової лінії до верхньої межі кегельного майданчики, як показано на рис.



**line-height:** Встановлює міжрядковий інтервал тексту, відлік ведеться від базової лінії шрифту. За звичайних обставин відстань між рядками залежить від виду та розміру шрифту і визначається браузером автоматично. Негативне значення міжрядкового відстані не допускається.

**font-family** Встановлює сімейство шрифту, яке використовуватиметься для оформлення тексту вмісту. Список шрифтів може включати одне або кілька назв, розділених комою. Якщо в імені шрифту містяться прогалини, наприклад, Trebuchet MS, його необхідно  брати в лапки.

Коли браузер зустрічає перший шрифт у списку, він перевіряє його наявність на комп'ютері користувача. Якщо такого шрифту немає, береться наступне ім'я зі списку і також аналізується на присутність. Тому кілька шрифтів збільшує ймовірність, що хоча б один з них буде виявлений на клієнтському комп'ютері. Закінчують список зазвичай ключовим словом, яке описує тип шрифту - serif, sans-serif, cursive, fantasy або monospace. Таким чином, послідовність шрифтів краще починати з екзотичних типів і закінчувати узагальненим ім'ям, яке задає вид накреслення.

***Приклад***  p { font: bold italic small-caps 12px/14px Forte,”Times New Roman”,Serif; }

**text-align:** center | justify | left | right | inherit – Вирівнювання тексту

**text-decoration**: [ blink | line-through | overline | underline ] | none | inherit – Додає оформлення тексту у вигляді його підкреслення, перекреслення, лінії над текстом і миготіння. Одночасно можна застосувати більше одного стилю, перераховуючи значення через пробіл.

**text-transform**: capitalize | lowercase | uppercase | none | inherit - Керує перетворенням тексту елемента

capitalize     Перший символ кожного слова буде великим. Інші символи свій вигляд не міняють.  
lowercase     Всі символи перетворюються в нижній регістр.  
uppercase     Всі символи перетворюються у верхній регістр.

**text-indent:** Встановлює величину відступу першого рядка блоку тексту (наприклад, для абзацу <p>). Впливу на всі інші рядки не відбувається. Допускається від'ємне значення для створення виступу першого рядка, але слід перевірити, щоб текст не виходив за межі вікна браузера.

**letter-spacing:** Визначає інтервал між символами в межах елемента. Браузери зазвичай встановлюють відстань між символами, виходячи з типу і виду шрифту, його розмірів і настройок операційної системи. Щоб змінити це значення і застосовується дана властивість. Допустимо використовувати від'ємне значення, але в цьому випадку треба переконатися, що зберігається читабельність тексту.

**word-spacing:** Встановлює інтервал між словами. Якщо для тексту задано вирівнювання по ширині через text-align, то властивість word-spacing ігнорується.

**list-style**: list-style-type || list-style-position || list-style-image | inherit

Комбінації значень повинні слідувати в зазначеному порядку: спочатку йде тип маркера, потім положення і картинка. Жодне значення не є обов'язковим, тому невживані можна опустити.

**list-style-type** Змінює вигляд маркера для кожного елемента списку. Ця властивість використовується тільки у випадку, коли значення list-style-image встановлено як none. Маркери розрізняються для маркірованого списку (тег <ul>) і нумерованого (тег <ol>).

Circle Маркер у вигляді кружка.

disc у вигляді точки.

square  у вигляді квадрата.

нумерований список

decimal Арабські числа (1, 2, 3, 4, ...).

lower-alpha Рядкові латинські літери (a, b, c, d, ...).

lower-greek Рядкові грецькі літери (α, β, γ, δ, ...).

lower-roman Римські числа в нижньому регістрі (i, ii, iii, iv, v, ...).

upper-alpha Великі латинські літери (A, B, C, D, ...).

upper-roman Римські числа в верхньому регістрі (I, II, III, IV, V, ...).  
none Скасовує маркери для списку.

**list-style-position:** inside | outside

Визначає, як буде розміщуватися маркер щодо тексту. Є два значення: outside - маркер винесений за межі елемента списку і inside - маркер обтікаєтся текстом.

**list-style-image:** Вказується шлях до зображення-маркеру.

**float:** left | right | none | inherit

Визначає, по якій стороні буде вирівнюватися елемент, при цьому інші елементи будуть обтікати його з інших сторін. Коли значення властивості float дорівнює none, елемент виводиться на сторінці як зазвичай, при цьому допускається, що один рядок тексту може бути на тій же лінії, що і сам елемент.

Щоб скасувати обтікання, треба після плаваючого елементу розташувати блок з властивістю clear: both;

**border:** [[border-width](http://htmlbook.ru/css/border-width) || [border-style](http://htmlbook.ru/css/border-style) || [border-color](http://htmlbook.ru/css/border-color)] | inherit

Універсальна властивість border дозволяє одночасно встановити товщину, стиль і колір кордону навколо елемента. Значення можуть йти в будь-якому порядку, розділяючись пробілом, браузер сам визначить, яке з них відповідає потрібному властивості. Для установки кордону тільки на певних сторонах елемента, скористайтеся властивостями border-top, border-bottom, border-left, border-right.

**border-width** - задає товщину межі. Зазвичай вказується в пікселях, але також можна вказувати ключовими словами thin (2px), medium (4px) і thick (6px).  
**border-color** - визначає колір межі.

**border-style** - визначає стиль межі. Вони можуть бути наступними:



**margin**. Встановлює величину відступу від краю елемента. Відступом є простір від кордону поточного елемента до внутрішньої межі його батьківського елементу або до найближчого елемента. Розмірність може бути як в значеннях довжини (px, pt, em і т.д.), так і в значенні auto.

**padding** встановлюють значення полів навколо вмісту елементу. Полем називається відстань від внутрішнього краю рамки елемента до уявного прямокутника, що обмежує його вміст.

Щоб встановлювати відступи і поля відповідно зверху, справа, знизу, зліва потрібно використовувати ключові слова top, right, bottom, left.

Наприклад: margin-top: 5px; або padding-bottom: 25px;

Значення зовнішніх відступів в деяких випадках може бути і від’ємним.

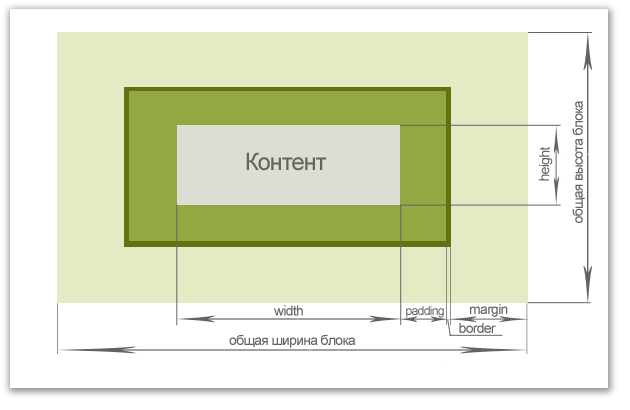
Існує скорочений запис:

margin: 5px; - застосовується до всіх полів;

margin: 5px 10px; - перше число означає відступ знизу і зверху, друге - ліворуч і праворуч;

margin: 5px 10px 4px; - перше число означає відступ зверху, друге - ліворуч і праворуч, третє - знизу;

margin: 5px 10px 4px 6px; - по порядку - зверху, справа, знизу, зліва.



**position**: absolute | fixed | relative | static | inherit Встановлює спосіб позиціонування елемента щодо вікна браузера або інших об'єктів на веб-сторінці.

**absolute**    Вказує, що елемент абсолютно позиціонується, при цьому інші елементи відображаються на веб-сторінці немов абсолютно позиціонувати елемента і немає. Положення елемента задається властивостями left, top, right і bottom, також на положення впливає значення властивості position батьківського елементу. Так, якщо в батька значення position встановлено як static або батька немає, то відлік координат ведеться від краю вікна браузера. Якщо у батька значення position задано як fixed, relative або absolute, то відлік координат ведеться від краю батьківського елементу.

**fixed**По своїй дії це значення близьке до absolute, але на відміну від нього прив'язується до зазначеної властивостями left, top, right і bottom точці на екрані і не міняє свого положення при прокрутці веб-сторінки.

**relative**   Положення елемента встановлюється щодо його вихідного місця. Додавання властивостей left, top, right і bottom змінює позицію елемента і зрушує його в ту або іншу сторону від первісного розташування.

**static**    Елементи відображаються як зазвичай. Використання властивостей left, top, right і bottom не призводить до якихось результатів.

**display**: block | inline | inline-block | inline-table | list-item | none | run-in | table | table-caption | table-cell | table-column-group | table-column | table-footer-group | table-header-group | table-row | table-row-group

багатоцільове властивість, яка визначає, як елемент повинен бути показаний в документі.

**block** Елемент показується як блоковий. Застосування цього значення для вбудованих елементів, наприклад тега <span>, змушує його вести подібно блокам - відбувається перенесення рядків на початку і в кінці вмісту.

**inline** Елемент відображається як вбудований. Використання блокових тегів, таких як <div> і <p>, автоматично створює перенос і показує вміст цих тегів з нового рядка. Значення inline скасовує цю особливість, тому вміст блокових елементів починається з того місця, де закінчився попередній **елемент.  
inline-block** Це значення генерує блоковий елемент, який обтекаєтся іншими елементами веб-сторінки подібно вбудованому елементу. Фактично такий елемент по своїй дії схожий на вбудовувані елементи (зразок тега <img>). При цьому його внутрішня частина форматується як блоковий елемент, а сам елемент - як вбудований.

**inline-table** Визначає, що елемент є таблицею як при використанні тега <table>, але при цьому таблиця є вбудованим елементом і відбувається її обтікання іншими елементами, наприклад, текстом.  
**list-item** Елемент виводиться як блоковий і додається маркер списку.

**none** Тимчасово видаляє елемент із документа. Займане ним місце не резервується і веб-сторінка формується так, немов елемента і не було. Змінити значення і зробити знову видимим елемент можна за допомогою скриптів, звертаючись до властивостей через об'єктну модель. У цьому випадку відбувається переформатування даних на сторінці з обліком знову доданого елемента.

**run-in** Встановлює елемент як блоковий або вбудований в залежності від контексту.

**table** Визначає, що елемент є блоковою таблицею подібно використанню тега <table>.

**table-caption** Задає заголовок таблиці подібно застосуванню тега <caption>.

**table-cell** Вказує, що елемент являє собою комірку таблиці (тег <td> або <th>).

**table-column** Призначає елемент колонкою таблиці, немов був доданий тег <col>.

**table-column-group** Визначає, що елемент є групою однієї чи більше колонок таблиці, як при використанні тега <colgroup>.

**table-footer-group** Використовується для зберігання однієї або кількох рядків осередків, які відображаються в самому низу таблиці. По своїй дії схоже з роботою тега <tfoot>.

**table-header-group** Елемент призначений для зберігання однієї або кількох рядків осередків, які представлені вгорі таблиці. По своїй дії схоже з роботою тега <thead>.

**table-row** Елемент відображається як рядок таблиці (тег <tr>).

**table-row-group** Створює структурний блок, що складається з декількох рядків таблиці аналогічно дії тега <tbody>.

**Задання розміру**

Ширина всього простору, що займає блок складається з ширини контенту (займає весь достуний простір рядка або задаємо за допомогою властивості width), внутрішнього відступу (відстань між контентом і рамкою) та зовнішніх відступів до інших елементів.

Висота аналогічно складається з висоти контенту (властивість height), внутрішнього відступу від контенту до рамки (padding), та зовнішнього відступу до інших елементів (margin)

Приклад 1.

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapper"**>     <**p**>       ipsum dolor sit amet,   consectetur adipiscing elit. Nam diam nulla, consectetur in purus sit amet, auctor dignissim nibh. venenatis. Nunc     </**p**>  <**p**> interdum varius orci. Donec eget volutpat urna. Duis ut odio ut leo ultrices tempus. Nunc massa massa quis maximus aliquot  </**p**>  </**div**> | .**wrapper**{      **border**: **black solid** 1**px**;  } |
|  | |

Додамо рамку для абзаців (також блочних елементів)

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapper"**>     <**p**>       ipsum dolor sit amet,  consectetur adipiscing elit. Nam diam nulla, consectetur in purus sit amet, auctor dignissim nibh. venenatis. Nunc     </**p**>  <**p**> interdum varius orci. Donec eget volutpat urna. Duis ut odio ut leo ultrices tempus. Nunc massa massa quis maximus aliquot  </**p**>  </**div**> | .**wrapper**{      **border**: **black solid** 1**px**;  }  **p**{      **border**: **black solid** 2**px**;  } |
|  | |

Додамно внутрішній відступ для зовнішнього блоку ( з класом «wrapper»)

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapper"**>     <**p**>       ipsum dolor sit amet,  consectetur adipiscing elit. Nam diam nulla, consectetur in purus sit amet, auctor dignissim nibh. venenatis. Nunc     </**p**>  <**p**> interdum varius orci. Donec eget volutpat urna. Duis ut odio ut leo ultrices tempus. Nunc massa massa quis maximus aliquot  </**p**>  </**div**> | .wrapper{      border: black solid 1px;     **padding: 30px;** }  p{      border: black solid 2px;  } |
|  | |

Додамо зовнішні відступи для зовнішнього блоку

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapper"**>     <**p**>       ipsum dolor sit amet,  consectetur adipiscing elit. Nam diam nulla, consectetur in purus sit amet, auctor dignissim nibh. venenatis. Nunc     </**p**>  <**p**> interdum varius orci. Donec eget volutpat urna. Duis ut odio ut leo ultrices tempus. Nunc massa massa quis maximus aliquot  </**p**>  </**div**> | .wrapper{      border: black solid 1px;     padding: 30px;  **margin: 40px;**  }  p{      border: black solid 2px;  } |
|  | |

Задамо розміри для абзаців (width, height) (можна задавати тільки для блочних елементів)

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapper"**>     <**p**>       ipsum dolor sit amet,  consectetur adipiscing elit. Nam diam nulla, consectetur in purus sit amet, auctor dignissim nibh. venenatis. Nunc     </**p**>  <**p**> interdum varius orci. Donec eget volutpat urna. Duis ut odio ut leo ultrices tempus. Nunc massa massa quis maximus aliquot  </**p**>  </**div**> | .wrapper{      border: black solid 1px;     padding: 30px;      margin: 40px;  }  p{      border: black solid 2px;  **width:300px;**  **height: 200px;**  } |
|  | |

При заданні висоти контент може не поміщатися всередині заданих меж блоку і спосіб відображення того контенту, який не поміщається можна задати за допомогою властивості overflow

|  |  |  |
| --- | --- | --- |
| overflow:hidden  **контент, який не поміщається не відображається** | overflow:scroll  **додається полоса прокрутки** | overflow:visible  **контент виходить за межі блоку** |
|  |  |  |

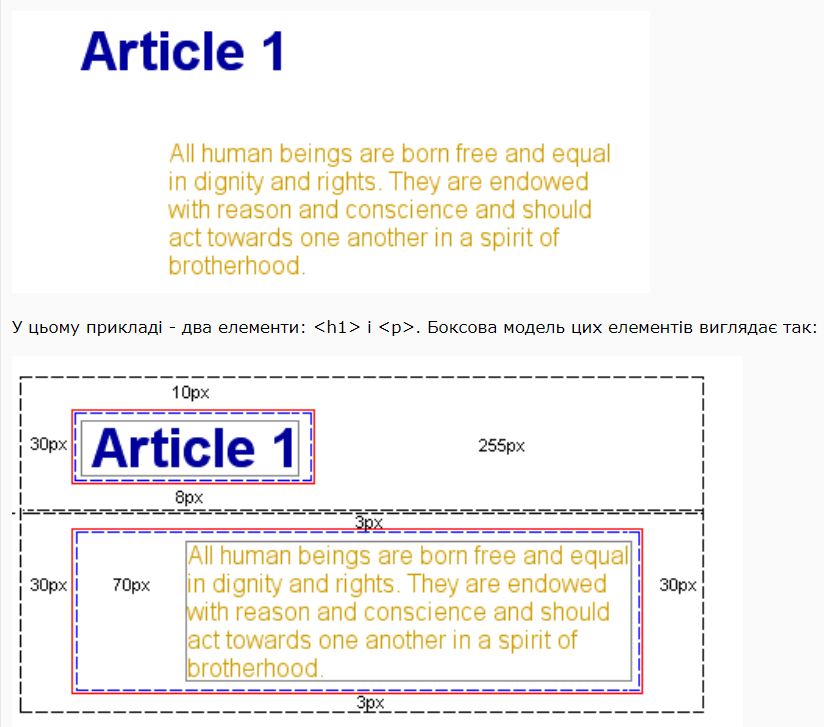
Вирівняємо контент по центру по горизонталі (відповідні розміри (зліва, справа) для  margin потрібно задати **auto**

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapper"**>     <**p**>       ipsum dolor sit amet,  consectetur adipiscing elit. Nam diam nulla, consectetur in purus sit amet, auctor dignissim nibh. venenatis. Nunc     </**p**>  <**p**> interdum varius orci. Donec eget volutpat urna. Duis ut odio ut leo ultrices tempus. Nunc massa massa quis maximus aliquot  </**p**>  </**div**> | .wrapper{      border: black solid 1px;     padding: 30px;      margin: 40px;  }  p{      border: black solid 2px;      width:300px;      height: 200px; **margin: 0 auto;** } |
|  | |

Вирівняємо контент всередині абзаців по центру

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapper"**>     <**p**>       ipsum dolor sit amet,  consectetur adipiscing elit. Nam diam nulla, consectetur in purus sit amet, auctor dignissim nibh. venenatis. Nunc     </**p**>  <**p**> interdum varius orci. Donec eget volutpat urna. Duis ut odio ut leo ultrices tempus. Nunc massa massa quis maximus aliquot  </**p**>  </**div**> | .wrapper{      border: black solid 1px;     padding: 30px;      margin: 40px;  }  p{      border: black solid 2px;      width:300px;      height: 200px; margin: 0 auto;  **text-align: center;** } |
|  | |

Приклад.

  margin:10px 255px 8px 30pxЗа замовчуванням усі елементи розташовуються у основному потоці (послідовності) елементів і поділяються на блочні і рядкові. Змінити спосіб відображення можна за допомогою властивості display. Розглянемо деякі із їх властивостей.

За замовчуванням (display:static)

При використанні значення static для блока предка блочні дочірні елементи займають весь доступний простір рядка і розташовуються один під одним. Рядкові ж елементи слідують один за одним у рядку.

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapp"**>      <**div**>           div1      </**div**>      <**div**>           div2      </**div**>      <**span**>          span1      </**span**>      <**span**>          span2      </**span**>    </**div**> | **div div**, **span**{  **border**:**black solid** 1**px**;  **margin**: 10**px**;  } |
|  | |

Перетворення рядкового у блочний (display:block)

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapp"**>      <**div**>           div1      </**div**>      <**div**>           div2      </**div**>      <**span**>          span1      </**span**>      <**span**>          span2      </**span**>    </**div**> | **div div**{  **border**:**black solid** 1**px**;  **margin**: 10**px**;  }  **span**{  **border**:**black solid** 1**px**;  **margin**: 10**px**;  **display**: **block**;  } |
|  | |

Перетворення блочного у рядковий (display:inline, display:inline-block)

|  |  |
| --- | --- |
| HTML | CSS |
| <**div class="wrapp"**>      <**div**>           div1      </**div**>      <**div**>           div2      </**div**>      <**span**>          span1      </**span**>      <**span**>          span2      </**span**>    </**div**> | **div div**{  **border**:**black solid** 1**px**;  **margin**: 10**px**;  **display**: **inline**;    }  **span**{  **border**:**black solid** 1**px**;  **margin**: 10**px**;  } |
|  | |

**ПОЗИЦІОНУВАННЯ**

За замовчуванням елементи на сторінці розташовуються згідно їх опису у основному потоці (один за одним).

Є декілька підходів до позиціонування елементів на сторінці. Розглянемо деякі із них:

* з викорстанням властивості position;
* з використанням властивості float;
* з використаннм flex-блоків;

**Використання position**

Властивість  position дозволяє змінити позицію елемента зі збереженням місця розташування і без збереження місця розташування у основному потоці. Розглянемо можливі значення цієї властивості:

* static – за замовчуванням елемент залишається на своєму місці у потоці;
* relative – змінює своє значення відносно поточного зі збереженням простору, який він займав у потоці;
* absolute – змінює своє значення відносно основного вікна або ж найближчого предка з властивістю position:relative чи absolute (втрачає місце у основному потоці)
* fixed – дозволяє зафіксувати позицію елемента на екрані і не змінювати її при прокручуванні (втрачає місце у основному потоці).

position:relative

При використанні даної позиції залишається місце під елемент у основному потоці і є можливість змістити його відносно початкового полощення. При цьому необхідно задати величину зміщення по осям за допомогою властивостей :

top – відстань від верхнього краю;

bottom – відстань від нижнього краю;

left– відстань від лівого краю;

right– відстань від правого краю.

Приклади. Розглянемо приклади зміщення на фрагменті

|  |  |  |
| --- | --- | --- |
| До зміщення |  |  |
|  | <**div class="item1"**>     div1 </**div**>      <**div class="item2"**>     div2 </**div**>    <**div class="item3"**>     div3 </**div**> | .**wrapp div**{     **border**:**black solid** 1**px**;     **width**: 150**px**;     **height**: 100**px**;     **margin**: 50**px**; } .**item1**{     **background**: **blue**; } .**item2**{     **background**: **purple**; } .**item3**{     **background**: **green**; } |

Приклади зміщення

|  |  |
| --- | --- |
| Вигляд на екрані | Код CSS |
|  | .**item1**{     **background**: **blue**; } .**item2**{     **background**: **purple**;     **position**: **relative**;     **left**:50**px**; } .**item3**{     **background**: **green**; } |
|  | .**item1**{     **background**: **blue**; } .**item2**{     **background**: **purple**;     **position**: **relative**;     **right**:50**px**; } .**item3**{     **background**: **green**; } |
|  | .**item1**{     **background**: **blue**; } .**item2**{     **background**: **purple**;     **position**: **relative**;     **top**:50**px**; } .**item3**{     **background**: **green**; } |
|  | .**item1**{     **background**: **blue**; } .**item2**{     **background**: **purple**;     **position**: **relative**;     **left**:40**px**;     **top**:50**px**; } .**item3**{     **background**: **green**; } |
|  | .**item1**{     **background**: **blue**; } .**item2**{     **background**: **purple**;     **position**: **relative**;     **bottom**:50**px**; } .**item3**{     **background**: **green**; } |
|  | .**item1**{     **background**: **blue**; } .**item2**{     **background**: **purple**;     **position**: **relative**;  **right**: 40**px**;  **bottom**:50**px**;     } .**item3**{     **background**: **green**; } |

Як бачимо, елемент зміщується, але місце, яке він займав у потоці залишається.



<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
    <**meta charset="UTF-8"**>  
    <**title**>Title</**title**>  
    <**style**>  
        .**w\_out**{  
            **background**: **lightskyblue**;  
            **width**:300**px**;  
            **padding-bottom**: 100**px**;  
            **padding-top**: 20**px**;  
            **border**:**black solid** 1**px**;  
            **border-radius**: 10**px**;  
        }  
        .**w\_in**{  
            **background**: **black**;  
            **border**:**black solid** 1**px**;  
            **border-radius**: 10**px**;  
            **margin-left**: 20**px**;  
            **margin-right**: 20**px**;  
            **padding-left**: 20**px**;  
            **padding-right**: 20**px**;  
        }  
        .**w\_in div**{  
            **text-align**: **center**;  
            **color**: **white**;  
        }  
        .**text2**{  
            **border**: **white solid** 1**px**;  
            **border-radius**: 10**px**;  
            **position**: **relative**;  
            **top**:30**px**;  
            **background**: **black**;  
        }  
  
    </**style**>  
</**head**>  
<**body**>  
<**div class="w\_out"**>  
    <**div class="w\_in"**>  
        <**div class="text1"**>  
            <**h1**>HTML</**h1**>  
            <**h2**>(sdgs  sdfg sd dfh dfh)</**h2**>  
            <**p**> dsfasdf aiusfh sav</**p**>  
        </**div**>  
        <**div class="text2"**>  
            <**p**>"dfsdfsdfds"</**p**>  
            <**p**>fsdfsdfdsf</**p**>  
            <**p**>fsdfsdfdsf</**p**>  
            <**p**>fsdfsdfdsf</**p**>  
        </**div**>  
    </**div**>  
</**div**>  
</**body**>  
</**html**>

position:absolute

Значення використовується тоді, коли потрібно спозиціонувати елемент-нащадок відносно деякого предка-контейнера (найближчого, який має властивість position:relative чи absolute (втрачає місце у основному потоці).

|  |  |
| --- | --- |
| Вигляд на екрані | Код CSS |
|  | .**wrapp**{     **margin**: 50**px**;     **position**: **relative**;     **overflow**: **hidden**; } .**item1**{     **background**: **blue**; } .**item2**{     **background**: **purple**;     **position**: **absolute**;     **left**: 40**px**;     **top**:50**px**; } .**item3**{     **background**: **green**; } |

position:fixed

Використовується тоді, коли необхідно задати положення фіксованого елемента на сторінці, який не змінює свого відображення під час прокрутки.

|  |  |
| --- | --- |
| На екрані | CSS |
|  | .**item1**{     **background**: **blue**; } .**item2**{     **background**: **purple**;     **position**: **fixed**;     **right**: 40**px**;     **top**:50**px**; } .**item3**{     **background**: **green**; } |

|  |  |
| --- | --- |
| На екрані | CSS |
|  | .**item1**{     **background**: **blue**;     **float**: **left**;     **height**: 100**px**;     **width**: 150**px**; } .**item2**{     **background**: **purple**;     **float**: **right**;     **height**: 100**px**;     **width**: 150**px**; } .**item3**{     **background**: **green**;  } |

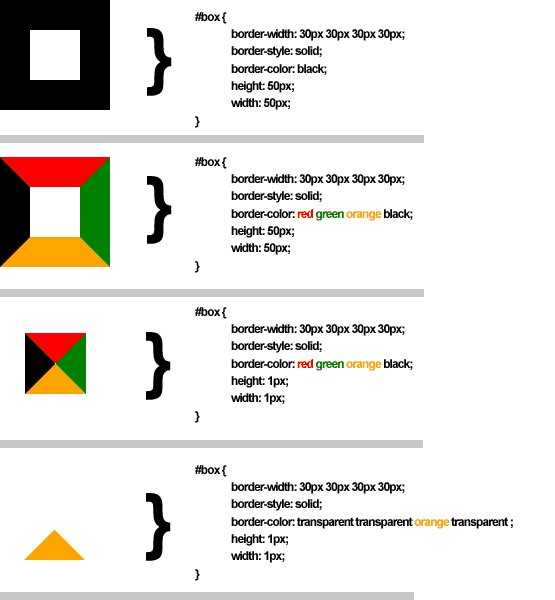
|  |  |
| --- | --- |
| **float** | |
| Значення |  |
| none | За замовчуванням. Нічого не змінює |
| left | Позиціонування по лівому краю блока-предка |
| right | Позиціонування по правому краю блока-предка |
| initial | Встановлення значення за замовчуванням |
| inherit | Успадкування значення від блоку-предка |

Відміна обтікання

Відміна обтікання здійснюється з викорстанням властивості clear

|  |  |
| --- | --- |
| **Можливі значення clear** | |
| Значення |  |
| left | Відміняє обтікання зліва |
| right | Відміняє обтікання справа |
| both | Відміняє обтікання з обох сторін |
| none | Значення за замовчуванням. Відміняє обтікання елементів однієї групи |
| initial | Встановлює значення за замовчуванням |
| inherit | Успадковує значення від предка |

|  |  |
| --- | --- |
| На екрані | CSS |
|  | .**item1**{     **background**: **blue**;     **float**: **left**;     **height**: 100**px**;     **width**: 150**px**;  } .**item2**{     **background**: **purple**;     **float**: **right**;     **height**: 100**px**;     **width**: 150**px**; } .**item3**{     **background**: **green**;     **overflow**: **hidden**; } |



CSS3 flexbox

**CSS flexbox** *(Flexible Box Layout Module)* — модуль гнучкого контейнера для позиціонування елементів.

Flexbox складається з **flex-контейнера** — контейнера-предка і **flex-элементів** — дочірних блоків.



Дочірні елементи можна розмістити у рядок чи стовбець, а інший вільний простір розподіляється між елементами деяким заданим способом.

Модуль flexbox дозволяє розв’язати такі задачі:

* розташувати елементи у одному із напрямків: зліва направо, справа наліво, зверху вниз, знизу вверх;
* перевизначати порядок виведення елементів;
* автоматично визначати розміри елементів, для того, щоб вони були виведені у доступній області;
* розв’язувати задачу горизонтального і вертикального вирівнювання;
* переносити елементи всередині контейнера, не допускаючи їх переповнення;
* створювати колонки однакової висоти;
* створювати притиснутий до низу сторінки footer.

В основі flexbox лежить ідея осей. Flexbox є инструментом двовимірної компоновки і використовує для роботи дві осі – горизонтальну  і вертикальну.

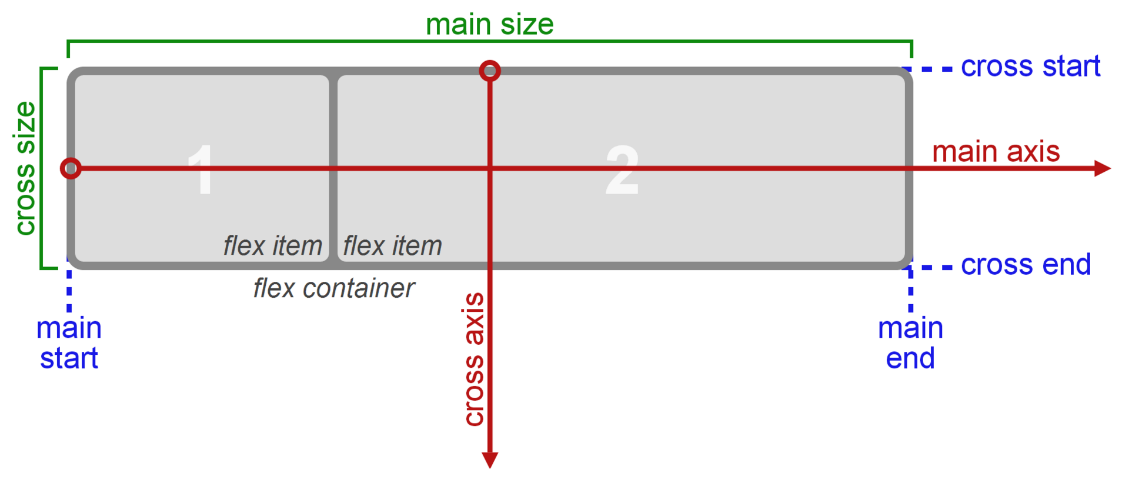


РИС. 1. МОДЕЛЬ FLEXBOX

Що таке flexbox

Підтримка браузерами

***IE:****11.0, 10.0 -ms-****Firefox:****28.0, 18.0 -moz-****Chrome:****29.0, 21.0 -webkit-****Safari:****6.1 -webkit-****Opera:****12.1 -webkit-****iOS Safari:****7.0 -webkit-****Opera Mini:****8****Android Browser:****4.4, 4.1 -webkit-****Chrome for Android:****44*

1. Властивості flex-контейнера

Flex-контейнер встановлює новий гнучкий контекст форматування для його контенту. Flex-контейнер не є блочним контейнером, тому для внутрішніх блоків не працюють такі властивості CSS, как float, clear, vertical-align. Також, на flex-контейнер не впливають властивості column-, які створюють стовпці в тексті і псевдоелементи ::first-line и ::first-letter.

1.1. Властивість display

Модель flexbox-розмітки пов’язана з вказаним значенням CSS-властивості display блока-контейнера html-елемента, який містить всередині себе дочірні елементи. Для керування елементами за допомогою цієї моделі потрібно вставити значення властивості display таким чином:

.flex-container {

display: -webkit-flex;

display: flex; /\*отображает контейнер как блочный элемент\*/

}

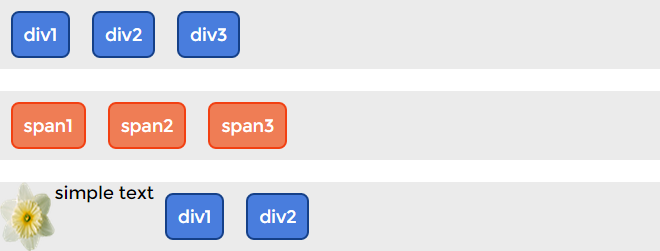
.flex-container {

 display: -webkit-inline-flex;

 display: inline-flex; /\*отображает контейнер как строчный элемент\*/

}

Після встановлення значення кожен дочірній елемент автоматично перетворюється в flex-елемент, і елементи розташовуються вздовж заданої основної осі. Усі колонки стають однакової висоти, що дорівнює висоті блоку контейнера. При цьому блочні і рядкові дочірні елементи поводяться себе однаково (ширина блоків дорівнює ширині їх контенту з врахування внутрішніх полів і рамок елемента).

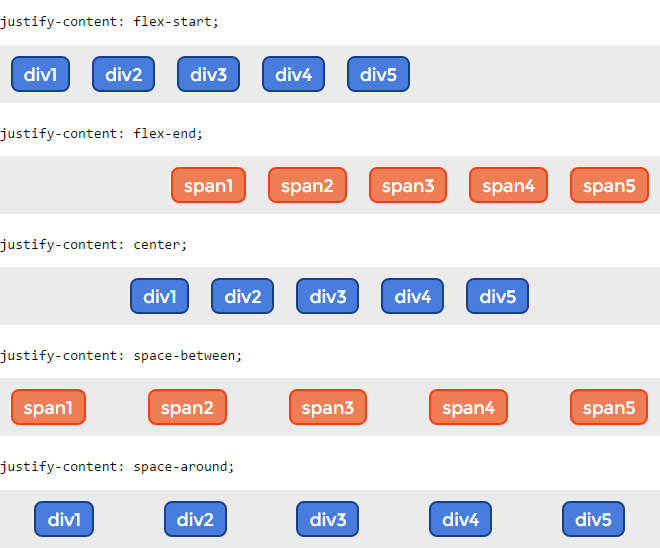
РИС. 2. Вирівнювання елементів в моделі FLEXBOX

Якщо блок-контейнер містить текст чи зображення без обгорток, то вони перетворюються в анонімні flex-елементи. Текст вирівнюється по верхньому краю блока-контейнера, а висота зображення дорівнюватиме висоті блоку (тобто деформується).

1.2. Вирівнювання дочірніх елементів по горизонталі justify-content

Властивість дозволяє вирівнювати flex-елементи по ширині flex-контейнера, розподіляючи вільний простір, незайнятий flex-елементами. Для вирівнювання елементів по вертикалі використовується властивість align-content. Властивість не успадковується.

|  |  |
| --- | --- |
| **justify-content** | |
| Значения: |  |
| flex-start | Значення за замовчуванням. Flex-елементи позиціонуються від початку flex-контейнера. |
| flex-end | Flex-элементи позиціонуються відносно правої границі flex-контейнера. |
| center | Flex-элементи виравнюються по центру flex-контейнера. |
| space-between | Flex-елементи вирівнюються по головній осі, вільний простір між ними розподідяється так:прший блок розташовується на початку flex-контейнера, останній блок – в кінці, всі інші блоки рівромірно розташовуються у просторі, що залишився. |
| space-around | Flex-елементи вирівнюються по головній осі, а вільний простір ділиться порівну, додаючи відступи зліва і справа. |
| initial | Встановлюються значення за замовчуванням. |
| inherit | Успадковується від предка. |

РИС. 3. Вирівнювання елементів по горизонталі JUSTIFY-CONTENT

**Синтаксис**

.flex-container {

display: -webkit-flex;

-webkit-justify-content: flex-start;

display: flex;

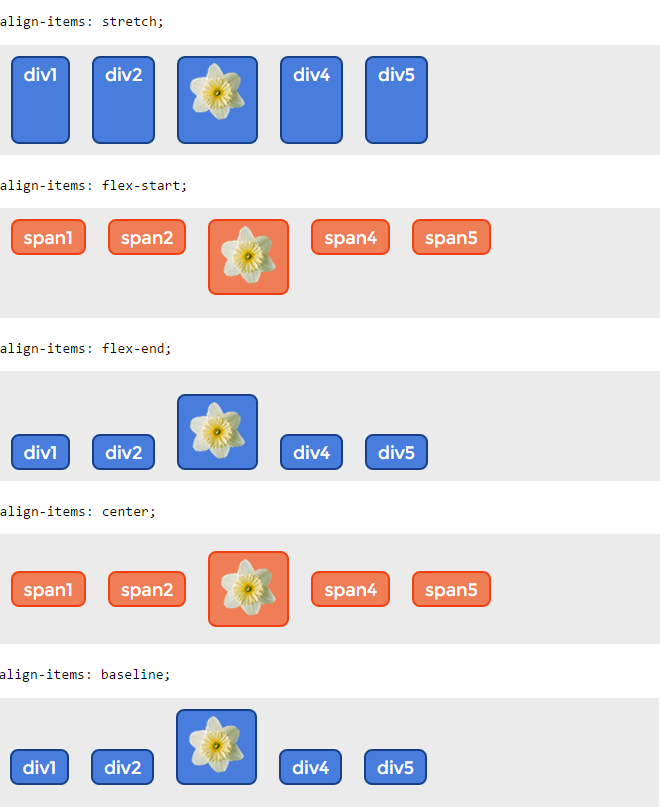
justify-content: flex-start;

}

1.3. Вирівнювання елементів по вертикалі align-items

Властивість вирівнює flex-елементи по верикальній осі (перпендикулярній до головної). Не успадковується.

|  |  |
| --- | --- |
| **align-items** | |
| Значения: |  |
| stretch | Значення за замовчуванням. Flex-елементи розтягуються, займаючи весь простір по висоті. |
| flex-start | Flex-елементи вирівнюються по верхньому краю края блока-контейнера. |
| flex-end | Flex-елементи вирівнюються по нижньому краю flex-контейнера. |
| center | Flex-елементи вирівнюються по центру flex-контейнера. |
| baseline | Flex-элементы вирівнюються по базовій лінії. |
| initial | Встановлюється значення за замовчуванянм. |
| inherit | Успадковується від предка. |

РИС. 4. Вирівнювання елементів по вертикалі

**Синтаксис**

.flex-container {

display: -webkit-flex;

-webkit-align-items: flex-start;

display: flex;

align-items: flex-start;

}

1.4. Напрямок головної осі

За замовчуванням вісь іде зліва направо. Властивість не успадковується.

|  |  |
| --- | --- |
| **flex-direction** | |
| Значения: |  |
| row | Значення за замовчуванням, елементи у рядку зліва направо |
| row-reverse | Елементи у рядку справа наліво. |
| column | Елементи у стовпці зверху вниз. |
| column-reverse | Елементи у стовпці знизу вверх. |
| initial | Встановлення значення за замовчуванням. |
| inherit | Успадковується від предка. |



РИС. 5. Значення властивості flex-direction

**Синтаксис**

.flex-container {

display: -webkit-flex;

-webkit-flex-direction: row-reverse;

display: flex;

flex-direction: row-reverse;

}

1.5. Багаторядковість елементів flex-wrap

За замовчуванням flex-елементи розташовуються в один рядок. При переповненні контейнера вміст контейнера буде виходити за межі границь flex-елементів. Використовуючи цю властивість можна дозволити елементам переходити на новий рядок.

|  |  |
| --- | --- |
| **flex-wrap** | |
| nowrap | Значення за замовчуванням. Flex-елементи не переносяться на новий рядок, розташовуючись один за одним. |
| wrap | Flex-елементи переносяться, розташовуючись у декілька горизонтальних рядів. |
| wrap-reverse | Flex-елементи переносяться, розміщуючись рядками знизу вверх. |
| initial | Встановлює значення за замовчуванням. |
| inherit | Успадковує значення предка. |

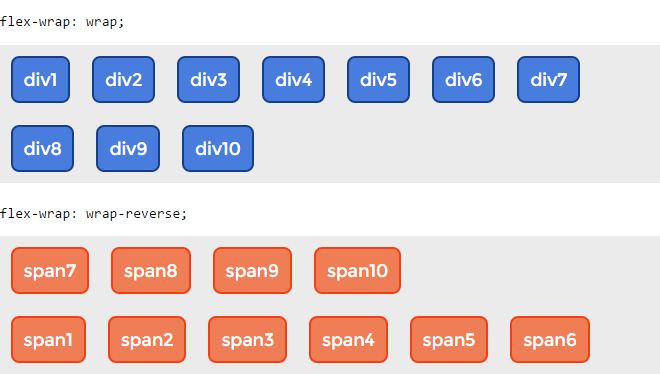


РИС. 6. Керування переносами елементів на новий рядок flex-wrap

**Синтаксис**

.flex-container {

display: -webkit-flex;

-webkit-flex-wrap: wrap;

display: flex;

flex-wrap: wrap;

}

1.6. Короткий запис напрямку і багаторядковості

|  |  |
| --- | --- |
| **flex-flow** | |
| напрямок | Вказує напрямок головної осі. За замовчуванням row. |
| багаторядковість | Задає багаторядковість поперечної осі. Значення за замовчуванням nowrap. |
| initial | Встановлюється значення за замовчуванням. |
| inherit | Успадковується від предка. |

**Синтаксис**

.flex-container {

display: -webkit-flex;

-webkit-flex-flow: row wrap;

display: flex;

flex-flow: row wrap;

}

1.7. Багаторядкове вирівнювання align-content

Властивість вирівнює рядки flex-елементів по вертикалі в flex-контейнері. Застосовується тільки у випадку, коли задано напрямок flex-flow: row/row-reverse/column/column-reverse wrap/wrap-reverse; і висота flex-контейнера. Не успадковується.

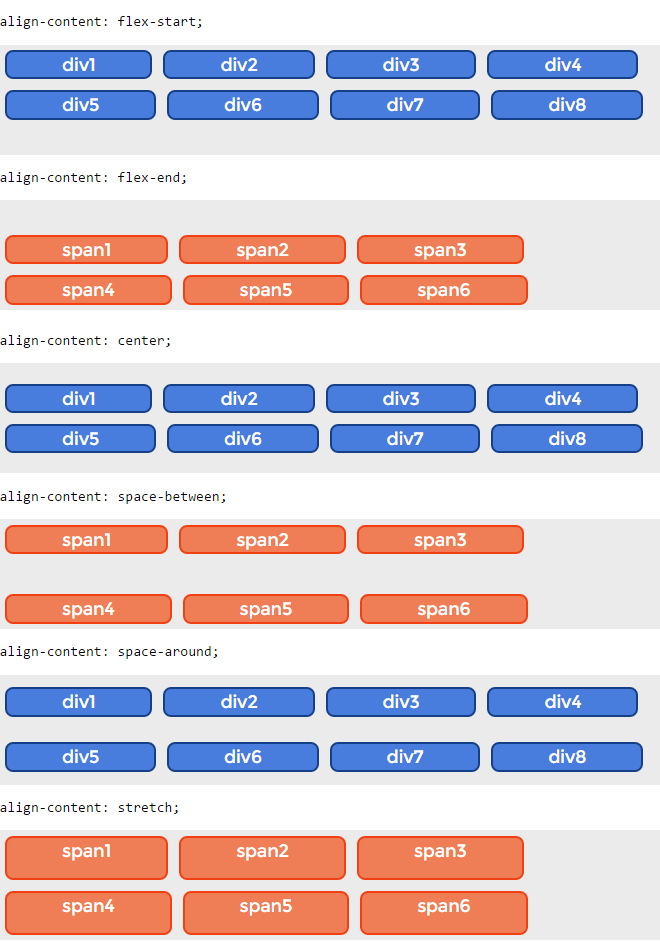


РИС. 7. Багаторядкове вирівнювання flex-елементів

**Синтаксис**

.flex-container {

display: -webkit-flex;

-webkit-flex-flow: row wrap;

-webkit-align-content: flex-end;

display: flex;

flex-flow: row wrap;

align-content: flex-end;

height: 100px;

}

2. Властивості flex-елементів

2.1. Порядок виведення елементів order

Дана властивість визначає порядок, у якому flex-елементи виводяться всередині flex-контейнера. За замовчуванням для усіх flex-елементів задано порядок order: 0; і вони слідують один за одним у порядку їх розташування у flex-контейнері в HTML. Щоб помістити flex-елемент на почато рядка потрібно призначити значення order: -1;, у кінець рядка — order: 1;. Властивість не успадковується.

|  |  |
| --- | --- |
| **order** | |
| Значения: |  |
| число | Задається ціле число, що відповідає за порядок виведення flex-елементів. За замовчуванням 0. |
| initial | Встановлює значення за замовчуванням. |
| inherit | Успадоковує значення від предка. |

**Синтаксис**

.flex-container {

display: -webkit-flex;

display: flex;

}

.flex-item {

-webkit-order: 1;

order: 1;

}

РИС. 8. Порядок виведення flex-елементів

2.2. Базова ширина елемента flex-basis

Властивість дозволяє задати базову (бажану) ширину flex-елемента, відносно якої буде відбуватися розтягування flex-grow або звуження flex-shrink елемента. Не успадковується.

|  |  |
| --- | --- |
| **flex-basis** | |
| Значения: |  |
| auto | Значення за замовчуванням. Ширина визначається на основі ширини контента (якщо вона не задана явно). |
| число | Ширина елемента задається в px, %, em  та ін. |
| initial | Встановлює початкове значення елемента. |
| inherit | Успадковується. |

**Синтаксис**

.flex-container {

display: -webkit-flex;

display: flex;

}

.flex-item {

-webkit-flex-basis: 100px;

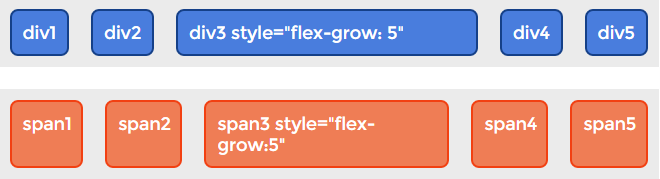
flex-basis: 100px;

}

2.3. Розтягнення елементів flex-grow

Властивість визначає коефіцієнт збільшення шириниflex-элемента відносно інших flex-елементів. Властивість не успадковується.

|  |  |
| --- | --- |
| **flex-grow** | |
| число | Додатне ціле або дробове число, що визначає коефіцієнт збільшення flex-елемента. Значення за замовчуванням 0. |
| initial | Встановлює значення за замовчуванням. |
| inherit | Успадковує значення від предка. |

РИС. 10. Розтягнення елементів у рядку

**Синтаксис**

.flex-container {

display: -webkit-flex;

display: flex;

}

.flex-item {

-webkit-flex-grow: 3;

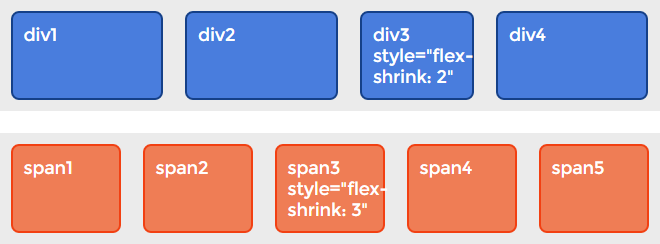
flex-grow: 3;

}

2.4. Звуження елементів flex-shrink

Властивість задає коефіцієнт зменшення ширини flex-елемента відносно інших flex-елементів. Діє тільки у випадку, коли для елемента задано ширину за допомогою властивості flex-basis або width. Властивість не успадковується.

|  |  |
| --- | --- |
| **flex-shrink** | |
| число | Додатне ціле або дробове число, яке встановлює коефіцієнт зменшення flex-елемента. Значення за замовчуванням 1. |
| initial | Встановлю значення за замовчуванням. |
| inherit | Успадковує значення від предка. |

РИС. 11. Звуження flex-елементів у рядку

**Синтаксис**

.flex-container {

display: -webkit-flex;

display: flex;

}

.flex-item {

-webkit-flex-shrink: 3;

flex-shrink: 3;

}

2.5. Короткий спосіб задання базової шири і трансформації елемента однією влативістю flex

Дана властивість представляє собою скорочений запис властивостей властивостей flex-grow, flex-shrink і flex-basis. Значення за замовчуванням: flex: 0 1 auto;. Можна вказати як одне, так і усі три значення властивостей. Не успадковується.

|  |  |
| --- | --- |
| **flex** | |
| коефіцієнт розтягнення | Коефіцієнт збільшення ширини flex-елемента відносно інших flex-елементів. |
| коефіцієнт звуження | Коефіцієнт зменшення ширини flex-елемента відносно інших flex-елементів. |
| базова ширина | Базова ширина flex-елемента. |
| auto | flex: 1 1 auto;. |
| none | flex: 0 0 auto;. |
| initial | Значення за замовчуванням. |
| inherit | Успадкування від предка. |

**Синтаксис**

.flex-container {

display: -webkit-flex;

display: flex;

}

.flex-item {

-webkit-flex: 3 1 100px;

-ms-flex:  3 1 100px;

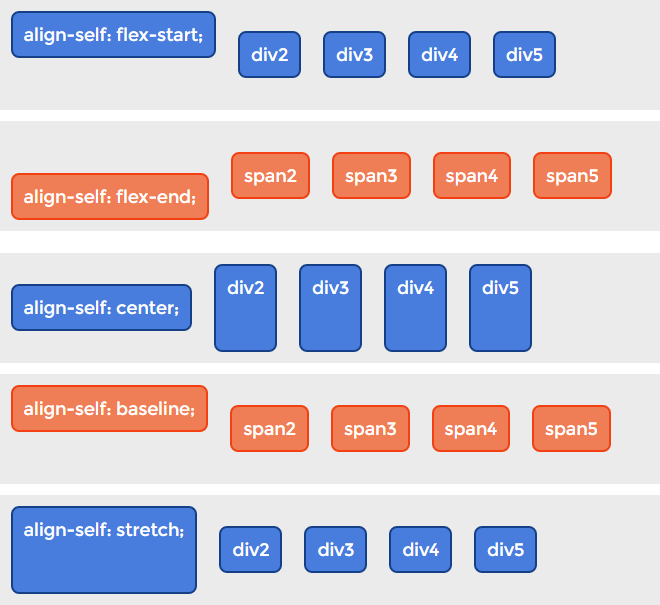
flex:  3 1 100px;

}

2.6. Вирівнювання окремих елементів align-self

Властивість відповідає за вирівнювання окремо взятого flex-элемента по висоті flex-контейнера. Перевизначає вирівнювання, задане align-items. Не успадковуєтсья.

|  |  |
| --- | --- |
| **align-self** | |
| Значения: |  |
| auto | Значення за замовчуванням. Flex-елемент використовує вирівнювання, вказане у властивості align-items flex-контейнера. |
| flex-start | Flex-елемент вирівнюється по верхньому краю flex-контейнера, відносно лівої границі. |
| flex-end | Flex-елемент вирівнюється по нижньому краю flex-контейнера, відносно лівої границі. |
| center | Flex-елемент вирівнюється по висоті по середині flex-контейнера, відносно лівої границі. |
| baseline | Flex-елемент вирівнюється по базовій лінії flex-контейнера, відносно лівої границі. |
| stretch | Flex-елемент розтягується на всю висоту flex-контейнера з врахуванням відступів і полів. |
| initial | Значення за замовчуваням. |
| inherit | Успадковується від предка. |

РИС. 12. Вирівнювання окремих flex-елементів

**Синтаксис**

.flex-container {

display: -webkit-flex;

display: flex;

}

.flex-item {

-webkit-align-self: center;

align-self: center;

}