**Що таке таблиця?**

Таблиця - це структурований набір даних, що складається з рядків і стовпців (табличних даних). Таблиці дозволяють швидко і легко подивитися значення, що показують деяку взаємозв'язок між різними типами даних, наприклад - людина і його вік, або розклад в плавальному басейні.

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

**Синтаксис**

<table>  
 ...  
</table>

Елемент **<tr>** (від англ. Table row - рядок таблиці) служить контейнером для створення рядки таблиці. Кожна клітинка в межах такого рядка встановлюється за допомогою елемента <th> або <td>.

**Синтаксис**

<table>  
 <tr>  
 ...  
 </tr>  
</table>

Елемент **<td>** (від англ. Table data cell - комірка з даними таблиці) повинен розміщуватися всередині контейнера <tr>, який в свою чергу розташовується всередині <table>. Призначений для створення однієї комірки таблиці.

**Синтаксис**

<table>  
 <tr>  
 <td>...</td>  
 </tr>  
</table>

Елемент **<th>** (від англ. Table header cell - комірка заголовка таблиці) призначений для створення однієї комірки таблиці, яка позначається як заголовна. Текст в такій клітинці відображається браузером зазвичай жирним шрифтом і вирівнюється по центру. Елемент <th> повинен розміщуватися всередині контейнера <tr>, який в свою чергу розташовується всередині <table>.

**Синтаксис**

<table>  
 <tr>  
 <th>...</th>  
 </tr>  
</table>

Елемент **<caption>** (від англ. Caption - заголовок) призначений для створення заголовка до таблиці і може розміщуватися лише всередині контейнера <table>, причому відразу після відкриваючого тега. Такий заголовок являє собою текст, за замовчуванням відображається перед таблицею і описує її зміст.

**Синтаксис**

<table>

<caption>Text</caption>  
 <tr>  
 <th>...</th>  
 </tr>  
</table>

Елемент **<thead>** (від англ. Table head - голова або шапка таблиці) призначений для зберігання одного чи кількох рядків, які представлені у верхній частині сторінки. Допустимо використовувати не більше одного елемента <thead> в межах однієї таблиці, і він повинен йти в вихідному коді відразу після відкриваючого тега <table> або <caption>.

**Синтаксис**

<table>

<caption> ... </caption>  
 <thead>  
 <th>  
 <td> ... </td>  
 </th>  
 </thead>  
 <tbody> ... </tbody>

<tfoot> ... </tfoot>  
</table>

Елемент **<tbody>** (від англ. Table body - тіло таблиці) призначений для зберігання одного чи кількох рядків таблиці. Це дозволяє створювати структурні блоки, до яких можна застосовувати єдине оформлення через стилі.

**Синтаксис**

<table>

<caption> ... </caption>  
 <thead> ... </thead>  
 <tbody>

<tr>  
 <td> ... </td>  
 </tr>

</tbody>

<tfoot> ... </tfoot>  
</table>

Елемент **<tfoot>** (від англ. Table foot - підвал таблиці) являє собою "підвал" таблиці і відображається внизу таблиці. Призначений для інформації про колонках таблиці.

**Синтаксис**

<table>

<caption> ... </caption>  
 <thead> ... </thead>  
 <tbody> ... </tbody>

<tfoot> ... </tfoot>  
</table>

Атрибут **colspan** встановлює число комірок, які повинні бути об'єднані по горизонталі. Цей атрибут має сенс для таблиць, що складаються з декількох рядків.

**Синтаксис**

<table>  
 <caption> ... </caption>  
 <thead> ... </thead>  
 <tbody>  
 <tr>  
 <td *colspan*="2"> TD1 </td>  
 <td> TD3 </td>  
 </tr>  
 <tr>  
 <td> TD1.1 </td>  
 <td> TD1.2 </td>  
 <td> TD1.3 </td>  
 </tr>  
 </tbody>  
 <tfoot> ... </tfoot>  
</table>

Атрибут **rowspan** встановлює число комірок, які повинні бути об'єднані по вертикалі. Цей атрибут має сенс для таблиць, що складаються з декількох рядків.

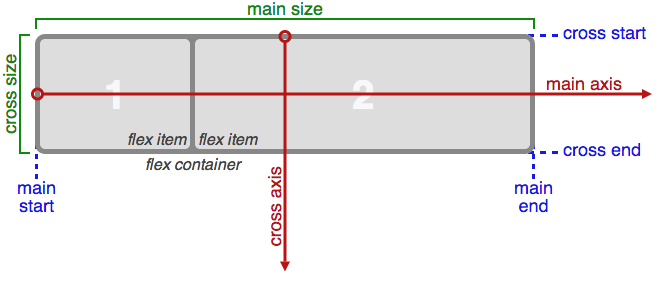
**Синтаксис**

<table>  
 <caption> ... </caption>  
 <thead> ... </thead>  
 <tbody>  
 <tr>  
 <td *colspan*="2"> TD1 </td>  
 <td *rowspan*="2"> TD3 </td>  
 </tr>  
 <tr>  
 <td> TD1.1 </td>  
 <td> TD1.2 </td>  
 </tr>  
 </tbody>  
 <tfoot> ... </tfoot>  
</table>

**Flexbox**

Оскільки flexbox - це цілий модуль, а не просто одиничне властивість, він об'єднує в собі безліч властивостей. Деякі з них мають застосовуватися до контейнера (батьківського елементу, так званому flex-контейнеру), в той час як інші властивості застосовуються до дочірніх елементів, або flex-елементам.

Якщо звичайний лейаут грунтується на напрямках потоків блокових і інлайн-елементів, то flex-лейаут грунтується на «напрямках flex-потоку». Ознайомтеся з цією схемою з специфікації, роз'яснює основну ідею flex-лейаутов.



В основному елементи будуть розподілятися або уздовж головної осі (від main-start до main-end), або вздовж поперечної осі (від cross-start до cross-end).

* **main-axis** - головна вісь, уздовж якого розташовуються flex-елементи. Зверніть увагу, вона необов'язково повинна бути горизонтальною, все залежить від якості flex-direction (див. Нижче).
* **main-start | main-end** - flex-елементи розміщуються в контейнері від позиції main-start до позиції main-end.
* **main size** - ширина або висота flex-елемента в залежності від вибраної базової величини. Основна величина може бути або шириною, або висотою елемента.
* cross axis - поперечна вісь, перпендикулярна до головної. Її напрямок залежить від напрямку головної осі.
* **cross-start | cross-end** - flex-рядки заповнюються елементами і розміщуються в контейнері від позиції cross-start і до позиції cross-end.
* **cross size** - ширина або висота flex-елемента в залежності від обраної розмірності дорівнює цій величині. Це властивість збігається з width або height елемента в залежності від обраної розмірності.

**Властивості**

**display: flex | inline-flex;**

**Застосовується до:** батьківського елементу flex-контейнера.

Визначає flex-контейнер (інлайновий або блоковий в залежності від обраного значення), підключає flex-контекст для всіх його безпосередніх нащадків.

display: other values | flex | inline-flex;

**Майте на увазі:**

* CSS-стовпці columns не працюють з flex-контейнером
* float, clear і vertical-align не працюють з flex-елементами

**flex-direction**

**Застосовується до:** батьківського елементу flex-контейнера.

Встановлює головну вісь main-axis, визначаючи тим самим напрямок для flex-елементів, що розміщуються в контейнері.

flex-direction: row | row-reverse | column | column-reverse;

* row (за замовчуванням): зліва направо для ltr, справа наліво для rtl;
* row-reverse: справа наліво для ltr, зліва направо для rtl;
* column: аналогічно row, зверху вниз;
* column-reverse: аналогічно row-reverse, від низу до верху.

**flex-wrap**

**Застосовується до:** батьківського елементу flex-контейнера.

Визначає, чи буде контейнер однорядковим або багаторядковим, а також напрямок поперечної осі, що визначає напрямок, в якому будуть розташовуватися нові рядки.

flex-wrap: nowrap | wrap | wrap-reverse;

* nowrap (за замовчуванням): однорядковий / зліва направо для ltr, справа наліво для rtl;
* wrap: багатостроковий / зліва направо для ltr, справа наліво для rtl;
* wrap-reverse: багатостроковий / справа наліво для ltr, зліва направо для rtl.

**flex-flow**

**Застосовується до:** батьківського елементу flex-контейнера.

Це скорочення для властивостей flex-direction і flex-wrap, разом визначають головну і поперечну осі. За замовчуванням приймає значення row nowrap.

flex-flow: <'flex-direction'> || <'flex-wrap'>;

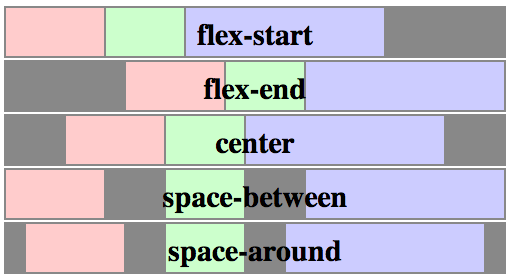
**justify-content**

**Застосовується до:** батьківського елементу flex-контейнера.

Визначає вирівнювання відносно головної осі. Сприяє ефективному розподілу вільне місце в разі, коли елементи рядки не «тягнуться», або тягнуться, але вже досягли свого максимального розміру. Також дозволяє до певної міри керувати вирівнюванням елементів при виході за межі рядка.

justify-content: flex-start | flex-end | center | space-between | space-around;

* flex-start (за замовчуванням): елементи зсуваються до початку рядка;
* flex-end: елементи зсуваються до кінця рядка;
* center: елементи вирівнюються по центру рядка;
* space-between: елементи розподіляються рівномірно (перший елемент на початку рядка, останній - в кінці);
* space-around: елементи розподіляються рівномірно з рівною відстанню між собою і кордонами рядки.



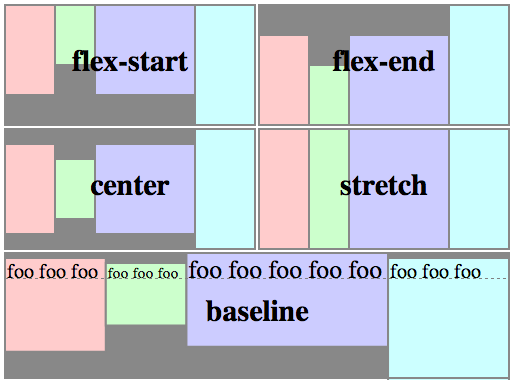
**align-items**

**Застосовується до:** батьківського елементу flex-контейнера.

Визначає поведінку за замовчуванням для того, як flex-елементи розташовуються відносно поперечної осі на поточному рядку. Вважайте це версією justify-content для поперечної осі (перпендикулярної до основної).

align-items: flex-start | flex-end | center | baseline | stretch;

* flex-start: межа cross-start для елементів розташовується на позиції cross-start;
* flex-end: межа cross-end для елементів розташовується на позиції cross-end;
* center: елементи вирівнюються по центру поперечної осі;
* baseline: елементи вирівнюються по своїй базової лінії;
* stretch (за замовчуванням): елементи розтягуються, заповнюючи контейнер (з урахуванням min-width / max-width).



**align-content**

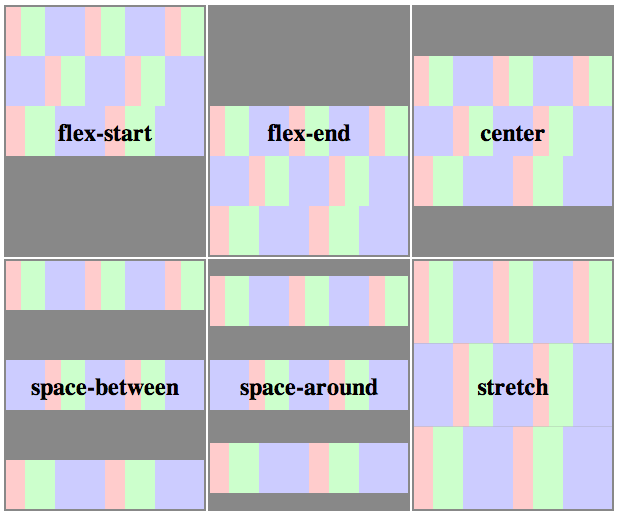
**Застосовується до:** батьківського елементу flex-контейнера.

Вирівнює рядки flex-контейнера при наявності вільного місця на поперечної осі аналогічно тому, як це робить justify-content на головній осі.

**Зауваження:** це властивість не працює з однорядковим flexbox.

align-content: flex-start | flex-end | center | space-between | space-around | stretch;

* flex-start: рядки вирівнюються щодо початку контейнера;
* flex-end: рядки вирівнюються щодо кінця контейнера;
* center: рядки вирівнюються по центру контейнера;
* space-between: рядки розподіляються рівномірно (перший рядок на початку рядка, остання - в кінці);
* space-around: рядки розподіляються рівномірно з рівною відстанню між собою;
* stretch (за замовчуванням): рядки розтягуються, заповнюючи вільний простір.



**order**

**Застосовується до:** дочірньому елементу / flex-елементу.

За замовчуванням flex-елементи розташовуються в початковому порядку. Проте, властивість order може управляти порядком їх розташування в контейнері.

order: <integer>;

**flex-grow**

**Застосовується до:** дочірньому елементу / flex-елементу.

Визначає для flex-елемента можливість «виростати» при необхідності. Приймає безрозмірне значення, що служить в якості пропорції. Воно визначає, яку частку вільного місця всередині контейнера елемент може зайняти.

Якщо у всіх елементів властивість flex-grow задано як 1, то кожен нащадок отримає всередині контейнера однаковий розмір. Якщо ви задали одному з нащадків значення 2, то він забере в два рази більше місця, ніж інші.

flex-grow: <number> (default 0);

Негативні числа не приймаються.

**flex-shrink**

**Застосовується до:** дочірньому елементу / flex-елементу.

Визначає для flex-елемента можливість стискатися при необхідності.

flex-shrink: <number> (default 1);

Негативні числа не приймаються.

**flex-basis**

**Застосовується до:** дочірньому елементу / flex-елементу.

Визначає розмір за замовчуванням для елемента перед розподілом простору в контейнері.

flex-basis: <length> | auto (default auto);

**flex**

**Застосовується до**: дочірньому елементу / flex-елементу.

Це скорочення для flex-grow, flex-shrink і flex-basis. Другий і третій параметри (flex-shrink, flex-basis) є обов'язковими. Значення за замовчуванням - 0 1 auto.

flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ];

**align-self**

**Застосовується до:** дочірньому елементу / flex-елементу.

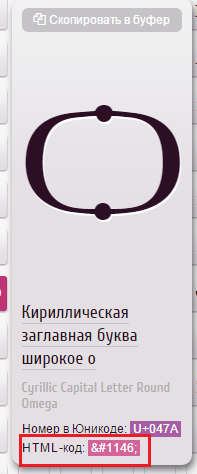
Дозволяє перевизначити вирівнювання, заданий за замовчуванням або в align-items, для окремих flex-елементів.

Зверніться до опису властивості align-items для кращого розуміння доступних значень.

align-self: auto | flex-start | flex-end | center | baseline | stretch;

<http://unicode-table.com/>

кожен символ має свій код і у разі введення його у свою html структуру він почне відображатись на сторінці



<span class="icon">&#1146;</span>

Також ці символи можливо скопіювати і вставити у текстовий редактор.



Додаткові джерела:

<http://flexboxfroggy.com/#uk>

<https://webref.ru/css/type/flex>

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

<http://unicode-table.com/ru/>