CSS Layouts Basic
<< units, standard blocks model >>
<< float, display, position, overflow >>

Единицы измерения

Для задания размеров различных элементов в **CSS** используются **абсолютные** и **относительные** единицы измерения.

Абсолютные единицы не зависят от устройства вывода

Относительные единицы определяют размер элемента относительно значения другого размера.

Абсолютные единицы измерения

Абсолютные единицы измерения имеют фиксированный размер. Если в вашем случае требуется указать точную длину, вам нужны абсолютные единицы измерения.
Например, это подходит для элементов, размер которых не должен меняться. Также абсолютные единицы могут быть полезны, если вы хотите определить ограничения для каких-то областей, чтобы они не могли стать слишком широкими или слишком узкими.
Те элементы, размеры которых вы задаете в абсолютных величинах, не меняются в зависимости от размеров экрана, ориентации в пространстве и прочих вещей.

cm	сантиметры. **1 cm = 1 cm**
mm	миллиметры. **10 mm = 1 cm**
in	дюймы (inches). **1 in = 96px = 2.54 cm**
px	пиксели. **1 px = 1/96 от 1 in**
pt	поинты (points). **1 pt = 1/72 от 1 in**
pc	пайки (пики, англ. pica). **1pc = 12 pt**

Относительные единицы измерения

Размеры в **CSS** можно указывать не только в абсолютных единицах, таких как пиксели, поинты или сантиметры, но и в относительных – в **процентах**, **em** или **rem**. Использование относительных единиц измерения также помогает придерживаться стандартов доступности. В большинстве браузеров по умолчанию установлен размер шрифта 16px. Это значение можно использовать при расчетах (например, 16px равны 1em, 1rem или 100%).

%	измерение в процентах.
em	размер шрифта относительно обычного, т. е., если шрифт имеет размер 2.5em, значит, он в 2,5 раза больше обычного шрифта.
rem	размер шрифта относительно корневого элемента документа
ch	ширина символа «0». В моноширинных шрифтах, где все символы имеют одинаковую ширину, 1ch это ширина одного символа.
ex	x-высота текущего шрифта, измеряется в высоте символа «x» в нижнем регистре.

EM vs REM

Единица измерения em
1em равен текущему размеру шрифта относительно родителя. Если у родителя не указан размер шрифта в пикселях, то **1em** равен приблизительно **16 пикселям**. Преимуществом использования **em**, состоит в том, что можно очень быстро изменить размеры шрифта во всем документе, поменяв размер шрифта в теге **body**. Но надо учитывать, что вложенный элемент, унаследует размер шрифта не у тега **body**, а у своего родителя.

Единица измерения rem
1rem равен текущему размеру шрифта корневого элемента, в документе есть только один корневой документ - это **html**. Для **rem** не имеет значение количество вложенных элементов в другие, **1rem** всегда будет иметь одинаковый размер в пикселях, как в корневом селекторе.

Разница между **em** и **rem** следующая. em зависит от размера шрифта родителя элемента и меняется вместе с ним, а rem привязан к корневому элементу, т. е. размеру шрифта заданного для элемента **html**.

Единицы измерения области просмотра (viewport-единицы)

Viewport-единицы представляют собой процентное отношение к текущей величине области просмотра браузера. От простого выражения в процентах viewport-единицы отличаются тем, что они всегда рассчитываются на основе текущего размера области просмотра. А размер, выраженный просто в процентах, вычисляется по отношению к родительскому элементу.

vw	1% от ширины области просмотра (50% это половина ширины области просмотра).
vh	1% от высоты области просмотра (50% это половина высоты области просмотра).
vmin	1% от меньшего размера ширины или высоты области просмотра (т. е., если ширина меньше высоты, то vmin рассчитывается от ширины, при этом 1 vmin = 1 vw).
vmax	1% от большего размера ширины или высоты области просмотра (т. е., если высота больше ширины, то vmax рассчитывается от высоты, при этом 1 vmax = 1 vh).

и

Тег ** ** и тег **** не имеют свойств по умолчанию для внешнего отображения, они представляют собой так называемые контейнеры, поэтому к ним можно применять всевозможные **CSS-стили**, чтобы элементы внутри этих тегов приняли желаемый вид.

Тег ** ** изначально был предназначен для деления веб-страницы на логические фрагменты, такие как нижний колонтитул, боковое меню и т.п. Но с появлением новой семантической разметки веб-страниц с помощью **HTML5**, необходимость в таком повсеместном использовании тега отпала.

В настоящее время тег ** ** используется для группирования блочных элементов. Тег также позволяет объединять любой набор таких элементов, как заголовок, насколько абзацев, список в единственном блоке, который потом можно позиционировать на веб-странице, создавая сложную схему разметки.

Тег **** применяется для выделения внутренних (**inline**) элементов, таких как отдельные слова и фразы, находящихся в пределах абзаца текста или заголовка.

Понятия потока документа

По умолчанию содержимое элемента уровня блока составляет **100% ширины** его родительского элемента, а высота равно его содержанию.

Ширина и высота строчных элементов равна содержимому.

Нельзя установить ширину или высоту для встроенных элементов - они просто находятся внутри содержимого элементов уровня блока.

Обычный поток разметки - это система, с помощью которой элементы размещаются внутри области просмотра браузера.

По умолчанию элементы уровня блока располагаются в направлении потока блоков сверху вниз - каждый из них будет отображаться в новой строке ниже последней, и они будут разделены любым внешними отступами (margin), которая установлена для них.

Строчные элементы ведут себя по-разному - они не появляются на новых строках, они располагаются на одной строке друг с другом и с любым смежным текстовым содержимым, до тех пор пока для них есть места. Если места нет, то переполненный текст или элементы переместятся вниз на новую строку.

```html
<h1>Basic document flow</h1>

<p>I am a basic block level element.
My adjacent block level elements sit on new
lines below me.</p>

<p>By default we span 100% of the width of our parent element,
 and we are as tall as our child content. Our total width and height
 is our content + padding + border width/height.</p>

<p>We are separated by our margins. Because of margin collapsing,
we are separated by the width of one of our margins, not both.</p>

<p>inline elements <span>like this one</span> and <span>this one</span>
sit on the same line as one another, and adjacent text nodes, if there is space
on the same line. Overflowing inline elements will <span>wrap into a new
line if possible (like this one containing text)</span>,
or just go on to a new line if not, much like this image will do:
<img src="https://mdn.mozillademos.org/files/13360/long.jpg">
</p>
```

```css
body {
   width: 500px;
   margin: 0 auto;
}
p {
   background: rgba(255,84,104,0.3);
   border: 2px solid rgb(255,84,104);
   padding: 10px;
   margin: 10px;
}
span {
   background: white;
   border: 1px solid black;
}
```

Basic document flow

I am a basic block level element. My adjacent block level elements sit on new lines below me.

By default we span 100% of the width of our parent element, and we are as tall as our child content. Our total width and height is our content + padding + border width/height.

We are separated by our margins. Because of margin collapsing, we are separated by the width of one of our margins, not both.

inline elements like this one and this one sit on the same line as one another, and adjacent text nodes, if there is space on the same line. Overflowing inline elements will wrap onto a new line if possible (like this one containing text), or just go on to a new line if not, much like this image will do:

display

Свойство **display** определяет, как элемент должен быть показан в документе.

none	Временно удаляет элемент из документа. Занимаемое им место не резервируется и веб-страница формируется так, словно элемента и не было.
block	Элемент показывается как блочный.
inline	Элемент отображается как строчный.
inline-block	Это значение генерирует блочный элемент, который обтекается другими элементами веб-страницы подобно строчному элементу.
list-item	Элемент выводится как блочный и добавляется маркер списка.
table	Определяет, что элемент является блочной таблицей подобно использованию тега **<table>**.
table-cell	Указывает, что элемент представляет собой ячейку таблицы

box-sizing

Применяется для изменения алгоритма расчета ширины и высоты элемента

Стандартная блочная модель (согласно спецификации CSS) – ширина блока складывается из ширины контента (**width**), значений отступов (**margin**), полей (padding) и границ (**border**). Аналогично обстоит и с высотой блока. Свойство **box-sizing** позволяет изменить этот алгоритм, чтобы свойства **width** и **height** задавали размеры не контента, а размеры блока.

content-box	Основывается на стандартах CSS, при этом свойства width и height задают ширину и высоту контента и не включают в себя значения отступов, полей и границ.
border-box	Свойства width и height включают в себя значения полей и границ, но не отступов (margin).
padding-box	Свойства width и height включают в себя значения полей, но не отступов (margin) и границ (border).

calc()

calc() - это функция CSS, которая дает возможность рассчитать значения свойств CSS во время их определения.

Функция **calc()** принимает в качестве параметра математическое выражение, результат вычисления которого можно использовать как значение CSS свойства. Выражение может включать операторы **+**, **-**, *****, **/** с использованием стандартных правил приоритета операторов.

Операнды в выражении могут быть разные единицы измерения для каждого из операндов. Вы также можете использовать скобки, чтобы указать порядок вычисления.

```
width: calc(100% - 30px);
```

Операторы **+** и **-** всегда должны быть по обеим сторонам отделены пробелом. Выражение **calc(50% -8px)** будет интерпретировано как величина в процентах и следующее за ним отрицательное число в пикселях (неверное выражение), в то время как **calc(50% - 8px)** - правильное выражение, будет интерпретировано как вычитание из процентов длины в пикселях. Операторы ***** и **/** не требуют отделения от операндов знаком пробела, но это не запрещено и даже приветствуется.

float

Определяет, по какой стороне будет выравниваться элемент, при этом остальные элементы будут обтекать его с других сторон. Когда значение свойства **float** равно **none**, элемент выводится на странице как обычно, при этом допускается, что одна строка обтекающего текста может быть на той же линии, что и сам элемент. Свойство **float “выбивает”** элемент из основного потока документа.

*Абсолютно позиционированные элементы игнорирует свойство **float**!*

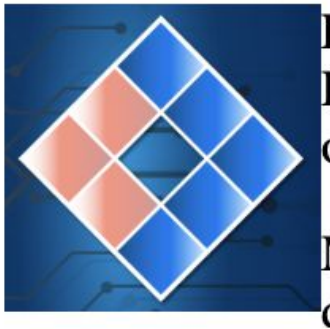
none	Элемент не “плавает”, (будет отображаться только там, где он встречается в тексте). Это значение по умолчанию
left	Элемент “плавает” слева от своего контейнера
right	Элемент “плавает” справа от своего контейнера
initial	Устанавливает для этого свойства значение по умолчанию.
inherit	Наследует это свойство от родительского элемента.

Свойство **clear** позволяет отменить обтекание плавающего элемента:

clear: left | right | both

```html
<img src="logo.png">
<p>Lorem, ipsum ...</p>
<p>Molestias, ...</p>
```

```css
img{
    width: 100px;
    float: left;
}
```



Lorem, ipsum dolor sit amet consectetur adipisicing elit. Dolorum in incidunt, eius nam hic vero enim animi obcaecati voluptatem debitis ullam sunt id vitae.

Molestias labore laborum, quos voluptate est impedit dolores maiores nam tempore exercitationem aliquid quidem dolor assumenda debitis! Facilis eius repudiandae neque, explicabo quae delectus fugiat nam odio animi, esse earum?

```
<img src="logo.png">
<p>Lorem, ipsum ...</p>

<p>Molestias, ...</p>
```

```
.clearfix{
    clear: left;
}
```

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Dolorum in incidunt, eius nam hic vero enim animi obcaecati voluptatem debitis ullam sunt id vitae.

Molestias labore laborum, quos voluptate est impedit dolores maiores nam tempore exercitationem aliquid quidem dolor assumenda debitis! Facilis eius repudiandae neque, explicabo quae delectus fugiat nam odio animi, esse earum?

position

Устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице.

static	Статическое позиционирование. Элементы отображаются как обычно. Использование свойств **left**, **top**, **right** и **bottom** не приводит к каким-либо результатам.
relative	Относительное позиционирование. Положение элемента устанавливается относительно его исходного места. Добавление свойств **left**, **top**, **right** и **bottom** изменяет позицию элемента и сдвигает его в ту или иную сторону от первоначального расположения.
absolute	Абсолютное позиционирование. Указывает, что элемент абсолютно позиционирован, при этом другие элементы отображаются на веб-странице словно абсолютно позиционированного элемента и нет. Положение элемента задается свойствами **left**, **top**, **right** и **bottom**, также на положение влияет значение свойства position родительского элемента. Так, если у родителя значение position установлено как static или родителя нет, то отсчет координат ведется от края окна браузера. Если у родителя значение **position** задано как **relative**, то отсчет координат ведется от края родительского элемента.

fixed	Фиксированное позиционирование. По своему действию это значение близко к **absolute**, но в отличие от него привязывается к указанной свойствами **left**, **top**, **right** и **bottom** точке на экране и не меняет своего положения при прокрутке веб-страницы.
sticky	Это сочетание относительного и фиксированного позиционирования. Элемент рассматривается как позиционированный относительно, пока он не пересекает определенный порог, после чего рассматривается как фиксированный. Обычно применяется для фиксации заголовка на одном месте, пока содержимое, к которому относится заголовок, прокручивается на странице.

При использование свойства **position** элементы выбиваются из основного потока документа

overflow

Свойство **overflow** управляет отображением содержания блочного элемента, если оно целиком не помещается и выходит за область заданных размеров.

visible	Отображается все содержимое элемента, даже за пределами установленной высоты и ширины.
hidden	Отображается только область внутри элемента, остальное будет скрыто.
scroll	Всегда добавляются полосы прокрутки.
auto	Полосы прокрутки добавляются только при необходимости.

overflow-x
Свойство **overflow-x** управляет отображением содержания блочного элемента по горизонтали, если контент целиком не помещается и выходит за область справа или слева от блока.

overflow-y
Свойство **overflow-y** управляет отображением содержания блочного элемента по вертикали, если контент целиком не помещается и выходит за область сверху или снизу от блока.