

Языковой псевдокласс `:lang()`

Данный псевдокласс используется когда в документе содержатся абзацы текста на разных языках. Чтобы браузер различал их, элементу с текстом добавляется атрибут `lang` с кодом языка, например, `lang="fr"`. В результате чего этот элемент может быть стилизован при помощи селектора `p:lang(fr) {css-стили}`.

<https://codepen.io/lostsou41216364/pen/YOrzyj>

Псевдокласс отрицания `:not()`

Функциональный селектор псевдокласса, который принимает в качестве значения простой селектор, а затем отбирает элементы, которые не содержат указанное значение аргумента.

Значениями аргументов могут быть только следующие:

селекторы элемента, например, `body :not(strong)`

селекторы класса и идентификатора, например, `p:not(.text)`

селекторы псевдокласса, например, `ul:not(:first-child)`

селекторы атрибута, например, `input:not([type="checkbox"])`

<https://codepen.io/lostsou41216364/pen/bxoGeb>

::backdrop

Псевдоэлемент отображается ниже самого верхнего элемента в стеке по оси Z, но выше всех остальных элементов на странице, если они имеются.

Обычно **::backdrop** применяется для затемнения страницы, чтобы акцентировать внимание на фотографии или диалоговом окне, которые выводятся поверх такого затемнения.

Селектор ::backdrop { ... }

<https://codepen.io/lostsou41216364/pen/gdGLEd>

::placeholder

Псевдоэлемент, с помощью которого задаётся стилевое оформление подсказывающего текста, созданного атрибутом [placeholder](#).

Допускается использовать свойства для изменения вида текста, например, задать шрифт и цвет.

```
Селектор::placeholder { ... }
```

<https://codepen.io/lostsou41216364/pen/XPepbp>

::selection

Псевдоэлемент **::selection** применяет стиль к выделенному пользователем тексту. В правилах стилей допускается использовать следующие свойства: color, background, background-color, cursor, outline и text-shadow.

```
Селектор::selection { ... }
```

<https://codepen.io/lostsou41216364/pen/aaLpZy>

Шрифты

CSS-шрифты — набор свойств для управления внешним видом текста веб-страниц. Используя различные шрифты для заголовков, абзацев и других элементов, можно задавать определенный стиль письменных сообщений.

Текст основного содержимого веб-страницы должен быть в первую очередь читабельным. Не рекомендуется использовать более двух шрифтов на странице.

font-family

```
@font-face {  
    font-family: Pompadur; /* Имя шрифта */  
    src: url(fonts/pompadur.ttf); /* Путь к файлу со шрифтом */  
}
```

Свойство используется для выбора начертания шрифта.

Поскольку невозможно предсказать, установлен тот или иной шрифт на компьютере посетителя вашего сайта, рекомендуется прописывать все возможные варианты однотипных шрифтов.

В таком случае браузер будет проверять их наличие, последовательно перебирая предложенные варианты. **Наследуется.**

Важно! Если в названии шрифта имеются пробелы или символы (например, #, \$, %), то оно заключается в кавычки. Это делается для того, чтобы браузер мог понять, где начинается и заканчивается название шрифта.

```
p {font-family: "Times New Roman", Georgia, Serif;}
```

font-style

Свойство позволяет выбрать стиль начертания для текста. При этом разница между курсивом и наклонным текстом заключается в том, что курсивное начертание вносит небольшие изменения в структуру каждой буквы, а наклонный текст представляет собой наклонную версию прямого текста. Наследуется.

`normal`

Значение по умолчанию, устанавливает для текста обычное начертание шрифта.

`italic`

Выделяет текст курсивом.

`oblique`

Устанавливает наклонное начертание шрифта.

`initial`

Устанавливает значение свойства в значение по умолчанию.

`inherit`

Наследует значение свойства от родительского элемента.

font-weight

```
h1 {font-weight: normal;}  
span {font-weight: bold;}  
span {font-weight: bolder;}  
span {font-weight: lighter;}  
h1 {font-weight: 100;}
```

normal

Значение по умолчанию, устанавливает нормальную насыщенность шрифта. Эквивалентно значению насыщенности, равной 400.

bold

Делает шрифт текста полужирным. Эквивалентно значению насыщенности, равной 700.

bolder

Насыщенность шрифта будет больше, чем у предка.

lighter

Насыщенность шрифта будет меньше, чем у предка.

100, 200,
300, 400, 500,
600, 700, 800,
900

Значение 100 соответствует самому легкому варианту начертания шрифта, а 900 — самому плотному. При этом, эти числа не определяют конкретной плотности, т.е. 100, 200, 300 и 400 могут соответствовать одному и тому же варианту слабой насыщенности начертания шрифта; 500 и 600 — средней насыщенности, а 700, 800 и 900 могут выводить одинаковое очень насыщенное начертание. Распределение плотности так же зависит от количества уровней насыщенности, определенных в конкретном семействе шрифтов.

initial

Устанавливает значение свойства в значение по умолчанию.

inherit

Наследует значение свойства от родительского элемента.

font-size

Свойство определяет размер (кегель) шрифта.

absolute-size

`xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`.

Абсолютные размеры определены относительно друг друга и коэффициент масштабирования между двумя соседними абсолютными размерами составляет примерно 1,5 при переходе от меньшего к большему и 0,66 при переходе от большего к меньшему. В качестве стандартного размера принимается `medium`.

relative-size

`smaller`, `larger`. Относительные размеры обуславливают изменение размера шрифта элемента относительно родителя. При этом размер шрифта может выйти за рамки размеров, предполагаемых для `xx-small` и `xx-large`.

длина

Размер шрифта устанавливается с помощью положительных значений единиц длины — `px`, как целых, так и дробных.

%

Относительное значение, вычисляется на основании любого размера, унаследованного от родительского элемента. Обеспечивает более точную настройку вычисляемого размера шрифта. Задание размеров шрифта с помощью `em` эквивалентно процентному значению.

initial

Устанавливает значение свойства в значение по умолчанию.

inherit

Наследует значение свойства от родительского элемента.

```
h3 {font-size: small;}  
h1 {font-size: xx-large;}, em {font-size: large;}  
p {font-size: 20px;}  
h3 {font-size: 120%;}
```


Цвета шрифтов

HEX	В данной системе используются арабские десятичные цифры от 0 до 9 и латинские буквы от A до F. Для веб-дизайна взяты 16 основных цветов, так называемый шестнадцатеричный код цвета <code>#RRGGBB</code> , где каждая пара отвечает за свою долю цвета: RR - красный, GG - зеленый и BB - синий. Каждая доля цвета находится в диапазоне от 00 до FF.
RGB	RedGreenBlue , обозначает количество соответствующего тона (красный,зеленый,синий) в получаемом цвете.
RGBA	Система цветопередачи RGB, расширенная параметром Alpha , который используется для управления смешиванием цветов. Значение поддерживается IE9+, Firefox 3+, Chrome, Safari, and in Opera 10+. Последнее число определяет степень прозрачности, задается значением от 0 до 1, где 0 соответствует полной прозрачности, а 1 — непрозрачности.

`initial`

Устанавливает значение свойства в значение по умолчанию.

`inherit`

Наследует значение свойства от родительского элемента.

HSL

Hue, Saturation, Lightness (Intensity) — оттенок (тон), насыщенность, светлота, цветовая модель описания цветов.

Первое значение — **оттенок** — определяется градусом поворота цветового спектра по часовой стрелке от 0° до 360° , где 60° — желтый, 120° — зеленый, 180° — голубой, 240° — синий, 300° — фиолетовый. Второе значение определяет **насыщенность** выбранного оттенка и указывается в процентах в диапазоне от 0% до 100%. Чем ближе данное значение к 100%, тем цвет более чистый и сочный.

Светлота или **яркость (Lightness)** указывается в процентах, чем выше процент, тем ярче становится цвет. Значения 0% и 100% обозначают соответственно чёрный (отсутствие света) и белый (засвеченный) цвета, в независимости от того, какой оттенок из цветового круга был выбран изначально. Оптимальное значение яркости цвета равняется 50%. Значение поддерживается IE9+, Firefox, Chrome, Safari, Opera 10+

HSLA

Задаёт прозрачность (через Альфа-канал) элемента.

Тон (от 0 до 360), насыщенность (от 0% до 100%), светлота (от 0% до 100%), прозрачность (от 0 до 1). Значение поддерживается IE9+, Firefox 3+, Chrome, Safari, and in Opera 10+.

Форматы веб-шрифтов

Формат WOFF (Web Open Font Format)

Форматы OTF/TTF (OpenType Font и TrueType Font)

Формат EOT (Embedded Open Type)

SVG/SVGZ (Scalable Vector Graphics)

```
@font-face {  
  font-family: 'Web font';  
  src: url('webfont.woff2') format('woff2'),  
       url('webfont.woff') format('woff');  
  font-weight: normal;  
  font-style: normal;  
}
```

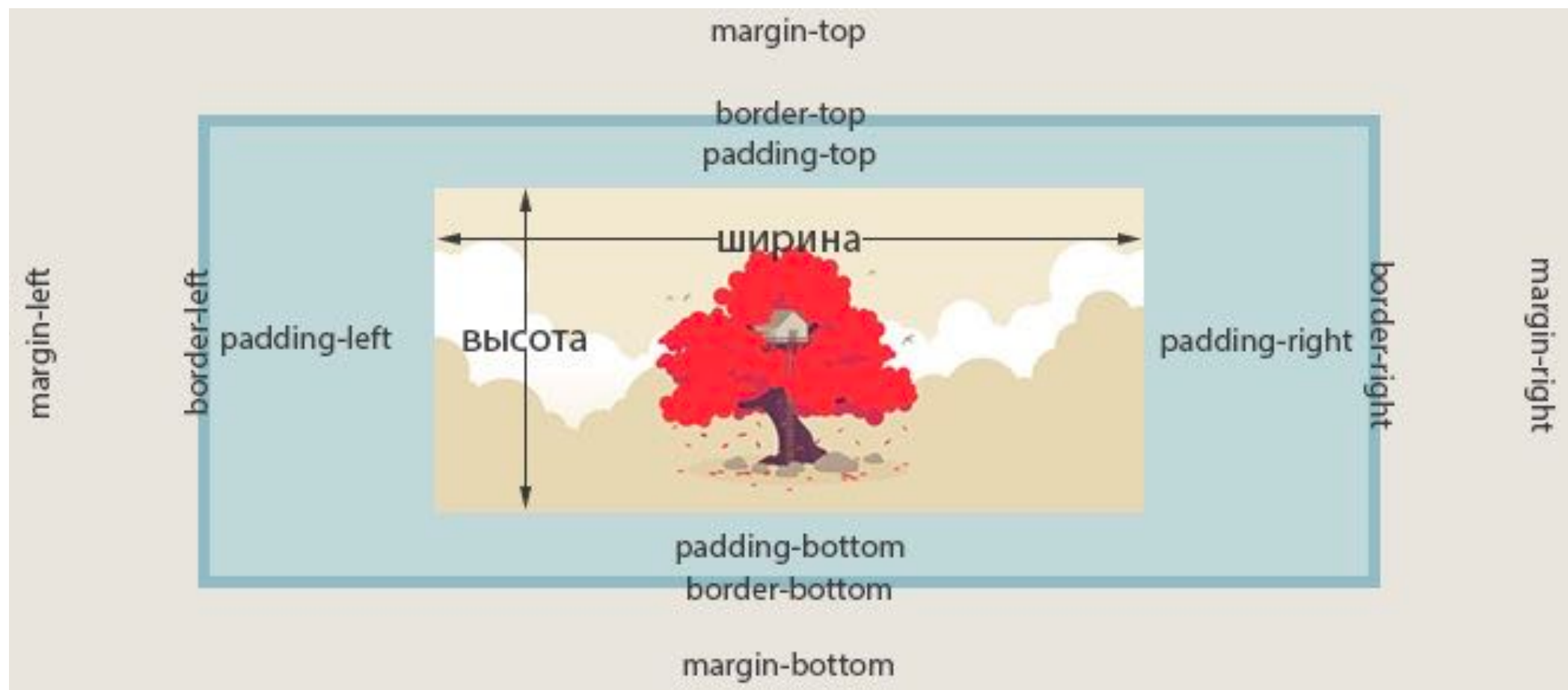
```
@font-face {  
  font-family: 'Web font';  
  src: url('webfont.eot');  
  src: url('webfont.eot?#iefix') format('embedded-opentype'),  
       url('webfont.woff2') format('woff2'),  
       url('webfont.woff') format('woff'),  
       url('webfont.ttf') format('truetype'),  
       url('webfont.svg#svgFontName') format('svg');  
  font-weight: normal;  
  font-style: normal;  
}
```

Box model

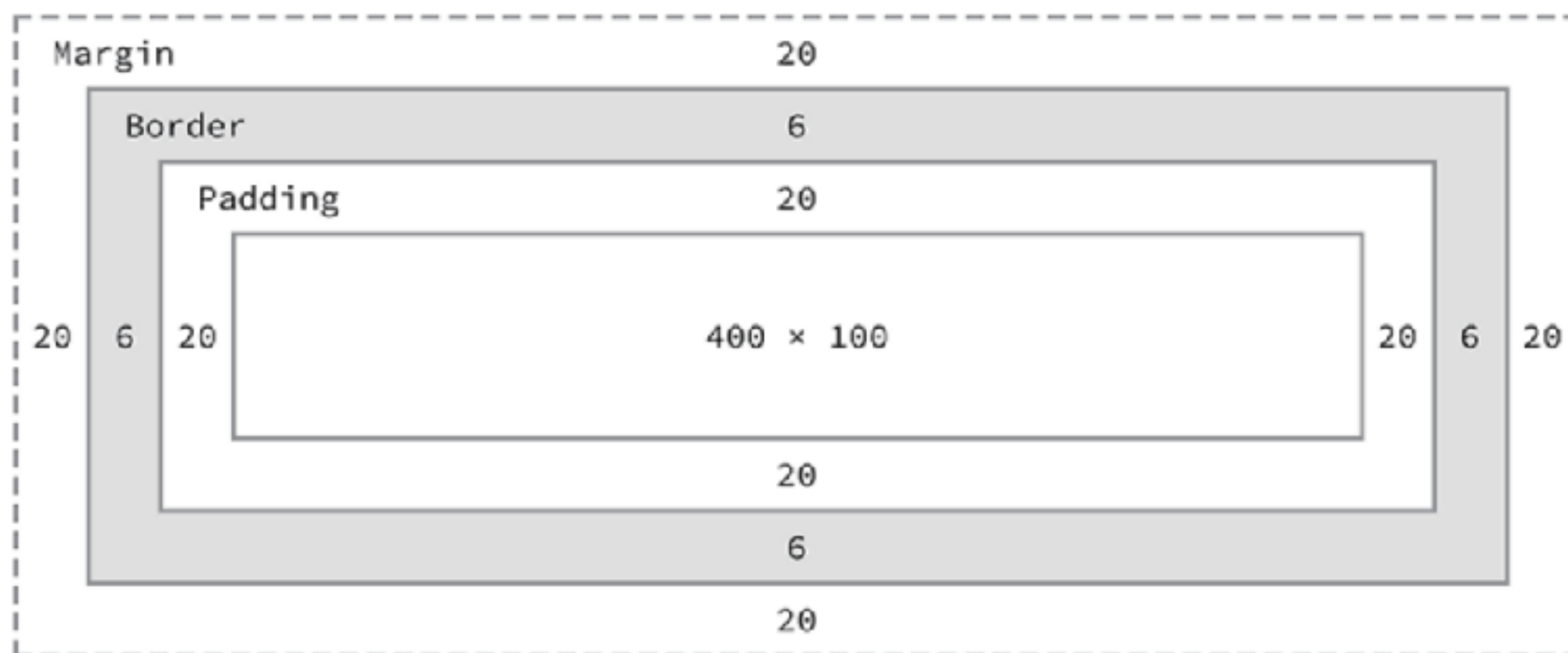
Vira Huskova

Блочная модель

В блочной модели элемент рассматривается как прямоугольный контейнер, имеющий область содержимого и необязательные рамки и отступы (внутренние внешние). Свойство **display** определяет тип контейнера элемента. Для каждого элемента существует значение браузера по умолчанию.



Область содержимого — это содержимое элемента, например, текст или изображение.



Ширина: $492\text{px} = 20\text{px} + 6\text{px} + 20\text{px} + 400\text{px} + 20\text{px} + 6\text{px} + 20\text{px}$

Высота: $192\text{px} = 20\text{px} + 6\text{px} + 20\text{px} + 100\text{px} + 20\text{px} + 6\text{px} + 20\text{px}$

Внутренний отступ задаётся свойством **padding**. Это расстояние между основным содержимым и его границей (рамкой).

Если для элемента задать фон, то он распространится также и на поля элемента. Внутренний отступ не может принимать отрицательных значений, в отличие от внешнего отступа.

Внешний отступ задаётся свойством **margin**.

Он добавляет расстояние снаружи элемента от внешней границы рамки до соседних элементов, тем самым разделяя элементы на странице.

Внешние отступы всегда остаются прозрачными и через них виден фон родительского элемента.

Значения **padding** и **margin** задаются в следующем порядке: верхнее, правое, нижнее и левое.

margin(padding): top, right, bottom, left;

Граница, или рамка элемента, задаётся с помощью свойства **border**.

Если цвет рамки не задан, она принимает цвет основного содержимого элемента, например, текста.

Если рамка имеет разрывы, то сквозь них будет проступать фон элемента.

Внешние, внутренние отступы и рамка элемента не являются обязательными, по умолчанию их значение равно нулю.

Тем не менее, некоторые браузеры добавляют этим свойствам положительные значения по умолчанию на основе своих таблиц стилей. Очистить стили браузеров для всех элементов можно при помощи универсального селектора:

```
* {  
  margin: 0;  
  padding: 0;  
}
```

Блочные элементы и блочные контейнеры

Блочные элементы — элементы высшего уровня, которые форматируются визуально как блоки, располагаясь на странице в окне браузера вертикально. Значения свойства **display**, такие как **block**, **list-item** и **table** делают элементы блочными.

Блочные элементы генерируют основной блок, который содержит только блок элемента.

Элементы со значением **display: list-item** генерируют дополнительные блоки для маркеров, которые позиционируются относительно основного блока.

**<address>, <article>, <aside>, <blockquote>, <dd>, <div>, <dl>, <dt>, <details>, <fieldset>, <figcaption>, <figure>, <footer>, <form>, <h1>-<h6>, <header>, <hr>, , <legend>, <nav>, <noscript>, , <output>, <optgroup>, <option>, <p>, <pre>, <section>, <summary>, <table>, **

Блочные элементы могут размещаться непосредственно внутри элемента `<body>`. Они создают разрыв строки перед элементом и после него, образуя прямоугольную область, по ширине занимающую всю ширину веб-страницы или блока-родителя (если для элемента не задано значение **width**).

Свойства **width** и **height** устанавливают ширину и высоту области содержимого элемента. Фактическая ширина элемента складывается из ширины полей (внутренних отступов) **padding**, границ **border** и внешних отступов **margin**.

Блочные элементы могут содержать как **строчные**, так и **блочные** элементы, но не оба типа элементов сразу. При необходимости, строки текста, принадлежащие блочному контейнеру, могут быть обернуты анонимными контейнерами, которые будут вести себя внутри блока как элементы со значением **display: block;**, а строчные элементы обернуты элементом `<p>`. Блочные элементы могут содержаться только в пределах блочных элементов.

Элемент `<p>` относится к блочным элементам, но он не может содержать внутри себя другой элемент `<p>`, а также любой другой блочный элемент.

До и после блочного элемента существует перенос строки.

Блочным элементам можно задавать ширину, высоту, внутренние и внешние отступы.

Занимают всё доступное пространство по горизонтали.

Строчные элементы и строчные контейнеры

Встроенные (строчные) элементы генерируют внутристрочные контейнеры. Они не формируют новые блоки контента. Значения свойства `display`, такие как `inline` и `inline-table` делают элементы строчными.

`<a>`, `<area>`, ``, `<bdo>`, `<bdi>`, `<cite>`, `<code>`, `<dfn>`, ``, ``, `<i>`,
`<iframe>`, ``, `<ins>`, `<kbd>`, `<label>`, `<map>`, `<mark>`, `<s>`, `<samp>`,
`<small>`, ``, ``, `<sub>`, `<sup>`, `<time>`, `<q>`, `<ruby>`, `<u>`, `<var>`

Строчные элементы являются потомками блочных элементов. Они игнорируют верхние и нижние `margin` и `padding`, но если для элемента задан фон, он будет распространяться на верхний и нижний `padding`, заходя на соседние строки текста.

Ширина и высота строчного элемента зависит только от его содержимого, задать размеры с помощью CSS нельзя. Можно увеличить расстояние между соседними элементами по горизонтали с помощью горизонтальных полей и отступов.

Для того чтобы верхние и нижние поля и отступы работали для строчного элемента, нужно использовать конструкцию `{display: inline-block}`. Элемент останется встроенным, но к нему можно будет полноценно применить поля, отступы, задать ширину и высоту.

```
span {padding: 10px;  
background: #c4c4c4;  
border: 2px dashed grey}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

```
span {padding: 10px;  
margin: 30px;  
background: #c4c4c4;  
border: 2px dashed grey;}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

До и после строчного элемента
отсутствуют переносы строки.

Ширина и высота строчного элемента
зависит только от его содержания, задать
размеры с помощью CSS нельзя.

Можно задавать только горизонтальные
отступы.

<https://codepen.io/lostsou41216364/pen/NOypGp>

Строчные элементы могут содержать только данные и другие строчные элементы.

Исключение составляет элемент `<a>`, который согласно спецификации HTML5 может оборачивать целые абзацы, списки, таблицы, заголовки и целые разделы при условии, что они не содержат другие интерактивные элементы — другие ссылки и кнопки.

Строчный -> блочный
Блочный -> строчный

В некоторых случаях бывает необходимо, чтобы строчный элемент вел себя как блочный и наоборот. Для этого необходимо установить соответствующее значение свойства **display**:

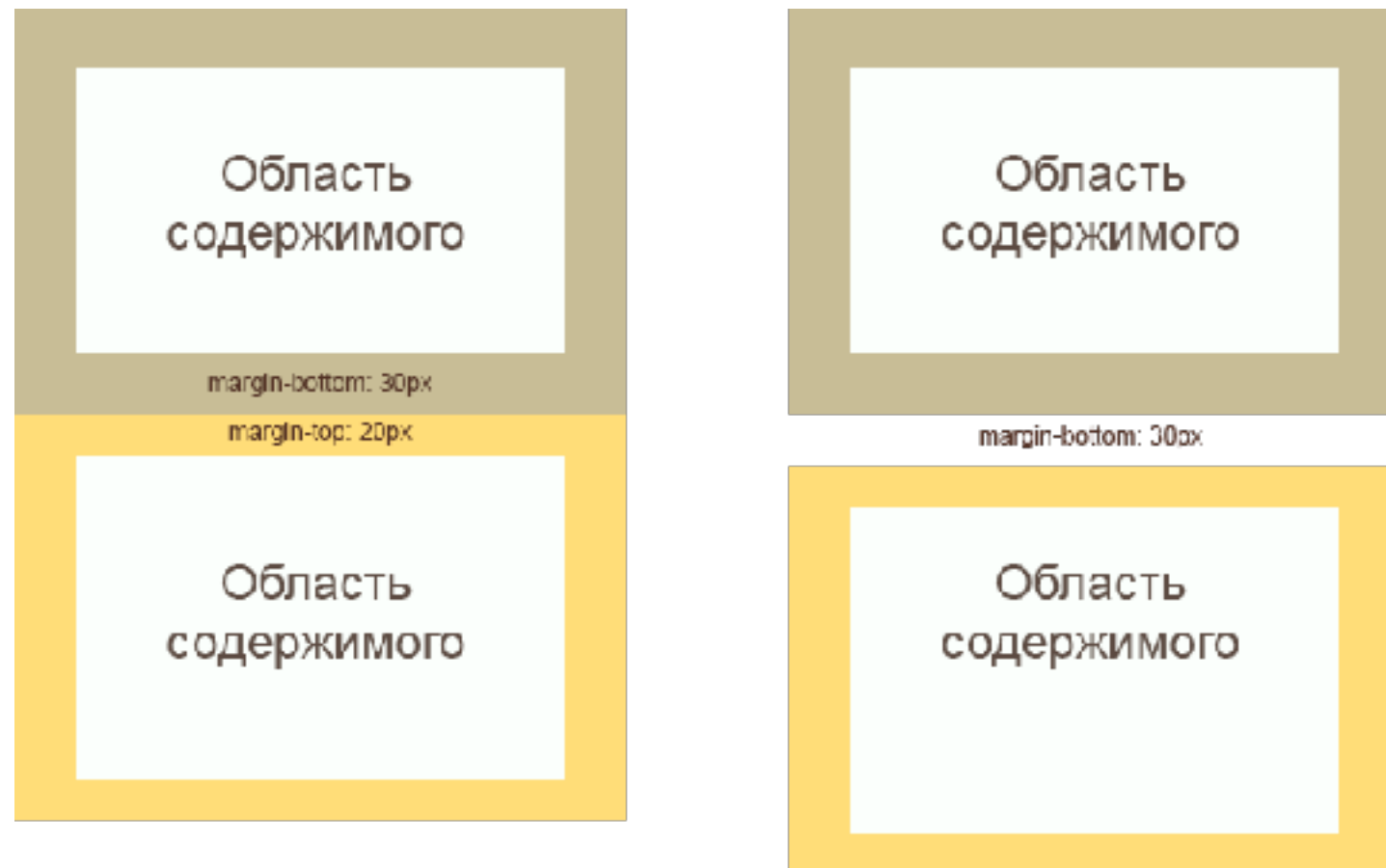
```
a { display: block; }
```

```
div { display: inline; }
```

В результате меняется только способ представления элемента браузером, при этом сам элемент свой тип не меняет.

Схлопывание

Соприкасающиеся вертикальные отступы `margin` объединяются. При этом ширина общего отступа равна ширине большего из исходных отступов.



Вертикальный отступ между двумя соседними элементами равен максимальному отступу между ними. Если отступ одного элемента равен 20px, а второго 40px, то отступ между ними будет 40px.

Горизонтальные отступы между элементами просто складываются. Например, горизонтальный отступ между двумя элементами с отступами 30px будет равен 60px.

Слияние выполняется только для блочных элементов в нормальном потоке документа. Внешние вертикальные отступы строчных, плавающих и абсолютно позиционированных элементов не сливаются.

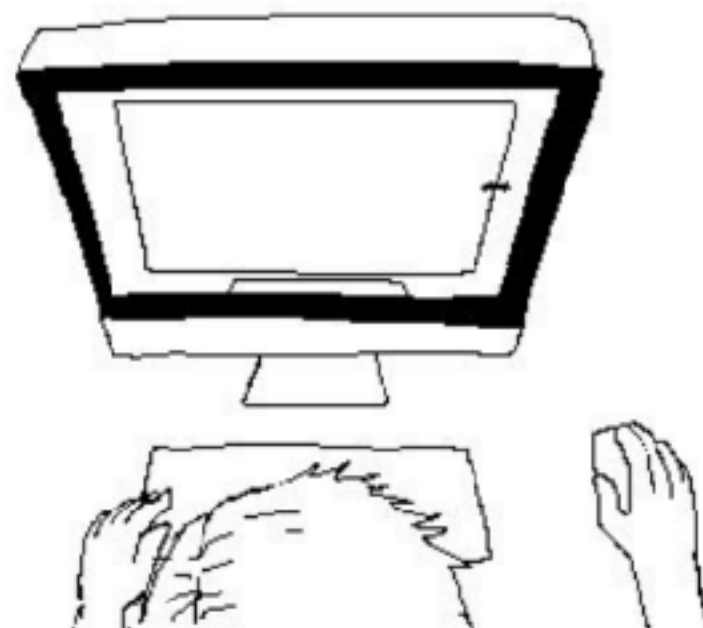
Чтобы получить желаемый промежуток, можно задать, например, для верхнего элемента `padding-bottom`, а для нижнего элемента — `margin-top`.

Если среди схлопывающихся отступов есть отрицательные значения, то браузер добавит отрицательное значение к положительному, а полученный результат и будет расстоянием между элементами.

Отрицательные отступы

Отрицательные отступы можно использовать, чтобы убрать пустые области между элементами. Например, необходимо расположить в ряд несколько элементов меню с заданными размерами.

Единицы измерения



Единицы измерения: "px", "em", "rem" и другие

Пиксель px – это самая базовая, абсолютная и окончательная единица измерения.

Количество пикселей задаётся в настройках разрешения экрана, один px – это как раз один такой пиксель на экране. Все значения браузер в итоге пересчитает в пиксели.

Достоинства

Главное достоинство пикселя – чёткость и понятность

Недостатки

Другие единицы измерения – в некотором смысле «мощнее», они являются относительными и позволяют устанавливать соотношения между различными размерами

Существуют также «производные» от пикселя единицы измерения: mm, cm, pt и pc, но они давно отправились на **свалку истории**.

1mm (мм) = 3.8px

1cm (см) = 38px

1pt (типографский пункт) = $\frac{4}{3}$ px

1pc (типографская пика) = 16px

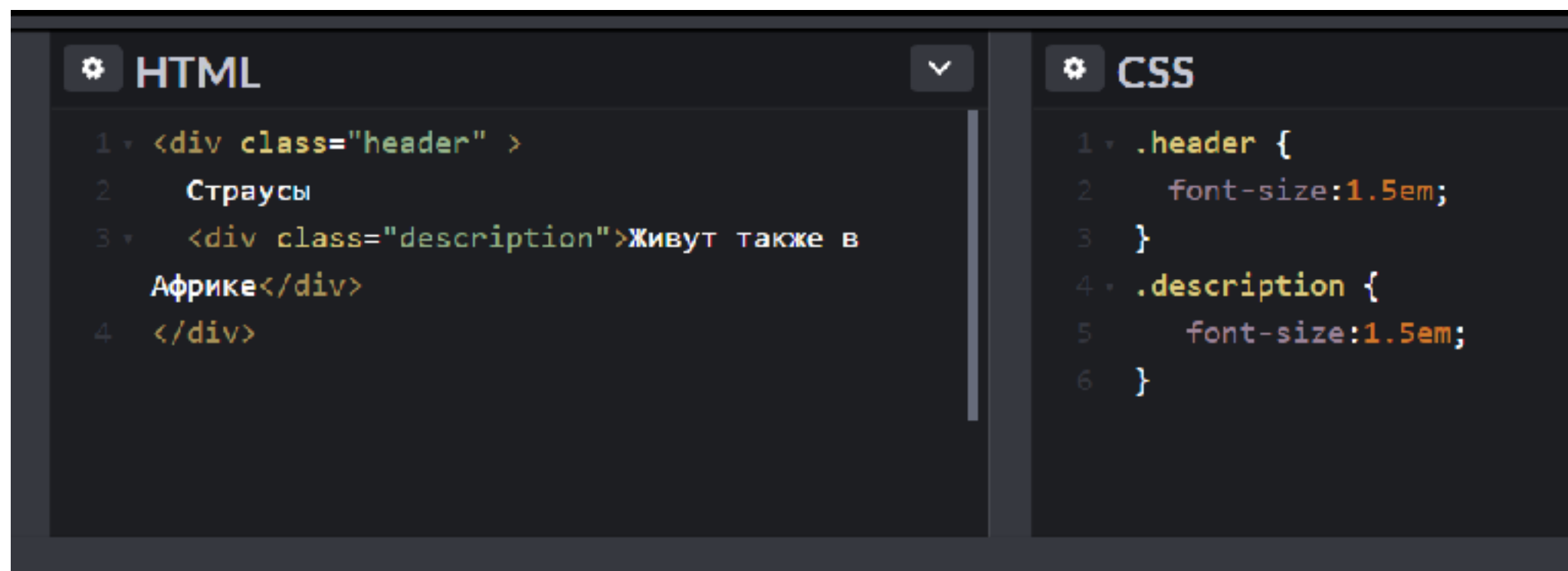
Так как браузер пересчитывает эти значения в пиксели, то смысла в их употреблении нет.

em

1em – текущий размер шрифта

Можно брать любые пропорции от текущего шрифта: 2em, 0.5em и т.п.

Размеры в em – *относительные*, они определяются по текущему контексту.



```
HTML
1 <div class="header" >
2   Страусы
3 <div class="description">Живут также в
4   Африке</div>
5 </div>

CSS
1 .header {
2   font-size:1.5em;
3 }
4 .description {
5   font-size:1.5em;
6 }
```

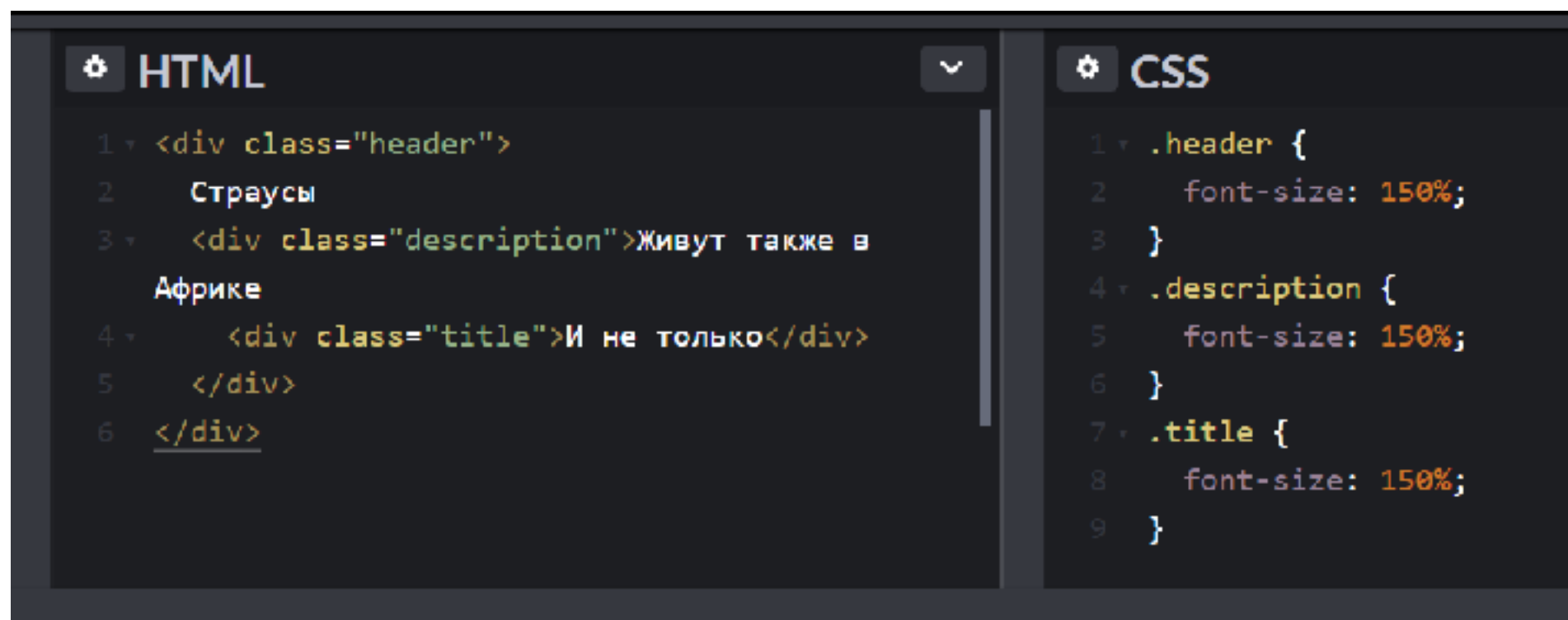
Страусы

Живут также в Африке

Проценты %, как и em – относительные единицы.

«Процент от чего?»

Как правило, процент будет от значения свойства родителя с тем же названием, но не всегда.



```
HTML
1 <div class="header">
2   Страусы
3   <div class="description">Живут также в
4     Африке
5     <div class="title">И не только</div>
6   </div>
7 </div>

CSS
1 .header {
2   font-size: 150%;
3 }
4 .description {
5   font-size: 150%;
6 }
7 .title {
8   font-size: 150%;
9 }
```

Страусы

Живут также в Африке

И не только

px – абсолютные, чёткие, понятные, не зависящие ни от чего.

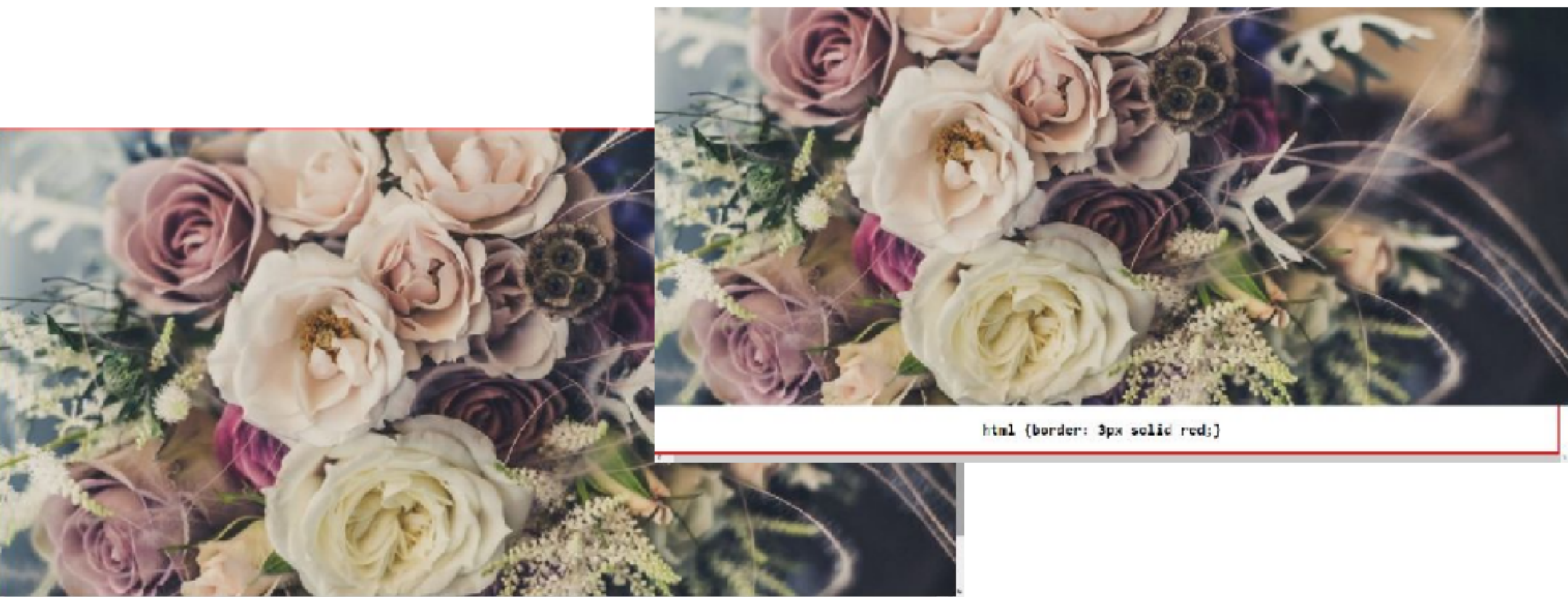
em – относительно размера шрифта.

% – относительно такого же свойства родителя.

В CSS3 были добавлены новые относительные единицы измерения `vh`, `vw`, `vmin`, `vmax`.

Эти единицы вычисляются относительно размеров окна браузера.

Для настольных компьютеров ширина окна браузера больше ширины области просмотра (добавляется ширина скроллбара), поэтому если для элемента установить ширину `100vw`, то он выйдет за пределы `html`.



```
html {border: 3px solid red;}
```


Единица	Описание
vh	Эквивалентно 1% высоты окна браузера.
vw	Эквивалентно 1% ширины окна браузера.
vmin	Эквивалентно 1% меньшего размера окна браузера по высоте или ширине.
vmax	Эквивалентно 1% большего размера окна браузера по высоте или ширине.

Единицы vh и vw

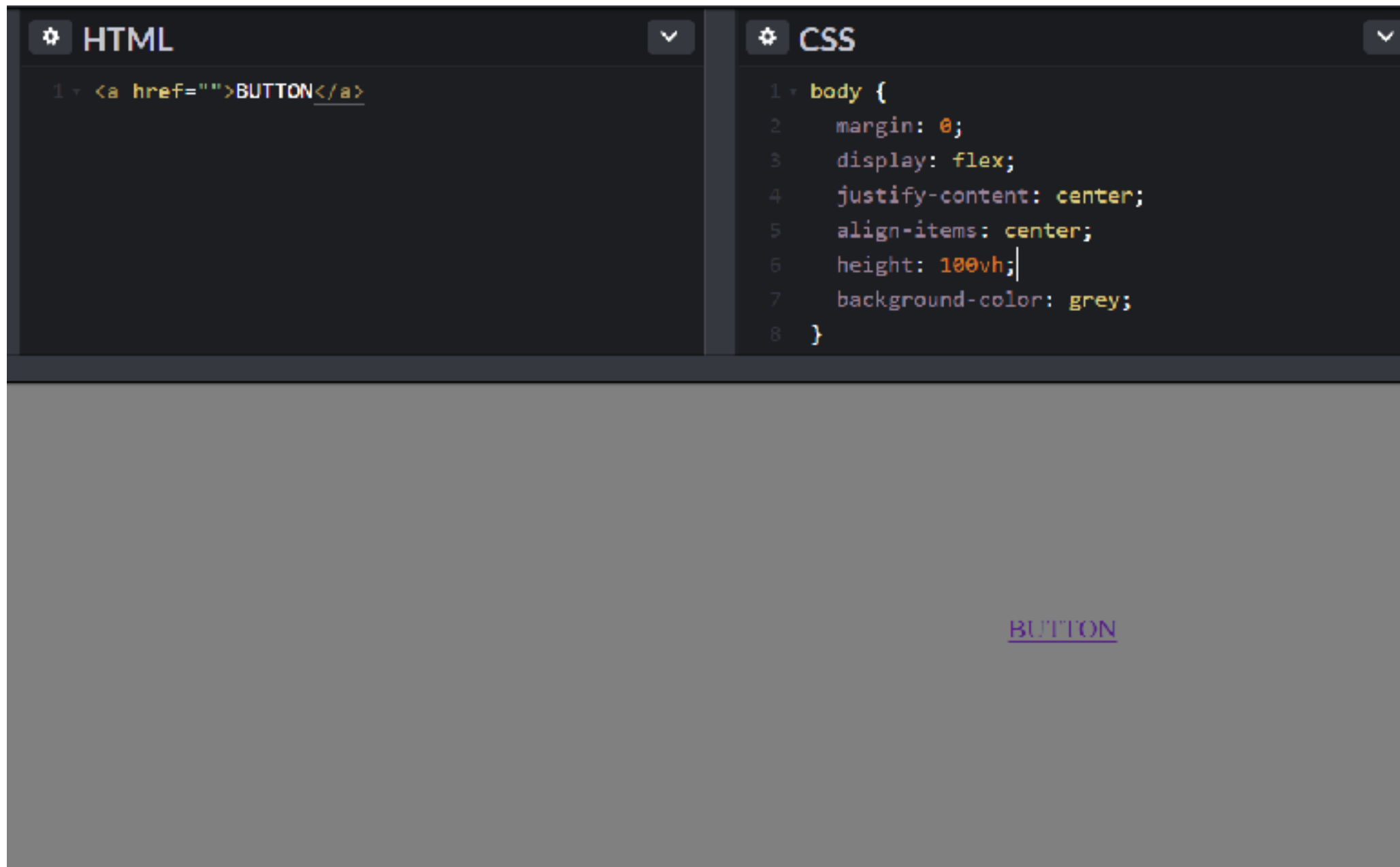
Проценты не лучшее решение для каждого случая, они вычисляются относительно размеров ближайшего родительского элемента.

Использовать высоту и ширину окна браузера - `vh` и `vw`.

Например, если высота окна браузера равна `900px`, то `1vh` будет равен `9px`. Аналогично, если ширина окна браузера равна `1600px`, то `1vw` будет равен `16px`.

Адаптивное полноэкранное фоновое изображение

Ширина элемента, указанная с помощью `100vw` больше ширины области просмотра, то для создания полноэкранных фоновых изображений лучше использовать ширину `100%`, которая будет равна ширине корневого элемента `html`.

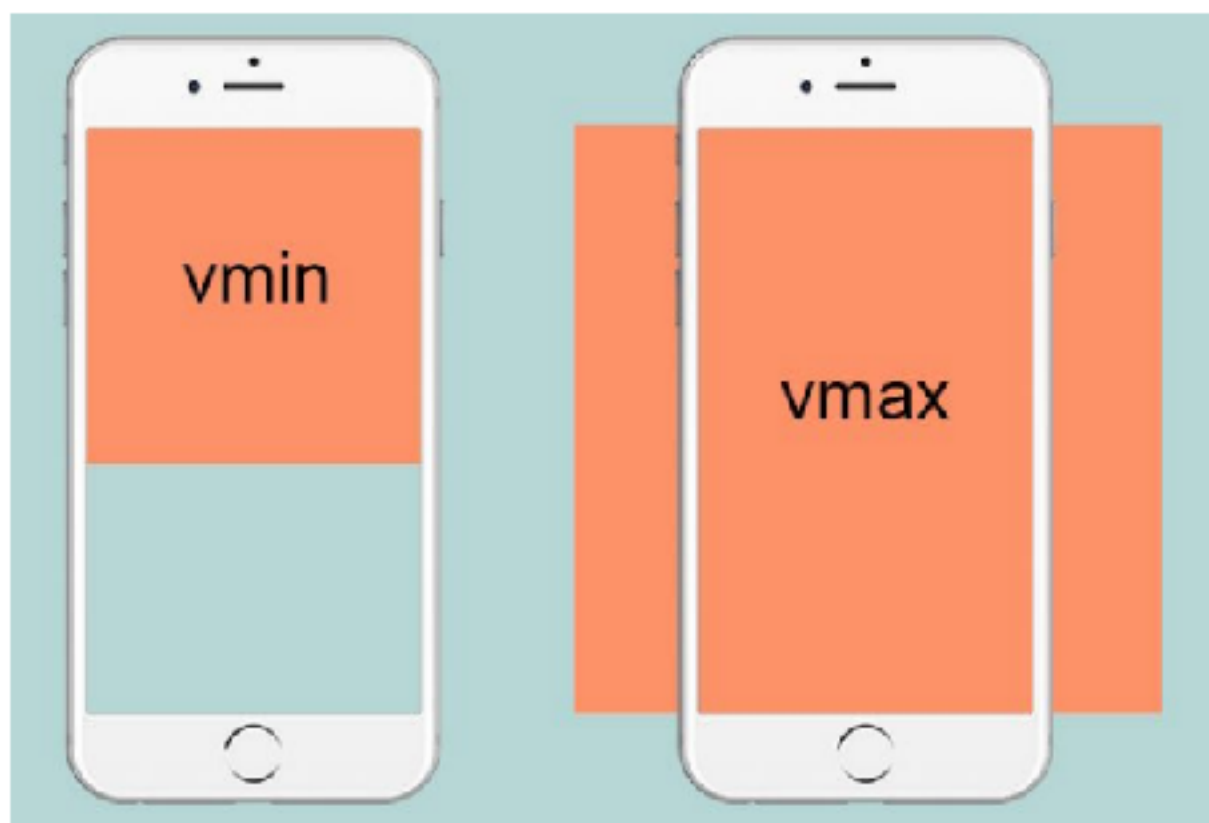


Единицы vmin и vmax

`vh` и `vw` всегда вычисляются относительно соответствующих размеров окна браузера

`vmin` и `vmax` определяются минимальным или максимальным значением высоты или ширины.

Например, если ширина окна браузера равна `1200px`, а высота `700px`, то `vmin` будет равен `7px`, а `vmax` — `12px`.

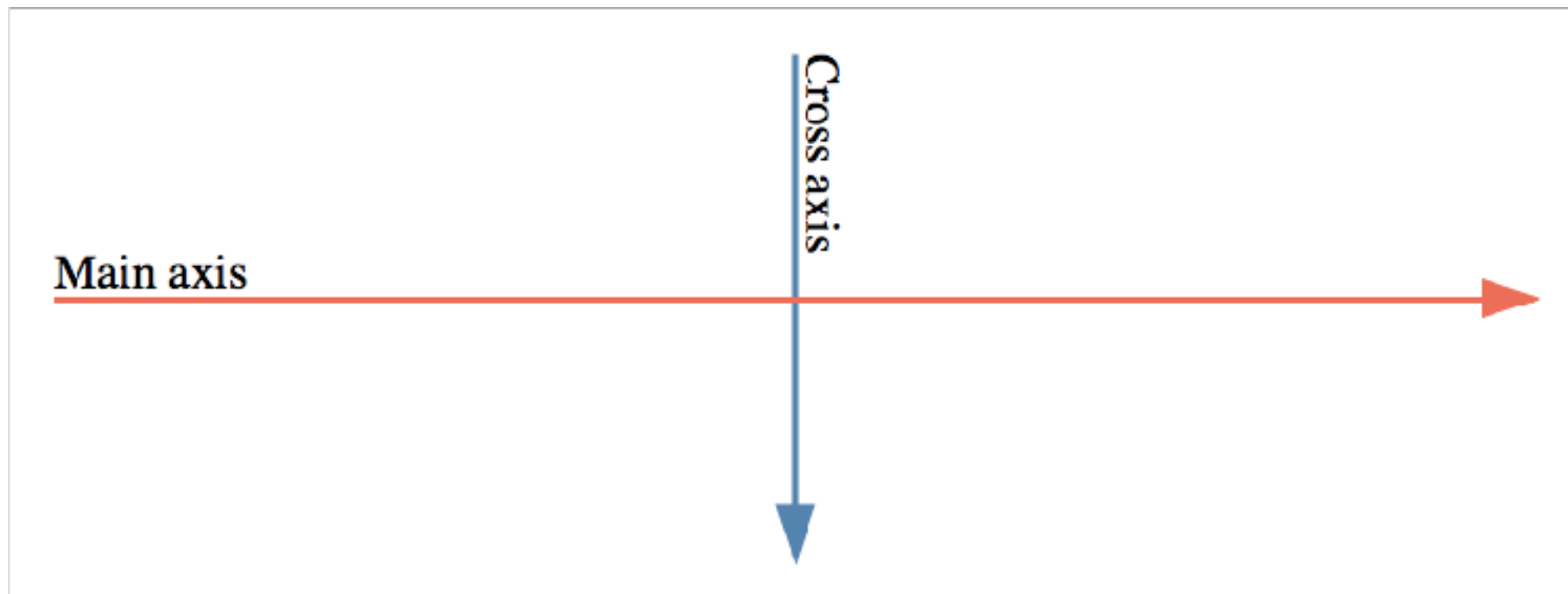


display: flex

Flexbox — это новый способ располагать блоки на странице. Это технология, созданная именно для раскладки элементов, в отличие от float-ов. С помощью **Flexbox** можно легко выравнивать элементы по горизонтали и по вертикали, менять направление и порядок отображение элементов, растягивать блоки на всю высоту родителя или прибивать их к нижнему краю.

Спецификация: <https://www.w3.org/TR/css-flexbox-1/>

Для начала надо знать, что flex-элементы располагаются по осям. По умолчанию элементы располагаются по горизонтали — вдоль **main axis** — главной оси.



При использовании **Flexbox** для внутренних блоков не работают float, clear и vertical-align

```

<body>
  <div class="flex-container">
    <div class="flex-item">
      Curabitur ac vestibulum mi
    </div>
    <div class="flex-item">
      In viverra dapibus
    </div>
    <div class="flex-item">
      Fusce tincidunt diam et
    </div>
    <div class="flex-item">
      Nulla in dui vel est
    </div>
    <div class="flex-item">
      at diam in lobortis
    </div>
  </div>

```

```

/* Decorations
----- */
BODY {
  padding: 20px;
  background: white;
}
.flex-container {
  padding: 10px;
  background: gold;
  border-radius: 10px;
}
.flex-item {
  margin: 10px;
  padding: 5px;
  background: tomato;
  border-radius: 5px;
  border: 1px solid #fff;
}

```

Curabitur ac
vestibulum mi

In viverra dapibus

Fusce tincidunt
diam et

Nulla in dui vel est

at diam in lobortis

Auto-run JS ☒

Run with JS

Curabitur ac
vestibulum mi

In viverra
dapibus

Fusce tincidunt
diam et

Nulla in dui vel
est

at diam in
lobortis

Родительскому элементу добавляем `display: flex;`.

Внутренние `div`-ы выстраиваются в ряд (вдоль главной оси) колонками одинаковой высоты, независимо от содержимого.

`display: flex;` делает все дочерние элементы резиновыми — `flex`, а не блочными — `block`, как было изначально.

Если родительский блок содержит картинки или текст без оберток, они становятся анонимными `flex`-элементами.

Curabitur ac
vestibulum mi

hendrerit
id magna

In viverra
dapibus

Fusce
tincidunt
diam et

Nulla in
dui vel est



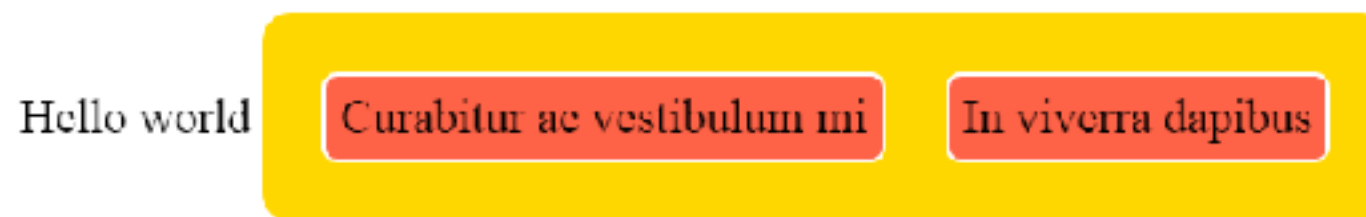
at diam in
lobortis

Свойство display для **Flexbox** может принимать два значения:

flex — ведёт себя как блочный элемент. При расчете ширины блоков приоритет у раскладки (при недостаточной ширине блоков контент может вылезать за границы).



inline-flex — ведёт себя как инлайн-блочный. Приоритет у содержимого (контент растопыривает блоки до необходимой ширины, чтобы строки, по возможности, поместились).



Flex-direction

Направление раскладки блоков управляется свойством flex-direction.

Возможные значения:

row — строка (значение по умолчанию);

row-reverse — строка с элементами в обратном порядке;

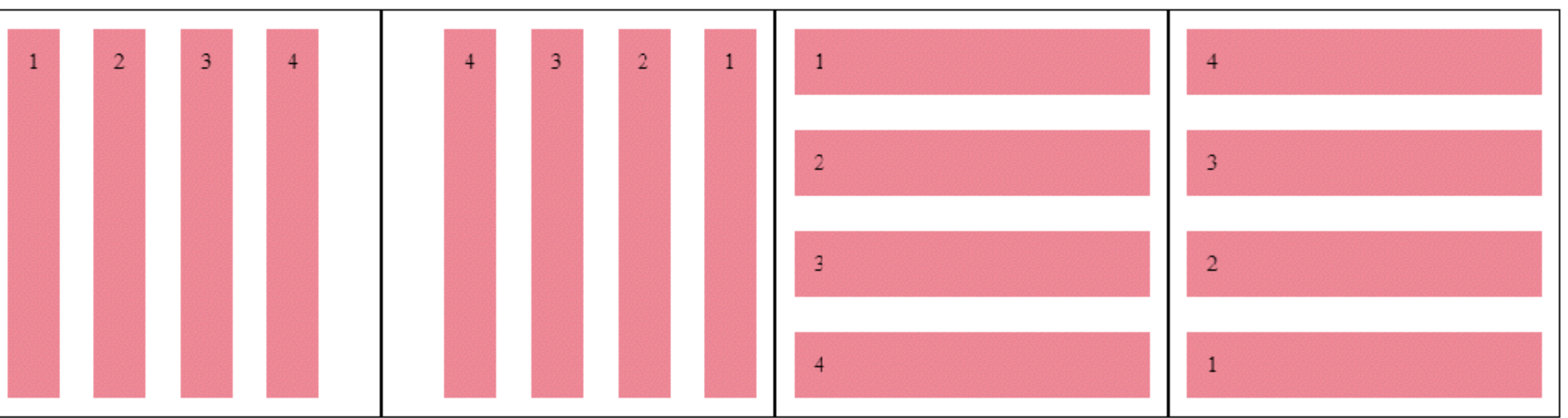
column — колонка;

column-reverse — колонка с элементами в обратном порядке.

row и row-reverse

<https://jsfiddle.net/kisnows/1ryjp43c/>

```
1  .wrap {
2    display: flex;
3  }
4  .container {
5    flex: 1;
6    display: flex;
7    border: 1px solid #000;
8    background: #FFF;
9  }
10 .item {
11   padding: 14px;
12   margin: 12px;
13   background: #ED8896;
14 }
15 .row {
16   flex-direction: row;
17 }
18 .row-reverse {
19   flex-direction: row-reverse;
20 }
21 .column {
22   flex-direction: column;
23 }
24 .column-reverse {
25   flex-direction: column-reverse;
26 }
```



Flex-wrap

В одной строке может быть много блоков. Переносятся они или нет определяет свойство flex-wrap.

Возможные значения:

nowrap — блоки не переносятся (значение по умолчанию);

wrap — блоки переносятся;

wrap-reverse — блоки переносятся и располагаются в обратном порядке.

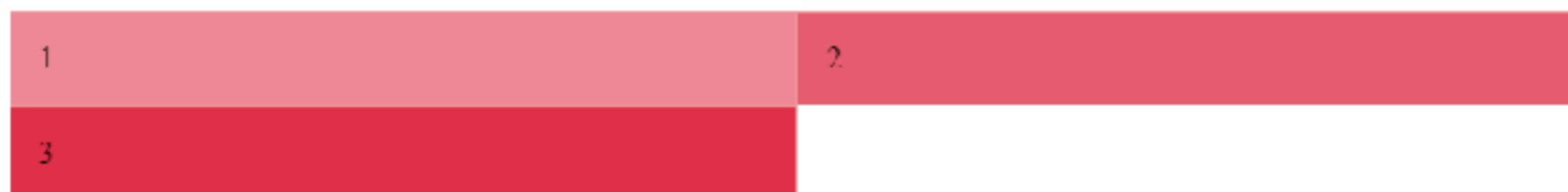
<https://jsfiddle.net/kisnows/3f7odnc6/>


```
27 ▾ .nowrap {  
28   flex-wrap: nowrap;  
29 }  
30  
31 ▾ .wrap {  
32   flex-wrap: wrap;  
33 }  
34  
35 ▾ .wrap-reverse {  
36   flex-wrap: wrap-reverse;  
37 }  
38
```

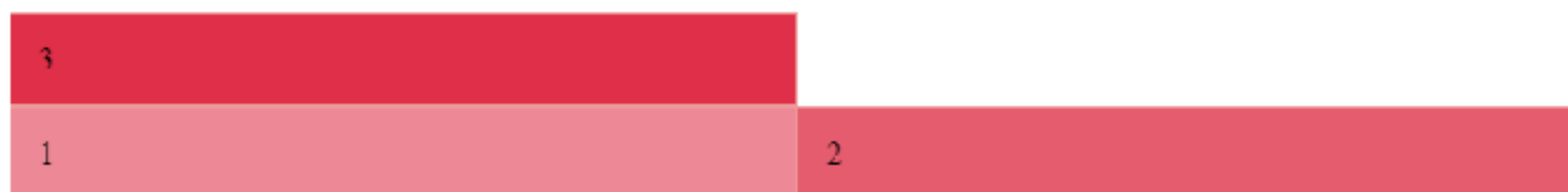
nowrap



wrap



wrap-reverse



Для короткой записи свойств flex-direction и flex-wrap существует свойство: flex-flow.

Возможные значения: можно задавать оба свойства или только какое-то одно.

Например:

flex-flow: column;

flex-flow: wrap-reverse;

flex-flow: column-reverse wrap;

<http://jsfiddle.net/mitrosin/w8rwmdww/>

```

1 * .flex-row {
2     webkit flex flow: row wrap;
3     -ms-flex-flow: row wrap;
4     flex flow: row wrap;
5 }
6
7 * .flex-column {
8     -webkit-flex-flow: column wrap;
9     -ms-flex-flow: column wrap;
10    flex-flow: column wrap;
11 }
12
13 * .flex-container {
14    width: 60%;
15    margin: 0 auto;
16    border: solid 1px #000000;
17    display: -webkit-flex;
18    display: -ms-flexbox;
19    display: flex;
20 }
21
22 * .flex-item {
23    border: solid 2px #FF0000;

```

Flex Flow : Row and Wrapping

Item One	Item Two	Item Three
----------	----------	------------

Flex Flow : Row and Wrapping

Item One
Item Two
Item Three

Order

Для управления порядком блоков служит свойство `order`.

Возможные значения: числа.

Чтобы поставить блок самым первым, задайте ему `order: -1`;

<https://jsfiddle.net/blayderunner123/h09geqz2/>

```
153 ▾ .flex-item:nth-child(1){
154     display:block;
155     width:33.333%;
156     text-align:center;
157     background: gold;
158     order:1;
159 }
160
161 ▾ .flex-item:nth-child(2){
162     display:block;
163     width:33.333%;
164     text-align:center;
165     background: hotpink;
166     order:2;
167 }
168
169 ▾ .flex-item:nth-child(3){
170     display:block;
171     width:33.333%;
172     text-align:center;
173     background: orange;
174     order:3;
175 }
176 }
```



Для выравнивания элементов есть несколько свойств:

`justify-content`, `align-items` и `align-self`.

`justify-content` и `align-items` применяются к родительскому контейнеру,

`align-self` — к дочерним.

`justify-content` отвечает за выравнивание по главной оси.

Возможные значения `justify-content`:

`flex-start` — элементы выравниваются от начала главной оси (значение по умолчанию);

`flex-end` — элементы выравниваются от конца главной оси;

`center` — элементы выравниваются по центру главной оси;

`space-between` — элементы выравниваются по главной оси, распределяя свободное место между собой;

`space-around` — элементы выравниваются по главной оси, распределяя свободное место вокруг себя.

flex-start



flex-end



center



space-between



space-around



Align-items

align-items отвечает за выравнивание по перпендикулярной оси.

Возможные значения align-items:

flex-start — элементы выравниваются от начала перпендикулярной оси;

flex-end — элементы выравниваются от конца перпендикулярной оси; center — элементы выравниваются по центру;

baseline — элементы выравниваются по базовой линии;

stretch — элементы растягиваются, занимая все пространство по перпендикулярной оси (значение по умолчанию).

flex-start



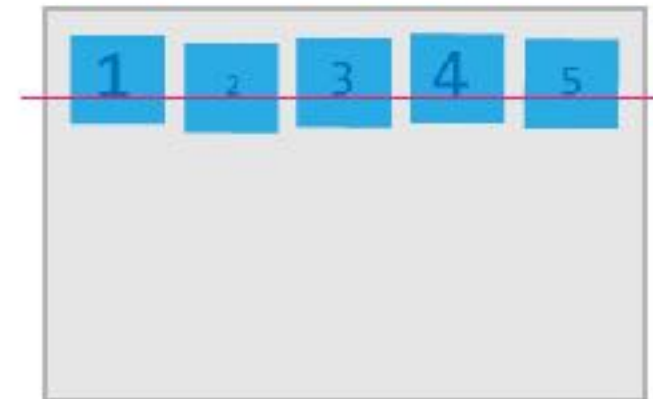
flex-end



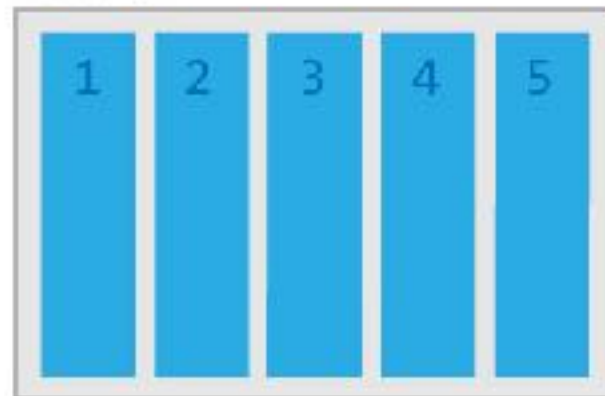
center



baseline



stretch



Align-Items

Align-self

align-self также отвечает за выравнивание по перпендикулярной оси, но задается отдельным flex-элементам.

Возможные значения align-self:

auto — значение по умолчанию. Означает, что элемент использует align-items родительского элемента;

flex-start — элемент выравнивается от начала перпендикулярной оси;

flex-end — элемент выравнивается от конца перпендикулярной оси;

center — элемент выравнивается по центру;

baseline — элемент выравнивается по базовой линии;

stretch — элемент растягивается, занимая все пространство по высоте.

Align-self



Align-content

Для управления выравниванием внутри многострочного flex-контейнера есть свойство align-content.

Возможные значения:

flex-start — элементы выравниваются от начала главной оси;

flex-end — элементы выравниваются от конца главной оси;

center — элементы выравниваются по центру главной оси;

space-between — элементы выравниваются по главной оси, распределяя свободное место между собой;

space-around — элементы выравниваются по главной оси, распределяя свободное место вокруг себя;

stretch — элементы растягиваются, заполняя всю высоту (значение по умолчанию).

align-content

stretch
(по умолчанию)

1	2	3	4	5
6	7	8	9	10

flex-start

1	2	3	4	5
6	7	8	■	10

flex-end

1	2	3	4	5
6	7	8	9	10

center

1	2	3	4	5
6	7	8	9	10

space-between

1	2	3	4	5
■	7	■	9	10

space-around

1	2	3	4	5
6	7	8	9	10