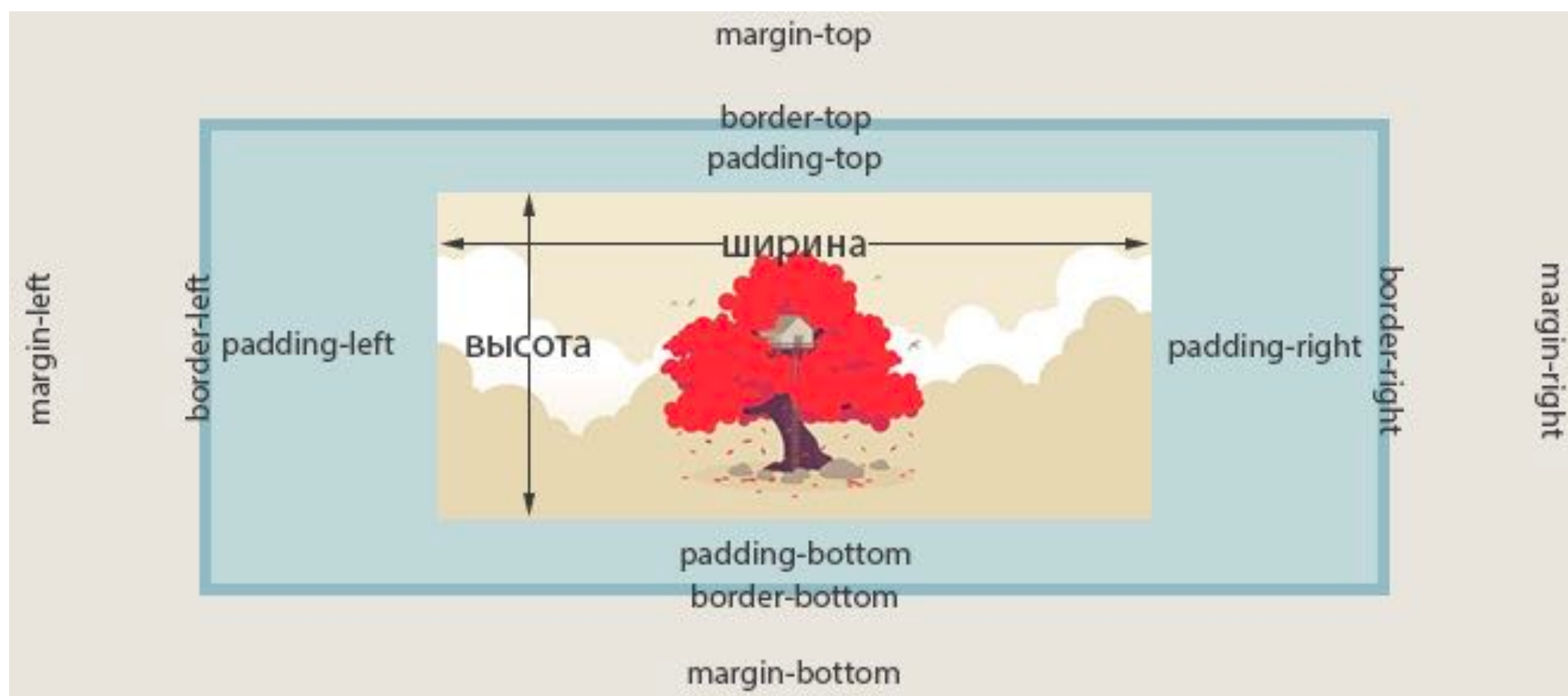


Box model

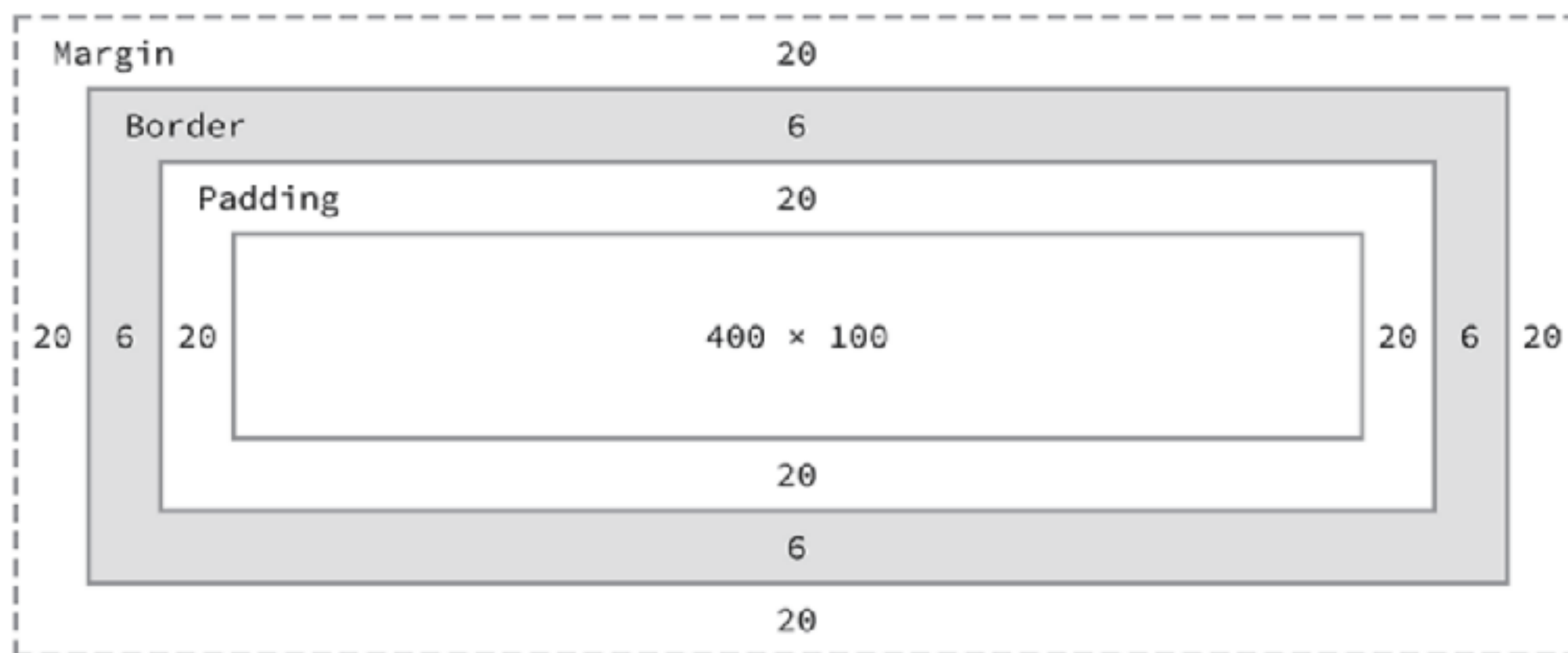
Vira Huskova

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.

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.

ые 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/anon/pen/BqJvOJ>

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

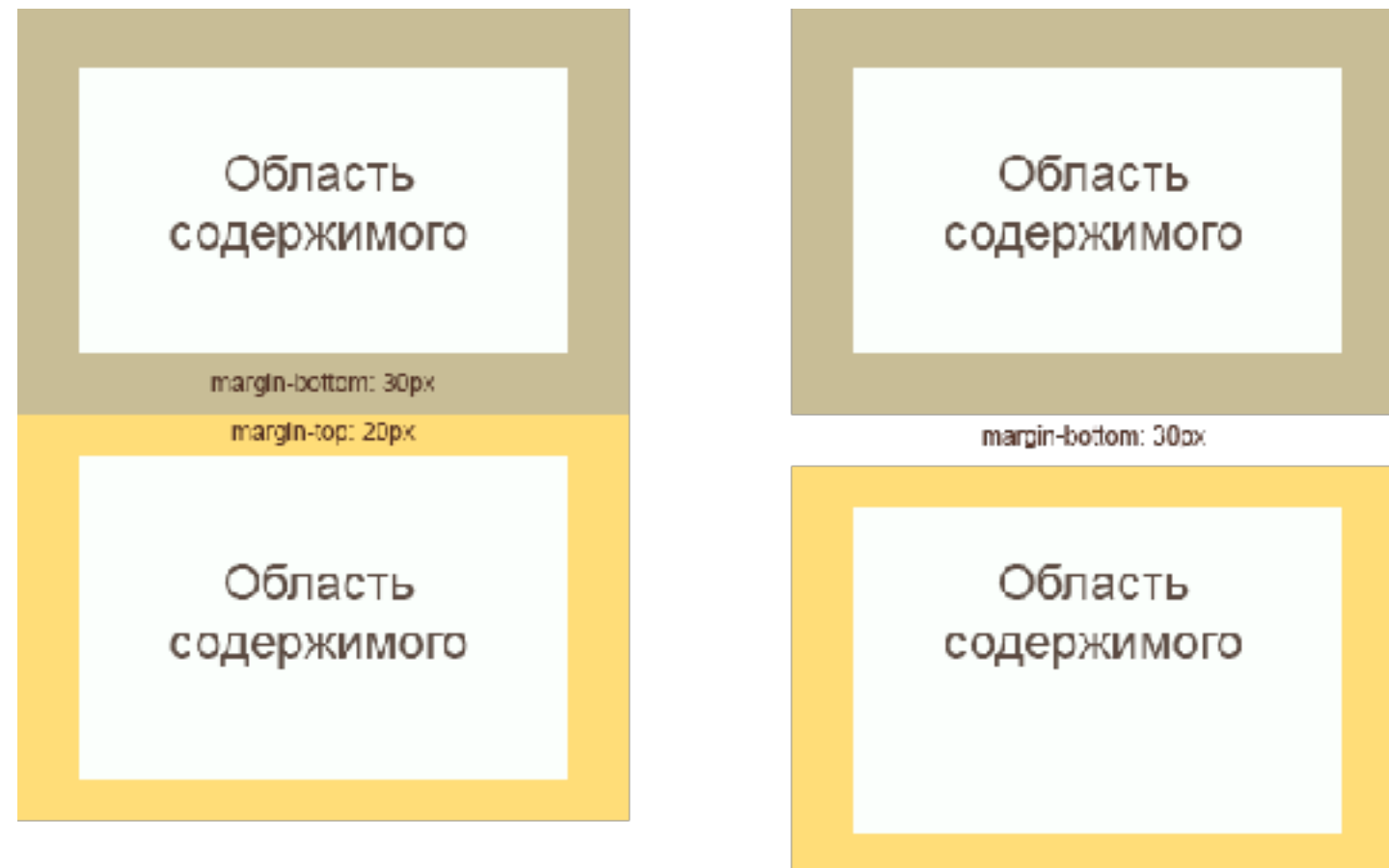
В некоторых случаях бывает необходимо, чтобы строчный элемент вел себя как блочный и наоборот. Для этого необходимо установить соответствующее значение свойства **display**:

```
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.

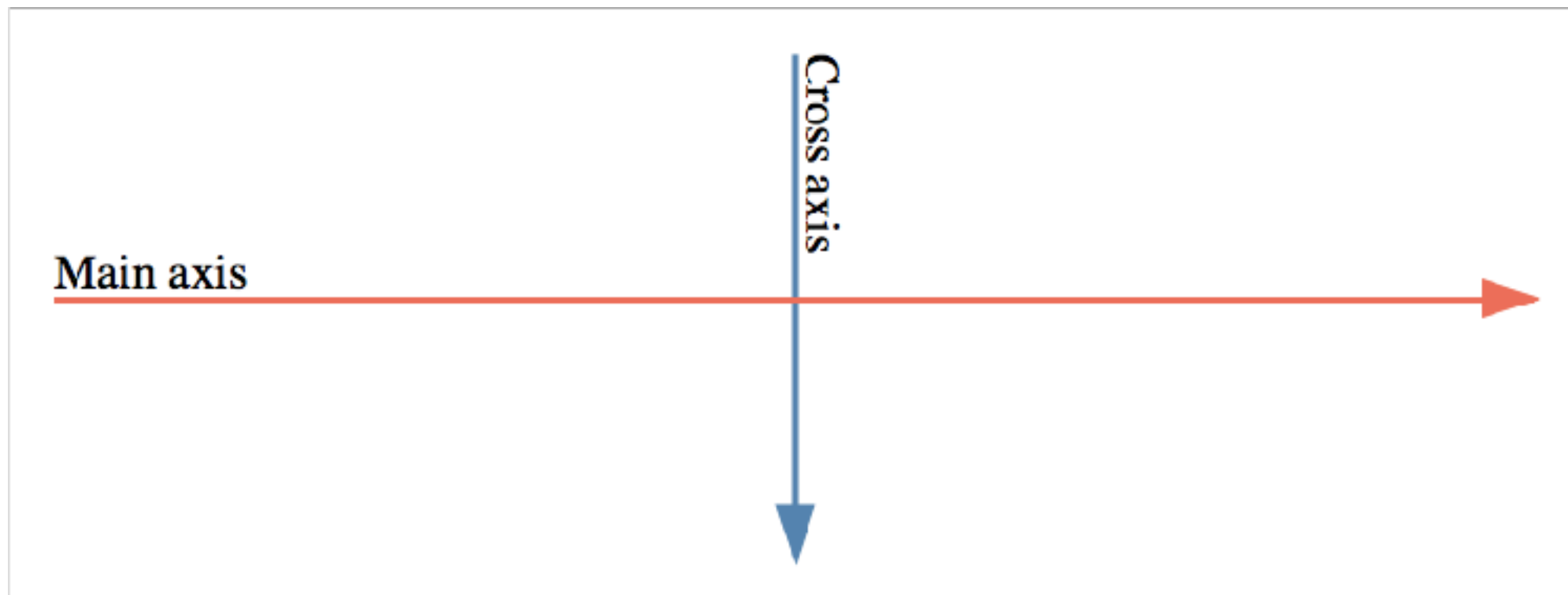
display: flex

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 отображения 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.

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

Curabitur ac
vestibulum mi

hendrerit
id magna

In viverra
dapibus

Fusce
tincidunt
diam et

Nulla in
dui vel est



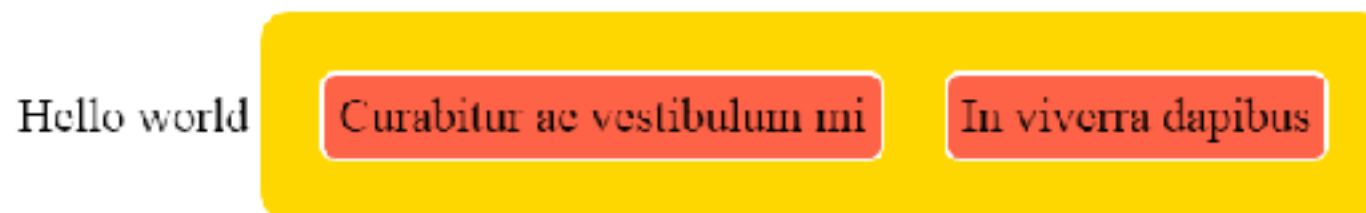
at diam in
lobortis

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). **Ширина внутренних блоков может выходить за рамки родителя (если размер родителя фиксированно задан).**



inline-flex — ведёт себя как инлайн-блочный. Приоритет у содержимого (контент растопыривает блоки до необходимой ширины, чтобы строки, по возможности, поместились). **Ширина родительского блока подстраивается под ширину дочерних.**



Flex-direction

Направление раскладки блоков управляется свойством flex-direction.

Возможные значения:

row — строка (значение по умолчанию);

row-reverse — строка с элементами в обратном порядке;

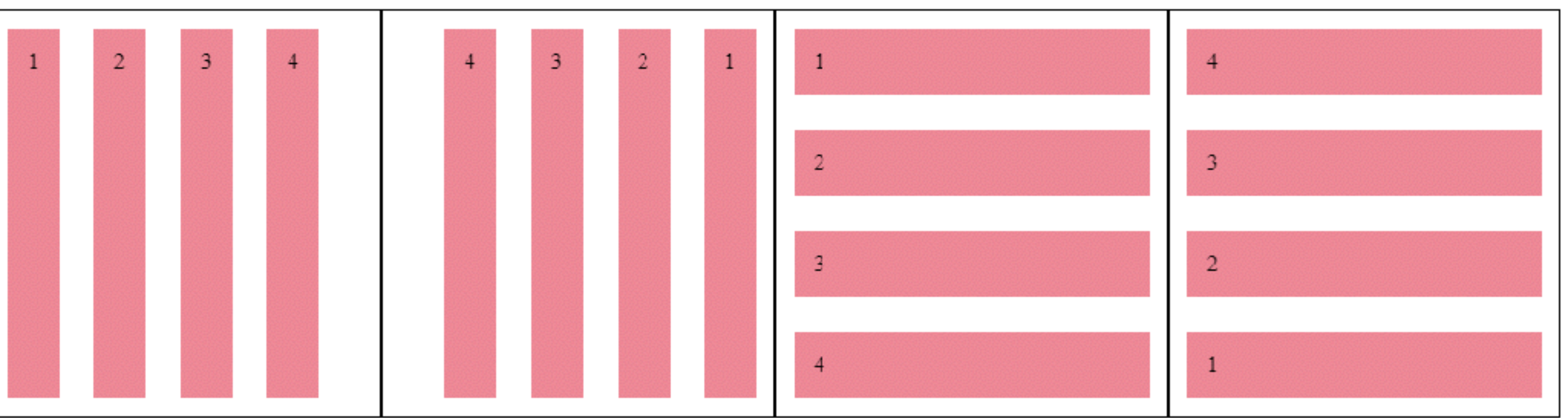
column — колонка;

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-grow: 0  
flex-shrink: 1
```

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.

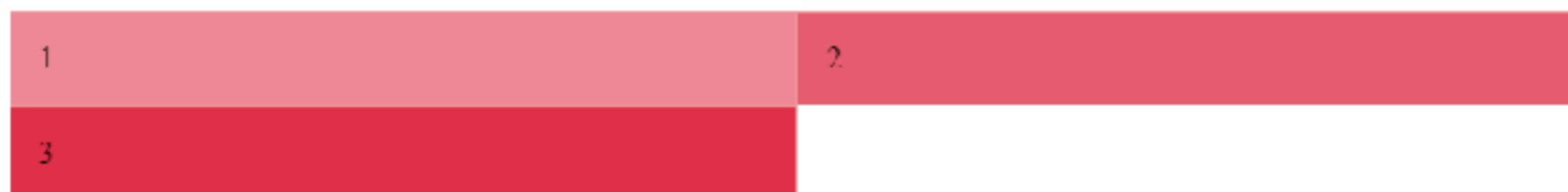
<https://jsfiddle.net/kisnows/3f7odnc6/>


```
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
```

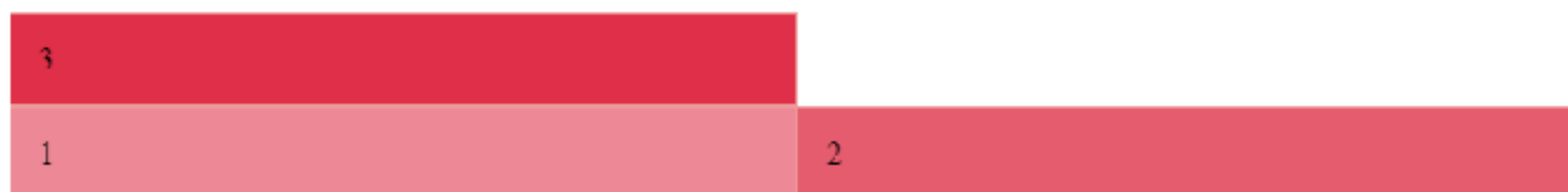
nowrap



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



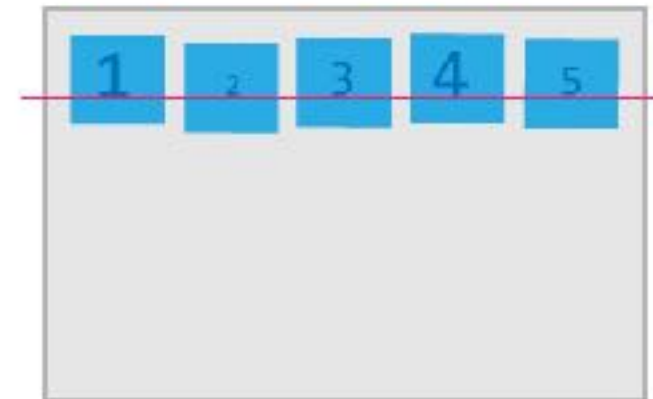
flex-end



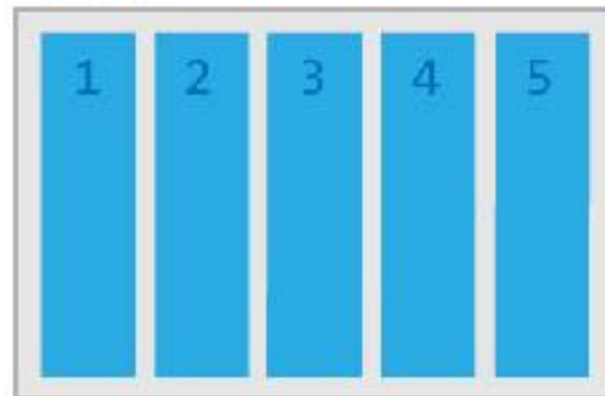
center



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

Flex generator

<http://bennettfeely.com/flexplorer/>

Свойство "float"

Свойство float в CSS занимает особенное место. До его появления расположить два блока один слева от другого можно было лишь при помощи таблиц.

float: left | right | none | inherit;

При применении этого свойства происходит следующее:

1. Элемент позиционируется как обычно
2. Затем *вынимается из документа потока* и сдвигается влево (для left) или вправо (для right) до того **как коснётся** либо границы **родителя**, либо другого **элемента с float**.

Если пространства по горизонтали не хватает для того, чтобы вместить элемент, то он сдвигается вниз до тех пор, пока не начнёт помещаться.

Другие непозиционированные **блочные элементы без float** ведут себя так, как **будто элемента с float нет**, так как он убран из потока.

Строки (**inline-элементы**), напротив, «**знают**» о **float** и обтекают элемент по сторонам.

Элемент при наличии float получает display:block.

То есть, указав элементу, у которого display:inline свойство float: left/right, мы автоматически сделаем его блочным.

В частности, для него будут работать width/height.

Исключением являются некоторые редкие display наподобие inline-table.

Ширина float-блока определяется по содержимому.

Вертикальные отступы margin элементов с float не сливаются с отступами соседей, в отличие от обычных блочных элементов.

Одно из первых применений float, для которого это свойство когда-то было придумано – это вёрстка текста с картинками, отжатыми влево или вправо.

Винни-Пух

Винни-Пух (англ. Winnie-the-Pooh) — плюшевый мишка, персонаж повестей и стихов Алана Александра Милна (цикл не только о Винни-Пухе, но и обычно тоже называется «Винни-Пух»), один из самых известных героев детской литературы. В 1960-е—1970-е годы, благодаря пересказу Бориса Заходера «Винни-Пух и все-все-все», а затем и фильмам «Винни-Пух и его друзья» и «Винни-Пух и день забот», выпущенным на «Союзмультфильм», где мишку озвучивал Евгений Леонидов, Винни-Пух стал очень популярен и в Советском Союзе.



Первый перевод «Винни-Пуха» в СССР вышел в 1958 году в Литве (лит. Mikė Pūkuotukas), его выполнил 20-летний литовский писатель Виргиліус Чепайтіс, пользовавшийся польским переводом Ирены Тувим. Впоследствии Чепайтіс, познакомившись с английским оригиналом, с переработкой переработал свой перевод, переиздававшийся неоднократно.

История Винни-Пуха в России начинается с того же 1958 года, когда с книгой познакомился Борис Владимирович Заходер. В студии «Союзмультфильм» под руководством Фёдора Хитрука было создано три мультфильма. Сценарий написал Хитрук в соавторстве с Заходером; работа соавторов не всегда шла гладко, что стало в конечном счёте причиной прекращения выпуска мультфильмов (первоначально планировалось выпустить сериал по всей книге). Текст и картинки взяты с Wikipedia

float: right



float: right



The Normal Document Flow



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.

Right-Floating Image

image margin ↓

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.



Right-Floating Image

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.



Если текста больше, то абзац будет обтекать картинку:

Right-Floating Image

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent euismod ultrices ante, ac laoreet nulla vestibulum adipiscing. Nam quis justo in augue auctor imperdiet. Curabitur aliquet orci sit amet est posuere consectetur. Fusce nec leo ut massa viverra venenatis. Nam accumsan libero a elit aliquet quis ullamcorper arcu tincidunt. Praesent purus turpis, consectetur quis congue vel, pulvinar at lorem. Vivamus varius condimentum dolor, quis ultricies ipsum porta quis.



Девять правил float-элементов:

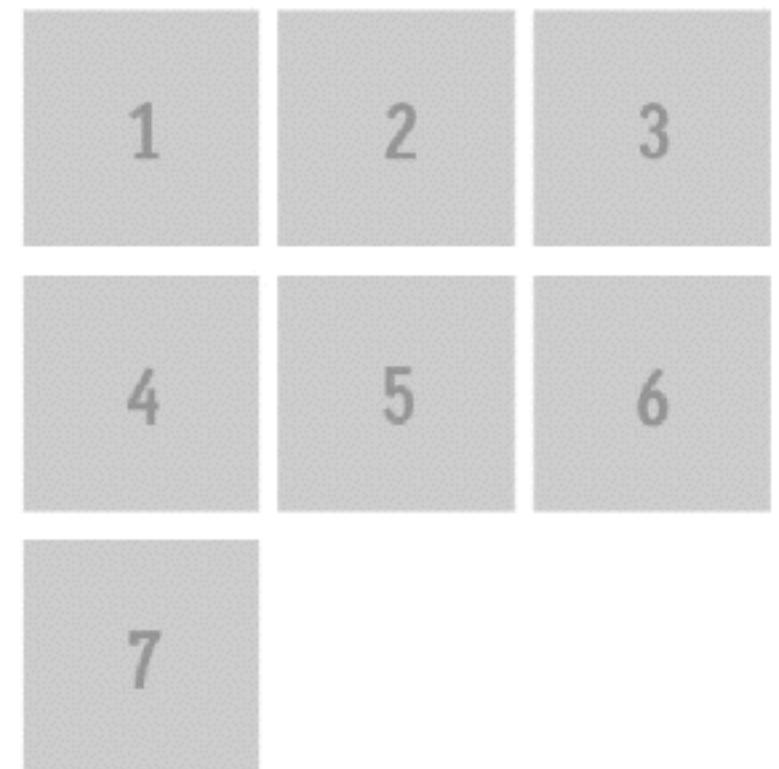
1. Плавающие элементы не могут выходить за край своего контейнера-родителя.
2. Каждый плавающий элемент будет отображаться справа или ниже от предыдущего при float:left, либо слева и ниже, при float:right.
3. Блок с float:left не может быть правее, чем блок с float:right.
4. Плавающий элемент не может выходить за пределы верхней границы своего контейнера.
5. Плавающий элемент не может располагаться выше, чем родительский блок или предыдущий плавающий элемент.
6. Плавающий элемент не может располагаться выше, чем предыдущая строка inline-элементов
7. Плавающий блок должен быть расположен как можно выше.
8. Один плавающий элемент, следующий за другим, не может выходить за пределы своего контейнера — происходит перенос на следующую строку.
9. Блок с float:left должен быть расположен как можно левее, а с float:right — как можно правее.

Пример галереи

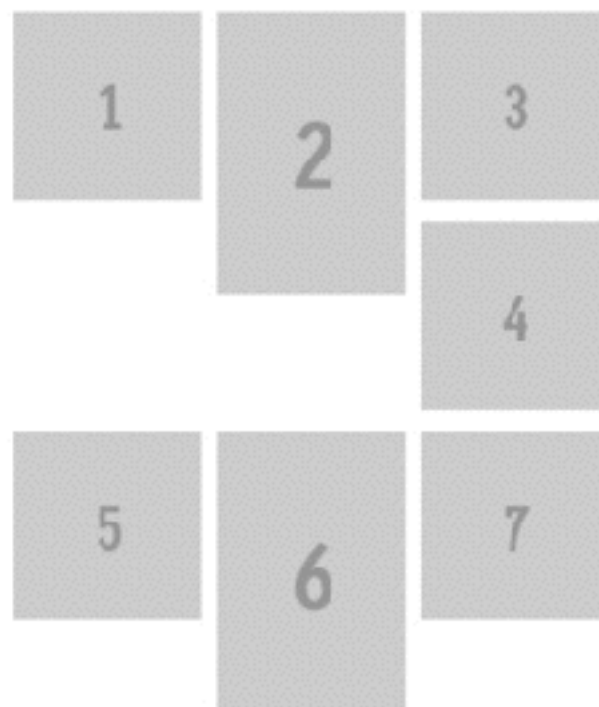
По умолчанию каждый элемент списка появится с новой строки. Если применить к каждому `float:left`, изображения встанут в один ряд с переносом строки:

```
<ul>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
</ul>
```

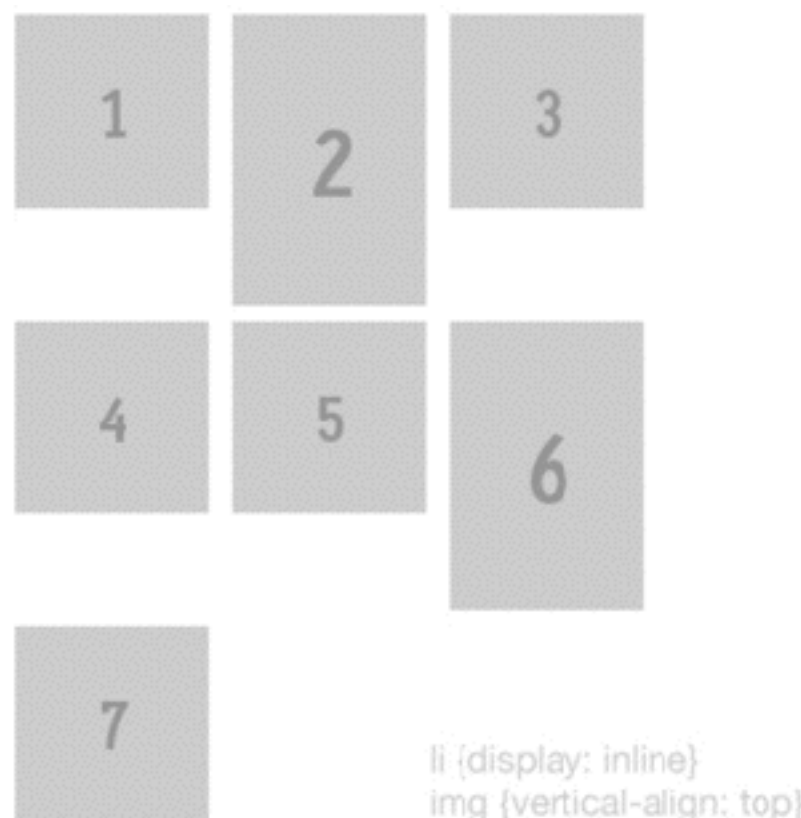
```
li {
  float: left;
  margin: 4px;
}
```



Если изображения разной высоты

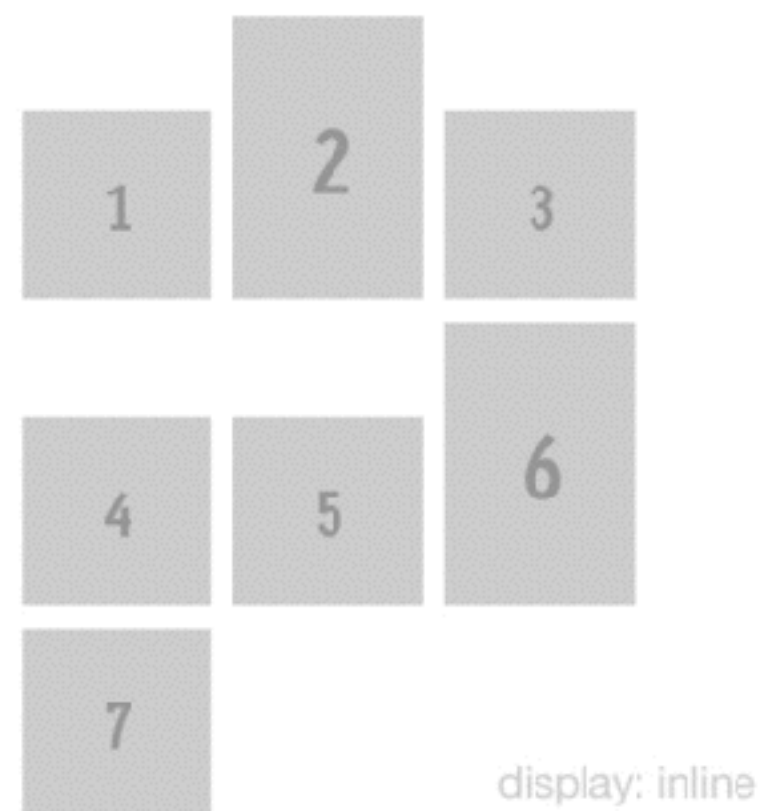


Выравниваем по вертикали



Добавим элементам списка отображение в одну строку

```
li {  
  display: inline;  
}
```



Свойство clear

clear: left | right | both;

Центрирование горизонтальное и вертикальное

В CSS есть всего несколько техник центрирования элементов.

Горизонтальное

text-align

Для центрирования инлайновых элементов – достаточно поставить родителю text-align: center (left/right);

```
1 <style>
2   .outer {
3     text-align: center;
4     border: 1px solid blue;
5   }
6 </style>
7
8 <div class="outer">Текст</div>
```

Текст

Для центрирования блока это уже не подойдёт, свойство просто не подействует. Например:


```
1 <style>
2   .outer {
3     text-align: center;
4     border: 1px solid blue;
5   }
6   .inner {
7     width: 100px;
8     border: 1px solid red;
9   }
10 </style>
11
12 <div class="outer">
13   <div class="inner">Текст</div>
14 </div>
```

Текст

margin: auto

Блок по горизонтали центрируется margin: auto

```
1 <style>
2   .outer {
3     border: 1px solid blue;
4   }
5   .inner {
6     width: 100px;
7     border: 1px solid red;
8     margin: auto;
9   }
10 </style>
11
12 <div class="outer">
13   <div class="inner">Текст</div>
14 </div>
```



Текст

В отличие от width/height, значение auto для margin само не появляется. Обычно margin равно конкретной величине для элемента, например 0 для DIV. Нужно поставить его явно.

Значение margin-left:auto/margin-right:auto заставляет браузер выделять под margin всё доступное сбоку пространство. А если и то и другое auto, то слева и справа будет одинаковый отступ, таким образом элемент окажется в середине.

Вертикальное

Вертикальное же изначальное не было предусмотрено в спецификации CSS

position:absolute + margin

```
1 <style>
2   .outer {
3     position: relative;
4     height: 5em;
5     border: 1px solid blue;
6   }
7
8   .inner {
9     position: absolute;
10    top: 50%;
11    border: 1px solid red;
12  }
13 </style>
14
15 <div class="outer">
16   <div class="inner">Текст</div>
17 </div>
```

Текст


Конечно, это не совсем центр. По центру находится верхняя граница. Нужно ещё приподнять элемент на половину своей высоты.

Высота центрируемого элемента должна быть известна. Родитель может иметь любую высоту.

Если мы знаем, что это ровно одна строка, то её высота равна line-height.

Приподнимем элемент на пол-высоты при помощи margin-top:

```
1 <style>
2   .outer {
3     position: relative;
4     height: 5em;
5     border: 1px solid blue;
6   }
7
8   .inner {
9     position: absolute;
10    top: 50%;
11    margin-top: -0.625em;
12    border: 1px solid red;
13  }
14 </style>
15
16 <div class="outer">
17   <div class="inner">Текст</div>
18 </div>
```



Текст

transform: translate(-50%, -50%)

```
1 <style>
2   .outer {
3     position: relative;
4     height: 5em;
5     border: 1px solid blue;
6   }
7
8   .inner {
9     position: absolute;
10    top: 50%;
11    transform: translate(-50%, -50%);
12    border: 1px solid red;
13  }
14 </style>
15
16 <div class="outer">
17   <div class="inner">Текст</div>
18 </div>
```

Текст

Одна строка: line-height

Вертикально отцентрировать одну строку в элементе с известной высотой `height` можно, указав эту высоту в свойстве `line-height`:

```
1 <style>
2   .outer {
3     height: 5em;
4     line-height: 5em;
5     border: 1px solid blue;
6   }
7 </style>
8
9 <div class="outer">
10   <span style="border:1px solid red">Текст</span>
11 </div>
```



Текст

Это работает, но лишь до тех пор, пока строка одна, а если содержимое вдруг переносится на другую строку, то контент начинает сдвигаться.

Таблица с vertical-align

У свойства **vertical-align**, которое управляет вертикальным расположением элемента, есть два режима работы.

В таблицах свойство vertical-align указывает расположение содержимого ячейки.

Его возможные значения:

Baseline Значение по умолчанию.
middle, top, bottom

Располагать содержимое посередине, вверху, внизу ячейки.

```
1 <style>
2   table { border-collapse: collapse; }
3   td {
4     border: 1px solid blue;
5     height: 100px;
6   }
7 </style>
8
9 <table>
10 <tr>
11   <td style="vertical-align: top">top</td>
12   <td style="vertical-align: middle">middle</td>
13   <td style="vertical-align: bottom">bottom</td>
14 </tr>
15 </table>
```

top	
	middle
	bottom

В ячейке с vertical-align: middle содержимое находится по центру. Таким образом, можно обернуть нужный элемент в таблицу размера width:100%;height:100% с одной ячейкой, у которой указать vertical-align:middle, и он будет отцентрирован.

Рассмотрим более красивый способ, который поддерживается во всех современных браузерах, и в IE8+.

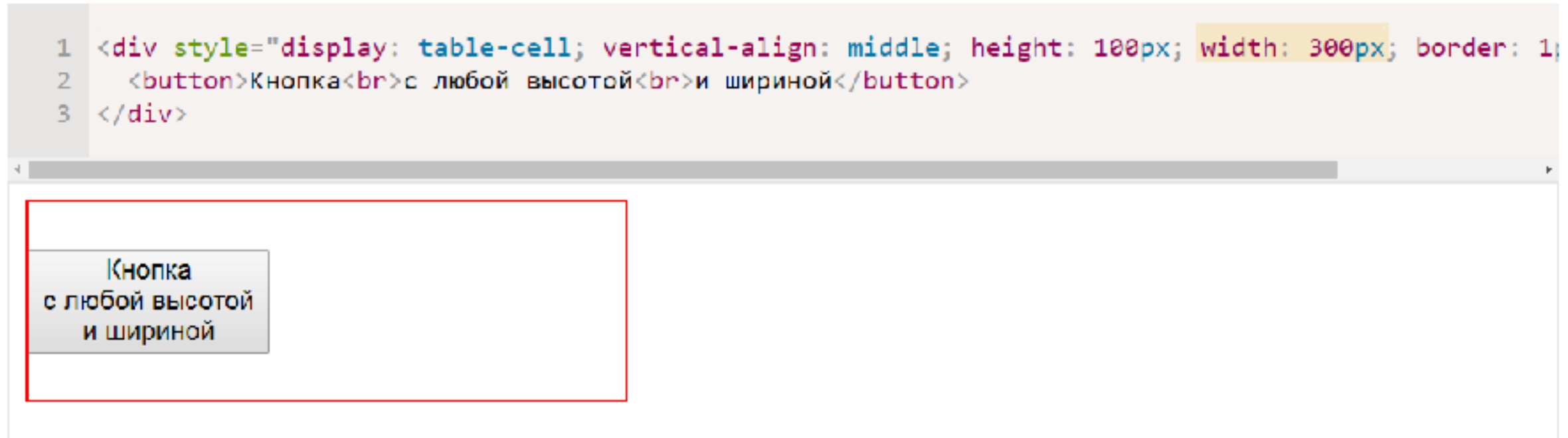
В них не обязательно делать таблицу, так как доступно значение display:table-cell. Для элемента с таким display используются те же алгоритмы вычисления ширины и центрирования, что и в TD. И, в том числе, работает vertical-align:



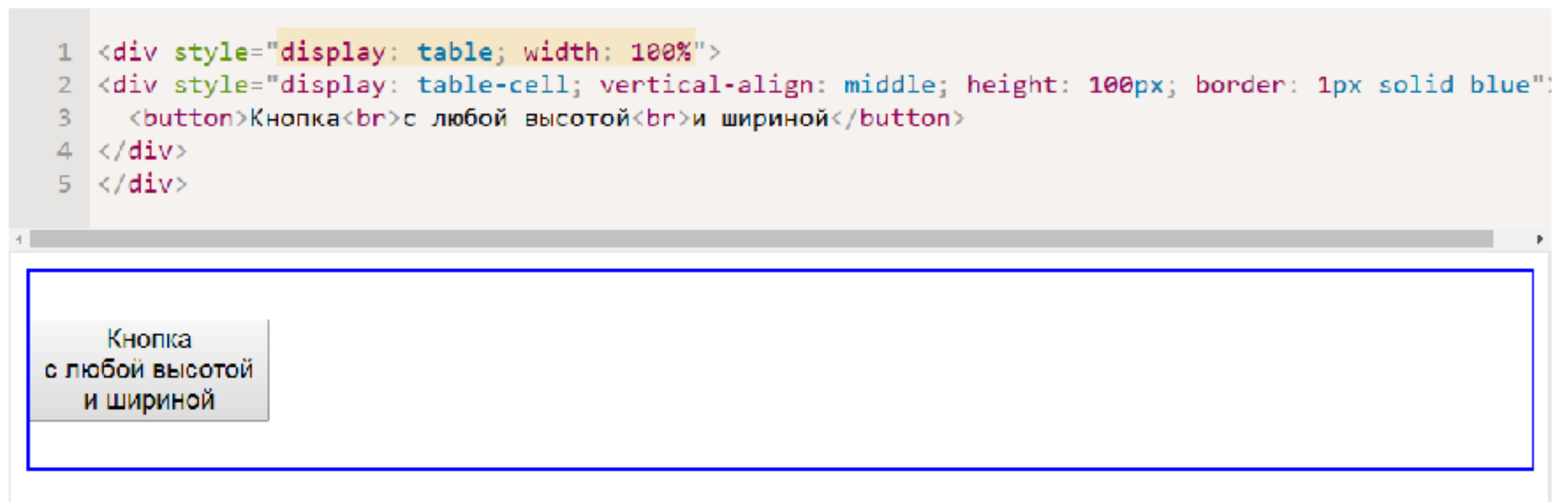
Этот способ замечателен тем, что он не требует знания высоты элементов.

Однако у него есть особенность. Вместе с vertical-align родительский блок получает табличный алгоритм вычисления ширины и начинает подстраиваться под содержимое. Это не всегда желательно.

Чтобы его растянуть, нужно указать width явно, например: 300px:



Можно завернуть «псевдоячейку» в элемент с display:table, которому и поставим ширину:



Если дополнительно нужно горизонтальное центрирование – оно обеспечивается другими средствами, например `margin: 0 auto` для блочных элементов или `text-align:center` на родителе – для других.

Центрирование в строке с `vertical-align`

Для инлайновых элементов (`display:inline/inline-block`), включая картинки, свойство `vertical-align` центрирует *сам инлайн-элемент в окружающем его тексте*.

Картинка размером в 30px, значения `vertical-align`:




The diagram illustrates the vertical alignment of a 30px image with different text elements using various `vertical-align` values. Each example shows the text and the image (a red square with the number 30) aligned according to the specified value:

- `baseline`(по умолчанию) 30: The image is aligned with the baseline of the text.
- `middle`(по середине) 30: The image is aligned with the middle of the text.
- `sub` 30: The image is aligned with the bottom of the text, as if it were a subscript.
- `super` 30: The image is aligned with the top of the text, as if it were a superscript.
- `text-top`(верхняя граница вровень с текстом) 30: The top of the image is aligned with the top of the text.
- `text-bottom`(нижняя граница вровень с текстом) 30: The bottom of the image is aligned with the bottom of the text.

Это можно использовать и для центрирования, если высота родителя известна, а центрируемого элемента – нет.

Допустим, высота внешнего элемента 120px. Укажем её в свойстве line-height:

```
1 <style>
2   .outer {
3     line-height: 120px;
4   }
5   .inner {
6     display: inline-block; /* центрировать..*/
7     vertical-align: middle; /* ..по вертикали */
8     line-height: 1.25; /* переопределить высоту строки на обычную */
9     border: 1px solid red;
10  }
11 </style>
12 <div class="outer" style="height: 120px;border: 1px solid blue">
13   <span class="inner">Центрирован<br>вертикально</span>
14 </div>
```




Центрирован
вертикально

Центрирование с vertical-align без таблиц

Если центрирование должно работать для любой высоты родителя и центрируемого элемента, то обычно используют таблицы или `display:table-cell` с `vertical-align`.

Если центрируются не-блочные элементы, например `inline` или `inline-block`, то `vertical-align` может решить задачу без всяких таблиц. Правда, понадобится вспомогательный элемент (можно через **:before**).

```
1 <style>
2 .before {
3   display: inline-block;
4   height: 100%;
5   vertical-align: middle;
6 }
7
8 .inner {
9   display: inline-block;
10  vertical-align: middle;
11 }
12 </style>
13
14 <div class="outer" style="height:100px;border:1px solid blue">
15   <span class="before"></span>
16   <span class="inner" style="border:1px solid red">
17     Центрированный<br>Элемент
18   </span>
19 </div>
```



Центрированный
Элемент

Перед центрируемым элементом помещается вспомогательный инлайн-блок before, занимающий всю возможную высоту.

Центрируемый блок выровнен по его середине.

```
1 <style>
2 .outer:before {
3   content: '';
4   display: inline-block;
5   height: 100%;
6   vertical-align: middle;
7 }
8
9 .inner {
10  display: inline-block;
11  vertical-align: middle;
12 }
13
14 /* добавим горизонтальное центрирование */
15 .outer {
16   text-align: center;
17 }
18 </style>
19
20 <div class="outer" style="height:100px; width: 100%; border:1px solid black">
21   <span class="inner" style="border:1px solid red">
22     Центрированный<br>Элемент
23   </span>
24 </div>
```

Центрированный
Элемент

В пример выше добавлено также горизонтальное центрирование `text-align: center`. Но видно, что на самом деле внутренний элемент не центрирован горизонтально, он немного сдвинут вправо.

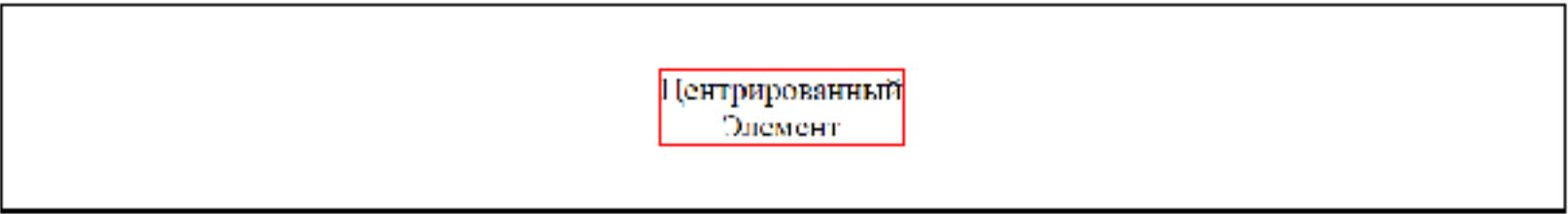
Это происходит потому, что центрируется *весь текст*, а перед `inner` находится пробел, который занимает место.

Избавиться:

Убрать лишний пробел между `div` и началом `inner`, будет `<div class="outer">....`

Оставить пробел, но сделать отрицательный `margin-left` у `inner`, равный размеру пробела, чтобы `inner` сместился левее.

```
9  .inner {
10    display: inline-block;
11    vertical-align: middle;
12    margin-left: -0.35em;
13  }
14
15  .outer {
16    text-align: center;
17  }
18  </style>
19
20  <div class="outer" style="height:100px; width: 100%; border:1px solid black">
21    <span class="inner" style="border:1px solid red">
22      Центрированный<br>Элемент
23    </span>
24  </div>
```



Центрированный
Элемент

Центрирование с использованием модели flexbox

```
1 <style>
2 .outer {
3     display: flex;
4     justify-content: center; /*Центрирование по горизонтали*/
5     align-items: center;     /*Центрирование по вертикали */
6 }
7 </style>
8
9 <div class="outer" style="height:100px; width: 100%; border:1px solid black">
10     <span class="inner" style="border:1px solid red">
11         Центрированный<br>Элемент
12     </span>
13 </div>
```

Центрированный
Элемент

Плюсы:

Не требуется знания высоты центрируемого элемента.

CSS чистый, короткий и не требует дополнительных элементов.

Минусы:

Не поддерживается IE9-, IE10 поддерживает предыдущую версию flexbox.

Отцентрировать

<https://codepen.io/anon/pen/bmojPL>

CSS-ссылки

CSS-ссылки содержат свойства, которые отвечают за внешний вид гипертекстовых ссылок HTML-документа. Ссылки представляют собой основной способ навигации по сайту, поэтому применение CSS-стилей для оформления улучшит их визуальное восприятие.

Основной способ оформления ссылок заключается в стилизации подчеркивания ссылки и изменении цвета текста ссылки. Также можно изменить внешний вид курсора с помощью свойства `cursor`.

Не посещенная — `a:link`

Посещенная — по которой уже выполнялся переход — `a:visited`

Не нажатая — над которой находится указатель мыши — `a:hover`

Нажатая — которая удерживается мышью — `a:active`

Оформление ссылок

Удаление подчеркивания:

```
a {text-decoration: none;}
```

Добавление подчеркивания только при наведении на ссылку:

```
a {text-decoration: none;}  
a:hover {text-decoration: underline;}
```

Внешний вид нижней границы ссылки:

```
a {  
  text-decoration: none;  
  border-bottom: 2px dashed DarkOrchid;  
  padding-bottom: 3px;  
}
```

<https://codepen.io/html5book/pen/RGWgBQ>

Внешний вид курсора мыши cursor

Курсор мыши может иметь различный вид, также можно установить пользовательское изображение в качестве курсора.

Значение по умолчанию `cursor: pointer;`.

Использование фонового изображения

Можно преобразовать внешний вид ссылки, добавив в качестве нижней границы фоновое изображение:

```
a {  
  text-decoration: none;  
  background: url(images/underline.png) repeat-x left bottom;  
  padding-bottom: 3px;  
}
```

Ссылки-кнопки

Благодаря свойствам `background-color`, `border` и `padding`, ссылкам можно придать вид прямоугольных кнопок, а, меняя отображение тех или иных свойств ссылок при наведении курсора мыши `a: hover`, добавить интересные эффекты.

