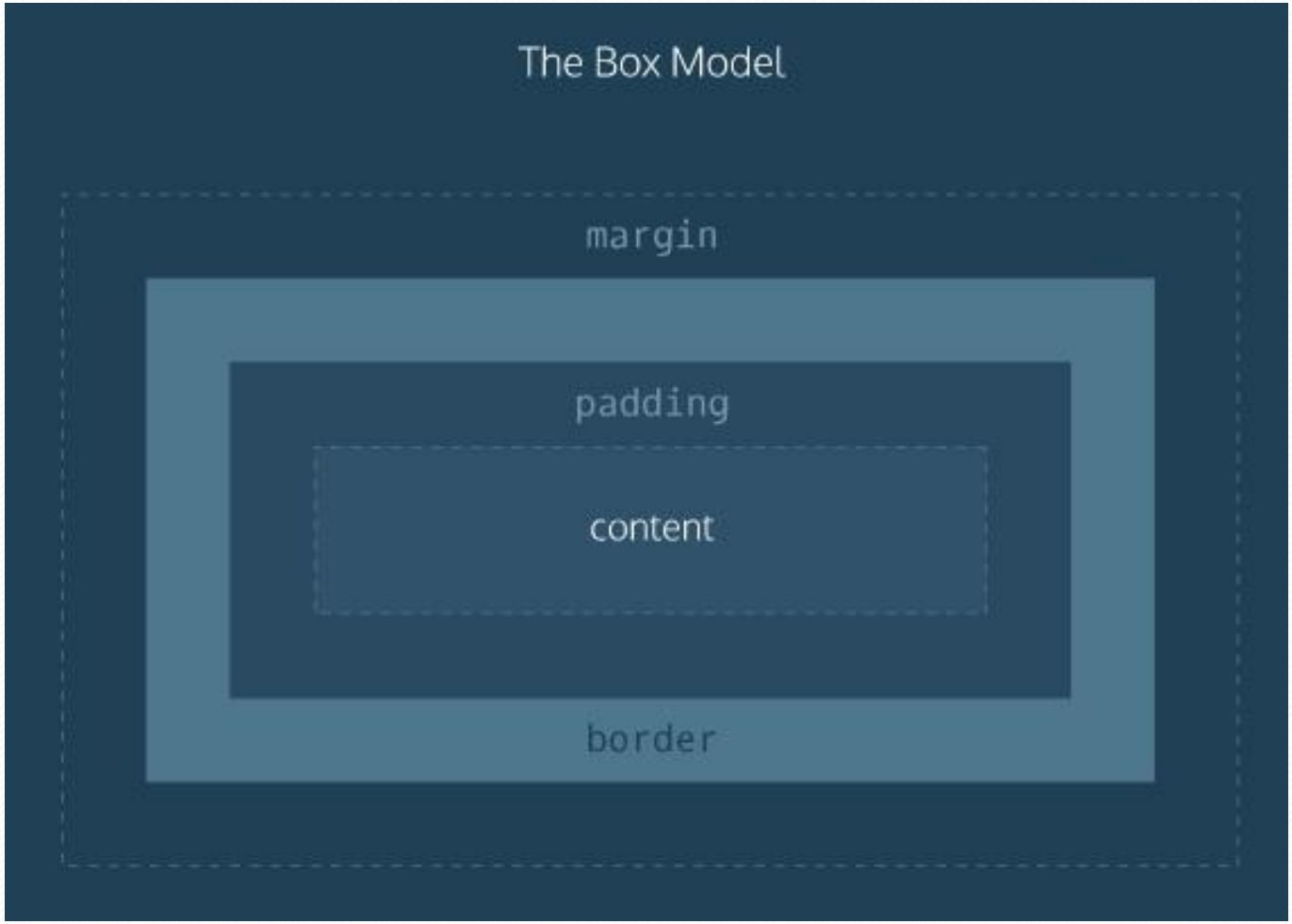
Блоковая модель, потоки, сетки

Блоковая модель



Блоковая модель

```
<html>
<head>
<style type='text/css'>
#wrap {
margin:0px;
padding:20px;
height:160px;
background-color:yellow;
}
.ex1 {
border:10px red solid;
margin-left:50px;
margin-right:10px;
padding:15px;
background-color:green;
color:white;
}
.ex2 {
border:5px brown solid;
margin-top:30px;
margin-left:250px;
margin-right:70px;
padding:15px;
background-color:green;
color:white;
}
```

Блоковая модель

Блочные элементы

Блочные элементы можно представлять как прямоугольные области на странице. Они имеют следующие особенности:

- До и после блочного элемента существует перенос строки.
- Блочным элементам можно задавать ширину, высоту, внутренние и внешние отступы.
- Занимают всё доступное пространство по горизонтали.

К блочным элементам относятся такие теги как: <p>, <h1>, <h2>, <ul>   и так далее.

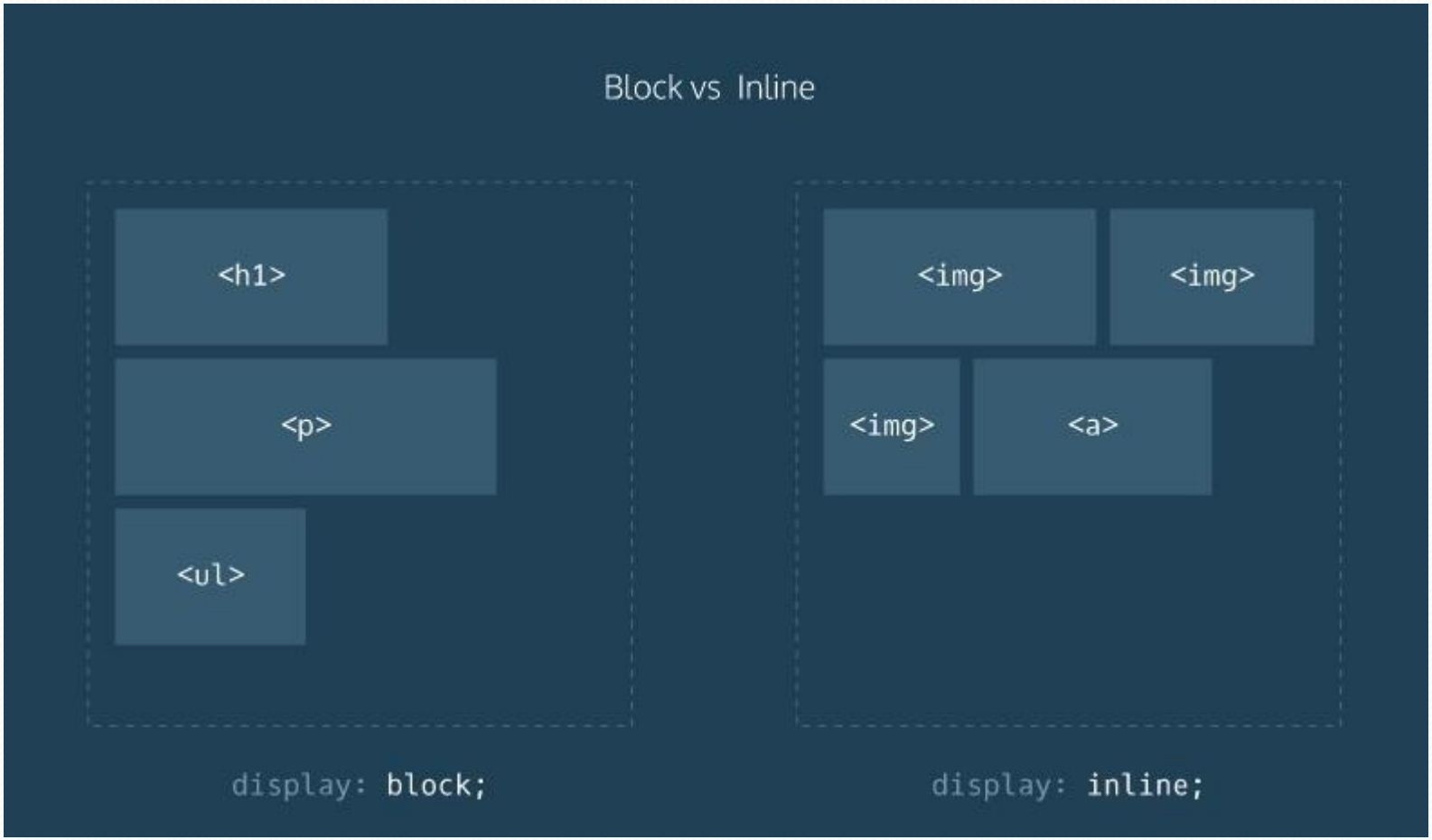
Строчные элементы

Строчные элементы располагаются друг за другом в одной строке, при необходимости строка переносится. Особенности строчных элементов:

- 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.

К строчным элементам относятся такие теги как: <a>, <b>, <i>, и так далее.

Блочные и строчные элементы



Ширина и высота

По умолчанию блочные элементы занимают всю доступную ширину, которая равна ширине родительского контейнера или окна браузера.
Высота по умолчанию блочных элементов зависит от их содержимого. Если задать блочному элементу ширину и высоту так, что содержимое элемента не будет в него помещаться, то оно как бы «выпадет» из него.

Строчные элементы **НЕ** реагируют на задание ширины и высоты в CSS.

Padding: внутренний отступ

```
<html>
<head>
<style type='text/css'>
.pad1
{
  border-style:solid;
  padding:20px;
}
.pad2
{
  border-style:solid;
  padding:10px;
}
.pad3
{
  border-style:solid;
  padding:5px;
}
</style>
</head>
<body>
<p class='pad1'>Элемент с внутренним о
<p class='pad2'>Элемент с внутренним о
<p class='pad3'>Элемент с внутренним о
</body>
```

Элемент с внутренним отступом равным 20 пикселям.

Элемент с внутренним отступом равным 10 пикселям.

Элемент с внутренним отступом равным 5 пикселям.

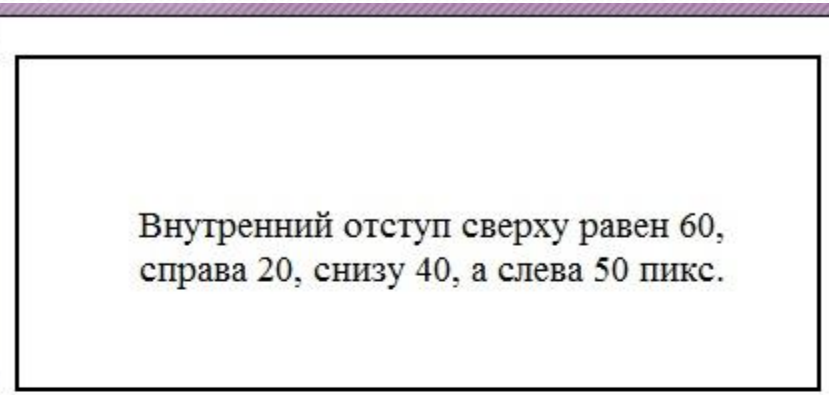
Padding с разных сторон

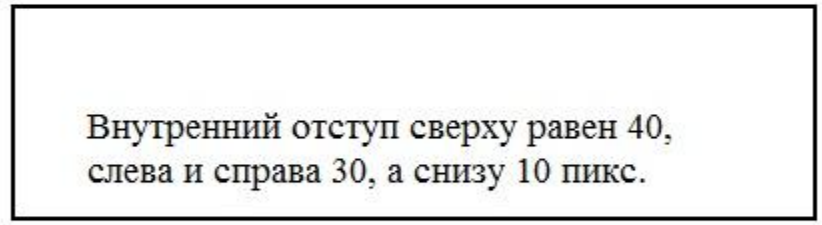
```
1  <html>
2  <head>
3  <style type='text/css'>
4  /* Отступ от содержимого элемента до его
5  .pad1
6  {
7    border-style:solid;
8    padding-top:30px;
9    padding-left:20px;
10   padding-bottom:10px;
11   padding-right:40px;
12 }
13 </style>
14 </head>
15 <body>
16 <p class='pad1'>Элемент с заданным внутр
17 </body>
18 </html>
19
```

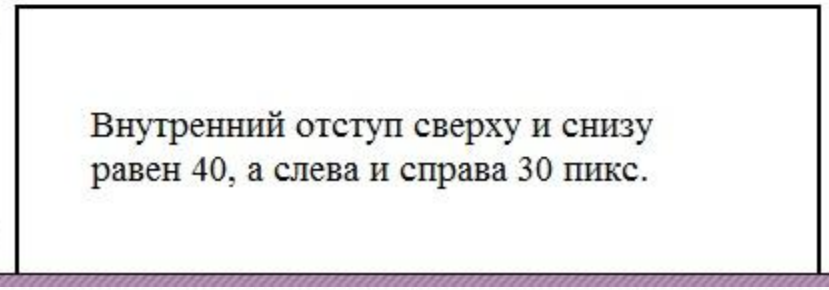
Элемент с заданным внутренним отступом.

Padding: краткая форма записи

```
<html>
<head>
<style type='text/css'>
/* Отступ от содержимого до границы эле
.pad1
{
border:2px solid;
padding:60px 20px 40px 50px;
}
/* Отступ от содержимого до границы эле
.pad2
{
border:2px solid;
padding:40px 30px 10px;
}
/* Отступ от содержимого до границы эле
.pad3
{
border:2px solid;
padding:40px 30px;
}
</style>
</head>
<body>
<p class='pad1'>Внутренний отступ свер
```







Margin: внешний отступ

```
<html>
<head>
<style type='text/css'>
.dv1
{
  border-style:solid;
  border-width:1px;
  background-color:red;
}
.mar1, .mar2, .mar3
{
  border-width:1px;
  border-style:solid;
  background-color:white;
}
.mar1
{
  margin:25px;
}
.mar2
{
  margin:10px;
}
.mar3
{
```

Обратите внимание: внешний отступ в данном примере для наглядности выделен красным цветом.

Выравнивание с margin

```
1  <html>
2  <head>
3  <style type='text/css'>
4  .ali1
5  {
6    margin-left:auto;
7    margin-right:auto;
8    width:50%;
9  }
10 </style>
11 </head>
12 <body>
13 <p class='ali1'>Данный абзац выра
14 </body>
15 </html>
16
```

Данный абзац
выравнен по центру.

Рамки

Рамка задаётся с помощью свойства border, которое состоит из трёх компонентов:

- ширина рамки;
- стиль рамки;
- цвет.

Например: **border: 5px solid red;**

Проблемы блочной модели

Проблемы начинаются, когда сетка резиновая и элементы в ней должны тянуться. Самый простой пример: форма, в которой поля должны занимать всю ширину контейнера, но при этом иметь фиксированные внутренние отступы, чтобы текст не прилипал к краям.

Box-sizing

Для решения описанной проблемы нужно, чтобы свойство width задавало не *ширину содержания*, а *общую ширину*.
К счастью, такая возможность была добавлена в CSS3 с помощью свойства **box-sizing**, которое уже поддерживается большинством современных браузеров.

Это свойство имеет два значения:

- **content-box** — значение по умолчанию, соответствует стандартной блочной модели.
- **border-box** — изменяет режим расчета ширины элемента на описанный выше.

Display: меняем тип элемента

Тип элемента не является чем-то вечным и неизменным, его можно изменять с помощью CSS. За это отвечает свойство display.

С его помощью, например, можно сделать абзацы и заголовки строчными, а спаны и стронги блочными.

У свойства display много значений, например:

- display:block —блочный элемент
- display:inline — строчный элемент
- display:inline-block — блочно-строчный элемент
- display:none — элемент вообще не видно, и он не занимает места

Display: inline-block

Особенности блочно-строчных элементов:

- им можно задавать размеры, рамки и отступы, как и блочным элементам;
- их ширина по умолчанию зависит от содержания, а не растягивается на всю ширину контейнера;
- они не порождают принудительных переносов строк, поэтому могут располагаться на одной строке, пока помещаются в родительский контейнер;
- элементы в одной строке выравниваются вертикально подобно строчным элементам.

Display: none

Значение **none** свойства display используется очень часто. С его помощью можно скрыть элемент, как будто его и не было. Скрытый элемент не отображается и не занимает места на странице.

Данное свойство применяется при создании выпадающих меню, динамических галерей, переключающихся вкладок и много где еще.

Float: обтекание элемента

```
<html>
<head>
<style type='text/css'>
.fl1
{
  float:right;
  margin:0px;
  margin-left:5px;
  width:340px;
  height:260px;
}
</style>
</head>
<body>
<img src='mountimg3.jpg' class='fl1'
<p><b>Ергаки (Пальцы)</b> — горный уз
<p>Ергаки находится на территории Кра
<p>Ергаки — название природного парка
</body>
</html>
```

Ергаки (Пальцы) — горный узел, хребет в Западном Саяне. Расположен в истоках рек Большой Кебеж, Большой Ключ, Тайгиш, Верхняя Буйба, Средняя Буйба и Нижняя Буйба.

Ергаки находится на территории Красноярского края, в пределах Ермаковского и Каратузского районов. Через Ергаки (в восточной части) проходит федеральная трасса *М-54*. Ближайшие

Поток

Поток — это порядок отображения элементов на странице. По умолчанию блочные элементы отображаются как прямоугольные области, идущие друг за другом сверху вниз, а строчные элементы располагаются сверху вниз и слева направо и при необходимости переносятся на новую строку.

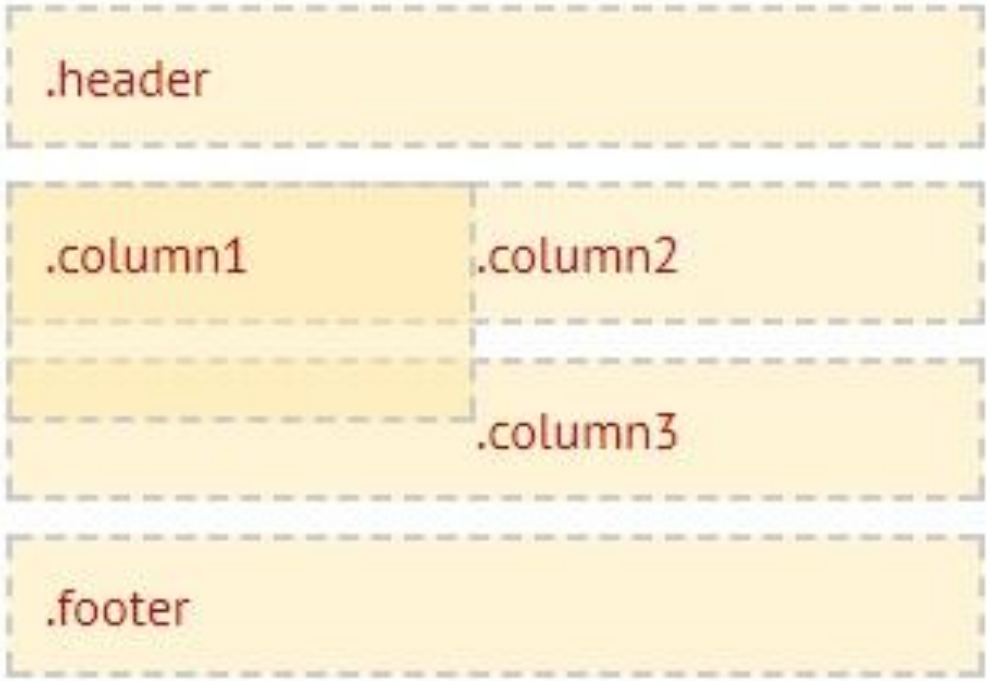
Потоком можно управлять и изменять привычное поведение элементов в потоке. Например, можно заставить блочные элементы двигаться не сверху вниз, а выстраиваться в несколько колонок.

Поток

```
<div class="header block">.header</div>
<div class="column1 block">.column1</div>
<div class="column2 block">.column2</div>
<div class="column3 block">.column3</div>
<div class="footer block">.footer</div>
```

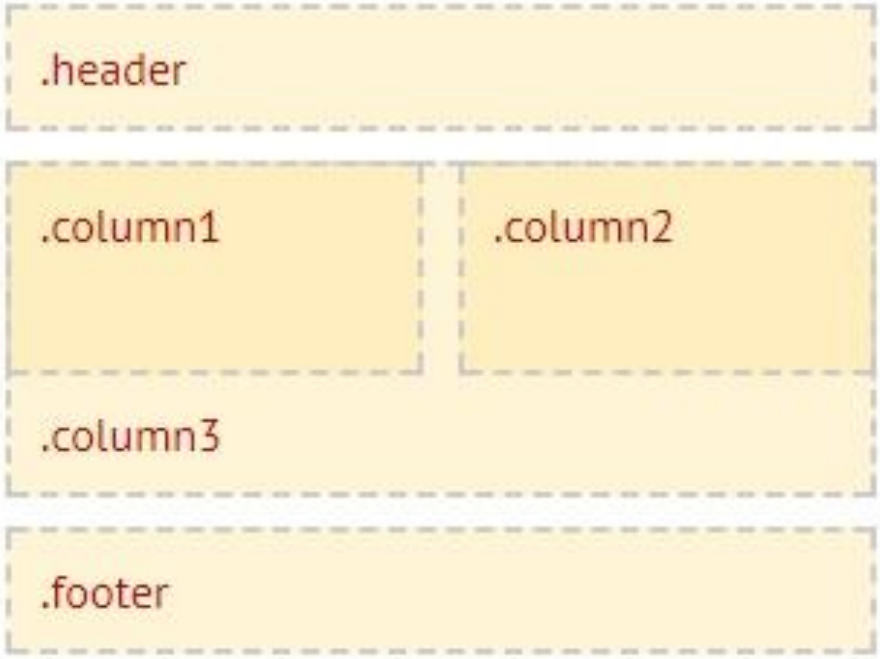
Float:left

Добавим блоку **.column1** свойство **float:left**
Обратите внимание, как он стал наезжать на последующие блоки.



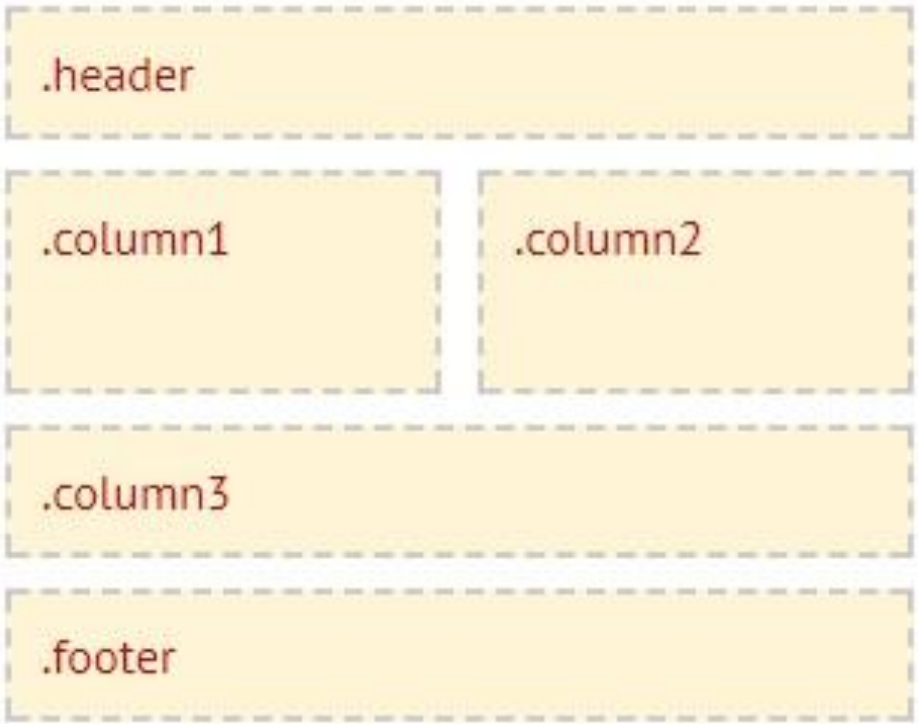
Float:right

Добавим блоку **.column2** свойство **float:right**
Теперь уже первые два блока наезжают на третий.



Clear:both

Добавим блоку **.column3** свойство **clear:both**
Теперь блоки выстроились правильно.



Float

Если мы задаём элементу свойство float:left или float:right, то он прижимается к левому или правому краю, а также начинает ужиматься по ширине под своё содержимое. С той стороны, которая не прижата к краю родителя, появляется свободное место. Это место может быть занято другими элементами.

Зафлоаченному элементу можно явно задавать размеры и отступы.

FlexBox

Во FlexBox есть два основных типа элементов:

- **Гибкий Контейнер (Flex Container)**
- **Гибкие Элементы (Flex Item)** — его дочерние элементы

Для инициализации контейнера достаточно присвоить, через css, элементу:
display: flex; или
display: inline-flex;

Разница между flex и inline-flex заключается лишь в принципе взаимодействия с окружающими контейнер элементами, подобно display: block; и display: inline-block;, соответственно.

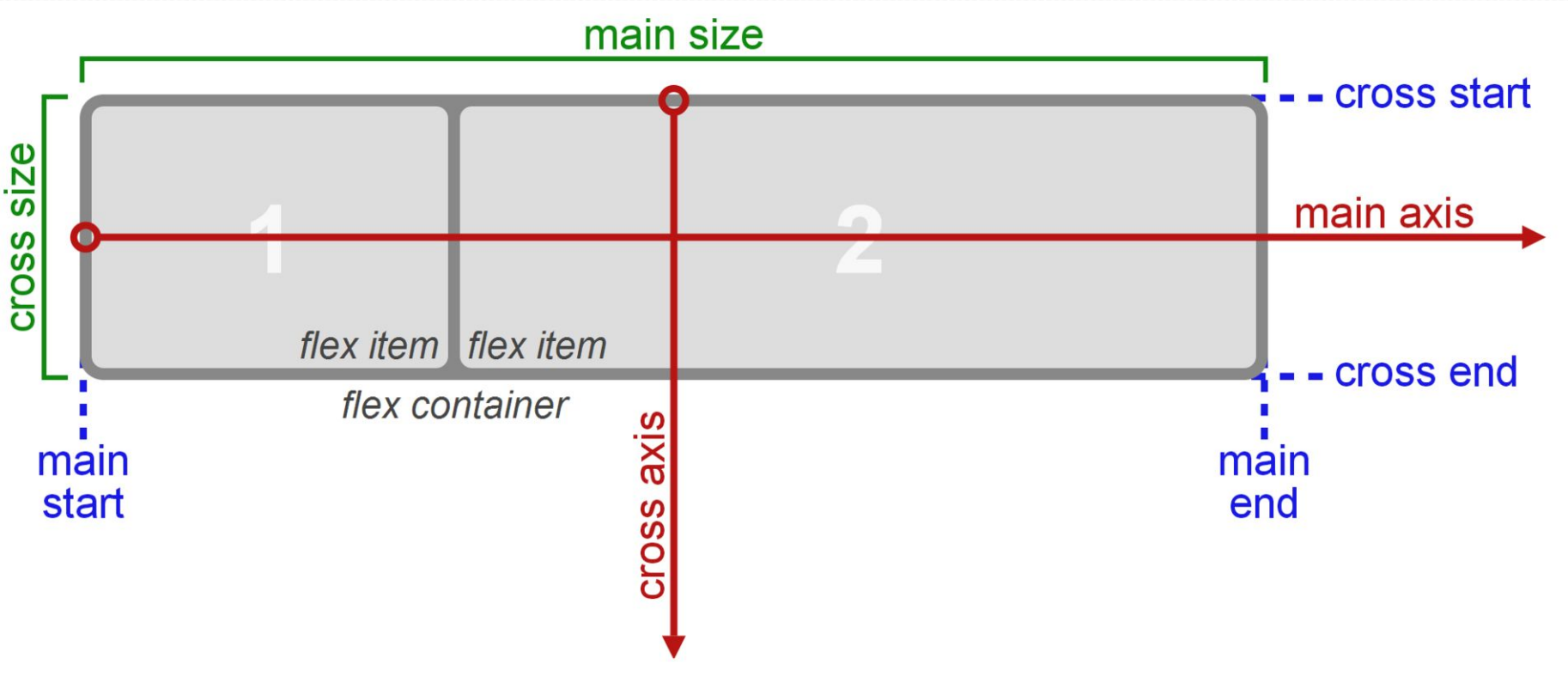
Оси FlexBox

Внутри гибкого контейнера создаются две оси:

- **главная ось (main-axis)**
- **перпендикулярная или кросс ось (cross axis).**

Преимущественно гибкие элементы выстраиваются именно по главной оси, а потом уже по кросс оси. По умолчанию главная ось горизонтальная и имеет направление слева направо, а кросс ось вертикальна и направлена сверху вниз.

Оси FlexBox

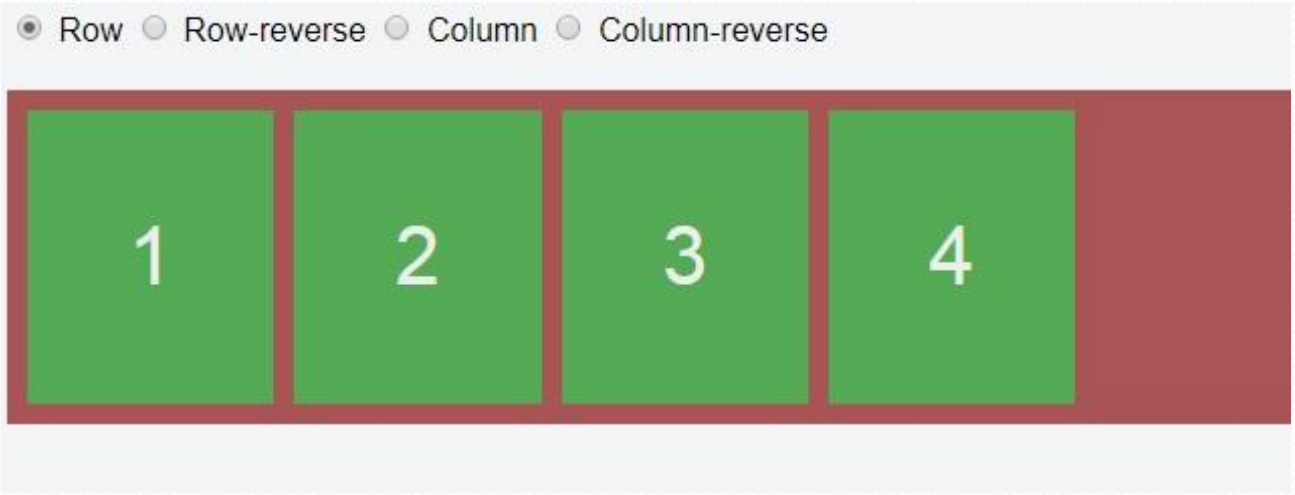

main size
cross start
cross size
main axis
1
2
flex item
flex item
cross end
flex container
main
start
cross axis
main
end

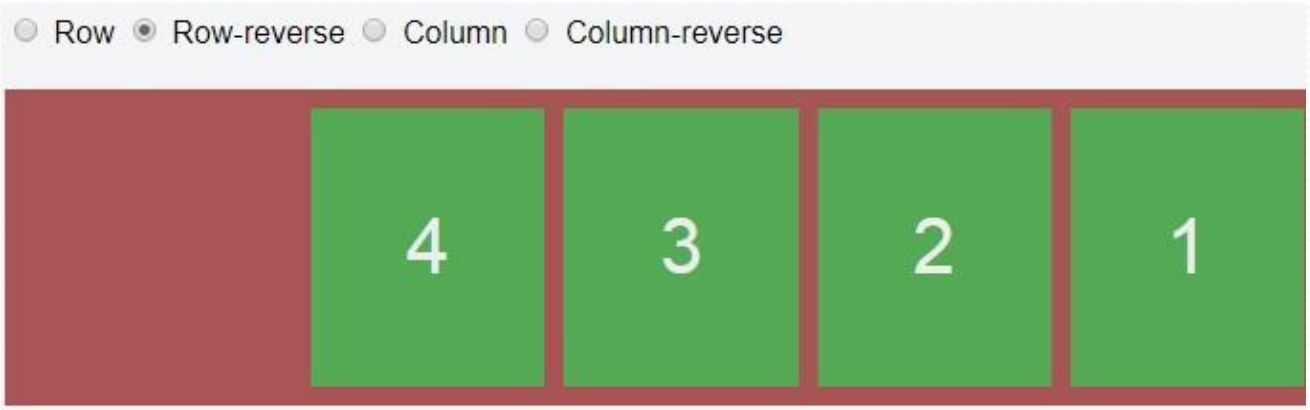
Направление осей

Направлением осей можно управлять с помощью css-свойства **flex-direction**. Его значения:

- **row** (default): Главная ось гибкого контейнера имеет ту же ориентацию, как и инлайн ось текущего режима.
- **row-reverse**: Все то же самое, что и в row только main-start и main-end меняются местами.
- **column**: так же само как и row, только теперь главная ось направлена сверху вниз.
- **column-reverse**: так же само как row-reverse, только главная ось направлена снизу вверх.

Направление осей



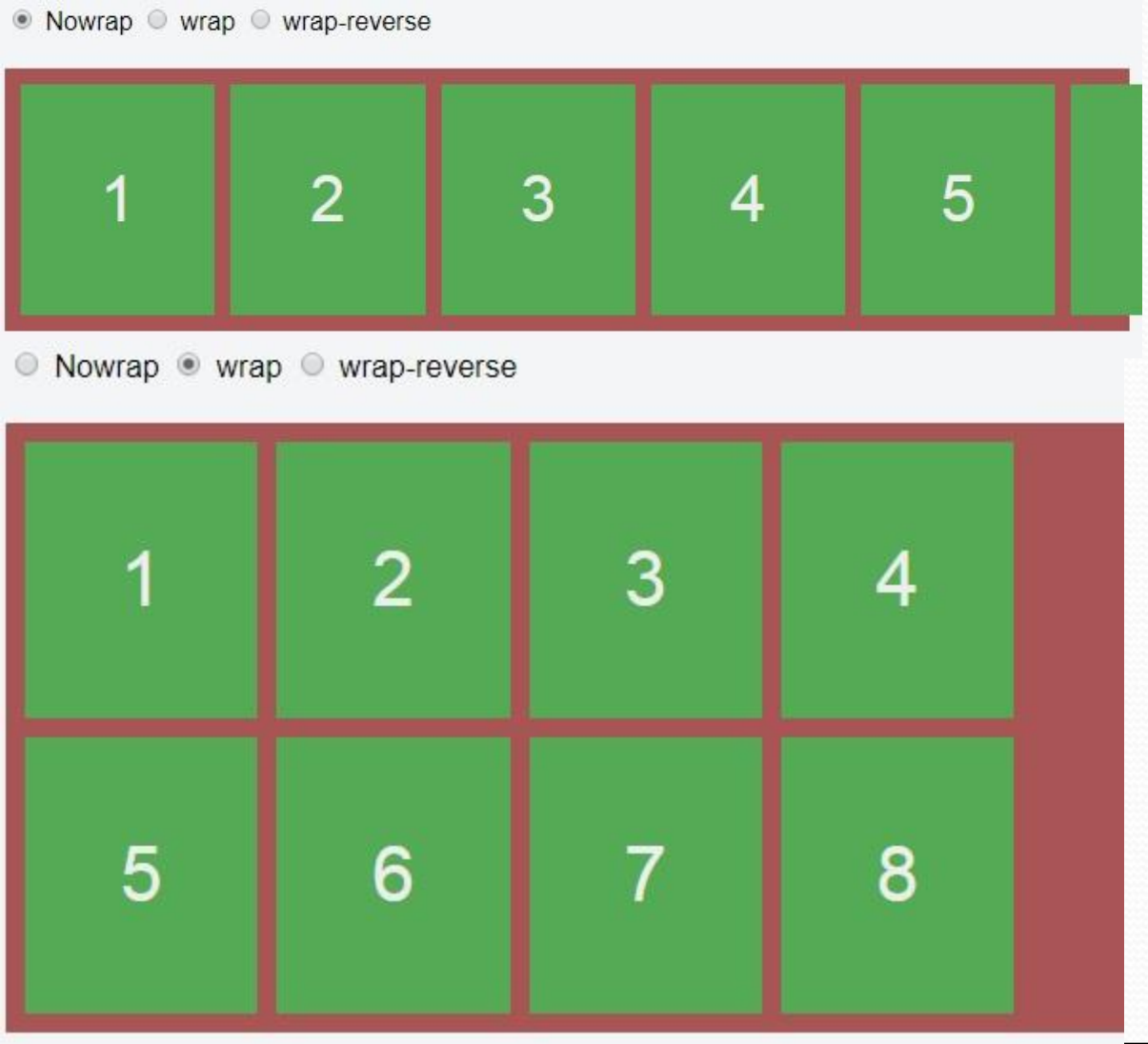


Flex-wrap

По умолчанию все гибкие элементы в контейнере укладываются в одну строку, даже если не помещаются в контейнер, они выходят за его границы. Данное поведение переключается с помощью свойства **flex-wrap**.

- **nowrap** (default): элементы выстраиваются в одну строку.
- **wrap**: гибкие элементы строятся в многострочном режиме, перенос осуществляется по направлению кросс оси, сверху вниз.
- **wrap-reverse**: так же как и wrap, но перенос происходит снизу вверх.

Flex-wrap


Nowrap wrap wrap-reverse
1
2
3
4
5
Nowrap wrap wrap-reverse
1
2
3
4
5
6
7
8

Justify-content

Элементы в контейнере поддаются выравниванию при помощи свойства **justify-content** вдоль главной оси. Это свойство принимает целых пять разных вариантов значений.

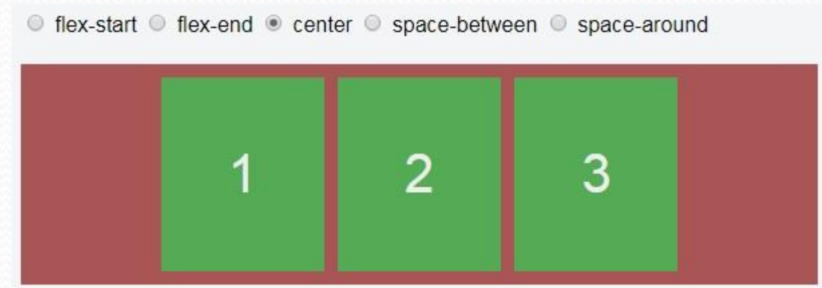
- **flex-start** (default): гибкие элементы выравниваются по началу главной оси.
- **flex-end**: элементы выравниваются по концу главной оси
- **center**: элементы выравниваются по центру главной оси

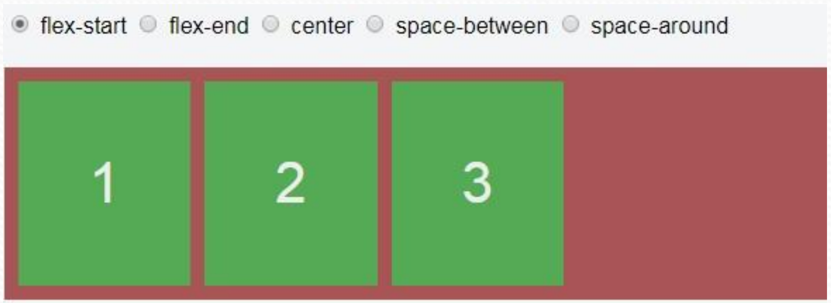
Justify-content

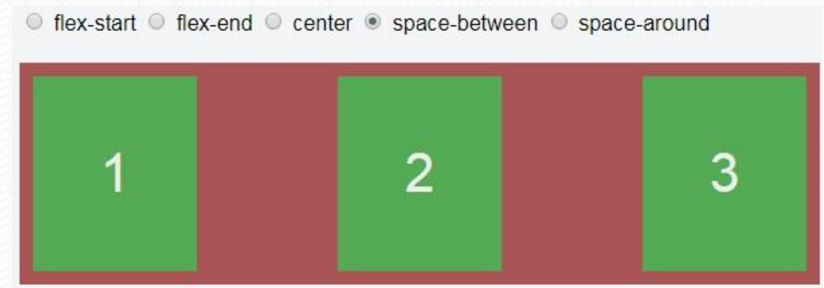
- **space-between**: элементы занимают всю доступную ширину в контейнере, крайние элементы вплотную прижимаются к краям контейнера, а свободное пространство равномерно распределяется между элементами.

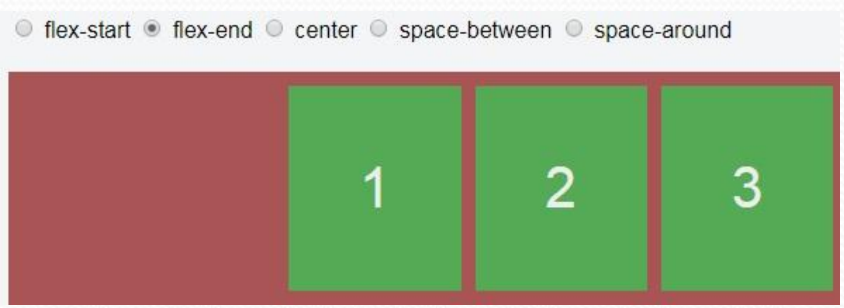
- **space-around**: гибкие элементы выравниваются таким образом, что свободное пространство равномерно распределяется между элементами. Но стоит отметить, что пространство между краем контейнера и крайними элементами будет в два раза меньше чем пространство между элементами в середине ряда.

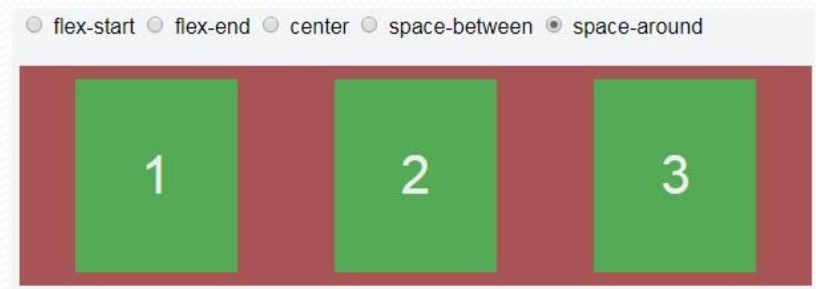
Justify-content











Align-items

Мы так же имеем возможность выравнивания элементов по кросс оси. Применив свойство **align-items**, которое принимает также пять разных значений, можно добиться интересного поведения. Это свойство позволяет выравнивать элементы в строке относительно друг друга.

flex-start: все элементы прижимаются к началу строки
flex-end: элементы прижимаются к концу строки
center: элементы выравниваются по центру строки
baseline: элементы выравниваются по базовой линии текста
stretch (default): элементы растягиваются заполняя полностью строку.

Align-content

Еще одно похожее свойство на предыдущее это **align-content**. Только оно отвечает за выравнивание целых строк относительно гибкого контейнера. Оно не будет давать эффекта если гибкие элементы занимают одну строку. Свойство принимает шесть разных значений.

flex-start: все линии прижимаются к началу кросс-оси

flex-end: все линии прижимаются к концу кросс-оси

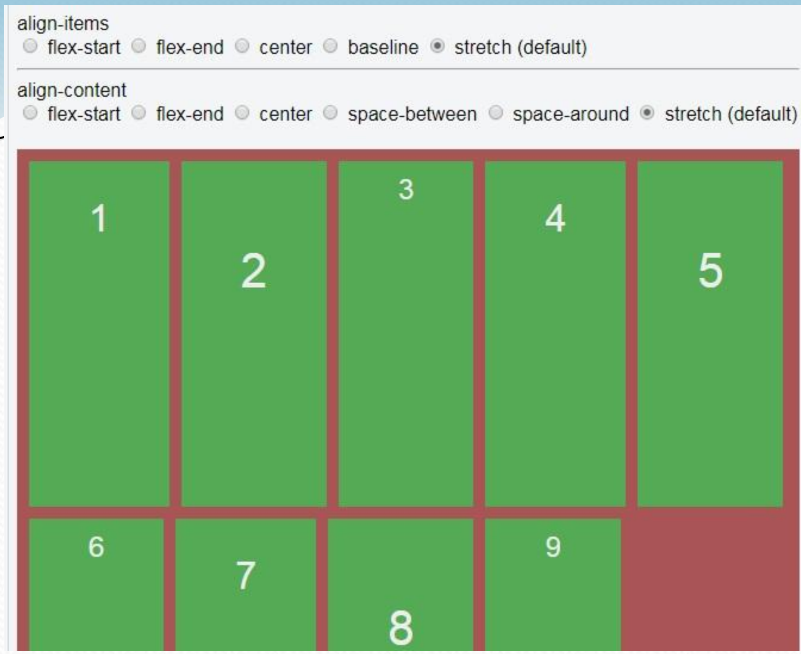
center: Все линии паком выравниваются по центру кросс оси

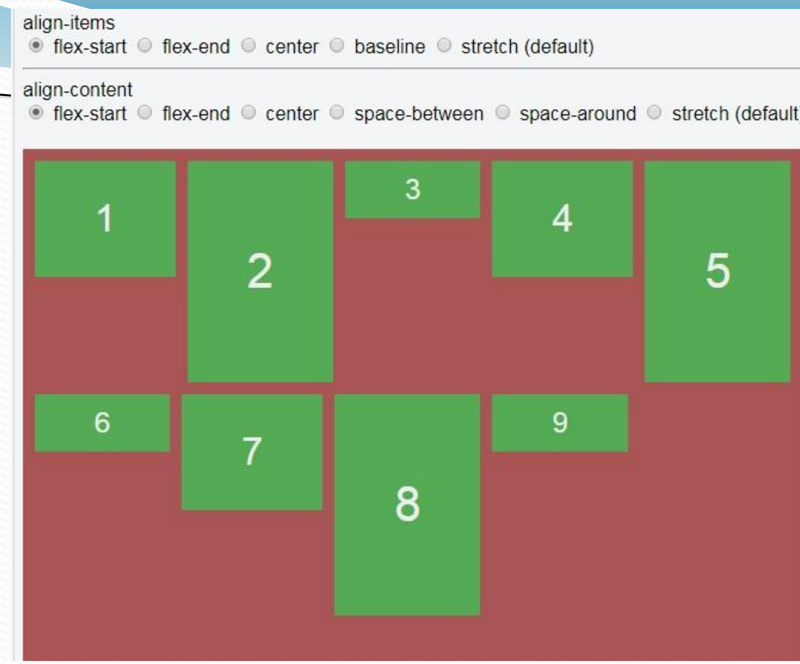
Align-content

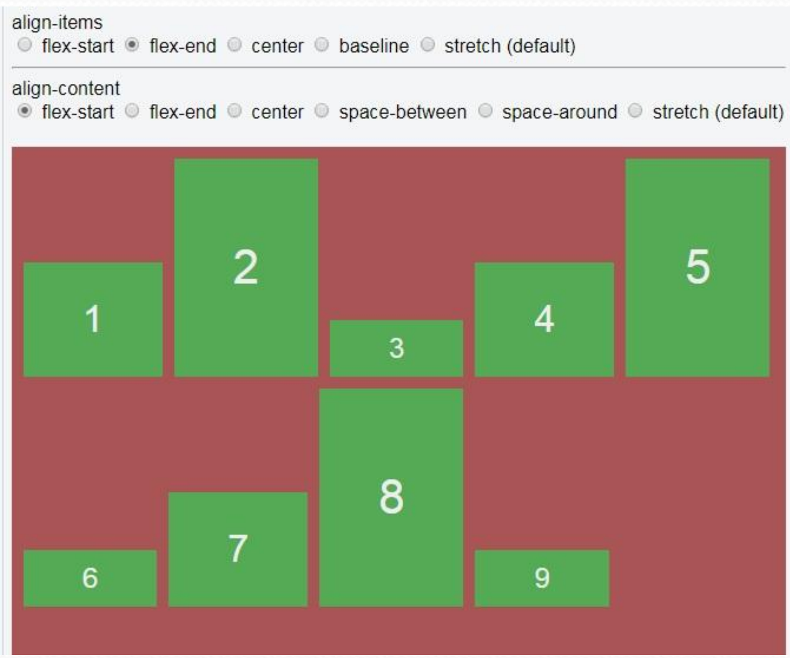
space-between: линии распределяются от верхнего края до нижнего оставляя свободное пространство между строками, крайние же строки прижимаются к краям контейнера.

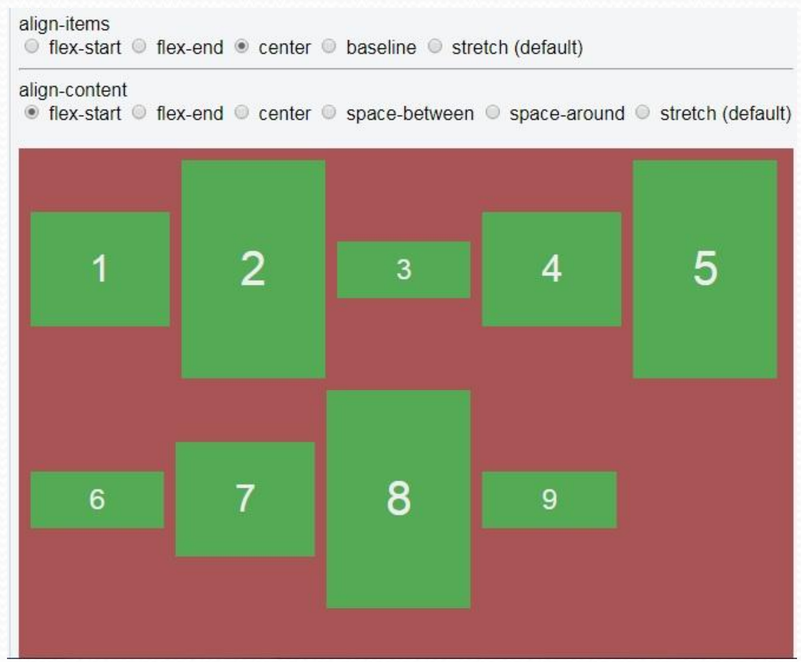
space-around: линии равномерно распределяются по контейнеру.

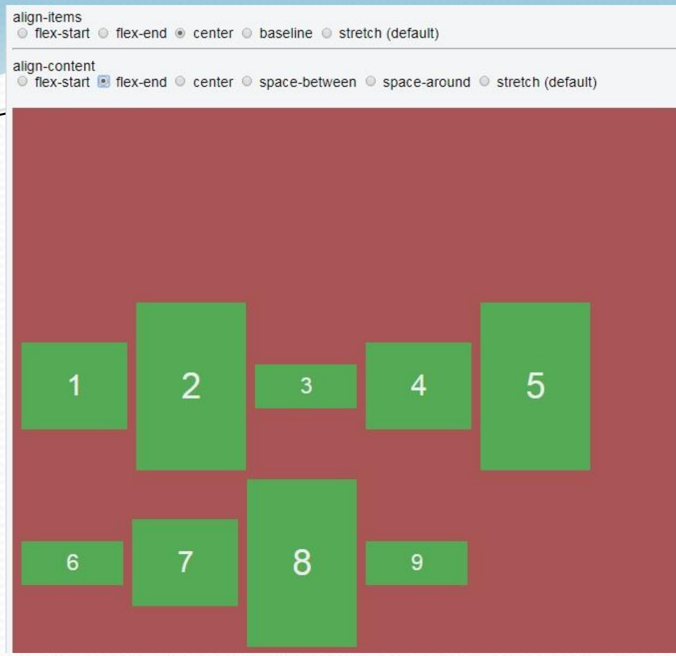
stretch (default): линии растягиваются занимая все доступное пространство.

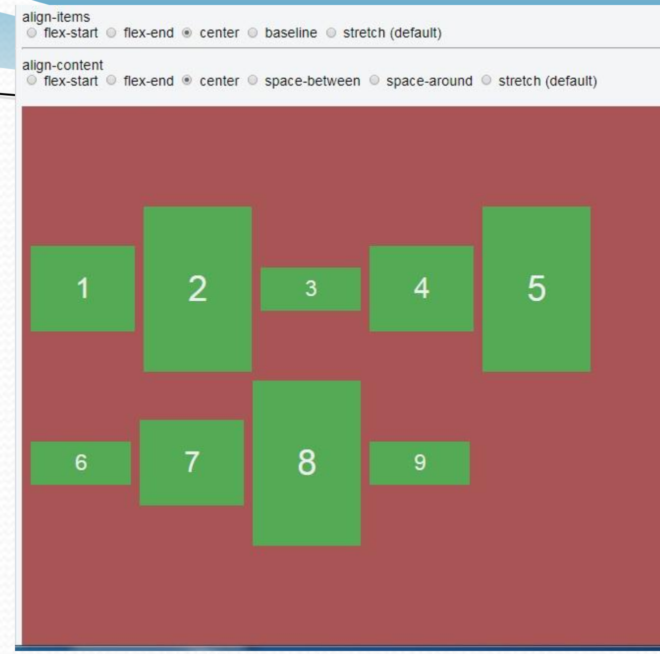

align-items
flex-start flex-end center baseline stretch (default)
align-content
flex-start flex-end center space-between space-around stretch (default)
1
2
3
4
5
6
7
8
9

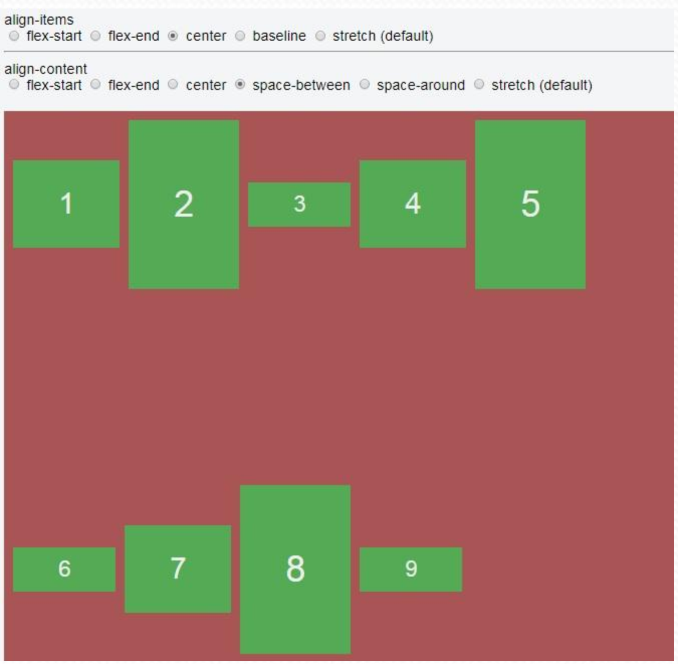

align-items
flex-start flex-end center baseline stretch (default)
align-content
flex-start flex-end center space-between space-around stretch (default)
1
2
3
4
5
6
7
8
9

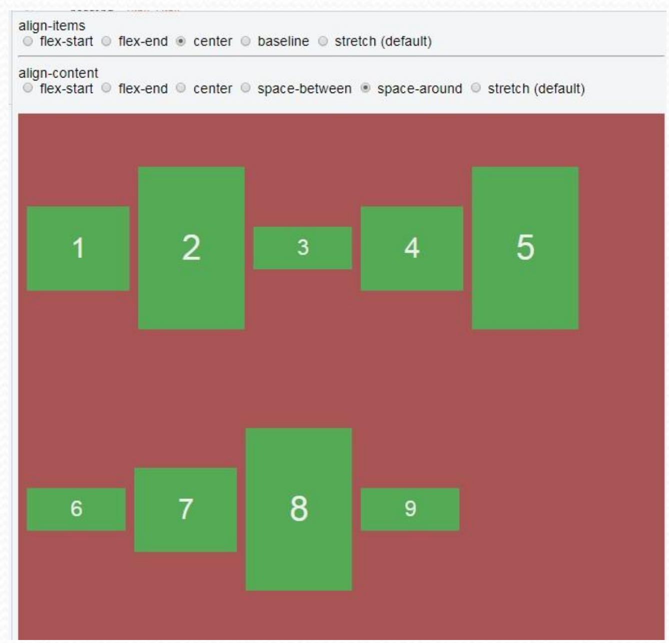

align-items
flex-start flex-end center baseline stretch (default)
align-content
flex-start flex-end center space-between space-around stretch (default)
2
5
1
3
4
8
7
6
9


align-items
flex-start flex-end center baseline stretch (default)
align-content
flex-start flex-end center space-between space-around stretch (default)
1
2
3
4
5
6
7
8
9


align-items
flex-start flex-end center baseline stretch (default)
align-content
flex-start flex-end center space-between space-around stretch (default)
1 2 3 4 5
6 7 8 9


align-items
flex-start flex-end center baseline stretch (default)
align-content
flex-start flex-end center space-between space-around stretch (default)
1 2 3 4 5
6 7 8 9


align-items
flex-start flex-end center baseline stretch (default)
align-content
flex-start flex-end center space-between space-around stretch (default)
1 2 3 4 5
6 7 8 9


align-items
flex-start flex-end center baseline stretch (default)
align-content
flex-start flex-end center space-between space-around stretch (default)
1 2 3 4 5
6 7 8 9

Гибкий элемент. Flex-basis

Одно из основных свойств гибкого элемента является **flex-basis**. С помощью этого свойства мы можем указывать базовую ширину гибкого элемента. По умолчанию имеет значение **auto**. Это свойство тесно связано с **flex-grow** и **flex-shrink**, о которых будет чуть позже. Принимает значение ширины в px, %, em и остальных единицах. По сути это не строго ширина гибкого элемента, это своего рода отправная точка. Относительно которой происходит растягивание или усадка элемента. В режиме auto элемент получает базовую ширину относительно контента внутри него.

Гибкий элемент. Flex-grow

flex-grow задает фактор увеличения элемента при наличии свободного места в контейнере. По умолчанию это свойство имеет значение 0. Давайте представим, что у нас есть гибкий контейнер, который имеет ширину 500px, внутри него есть два гибких элемента, каждый из которых имеет базовую ширину 100px. Тем самым в контейнере остается еще 300px свободного места. Если первому элементу укажем flex-grow: 2;, а второму элементу укажем flex-grow: 1;. В результате эти блоки займут всю доступную ширину контейнера, только ширина первого блока будет 300px, а второго только 200px. Что же произошло? А произошло вот что, доступные 300px свободного места в контейнере распределились между элементами в соотношении 2:1

Гибкий элемент. Flex-shrink

flex-shrink так же задает фактор на изменение ширины элементов, только в обратную сторону. Если контейнер имеет ширину **меньше,** чем сумма базовой ширины элементов, то начинает действовать это свойство. Например контейнер имеет ширину 600px, а flex-basis элементов по 300px. Первому элементу укажем flex-shrink: 2;, а второму flex-shrink: 1;. Теперь сожмем контейнер на 300px. Следовательно сумма ширины элементов на 300px больше чем контейнер. Эта разница распределяется в соотношении 2:1, получается от первого блока отнимаем 200px, а от второго 100px. Новый размер элементов получается 100px и 200px, у первого и второго элемента, соответственно.

Таблицы. Пример

Страна	Население (млн.)
Россия	141
США	309
Китай	1338
Великобритания	61

Таблицы

<table> Создание таблицы **<table>**

<tr> Создание строки **</tr>**

<td> Создание ячейки **</td>**

Таблицы. Пример

```
<table border="1">
    <tr>
        <td>Россия</td> <td>141</td>
    </tr>
    <tr> <td>США</td> <td>309</td> </tr>
    <tr> <td>Китай</td> <td>1338</td> </tr>
    <tr> <td>Великобритания</td> <td>61</td>
    </tr>
</table>
```

Растягивание ячеек

```
1  <html>
2  <body>
3  <table border='1'>
4  <tr>
5  <td> Ячейка 1 </td>
6  <td> Ячейка 2 </td>
7  </tr>
8  <tr>
9  <td colspan='2'> Ячейка 3 растяну
10 </tr>
11 </table>
12 </body>
13 </html>
14
```

Ячейка 1	Ячейка 2
Ячейка 3 растянутая на 2 столбца	

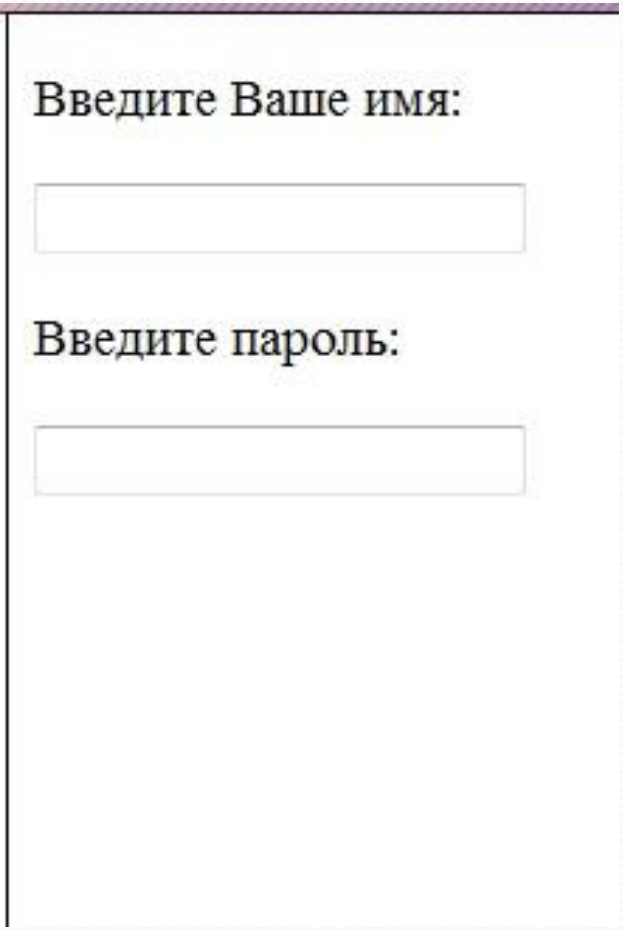
Формы

- Текстовые поля
- Флажки
- Радио-кнопки
- Кнопки

и другие

Формы

```
1  <html>
2  <body>
3  <form>
4    <p>Введите Ваше имя:</p>
5    <input type='text' />
6
7    <p>Введите пароль: </p>
8    <input type='password' />
9
10 </form>
11 </body>
12 </html>
13
```



Текстовое поле

<input type="text" />

HTML

```
<html>
<body>
<form>
<p> Введите ФИО в поля ниже: </p> <br />
Имя: <input type='text' name='firstname' /><br />
Фамилия: <input type='text' name='lastname' /><br />
Отчество: <input type='text' name='lastname' />
</form>
</body>
</html>
```

Результат

Введите ФИО в поля ниже:

Имя:

Фамилия:

Отчество:

Флажки

<input type="checkbox" />

HTML

```
<html>
<body>
<form>
  <p> Как вы относитесь к полетам в космос? </p>

  <input type='checkbox' name='space' value='1' />
       Положительно, всегда хотел полететь в космос.<br />
  <input type='checkbox' name='space' value='2' />
       Безразлично, никогда не думал об этом серьезно. <br />
  <input type='checkbox' name='space' value='3' />
       Отрицательно, меня с детства отталкивают мысли о космосе. <br />

</form>
</body>
</html>
```

Результат

Как вы относитесь к полетам в космос?

- ☐ Положительно, всегда хотел полететь в космос.
- ☐ Безразлично, никогда не думал об этом серьезно.
- ☐ Отрицательно, меня с детства отталкивают мысли

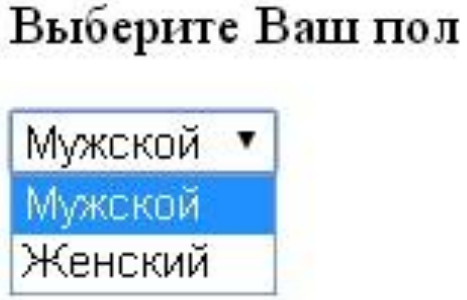
Выпадающий список

<select> и <option>

HTML

```
<html>
<body>
<p>Выберите Ваш пол: </p>
<form>
  <select name='sex' >
    <option value='m'> Мужской </option>
    <option value='f'> Женский </option>
  </select>
</form>
</body>
</html>
```

Результа



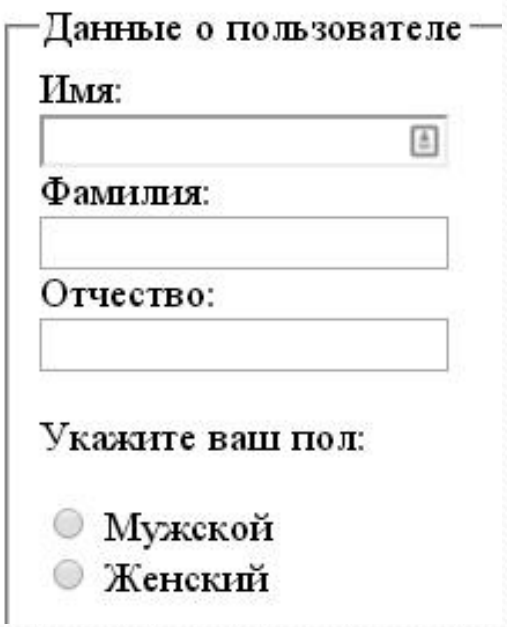
Группы в формах

<fieldset> и <legend>

HTML

```
<html>
<body>
<form>
<fieldset>
  <legend>Данные о пользователе</legend>
  Имя: <br /><input type='text' name='firstname' /><br />
  Фамилия:<br /> <input type='text' name='lastname' /><br />
  Отчество: <br /><input type='text' name='lastname' /><br />
  <p>Укажите ваш пол:</p>
  <input type='radio' name='s' value='m' /> Мужской<br />
  <input type='radio' name='s' value='f' /> Женский
</fieldset>
</form>
```

Результа



Скрытие элементов

В CSS скрыть элементы можно двумя способами:

1. С помощью свойства **visibility:hidden**;
2. С помощью **display:none**.

Элемент скрытый первым способом будет невидим, но будет по прежнему занимать место на странице.

Пример

```
/* Абзац стал невидим, но все еще занимает место на странице */
p.dis1
{
visibility:hidden;
}
```

Второй способ позволяет полностью скрыть элемент, чтобы он больше не занимал места на странице.

Пример

```
/* Абзац стал невидим и не занимает места */
p.dis1
{
display:none;
}
```

Скрытие элементов

```
<html>
<head>
<style type='text/css'>
#dis1
{
  visibility:hidden;
}
</style>
</head>
<body>
<p id='dis1'>Данный абзац невидим, но за
<p>Это второй абзац.</p>
<b>Первый абзац не видим, но занимает ме
</body>
</html>
```

Это второй абзац.

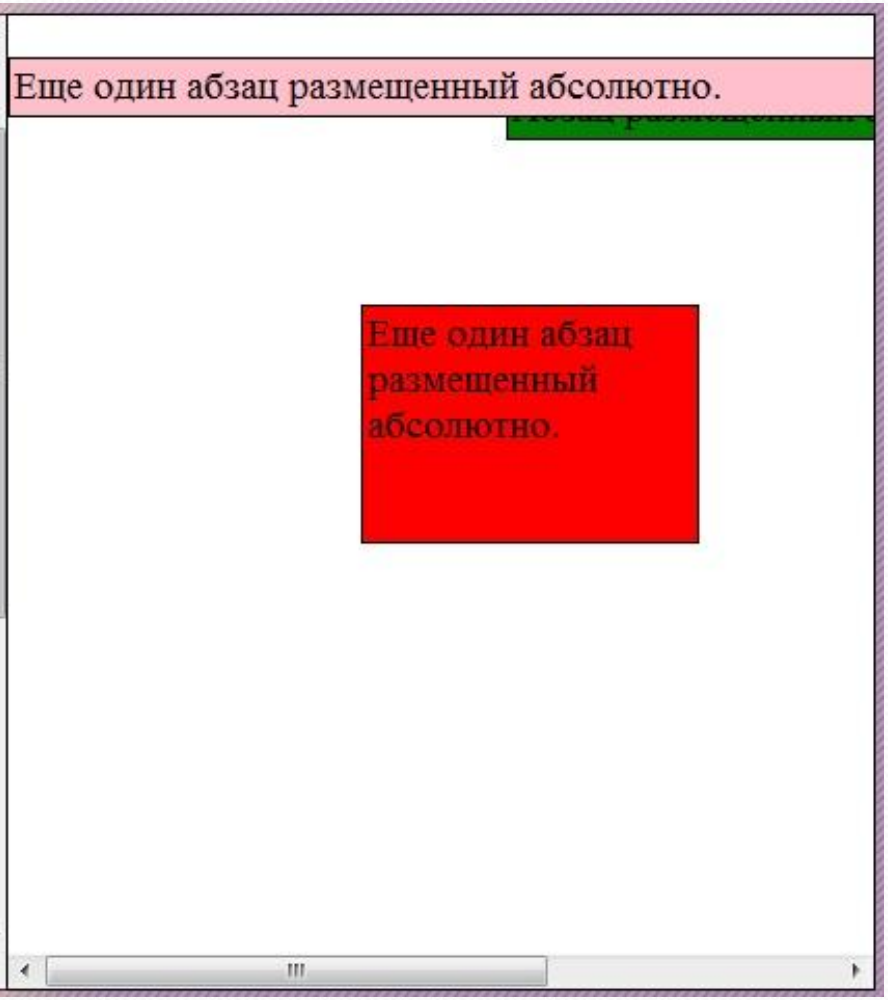
Первый абзац не видим, но занимает место.

Размещение элементов

```
#pos1
{
    position:absolute;
    top:10px; left:200px;
}
```

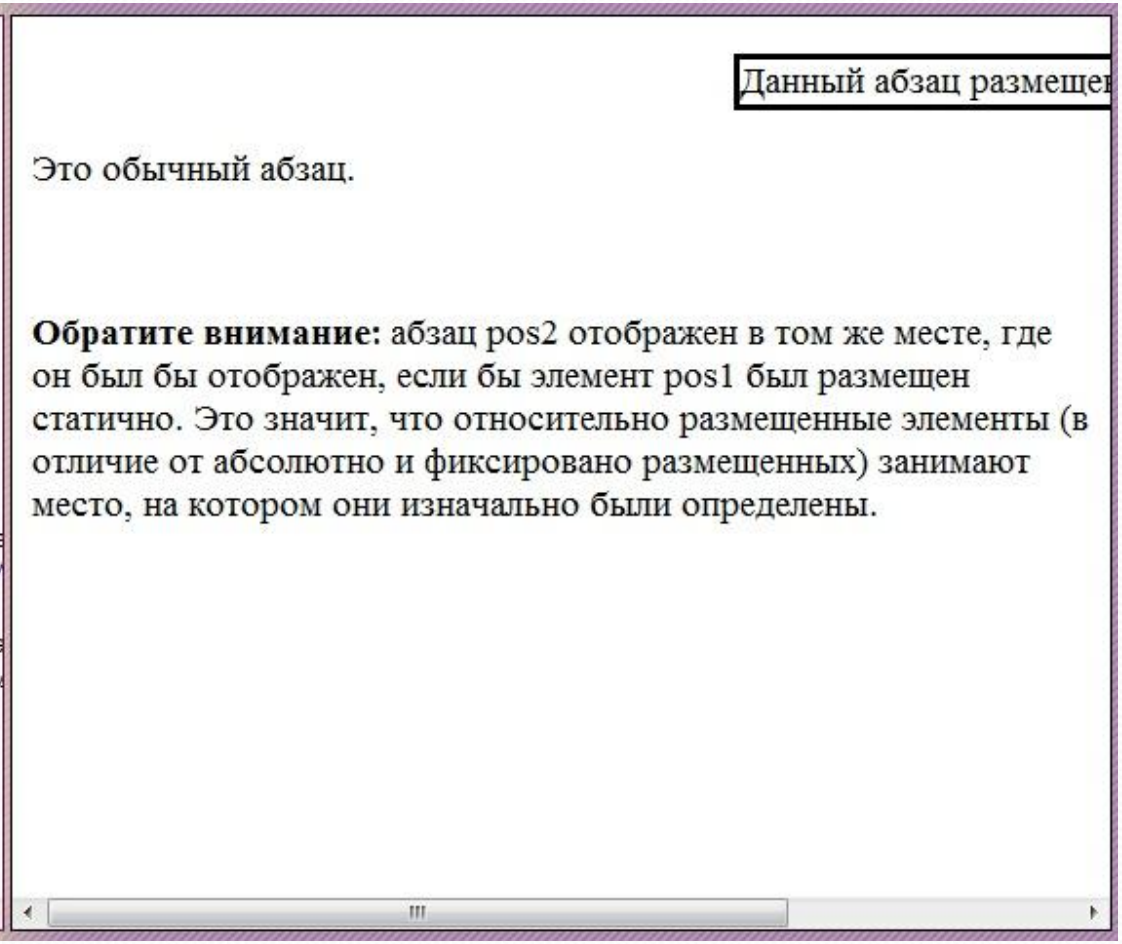
Размещение элементов

```
#pos1 {
  position:absolute;
  top:10px;
  left:200px;
  background-color:green;
  padding:2px;
  width:300px;
  border:1px solid;
}
#pos2 {
  position:absolute;
  top:1px;
  left:0px;
  background-color:pink;
  padding:2px;
  width:400px;
  border:1px solid;
}
#pos3  {
  position:absolute;
  top:100px;
  right:70px;
  background-color:red;
  padding:2px;
  height:90px;
```



Относительное размещение

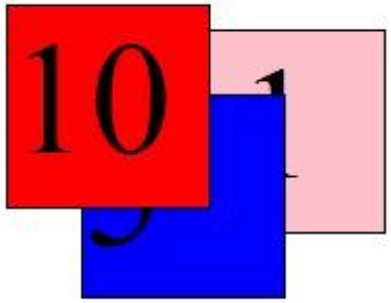
```
 <html>
<head>
<style type='text/css'>
#pos1
{
  position:relative;
  top:0px;
  left:300px;
  border-style:solid;
  width:300px;
}
</style>
</head>
<body>
<p id='pos1'>Данный абзац размеще
<p id='pos2'>Это обычный абзац.</
<br />
<p><b>Обратите внимание:</b> абза
Это значит, что относительно разм
</body>
</html>
```



Наложение элементов

```
<style type='text/css'>
#pos1
{
  position:absolute;
  top:0px;
  left:100px;
  border:1px solid;
  width:80px;
  height:80px;
  background-color:red;
  font-size:4em;
  z-index:10;
}
#pos2
{
  position:absolute;
  top:36px;
  left:130px;
  border:1px solid;
  width:80px;
  height:80px;
  background-color:blue;
  font-size:4em;
  z-index:5;
}
```

Элемент с *z-index:10* - красный, с *z-index:5* - синий, а с *z-index:-1* - розовый.



HTML5

- **Теги-контейнеры:** header, footer, article, nav, section, aside
- **Теги работы с текстом:** dialog, mark, time, progress
- **Мультимедиа:** audio, video
- **Интерактивные элементы:** canvas, menu, datagrid

HTML5

HTML5 - это новая версия HTML.
Предыдущая версия HTML была выпущена в 1999 году, за это время всемирная паутина сильно изменилась и поэтому изменения требовались и от HTML.
В HTML5 было добавлено много нового, поэтому некоторые возможности, которые были доступны в HTML4 только с использованием внешних плагинов (*таких как Adobe Flash*) или клиентских скриптов теперь доступны с помощью обычных разметочных тэгов.

HTML5. Новые возможности

- Поддержка видео и аудио (*элементы video и audio*);
- Возможности рисования на веб-страницах произвольных объектов (*элемент canvas*);
- Улучшение форм (*новые значения type для <input> и множество новых элементов и атрибутов*);
- Добавление семантических тэгов, позволяющих сделать веб-страницы более понятными для поисковых систем, браузеров и других программ и устройств читающих веб-страницы (*элементы footer, header, nav, article*);
- DOM хранилища - более функциональная альтернатива cookie.

DOCTYPE

Все HTML5 документы должны начинаться с объявления DOCTYPE.
Предыдущие версии HTML имели несколько типов DOCTYPE. HTML5 имеет только один:

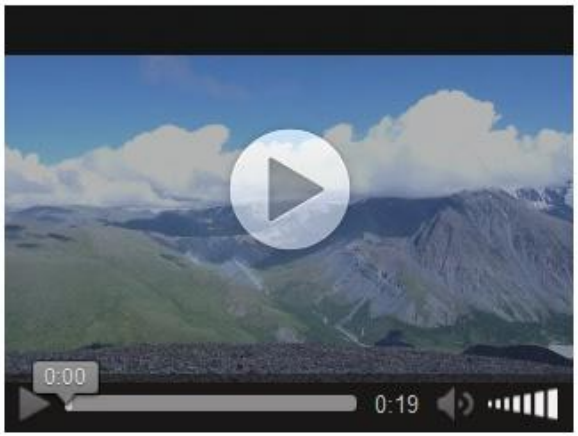
<!DOCTYPE html>

Данное объявление переводит все браузеры в нормальный режим. Браузеры не поддерживающие HTML5 в данном режиме будут интерпретировать старые теги и игнорировать новые, которые они не поддерживают.

Видео

```
<html>
<body>
<p>Панорама горы Белуха с перевала Кара-Тюрек.</p>
<video  src="mountvideo.ogv"
        width="300" height="200" controls="controls">
</video>
</body>
</html>
```

Панорама горы Белуха с перевала Кара-Тюрек.



Браузеры: поддержка форматов видео

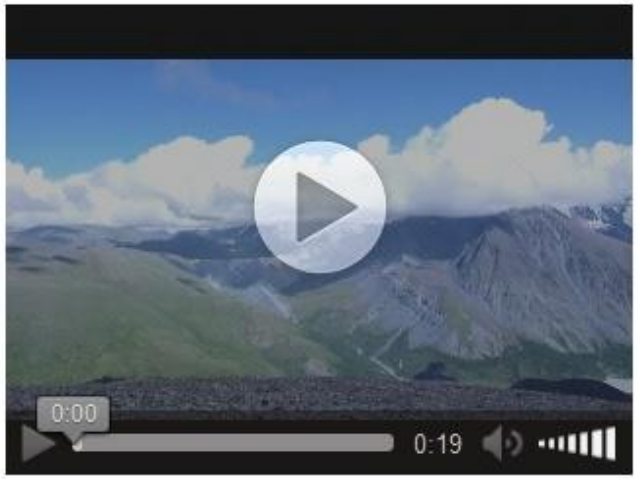
- Opera и Firefox: Ogg и WebM
- Internet Explorer и Safari: MPEG4
- Chrome: все.

Таким образом для того, чтобы видео нормально воспроизводилось во всех браузерах необходимо добавить источники в формате Ogg и MPEG4 (*или WebM и MPEG4*).

Видео: разные форматы

```
<html>
<body>
<p>Панорама горы Белуха с перевала Кара-Тюрек.</p>
<video width="300" height="200" controls="controls">
  <source src="mountvideo.ogv" type="video/ogg" />
  <source src="mountvideo.mp4" type="video/mp4" />
  <source src="mountvideo.webm" type="video/webm" />
</video>
</body>
</html>
```

Панорама горы Белуха с перевала Кара-Тюрек.



Аудио

```
1  <html>
2  <body>
3  <p>Трек: <b>Stop (<i>blue mix</i>)</b>.<br />
4  Автор: <b>Ghost_k (<i>Ghost Kollective</i>) (
5
6  <audio src="ghost_k-stop.ogv"
7    controls="controls">
8  </audio>
9
10 </body>
11 </html>
12
```

Трек: **Stop** (***blue mix***).
Автор: **Ghost_k** (***Ghost Kollective***) (***feat. hymn***).

Браузеры: поддержка форматов аудио

В предыдущем примере используется только формат Ogg, поэтому пример будет работать только в Firefox, Opera и Chrome.

Для того, чтобы файл нормально проигрывался в Safari и Internet Explorer необходимо добавить также файл в формате MP3.

Аудио: разные форматы

```
<html>
<body>
<p>Трек: <b>Stop (<i>blue mix</i>)</b>.<br />
Автор: <b>Ghost_k (<i>Ghost Kollective</i>) (<i>feat.

<audio controls='controls'>
  <source src="ghost_k-stop.ogv" type="audio/ogg" />
  <source src="ghost_k-stop.mp3" type="audio/mpeg" />
  Данный текст будет выведен если браузер пользовател
</audio>

</body>
</html>
```

Трек: **Stop** (***blue mix***).
Автор: **Ghost_k** (***Ghost Kollective***) (***feat. hymn***).

Аудио: autoplay

```
1  <html>
2  <body>
3  <p>Трек: <b>Stop (<i>blue mix</i>)</b>.<br />
4  Автор: <b>Ghost_k (<i>Ghost Kollective</i>) (<
5
6  <audio autoplay='autoplay'>
7    <source src="ghost_k-stop.ogv" type="audio/o
8    <source src="ghost_k-stop.mp3" type="audio/m
9    Данный текст будет выведен если браузер поль
10 </audio>
11
12 </body>
13 </html>
14
```

Трек: **Stop** (***blue mix***).
Автор: **Ghost_k** (***Ghost Kollective***) (***feat. hymn***)

Конвертация видео и аудио

Miro Video Converter

Семантические теги

В HTML4 благодаря использованию CSS Вы могли создавать страницы с хорошо понятной для пользователей визуальной структурой, но были ли эти страницы также понятны для поисковых систем или браузеров?

Например, как поисковый робот может отличить содержимое документа от навигационного меню если они размечены с помощью одинаковых div элементов?
Для того, чтобы разрешить эту проблему в HTML5 были введены **семантические тэги**. С помощью семантических тэгов Вы можете сделать страницы сайтов более понятными для поисковых систем и браузеров.

Семантические теги

Тэг	Описание
<footer>	Определяет футер.
<header>	Определяет заголовочный блок сайта.
<nav>	Определяет навигационное меню.

HTML4: структура страницы

HTML5: структура страницы

Группировка содержимого

С помощью тэга **<section>** Вы можете группировать логически связанное содержимого в документе.

Если логически связанное содержимое является автономным (*может использоваться в других документах независимо от остального содержимого на странице*) необходимо использовать вместо <section> тэг **<article>**.

Группировка содержимого

```
<article>
<h1>Титаник</h1>
<p>«Титаник» — британский пароход компании «Уайт Стар Лайн». Во время первого рейса 14 апреля 1912 года столкнулся с айсбергом и через 2 часа 40 минут затонул. На борту находилось 1316 пассажиров и 908 членов экипажа, всего 2224 человека. </p>
<section>
<h2>Постройка</h2>
<p>Заложен 31 марта 1909 года на верфях судостроительной компании «Харланд энд Вольф» в Куинс-Айленд (Белфаст, Северная Ирландия), спущен на воду 31 мая 1911 года, прошёл ходовые испытания 2 апреля 1912 года.</p>
</section>
<section>
<h2>Местонахождение</h2>
<p>1 сентября 1985 года экспедиция под руководством директора Института океанологии города Вудс-Холл, штат Массачусетс, доктора Роберта Балларда (Robert D. Ballard) обнаружила место залегания «Титаника» на дне Атлантического океана на глубине 3750 метров.</p>
</section>
</article>
```

Тег <aside>

Тэг aside используется для выделения элементов, которые не являются частью содержимого, но косвенно с ним связаны.

Данным тэгом могут выделяться: цитаты, дополнительная информация к статье, словарь с терминами, список ссылок и т.д.

```
<article>
<h1>Титаник</h1>
<p>«Титаник» — британский пароход компании «Уайт Стар Лайн». Во время первого рейса 14
апреля 1912 года столкнулся с айсбергом и через 2 часа 40 минут затонул. На борту
находилось 1316 пассажиров и 908 членов экипажа, всего 2224 человека. </p>
<aside>Гибель Титаника - это одно из самых крупных кораблекрушений в истории человечества.</a
</article>
```

Подсветка важного текста

```
1  <html>
2  <body>
3
4  <p>Я считаю, что <mark>HTML5</mark>
5     облегчает жизнь веб-разработчиков.
6  </p>
7
8  </body>
9  </html>
10
```

Я считаю, что HTML5 облегчает жизнь веб-разработчиков.

Подписи для иллюстраций

```
<html>
<body>
<html>
<body>
<p>Западный Саян ограничивается с запада Шапшальским хре
<p>Он тянется в широтном направлении полосой, постепенно

<figure>
  <img src='mountimg3.jpg' width='300' height='230' />
  <figcaption>
        Скала "Братья", Западный Саян
  </figcaption>
</figure>

</body>
</html>
```

Восточного Алтая и Абаканским хребтом Кузнецкого Алатау.

Он тянется в широтном направлении полосой, постепенно сужающейся с 200 до 80 км, от верховьев реки Абакан до стыка с хребтами Восточного Саяна в верховьях рек Казыр, Уда и Кижи-Хем.

Скала "Братья", Западный Саян

Новые типы input

input type=email

input type=number

input type=url

input type=range

input type=tel

input type=search

```
<input name='email' type='email' value='Не email' />
<input name='url' type='url' value='Не url' />
<input name='tel1' type='tel' pattern='8[0-9]{10}' />
<input name='tel2' type='tel' pattern='[0-9]{2,3}-[0-9]{2}-[0-9]{2}' />
<input name='number' type='number' value='10' />
<input name='range' type='range' min='1' max='5' />
<input name='search' type='search' value='Поиск по сайту' />
<input name='color' type='color' />
```

Проверка валидатором

http://validator.w3.org/

- **Validate By URL** (проверить по URL). Для проверки существующего сайта в Интернете
- **Validate By File Upload** (проверить с помощью загрузки файла). Для проверки страницы еще не загруженного сайта
- **Validate By Direct Input** (проверить с помощью прямого ввода). Для проверки создаваемого кода

Оформление таблиц

```
<html>
<head>
<style type='text/css'>
table, th, td
{
border-style:solid;
border-width:1px;
border-collapse:collapse;
padding:2px;
}
th
{
height:28px;
background-color:#f892dc;
color:white;
border-color:black;
}
.ts1 {background-color:#ffeffb;}
</style>
</head>
<body>
<h4> Самые высокие небоскребы по контин
<table>
<tr>
<th> Континент </th>
```

Самые высокие небоскребы по континентам:

Континент	Название	Город	Высота, м
Австралия и Океания	Q1	Голд-Кост	323
Азия	Дубайская башня	Дубай	828
Африка	Карлтон Сентр Офис Тауэр	Йоханнесбург	223
Европа	Башня 'Москва'	Москва	302
Северная Америка	Сирс Тауэр	Чикаго	443
Южная Америка	Парке Сентраль	Каракас	225

Способы подключения к HTML

- Внутренние стили (внутри тегов, атрибут style)
- Глобальные стили (указанные внутри тега style)
- Связанные стили (отдельный файл CSS, подключение в html через link)
- Импорт стилей (еще один вариант подключения внешних CSS-файлов)

Строковое подключение стиля

```
<p style="font-size:1.3em">
    Абзац оформленный с помощью CSS.
</p>
```

Внутренние стили

```
<head>
    <style type='text/css'>
        h1 {color:red;}
        p {margin-right:38px;}
        div {float:left;}
    </style>
</head>
```

Подключение внешнего файла

```
<head>
    <link rel="stylesheet" type="text/css"                    href="
  адрес_внешнего_файла_стилей" />
</head>
```