УРОК №6. Формирование блочной модели

Содержание урока

- Основные теги для верстки (div и span)
- Свойство display
- Формирование блочной модели
- Обтекаемые элементы
- Позиционирование блоков

Основные теги для верстки

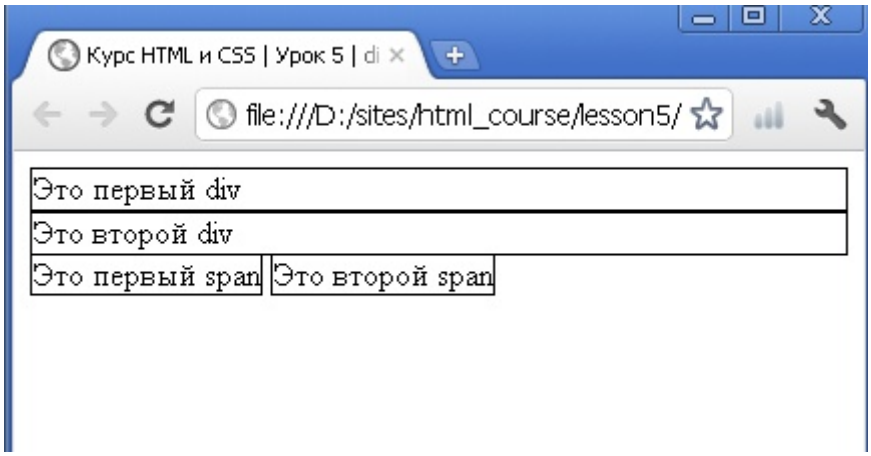
При верстке сайта при помощи слоев, самый часто используемый html тег называется , который как раз и формирует слой на веб-странице и он является блочным тегом. Второй тег, который используется при верстке – это строчный тег . Сами по себе эти теги ничего на экране не отображают, и оформляются они стилями css.

```
<div>Это блочный элемент</div>

<span>Это строчный элемент</span>
```

Для демонстрации работы тегов и рассмотрим небольшой пример.

```
div, span {

    border: 1px solid #000;

}

<div>Это первый div</div>

<div>Это второй div</div>

<span>Это первый span</span>

<span>Это второй span</span>
```

На странице находятся два элемента и два элемента . При помощи стилей css всем элементам задана сплошная граница черного цвета, толщиной 1 px. Если запустить этот пример в браузере, то получится следующее:



Элементы занимают всю ширину окна браузера, и располагаются соответственно друг под другом. А ширина элементов получается ровно столько, сколько занимает их содержимое, и ведут они себя так же, как и другие строчные элементы HTML.

Рассмотрим еще одно отличие от .

```
div, span {
    border: 1px solid #000;
    width: 400px;
    height: 50px;
}
<div>Это первый div</div>
<div>Это второй div</div>
<span>Это первый span</span>
<span>Это второй span</span>
```

В предыдущий пример добавим элементам и ширину = 400 px и высоту = 50 px.

Теперь видно, что стили применились только к элементам . А теги остались при своих размерах. Можно сделать вывод, что ширину и высоту нельзя задать строчным элементам.

Давайте вспомним, какие основные элементы html относятся к блочным, а какие к строчным.

Блочные элементы

- <form>
- <h1>
- <p>
- <table>
-

Строчные элементы

- <img />
- <a>
-
-

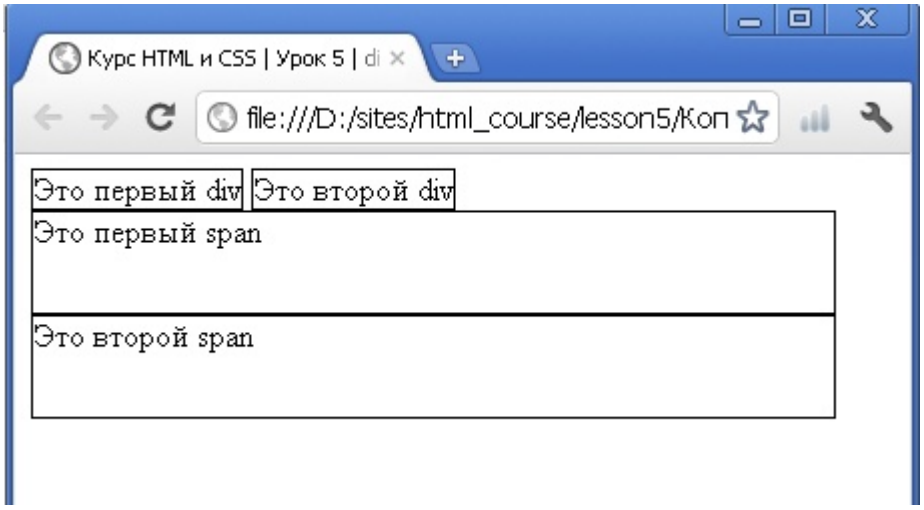
В этом списке тег картинки <img> является замещаемым строчным элементом, т.е. при помощи замещаемых элементов указывается, что в данном месте должен быть какой-то сторонний объект, в данном случае картинка. И замещаемому элементу можно задать и ширину и высоту. Но все равно замещаемый элемент будет являться строчным.

Свойство display

При помощи css можно изменить тип элемента, т.е. блочный тег можно сделать строчным, а строчный - блочным. Для этого существует css свойство - display. Если вернуться к предыдущему примеру и для элементов задать значение свойства display: inline;, а для - значение display: block;.

```
div, span {
	border: 1px solid #000;
	width: 400px;
	height: 50px;
}
div {
	display: inline;
}
span {
	display: block;
}

<div>Это первый div</div>
<div>Это второй div</div>
<span>Это первый span</span>
<span>Это второй span</span>
```

В этом случае получается, что элементы поменялись местами, стал строчным элементом, и ему теперь невозможно задать ни ширину, ни высоту, а стал блочным, и ему теперь можно задать и ширину и высоту.



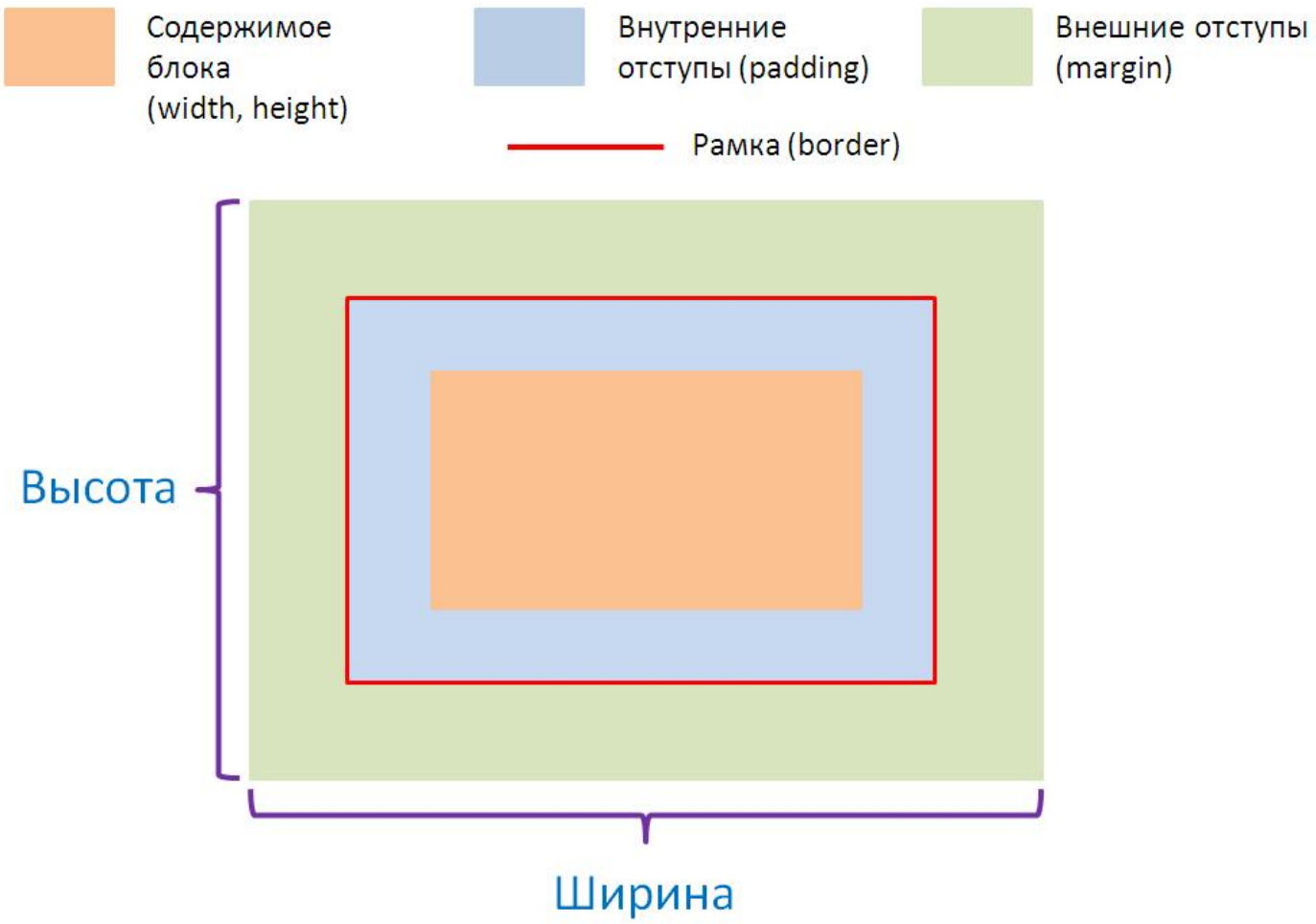
Некоторые часто используемые значения свойства display:

- none
- block
- inline
- inline-block
- table-cell

Значение none делает блок невидимым, точнее совсем убирает блок из документа и получается, что его как будто там и нет. Вернуть блок на страницу можно при помощи языка программирования javascript. Значения inline делает элемент строчным и наоборот, значение block – блочным. inline-block представит элемент подобно замещаемому элементу <img />, т.е. элемент будет строчным, но внутреннее содержимое блочным, и при этом можно изменять размеры блока. Значение table-cell установит элемент в виде ячейки таблицы <td>, что иногда оказывается полезным, например при задании вертикального выравнивания у блока.

Формирование блочной модели

На первый взгляд может показаться что, width – это окончательная ширина элемента, height - это окончательная высота элемента. На самом деле это не так, width и height - это не окончательные размеры элемента. Для того, чтобы вычислить размеры, необходимо учитывать следующие моменты.



Если внимательно ознакомиться с данной схемой, то можно сделать вывод, что ширина блока складывается из следующих свойств:

```
margin-left +
border-left +
padding-left +
width +
padding-right +
border-right +
margin-right
```

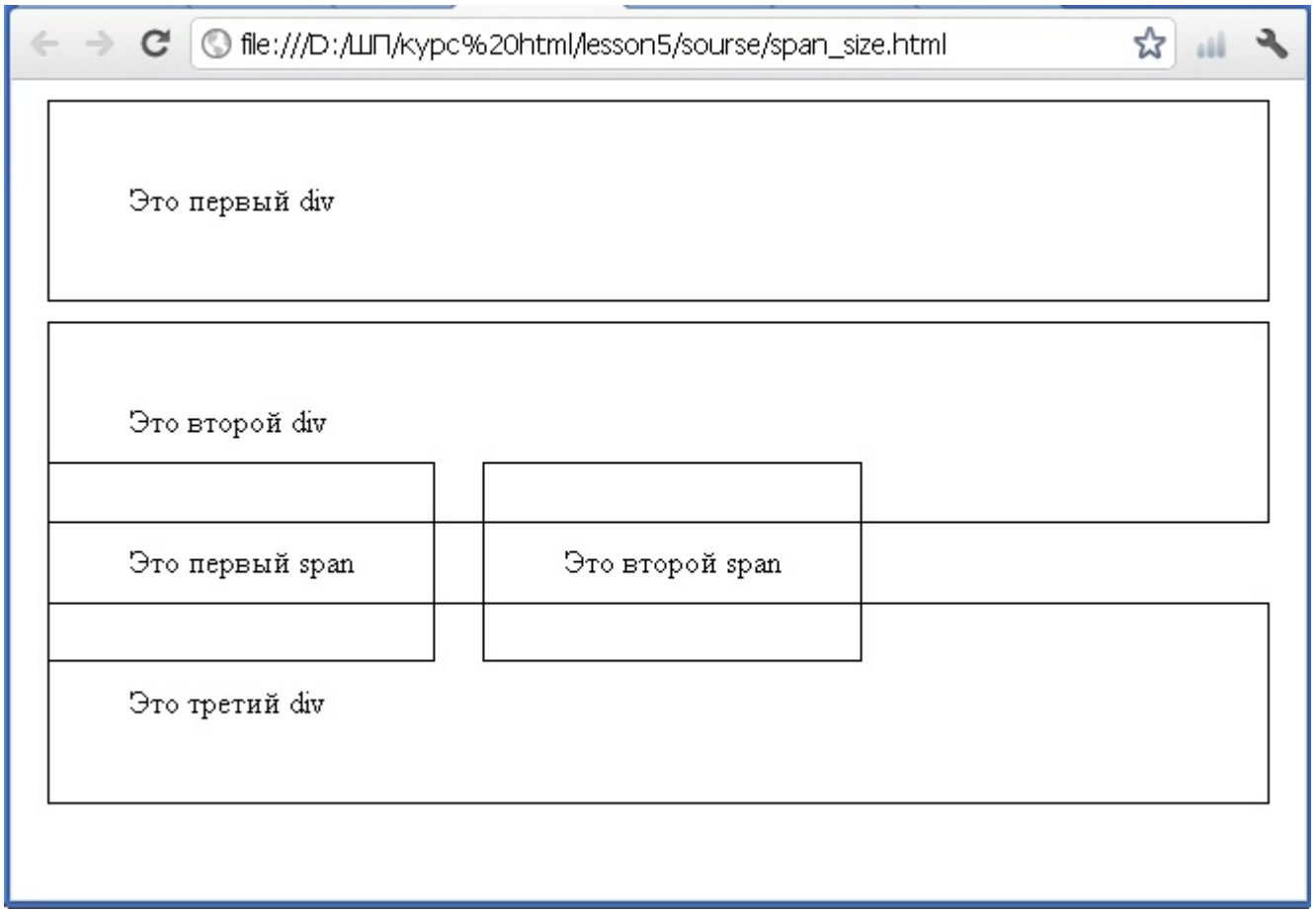
Соответственно, высота из следующих:

```
margin-top +
border-top +
padding-top +
height +
padding-bottom +
border-bottom +
margin-bottom
```

Рассмотрим, как формируется размер у строчного элемента:

```
div, span {
	border: 1px solid #000;
	margin: 10px;
	padding: 40px;
}

<div>Это первый div</div>
<div>Это второй div</div>
<span>Это первый span</span>
<span>Это второй span</span>
<div>Это третий div</div>
```

В данном примере для элементов и задаются одинаковые свойства css. Посмотрим на действие данного примера в браузере:



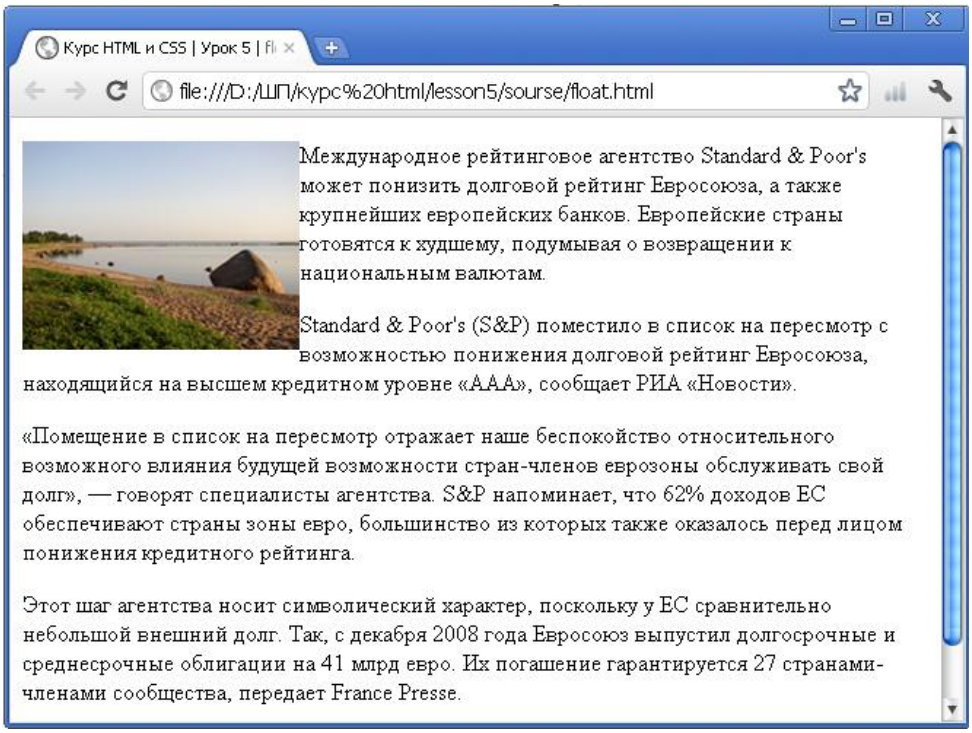
К элементы применились все заданные стили css . У них есть и внешние отступы margin со всех 4 сторон, и все 4 внутренних отступа padding. И Дивы располагаются друг под другом. А элементы накладываются на второй и третий . Это происходит потому, что внутренние отступы padding есть у всех четырех сторон строкового элемента, а свойство margin действует на

строковый элемент только слева и справа, т.е. у строкового элемента есть margin-left и margin-right, а margin-top и margin-bottom игнорируются.
Еще один важный момент, всем элементам были заданы одинаковые отступы, как внутренние, так и внешние. В данном примере между первым и вторым элементами , расстояние будет равняться 10 px. В этом случае применяется правило так называемых *схлопывающихся отступов*. Его суть в том, что вертикальные внешние отступы у блоков не суммируются, а берется тот внешний отступ, у которого значение больше, и это значение как раз и будет расстоянием между блоками.

Обтекаемые элементы

Обтекаемые элементы, или как их еще называют «плавающие», используются для реализации обтекания текстом изображений, создании врезок, и даже создания много-столбцовых компоновок. Также обтекаемые элементы активно используются при верстке веб-страниц и при помощи них возможно заменить табличную верстку на верстку слоями.
Для того, чтобы задать обтекание, в CSS существует только одно свойство float, которое может принимать всего два значения - это left и right.

В следующем примере картинке, т.е. тегу <img />, заданно свойство float: left;



В этом случае картинка займет положение слева и позволит любым элементам, будь они строчные или блочные, обтекать себя справа.

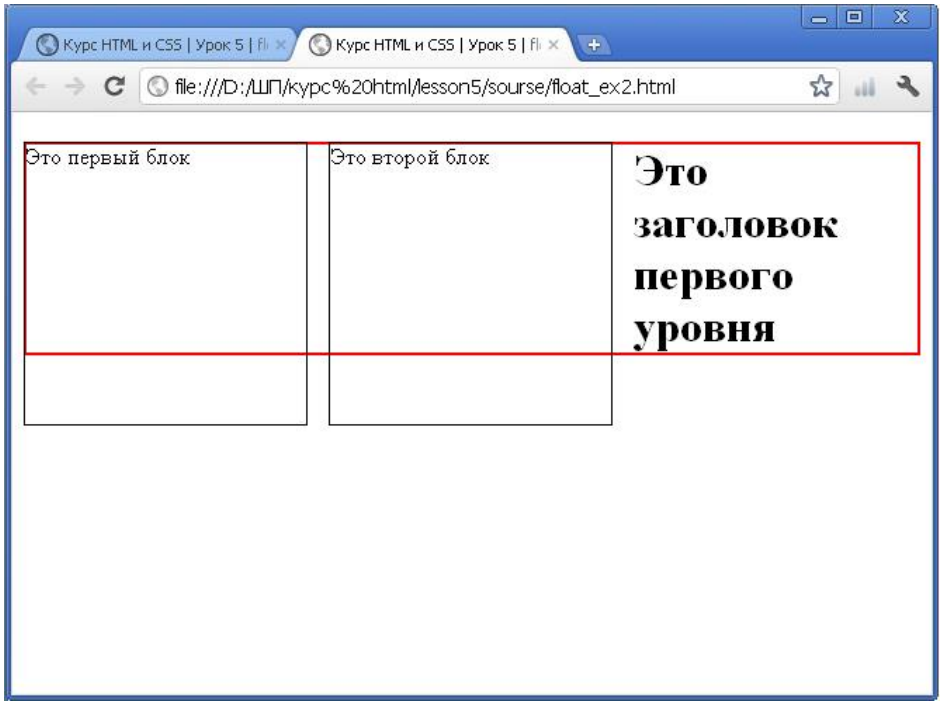
В следующем примере картинке задано свойство float: right;

И то тогда происходит та же ситуация, только картинка становится справа, и позволяет обтекать себя слева.

Далее рассмотрим следующий пример. Создадим два элемента и один заголовок первого уровня <h1>

```
div {
	width: 200px;
	height: 200px;
	margin-right: 15px;
	border: 1px solid #000;
	float: left;
}
h1 {
	border: 2px solid #f00;
}

<div>Это первый блок</div>
<div>Это второй блок</div>
<h1>Это заголовок первого уровня</h1>
```

У обоих элементов задано свойство float: left;, т.е. они должны занимать левое положение и позволять обтекать себя справа. Посмотрим, на действие этого примера в браузере.



Разберемся, что же произошло. Элементы находятся на одной линии по горизонтали, что и ожидаемо, т.к. у них задано свойство float: left;. Первый занял положение слева, позволил обтекать себя справа. Второй , соответственно, в свою очередь также позволил обтекать себя справа. Заголовок первого уровня находится справа второго элемента , но его рамка обрамляет также оба элемента . Это происходит потому, что у свойства float есть особенность: элементы, которым заданно это свойство начинают притягивать к себе все близлежащие элементы, и заставляют их тоже участвовать в обтекании. Но с этим можно бороться.

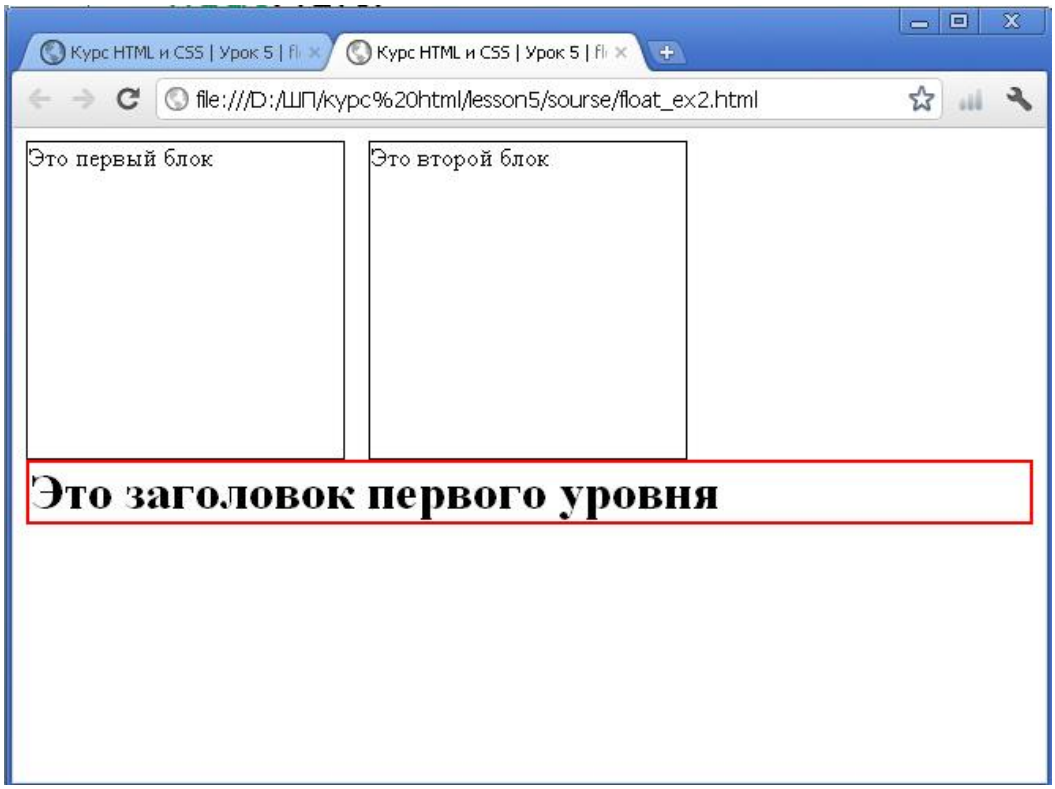
Рассмотрим две ситуации.

1. Заголовок не должен участвовать в обтекании и должен находиться под элементами .

В этом случае необходимо применить запрет на обтекание. Для этого в css существует свойство clear. Оно может принимать три значения - это left, отменяющее обтекание с левого края, right - с правого края, и значение both - которое отменяет обтекание с обеих сторон. Добавим свойство clear со значением both для заголовка первого уровня.

```
div {
	width: 200px;
	height: 200px;
	margin-right: 15px;
	border: 1px solid #000;
	float: left;
}
h1 {
	border: 2px solid #f00;
	clear: both;
}

<div>Это первый блок</div>
<div>Это второй блок</div>
<h1>Это заголовок первого уровня</h1>
```

Если запустить данный код в браузере, то заголовок уже не будет участвовать в обтекании, а будет находиться под элементами .

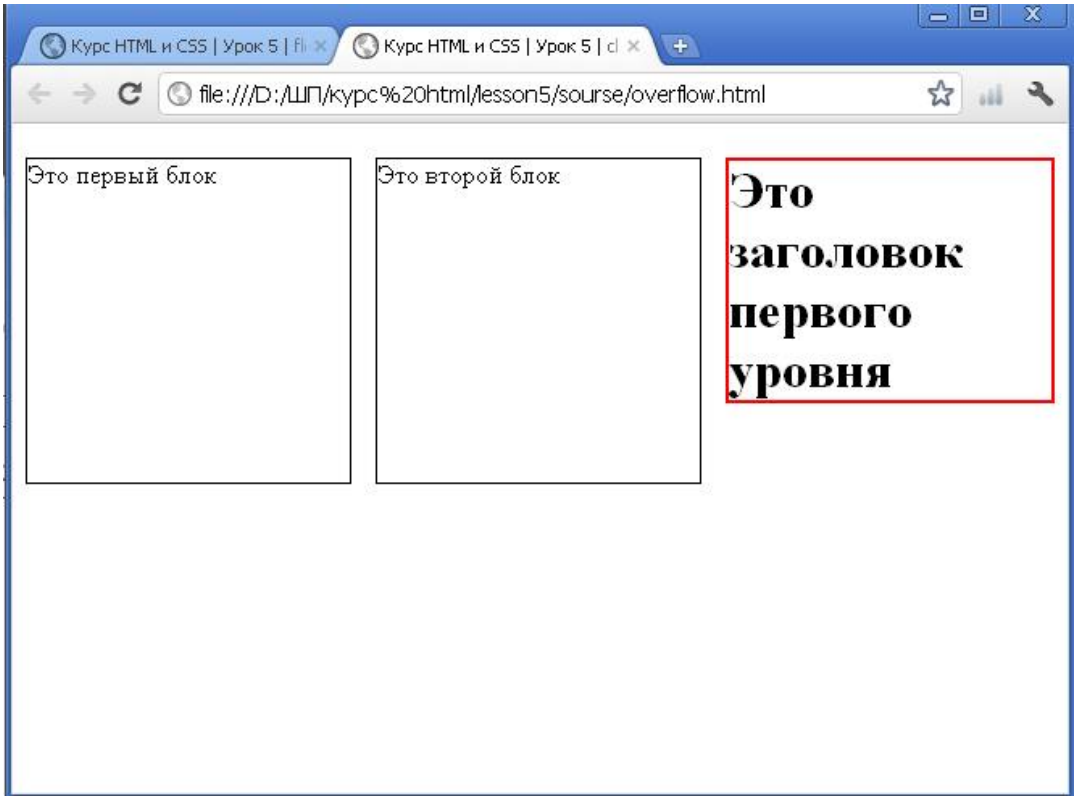


2. Заголовок остается на том же месте, где он сейчас находится, но рамка должна обрамлять только сам заголовок.

Для решения этой задачи, поможет css свойство overflow. Оно определяет, как будет вести себя блочный элемент в случае его переполнения, и при значении hidden, отображает только содержимое этого элемента.

```
div {
      float: left;
      width: 200px;
      height: 200px;
      margin-right: 15px;
      border: 1px solid #000;
}
h1 {
      border: 2px solid #f00;
      overflow: hidden;
}

<div>Это первый блок</div>
<div>Это второй блок</div>
<h1>Это заголовок первого уровня</h1>
```

Если запустить этот пример в браузере, то заголовок остается на том же месте, и рамка теперь обрамляет только элемент <h1>.



Позиционирование блоков

Идея, лежащая в основе позиционирования довольно проста. Позиционирование позволяет точно определить, где появятся блоки относительно другого элемента или относительно окна браузера. Свойство css, которое позволяет управлять позиционированием, называется - position. Оно может принимать 4 значения, таким образом, существуют 4 вида позиционирования.

static (значение по умолчанию) - соответствует нормальному потоку документа, т.е. элементы будут идти в том порядке, в котором вы их зададите.

relative (относительное позиционирование) - элемент будет смещаться относительно его определенного в настоящее время положения, и при этом его место будет оставаться не заполненным.

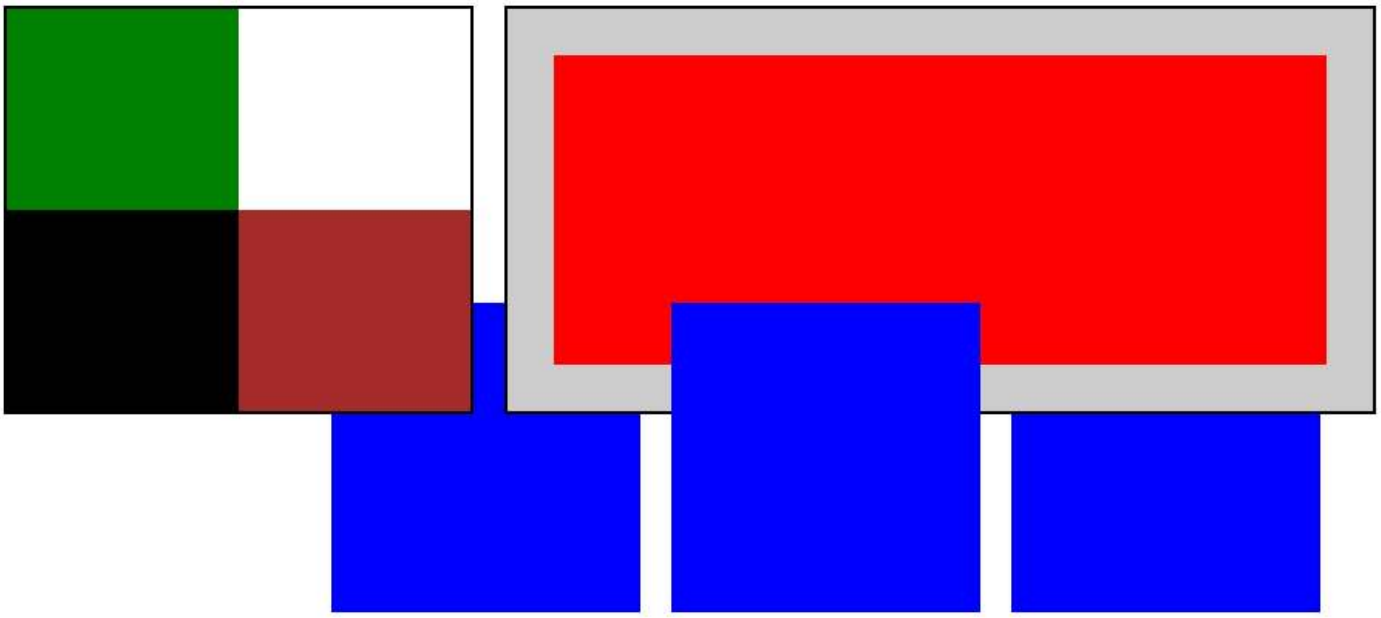
absolute (абсолютное позиционирование) - элемент вообще исключается из потока документа, как будто его там вообще не существует.

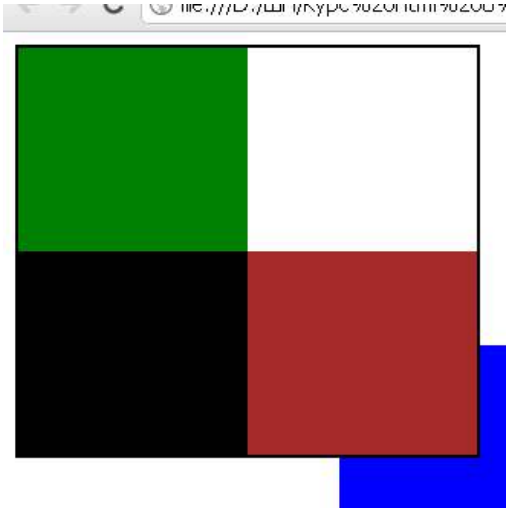
fixed (фиксированное позиционирование) - похоже на абсолютное, только элемент будет оставаться неподвижен при прокрутке окна просмотра.

Совершенно бесполезным будет свойство position без свойств смещения. Эти свойства называются left, right, top, bottom, и применяются они для смещения блоков в соответствующие стороны. В качестве значений они могут принимать любые единицы измерения css. Свойства смещения не применяются для элементов, позиционированных в нормальном потоке, т.е. для свойства position со значением static, и будут проигнорированы.

Домашнее задание

Немного отвлечемся от наших проектов, и в этом задании соберите следующий «конструктор» из блоков.





1. Блок слева состоит из 5 блоков: 1 родительский и в нем 4 блока разных цветов со свойством position: relative.

2. Красному блоку, который вложен в правый блок серого цвета задайте только 3 свойства: ширину, высоту и цвет фона.
3. Синие блоки одинаковых размеров и разместите их используя свойство position: absolute.

И вторая часть задания: сверстайте такое же меню, как на рисунке ниже. Используйте при этом только слои. Пункты меню должны быть ссылками. При наведении на определенный пункт меню, он должен изменить цвет. При наведении на блок с пунктом меню и картинкой, должна появиться рамка (см. рис.).


Пункт 1
Пункт 2
Пункт 3
Пункт 4
Пункт 5