

HyperText Markup Language

HTML & CSS



Урок 6.1

Блочная модель элементов

Оглавление

Блочная модель элементов	3
Ширина элемента.....	4
Высота элемента.....	7
Свойство overflow.....	16
Границы (рамки) элемента	21
Использование свойств блочной модели	35
Инструменты разработчика, или Инспектор свойств	46
Отмена обтекания	65
Класс clearfix.....	72
Использование свойств блочной модели для верстки html-страницы	75
Домашнее задание.....	117

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Блочная модель элементов

Box-model, или блочная модель элементов, подразумевает набор нескольких CSS-свойств, которые назначают именно для блочных элементов, так как к строчным они либо совсем не применяются, либо применяются не так, как вы прогнозируете. С частью этих свойств вы познакомились на прошлом занятии — это внешние отступы (`margin`) и внутренние (`padding`). Кроме того, к свойствам блочной модели относятся ширина (`width`) и высота (`height`) элемента, а также его границы — свойство `border` (рис. 1).

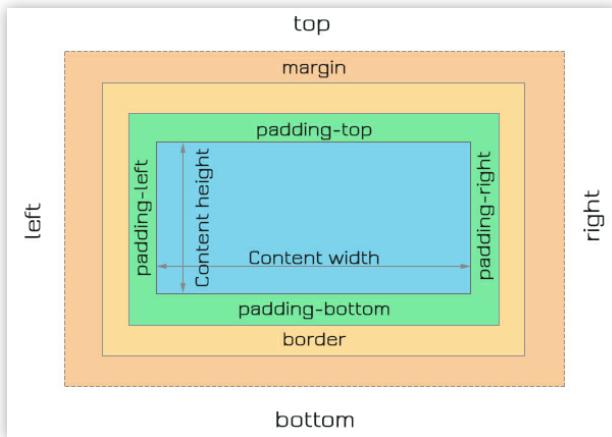


Рисунок 1

На рисунке видно, что блочная модель элемента отталкивается от его содержимого, ограниченного свойствами `width` и `height`, а все остальные свойства распространяются во все стороны от контента. Поэтому для отступов (`padding` и `margin`), а также для границы элемента (`border`) важны еще названия сторон, для которых они

назначается. Для всех этих свойств можно указать, что они применяются только к верхней стороне блока (`top`), нижней (`bottom`), левой (`left`) или правой (`right`).

Любой блочный элемент, как правило, содержит некоторый контент, т.е. текст, изображения, видео и т.п. Поэтому первым важным свойством для него будет ширина — `width`. Рассмотрим подробнее, каким образом можно назначить в CSS ширину элементов.

Ширина элемента

Для блочного элемента можно задать CSS-свойство `width`, по умолчанию имеющее значение `auto`, а это значит, что любой блок занимает автоматически все доступное пространство внутри родительского элемента или `body`.

Также для `width` можно назначить значения в `px`, `em`, `vw`, `cm`, `mm` и др. единицах или в %. Когда ширина задается в процентах, ее расчет выполняется браузером всегда в пикселях относительно размера родительского элемента. Например, если в элементе `main` (основное содержимое страницы) задать

```
main { width: 80%; }
```

то при ширине окна браузера в 1300px ширина элемента `main` будет 1040px ($1340\text{px} \times 80/100 = 1040\text{px}$), а при уменьшении окна браузера до 600px ширина этого элемента составит уже 480px ($600\text{px} \times 80/100 = 480\text{px}$).

Для управления размерами элемента на маленьких и больших экранах и не только можно использовать такие CSS-свойства, как `max-width` и `min-width`. Свойство `max-width` определяет максимальную ширину элемента, а `min-`

width — минимальную его ширину. Можно сказать, что **max-width** ограничивает максимальное пространство, занимаемое элементом по ширине, а **min-width** не позволяет ему иметь ширину меньше той, что указана в этом свойстве. Оба свойства могут быть заданы с такими значениями:

```
max-width: размер в px, pt, em, in, % и др. единицах |  
none | inherit min-width: размер в px, pt, em, in, %  
и др. единицах | inherit
```

Для **min-width** значение по умолчанию 0, для **max-width** — none (рис. 2).

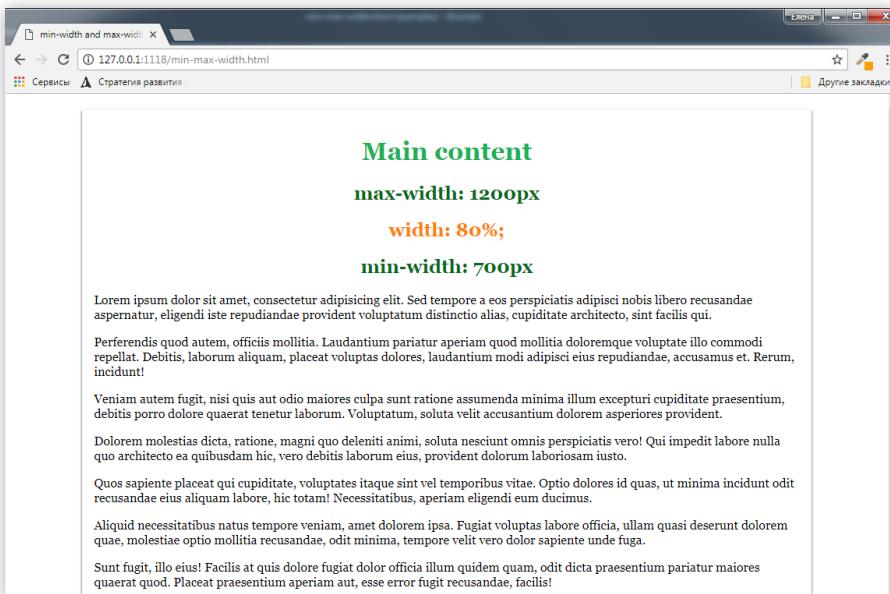


Рисунок 2

В файле примера `min-max-width.html` использованы все 3 значения для указания ширины основного контента. На широких экранах размер элемента не может быть

больше 1200px, указанных в свойстве `max-width`, на средних экранах он будет определяться свойством `width: 80%`, а на маленьких экранах появится горизонтальная полоса прокрутки, т.к. ширина не может быть менее 700px, указанных в свойстве `min-width` (рис. 3).



Рисунок 3

Полезные ссылки:

1. <https://developer.mozilla.org/en-US/docs/Web/CSS/width>.
2. https://www.w3schools.com/cssref/pr_dim_width.asp.

3. https://www.w3schools.com/cssref/pr_dim_min-width.asp.
4. <https://developer.mozilla.org/en-US/docs/Web/CSS/min-width>.
5. https://www.w3schools.com/cssref/pr_dim_max-width.asp.
6. <https://developer.mozilla.org/en-US/docs/Web/CSS/max-width>.
7. <https://webref.ru/css/width>.
8. <https://webref.ru/css/min-width>.
9. <https://webref.ru/css/max-width>.
10. <http://html-plus.in.ua/box-model-elementa-nyuansy/#box-width>.

Высота элемента

Как и в свойстве `width` для высоты элемента, определяемой свойством `height`, по умолчанию задано значение `auto`, которое подразумевает, что высота любого блока определяется в зависимости от высоты его содержимого (контента). Т.е. при большом количестве текста и картинок высота элемента будет большой, а для пары абзацев и заголовка высота будет иметь небольшой значение.

Чаще всего такой подход менять не стоит, т.к. количество текста в элементе определяется заказчиком и, кроме того, со временем может меняться, особенно в CMS — системах управления контентом. Поэтому жестко ограничивать его, как правило, не стоит. Тем не менее, в CSS предусмотрены 3 свойства, позволяющие задать высоту элемента явно. Это свойства `height`, `min-height` и `max-height`.

Доступные значения свойств:

```
height: размер в px, pt, em, in, % и др. единицах |  
        auto | inherit
```

```
max-height: размер в px, pt, em, in, % и др. единицах |  
        none | inherit  
min-height: размер в px, pt, em, in, % и др. единицах |  
        inherit
```

Для `min-height` значение по умолчанию 0, для `max-height` — `none`.

Если значение свойства `height` будет меньше, чем его содержимое, то содержимое будет выходить за пределы своего контейнера и наползать на текст, идущий следом.

В файле примера `height.html` верхнему элементу `div` назначен класс `fixed`, для которого задана высота `height: 200px`. Текст в этом `div` не помещается в эти 200px, поэтому накладывается на текст следующего `div`-а (рис. 4).

height: 200px;

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quo inventore temporibus, voluptate mollitia recusandae aut, dolorem, dolorum ex adipisci quibusdam autem deleniti necessitatibus aliquid hic dicta maxime maiores sequi pariatur! Magni minima ex sit, nulla dolore! Dolore maxime nesciunt facilis excepturi voluptates reiciendis odio itaque.

Ipsum inventore quasi, possimus eligendi fugiat eveniet cum perferendis, doloremque, est veritatis nemo nam ipsa! Omnis in voluptate et, iste, vitae autem modi sequi numquam commodi, enim at Nihil tempora, vitae recusandae eum optio possimus, excepturi voluptas magnam, minus unde aut, enim. Voluptas magnam, quia.

Iure, aliquid ab! Quos amet molestiae, nemo delectus natus quaerat recusandae doloribus neque soluta, culpa quam quod nesciunt ipsa odit. Iusto distinctio sequi lure doloremore. Voluptas magnam, eligendi qui. Illum dolore eaque, expedita cumque harum vero quam minus et quas praesentium modi nesciunt consectetur ullam?

Et consequuntur dolore quidem, temporibus vel ipsa id quod. Temporibus ex vite enim ad hunc punctum ducuntur tales adipisci in vel maxime dolores. Earum, diliq. difficile optinere atque, extenuatibus capi fabri non expedita, sed excepturi magni dolores. Iusto deleniti. Ut tempore dolore facilius rei quam que cillis amet temporibus. Voluptas magnam nobis asperiores. Ea tempore optio nostrum quaerat sunt neque.

Voluptatibus molestias vero perspicatis, odit perferendis assumenda quo ratione aut consectetur illum, suscipit deserunt neque exercitationem autem tenetur vitae ipsa! Quae ipsum officia fuga, dolor voluptatem minima at ducimus ipsam id dolorem hic recusandae, quo beatiae voluptates explicabo eum, aut accusantium voluptatibus fugiat vel Laborum.

Iusto deleniti, temporibus, a, ab recusandae modi repudiandae iste, atque similique animi quo voluptatibus. Illum praesentium veritatis aperiam numquam maxime vel autem, voluptatem quae reprehenderit minus! Magnam nisi quos deleniti consectetur hic tempore illo deserunt illum, est quisquam. Ex tempore magni autem labore illum quos.

Doloremque a laboriosam reiciendis, dolore voluptatum iste saepè. Nobis impedit saepè eveniet iste, quis recusandae repellat laudantium, similique explicabo dolores dolorum rem accusamus quos maiores, lute repellendus facere excepturi necessitatibus delectus totam commodi quod. Repellendus placeat cupiditate nesciunt illum sed, alias blanditiis nulla quos possimus.

Рисунок 4

Второй пример в этом же файле `height.html` демонстрирует разницу между 2-мя рядом расположены-

ми div-ми с белым цветом фона. Левый имеет высоту по умолчанию (height: auto), а в правом задана высота в 200px. Поэтому цвет фона распространяется только на эти 200px, а не поместившийся в них текст оказался на сером фоне родительского элемента (рис. 5).



Рисунок 5

В этом примере назначение **height** явно выглядит некрасиво. Это получилось потому, что высота блока имеет слишком маленькую величину. Если указать достаточное значение, явно заданное свойство **height** может улучшить внешний вид элементов.

Предположим, у нас есть 3 блока с разным количеством контента в каждом из них (рис. 6).



Рисунок 6

Смотрится не очень красиво, не так ли? Здесь и может помочь назначение высоты блока в пикселях не меньше, чем высота самого большого блока:

```
height: 350px;
```

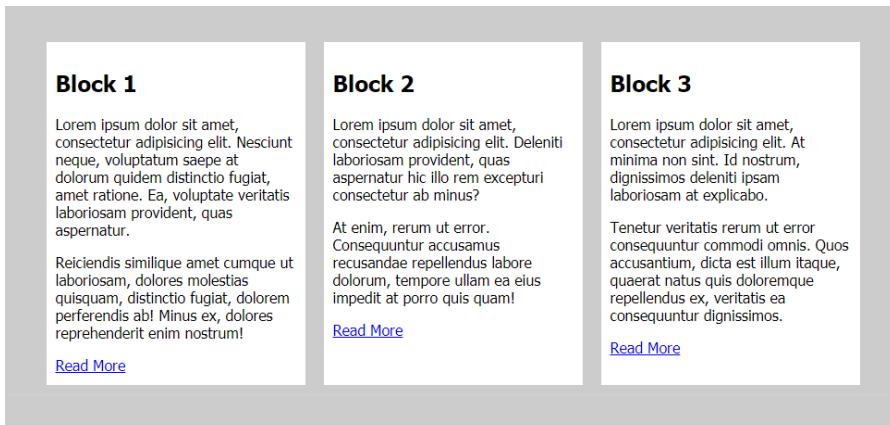


Рисунок 7

Вы можете посмотреть на работу этого свойства в файле *height-2.html*, используя css-комментарии для свойства **height** для селектора класса **box**:

```
.box {  
    width: 260px;  
    float: left;  
    margin: 10px;  
    padding: 10px;  
    background-color: #fff;  
    /* height: 350px; */  
}
```

Рисунок 8

Напомню: комментарии можно добавить и убрать в Brackets сочетанием клавиш **Ctrl + /**.

Рассмотрим еще один вариант. Предположим, мы не можем задать для 3-х рядом расположенных блоков фиксированную ширину в px, как это было в примере height-2.html (см. скриншот выше). Ширина должна быть в %, т.к. родительский элемент также имеет ширину в % относительно body (рис. 9).

```
.wrap {  
    width: 90%;  
    margin: 20px auto;  
}  
.box {  
    width: 33.3%;  
    float: left;  
}
```

Рисунок 9

Если в этом случае мы зададим фиксированное значение height в размере тех же 350px, что и в предыдущем примере, то при определенной ширине браузера никакой разницы не увидим (рис. 10).

Block 1
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nesciunt neque, voluptatum saepe at dolorum quidem distinctio fugiat, amet ratione. Ea, voluptate veritatis laboriosam provident, quas aspernatur.
Reiciendis similique amet cumque ut laboriosam, dolores molestias quisquam, distinctio fugiat, dolorem perferendis ab! Minus ex, dolores reprehenderit enim nostrum!
[Read More](#)

Block 2
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Deleniti laboriosam provident, quas aspernatur hic illo rem excepturi consectetur ab minus?
At enim, rerum ut error. Consequuntur accusamus recusandae repellendus labore dolorum, tempore ullam ea eius impedit at porro quis quam!
[Read More](#)

Block 3
Lorem ipsum dolor sit amet, consectetur adipisicing elit. At minima non sint. Id nostrum, dignissimos deleniti ipsam laboriosam at explicabo.
Tenetur veritatis rerum ut error consequuntur commodi omnis. Quos accusantium, dicta est illum itaque, quaerat natus quis doloremque repellendus ex, veritatis ea consequuntur dignissimos.
[Read More](#)

Рисунок 10

Если же ширину браузера уменьшить, то внешний вид блоков опять будет испорчен:

The screenshot shows three separate blocks labeled Block 1, Block 2, and Block 3. Each block contains placeholder text (Lorem ipsum...) and a 'Read More' link. The blocks are aligned horizontally and have equal height, which causes them to overlap when the browser window is very narrow.

Block 1

Placeholder text for Block 1.

Placeholder text for Block 1.

[Read More](#)

Block 2

Placeholder text for Block 2.

Placeholder text for Block 2.

[Read More](#)

Block 3

Placeholder text for Block 3.

Placeholder text for Block 3.

[Read More](#)

Рисунок 11

Здесь нам понадобится свойство min-height:

```
.box-inner {  
    background-color: #fff;  
    margin: 10px;  
    padding: 10px;  
    /* height: 350px; */  
    min-height: 350px;  
}
```

Рисунок 12

Вы не заметите разницы между height и min-height на широком экране. При уменьшении ширины браузера высота блоков будет интерпретироваться как auto, но

не будет меньше указанных 350px. Блоки будут иметь рваные края — это можно исправить только с помощью медиа-запросов. Однако текст будет находиться в пределах блока на белом фоне, а не за его пределами.

Block 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nesciunt neque, voluptatum saepe at dolorum quidem distinctio fugiat, amet ratione. Ea, voluptate veritatis laboriosam provident, quas aspernatur.

Reiciendis similique amet cumque ut laboriosam, dolores molestias quisquam, distinctio fugiat, dolorem preferendis ab! Minus ex, dolores reprehenderit enim nostrum!

[Read More](#)

Block 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Deleniti laboriosam provident, quas aspernatur hic illo rem excepturi consectetur ab minus?

At enim, rerum ut error. Consequuntur accusamus recusandae repellendus labore dolorum, tempore ullam ea eius impedit at porro quisquam!

[Read More](#)

Block 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit. At minima non sint. Id nostrum, dignissimos deleniti ipsam laboriosam at explicabo.

Tenetur veritatis rerum ut error consequuntur commodi omnis. Quos accusantium, dicta est illum itaque, quaerat natus quis doloremque repellendus ex, veritatis ea consequuntur dignissimos.

[Read More](#)

Рисунок 13

Еще одно замечание относительно свойств `height` и `min-height`: если вы задаете значение `height` в %, оно будет рассчитываться относительно высоты родительского блока. Однако если родителю задать `min-height` вместо `height`, то расчет работать не будет. Это и понятно с точки зрения браузера: ему нужно как-то сообщить реальную высоту блока, чтобы он мог рассчитать проценты. Проблема решается простым способом — нужно назначить родительскому элементу `height` в 1px:

```
.parent-element {  
    min-height: 500px;  
    height: 1px;  
}  
.child-element {  
    height: 100%;  
}
```

Максимальную высоту элемента можно задавать в том случае, если ширина элемента меняется в зависимости от ширины родительского элемента и размеров окна браузера. Например, в файле *max-height.html* на широком экране свойство `height: 320px` увеличивает пустое пространство после последнего абзаца.

max-height

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Veritatis molestiae, officiis alias iusto eligendi eaque illum repellat laboriosam corporis excepturi deserunt aperiam culpa repellendus eum expedita aliquam rerum ipsam.

In asperiores, nesciunt molestiae debitis iure modif Quaerat nihil magnam qui cupiditate numquam obcaecati odio. Voluptatem perspiciatis repudiandae rem, repellat, a ullam dicta sapiente deleniti, debitis eos atque repellendus facere?

Quo, voluptates, at. Dolores quaerat dignissimos vero debitis asperiores iusto incident, consequuntur iste ratione ex, temporibus voluptatem modi neque nemo aperiam sint eius nostrum quisquam itaque dolor nihil eveniet! Corporis.

Aperiam possimus, animi dicta obcaecati et ut natus non accusamus explicabo est cumque eum ex similiqe repudiandae aliquam optio, omnis qui amet veniam fuga temporal Similique, officia?

Рисунок 14

Заменим его на `max-height: 320px` и получим уменьшение высоты до соответствующей значению `auto`.

max-height

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Veritatis molestiae, officiis alias iusto eligendi eaque illum repellat laboriosam corporis excepturi deserunt aperiam culpa repellendus eum expedita aliquam rerum ipsam.

In asperiores, nesciunt molestiae debitis iure modif Quaerat nihil magnam qui cupiditate numquam obcaecati odio. Voluptatem perspiciatis repudiandae rem, repellat, a ullam dicta sapiente deleniti, debitis eos atque repellendus facere?

Quo, voluptates, at. Dolores quaerat dignissimos vero debitis asperiores iusto incident, consequuntur iste ratione ex, temporibus voluptatem modi neque nemo aperiam sint eius nostrum quisquam itaque dolor nihil eveniet! Corporis.

Aperiam possimus, animi dicta obcaecati et ut natus non accusamus explicabo est cumque eum ex similiqe repudiandae aliquam optio, omnis qui amet veniam fuga temporal Similique, officia?

Рисунок 15

При уменьшении ширины окна браузера свойство `max-height: 320px` будет ограничивать высоту блока 320 пикселями вне зависимости от того, поместится ли текст в пределы контейнера по высоте.

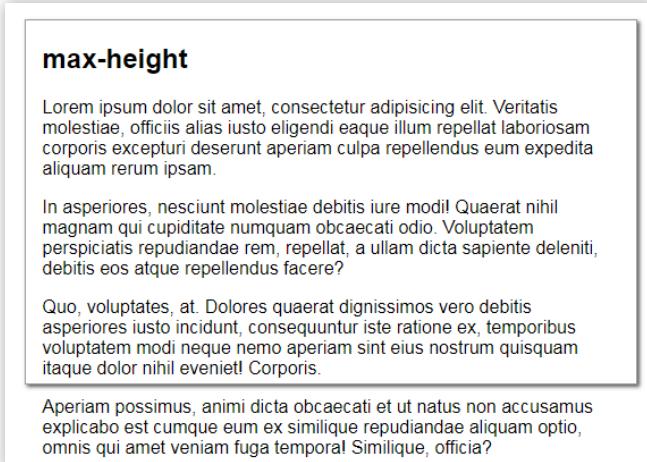


Рисунок 16

Для того чтобы текст находился в пределах блока даже при переполнении его контентом, необходимо использовать свойство `overflow`.

Полезные ссылки:

1. https://www.w3schools.com/cssref/pr_dim_height.asp
2. <https://developer.mozilla.org/en-US/docs/Web/CSS/height>
3. <https://developer.mozilla.org/en-US/docs/Web/CSS/max-height>
4. https://www.w3schools.com/cssref/pr_dim_max-height.asp
5. <https://developer.mozilla.org/en-US/docs/Web/CSS/min-height>
6. https://www.w3schools.com/cssref/pr_dim_min-height.asp

7. [https://webref.ru/css/height.](https://webref.ru/css/height)
8. [https://webref.ru/css/min-height.](https://webref.ru/css/min-height)
9. [https://webref.ru/css/max-height.](https://webref.ru/css/max-height)
10. [https://habrahabr.ru/post/189252/.](https://habrahabr.ru/post/189252/)
11. [http://html-plus.in.ua/box-model-elementa-nyuansy/#-box-height.](http://html-plus.in.ua/box-model-elementa-nyuansy/#-box-height)

Свойство overflow

Это свойство определяет, как будет вести себя контент блочного элемента, если он целиком не помещается в области, задаваемой шириной и высотой элемента. Возможные значения:

```
overflow: visible | auto | hidden | scroll | inherit
```

Значение **visible** по умолчанию показывает все содержимое элемента, даже, если оно выходит за пределы блока — собственно, это то, что мы с вами наблюдали в нескольких примерах выше.

max-height

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Veritatis molestiae, officiis alias iusto eligendi eaque illum repellat laboriosam corporis excepturi deserunt aperiam culpa repellendus eum expedita aliquam rerum ipsam.

In asperiores, nesciunt molestiae debitis iure modi! Quaerat nihil magnam qui cupiditate numquam obcaecati odio. Voluptatem perspiciatis repudiandae rem, repellat, a ullam dicta sapiente deleniti, debitis eos atque repellendus facere?

Quo, voluptates, at. Dolores quaerat dignissimos vero debitis asperiores iusto incident, consequuntur iste ratione ex, temporibus voluptatem modi neque nemo aperiam sint eius nostrum quisquam itaque dolor nihil eveniet! Corporis.

Aperiam possimus, animi dicta obcaecati et ut natus non accusamus explicabo est cumque eum ex similique repudiandae aliquam optio, omnis qui amet veniam fuga temporal! Similique, officia?

Рисунок 17

Значение `auto` добавляет полосу (или 2 полосы) прокрутки, когда они необходимы. Именно это значение используем в файле `max-height.html`, чтобы прокручивать контент в области, ограниченной свойством `max-height` (рис. 17).

Две полосы прокрутки добавятся, если ограничены будут и ширина, и высота контента, причем таким образом, что текст внутри контейнера нельзя перенести на следующую строку. Обычно таким образом оформлены блоки кода на сайтах или форумах программистов, когда перенос строки при копировании кода может привести к ошибке в программе:

overflow: auto

```
<head>
    <title>JavaScript Animation</title>

    <script>
        var imgObj = null;

        function init() {
            imgObj = document.getElementById('myImage');
            imgObj.style.position = 'relative';
            imgObj.style.left = '0px';
        }

        function moveRight() {
            imgObj.style.left = parseInt(imgObj.style.left)
        }

        window.onload = init;
    </script>

</head>
```

Рисунок 18

```
.programm-code {  
    background-color: #e4fdf4;  
    white-space: pre;  
    width: 500px;  
    height: 400px;  
    overflow: auto;  
}
```

Рисунок 19

Значение **hidden** в соответствии со своим переводом с английского языка прячет контент, который не поместился в пределы элемента:

overflow: hidden

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Doloremque animi deserunt temporibus, suscipit id reiciendis eligendi modi tempore tenetur! Sit placeat excepturi distinctio ut a et quasi, esse iste. Aut.

Perspiciatis tenetur sequi delectus voluptatibus aperiam impedit, vitae doloribus minus? Nihil aspernatur at itaque consequatur, distinctio error. Perferendis dolor voluptates

Рисунок 20

```
.hidden-content {  
    background-color: #84fc89;  
    font-size: 1.16em;  
    padding: 0 15px;  
    width: 500px;  
    height: 200px;  
    overflow: hidden;  
}
```

Рисунок 21

В нижней части блока видны частично обрезанные буквы текста, который не поместился в заданную высоту элемента.

Значение scroll добавляет две полосы прокрутки (горизонтальную и вертикальную) вне зависимости от того, нужны они или нет (рис. 22).

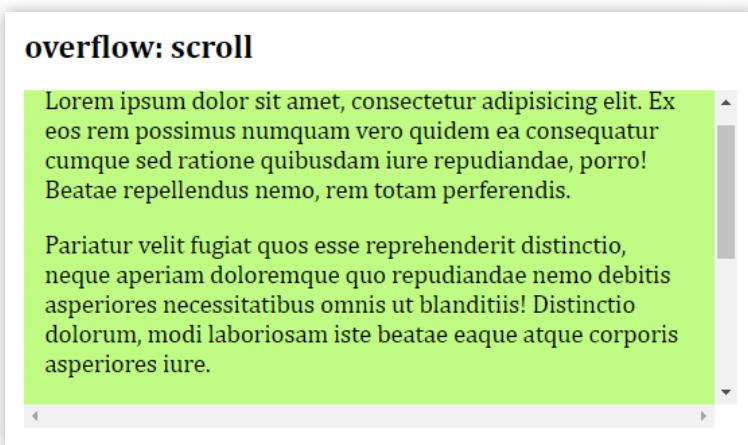


Рисунок 22

Значение inherit наследует значение родительского элемента. В примере свойство overflow: auto задано для родительского контейнера с классом parent-block вместе с желтым цветом фона. При этом ни ширина, ни высота для него не назначены, соответственно, и полос прокрутки в этом блоке вы не увидите.

Для дочерних элементов с классом inherit-property свойство overflow имеет значение inherit, т.е. фактически наследует родительское значение auto. Поэтому при указании высоты в 200px контент в этих блоках получает вертикальную полосу прокрутки, т.к. не помещается полностью внутри заданного размера (рис. 23, 24).

Все примеры указанные здесь, вы можете найти в файле *overflow.html*.

```
.parent-block {  
    background-color: #fcfc66;  
    overflow: auto;  
}  
.inherit-property {  
    float: left;  
    height: 200px;  
    width: 30%;  
    margin: 1.5%;  
    overflow: inherit;  
}
```

Рисунок 23

overflow: inherit



Рисунок 24

Свойство `overflow` имеет разновидности: `overflow-x` и `overflow-y`, которые управляют отображением контента элемента, если оно не помещается по горизонтали или вертикали, соответственно. Их значения таковы:

```
overflow-x: visible | auto | hidden | scroll  
overflow-y: visible | auto | hidden | scroll
```

Полезные ссылки:

1. <https://developer.mozilla.org/en-US/docs/Web/CSS/overflow>.
2. https://www.w3schools.com/cssref/pr_pos_overflow.asp.
3. https://www.w3schools.com/cssref/css3_pr_overflow-x.asp.

4. <https://developer.mozilla.org/en-US/docs/Web/CSS/overflow-x>.
5. <https://developer.mozilla.org/en-US/docs/Web/CSS/overflow-y>.
6. https://www.w3schools.com/cssref/css3_pr_overflow-y.asp.
7. <http://htmlbook.ru/css/overflow>.
8. <http://htmlbook.ru/css/overflow-x>.
9. <http://htmlbook.ru/css/overflow-y>.
10. <http://html-plus.in.ua/box-model-elementa-nyuansy/#overflow>.

Границы (рамки) элемента

Исходя из блочной модели, сразу после внутренних отступов в блоке могут быть назначены границы (рамки). По умолчанию блочные элементы границ не имеют.

Для того чтобы задать границы, используют 3 свойства: **border-style**, **border-width** и **border-color** или одно универсальное свойство **border**, которое включает все перечисленные.

Свойство border-style

Свойство **border-style** определяет стиль границы (рамки) элемента и имеет такие значения:

```
border-style: solid | dashed | dotted | double | ridge  
| groove | inset | outset | hidden | none | inherit
```

Большая часть значений представлена на рисунке 25. Значения **none** и **hidden** убирают рамку, отменяя или скрывая ее.

border-style

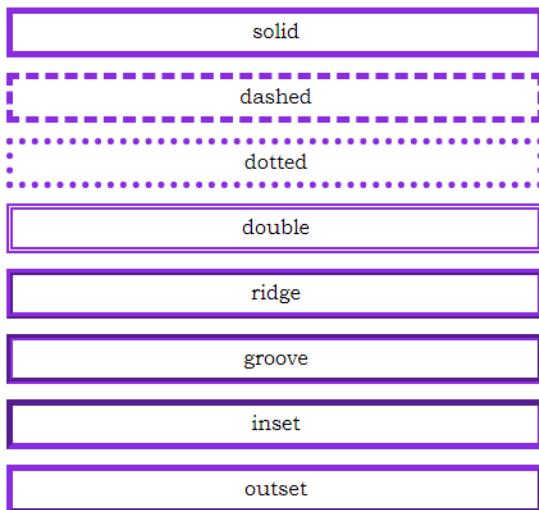


Рисунок 25

Вы можете использовать для назначения различным сторонам разных стилей рамки следующие свойства:

```
border-top-style: double;  
border-right-style: dotted;  
border-bottom-style: solid  
border-left-style: ridge;
```

Также для свойства `border-style` можно назначить различное количество стилей: одно — для всех сторон, 2 — для горизонтальных и вертикальных сторон, 3 — для верхней, левой и правой, нижней сторон, 4 — поочередно для верхней, правой, нижней и левой сторон.

```
border-style: double;  
border-style: double solid;
```

```
border-style: dashed solid dotted;  
border-style: solid ridge double none;
```

1-4 values

```
border-style: double;
```

```
border-style: double solid;
```

```
border-style: dashed solid dotted;
```

```
border-style: solid ridge double none;
```

Рисунок 26

Аббревиатура Emmet:

```
bds, bds:s, bds:dt, bds:ds, bds:d, bds:r, bds:o,  
bds:i, bds:n, bds:h, bdrs, bdls, bdts, bdbs
```

Свойство border-width

Свойство border-width определяет размер, а именно толщину границы (рамки) элемента и имеет такие значения:

```
border-width: thin | medium | thick | значения в px,  
pt, em, mm, cm, in, но не в % | 0 | inherit
```

Как правило, размер рамки указывается в px, но вы можете использовать любые единицы, кроме процентов, или ключевые слова **thin** — тонкая (1px), **medium** — средняя (3px), **thick** — толстая (5px). Нулевое значение уберет границу, значение **inherit** позволит повторить толщину границы родительского элемента.

border-width

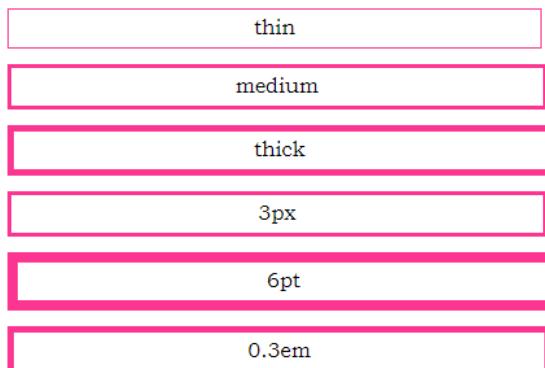


Рисунок 27

Можно задавать толщину границы с помощью отдельных свойств:

```
border-left-width: 5px;  
border-right-width: thick;  
border-top-width: 1em;  
border-bottom-width: 2pt;
```

Ключевые слова `left`, `right`, `top` и `bottom` указывают, что толщина рамки назначается для левой, правой, верхней или нижней стороны элемента.

Так же, как и в свойствах `padding` или `margin`, `border-width` можно задавать в виде одной, 2-х, 3-х или 4-х цифр. Одна цифра или ключевое слово задает толщину границы для всех сторон элемента, 2 цифры — для верхней и нижней, а затем для левой и правой сторон. 3 цифры описывают рамку для верхней, одновременно левой и правой, и нижней границы, 4 — последовательно для верхней, правой, нижней и левой стороны.

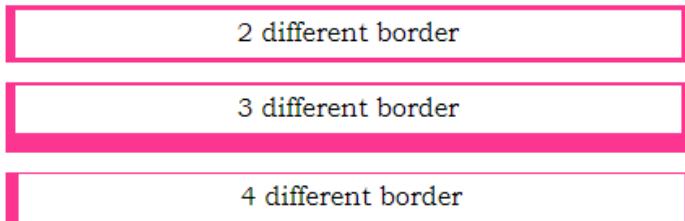


Рисунок 28

```
border-width: 3px 6px;  
border-width: 2px 6px 12px;  
border-width: 1px 4px 2px 8px;
```

Аббревиатура Emmet:

```
bdw, bdw5, bdrw, bdlw, bdtw, bdbw
```

Свойство border-color

Свойство **border-color** определяет цвет границы элемента и назначается так же, как и другие свойства, управляющие цветом:

```
border-color: red | #ff0000 | #f00 | rgb(255, 0, 0) |  
rgba(255, 0, 0, .8) | hsl(0, 100%, 50%) | transparent
```

Со значением **transparent** вы получаете прозрачную рамку, которая визуально не видна, но, тем не менее, существует и может быть использована для анимации или для создания треугольников.

Как и в случае с толщиной рамки, цвет можно задать одной, двумя, тремя или четырьмя цифрами для сторон в той же последовательности.

border-color

One color (border-color: crimson;)

Two colors (border-color: #f00 #ff0;)

Three colors (blueviolet #f7079b rgb(0, 235, 255);)

Four colors (border-color: blue rgb(0, 235, 255) hsl(95, 91%, 26%) #f40;)

Рисунок 29

Аббревиатура Emmet:

bdc, bdc:t, bdrc, bdrc, bdtc, bdrc

Составное (универсальное) свойство border

Это свойство состоит из 3-х перечисленных выше, указанных через пробел в любой последовательности:

border: border-width border-style border-color;

Например,

border: 2px solid #ccc;

В этом случае граница будет создана вокруг всех сторон элемента.

Можно добавить границу только для одной стороны, используя ключевые слова top, right, left, bottom:

```
border-top: 3pt solid #ccc;  
border-right: .3em dotted #f00;  
border-bottom: 7px dashed rgb(0, 235, 255);  
border-left: thick outset crimson;
```

border

```
border-top: 3pt solid #ccc;
```

```
border-right: .3em dotted #f00;
```

```
border-bottom: 7px dashed rgb(0, 235, 255);
```

```
border-left: thick outset crimson;
```

Рисунок 30

Можно использовать 2 свойства: сначала общее для всех сторон, а потом переопределить вид рамки или какого-то одного из свойств для одной стороны:

- изменение цвета для нижней границы:

```
border: 2px solid #ddd;  
border-bottom-color: #666;
```

- изменение всех свойств левой границы:

```
border: 1px solid #7ef88;  
border-left: 6px double #1b9f26;
```

```
border: 2px solid #ddd;  
border-bottom-color: #666;
```

```
border: 1px solid #7ef88;  
border-left: 6px double #1b9f26;
```

Рисунок 31

Аббревиатура Emmet:

```
bd, bd10, bdr, bdl, bdt, bdb
```

Полезные ссылки:

1. https://www.w3schools.com/cssref/pr_border.asp.
2. <https://developer.mozilla.org/en-US/docs/Web/CSS/border>.
3. <http://htmlbook.ru/css/border>.
4. <http://htmlbook.ru/css/border-style>.
5. <http://htmlbook.ru/css/border-width>.
6. <http://htmlbook.ru/css/border-color>.
7. <http://html-plus.in.ua/box-model-elementa-nyuansy/#border>.

Свойство border-radius

В CSS3 стало возможным закруглять углы блочного элемента. Для этого используется свойство border-radius:

```
border: 2px solid #09c77e;  
border-radius: 10px;
```

Bordered content

border-radius: 10px

 Lorem ipsum dolor sit amet, consectetur adipisicing elit.
 Fugit, quaerat.

 Porro, adipisci? Non aliquam deserunt ipsam adipisci,
 praesentium accusamus inventore.

 Beatae a quas quo rem, necessitatibus, error nostrum
 voluptatem sint.

Рисунок 32

В этом случае закруглены будут все углы.

Интересно, что это свойство можно использовать без назначения свойства border. В этом случае закругление будет видно благодаря цвету фона элемента.

```
background-color: #81f89d;  
border-radius: 10px;
```

Some content

border-radius: 10px

Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Fugit, quaerat.

Porro, adipisci? Non aliquam deserunt ipsam adipisci,
praesentium accusamus inventore.

Beatae a quas quo rem, necessitatibus, error nostrum
voluptatem sint.

Рисунок 33

Указывать скругление можно в любых единицах (px, pt, em, cm, mm и др.), в том числе и в процентах. Проценты по горизонтальной оси вычисляются относительно ширины элемента, а проценты по вертикальной оси — относительно его высоты. Отрицательные значения не допускаются. По умолчанию значение **border-radius** равно нулю, т.е. скругления нет — элемент имеет прямые углы.

Percent value

border-radius: 10%

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Excepturi,
cupiditate, reiciendis. Esse, ea. Quibusdam, accusantium!

Minima alias voluptate voluptatibus perspiciatis voluptatem sequi
itaque nihil dolorum quam amet, reiciendis harum culpa.

Obcaecati ipsum ratione tenetur commodi, accusamus tempora illum
inventore ipsam laboriosam, quod deleniti repellendus temporibus!

Рисунок 34

Очень удобно использовать `border-radius` со значением 50%, для того чтобы получить круг. Однако, важно понимать, что ширина и высота элемента должны быть одинаковыми:

```
.bdr-percent {  
    width: 200px;  
    height: 200px;  
    border-radius: 50%;  
    text-align: center;  
}
```

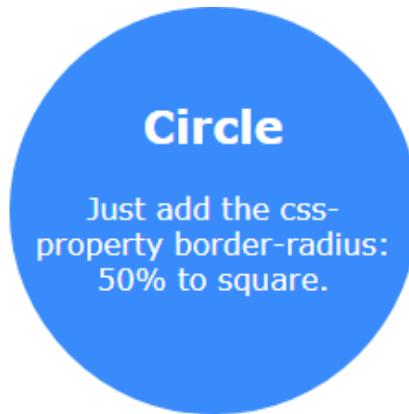


Рисунок 35

Как правило, в круге нужно размещать текст по центру, чтобы он хорошо выглядел. Сдвинуть в нем контент несколько вниз от верхней границы можно либо с помощью свойства `padding-top` для этого элемента, либо с помощью `margin-top` для какого-либо дочернего элемента (в нашем случае заголовка второго уровня):

```
.bdr-percent h2 {margin-top: 40px;}
```

При желании вы можете добавить свойства `border` или `box-shadow` к вашему кругу, как в файле `border-radius.html`.

Аббревиатура Emmet:

```
bdrs, bdtrs10, bdtrs:5, bdtrs:50%
```

Как и в других свойствах, можно назначить различные закругления для разных углов блока, используя 2, 3 или 4 цифры. Два значения определяют закругления углов в такой последовательности: первое значение — для радиуса верхнего левого и нижнего правого углов, второе значение — для верхнего правого и нижнего левого углов. На скриншоте это выглядит понятнее.

```
border-radius: 20px 5px;
```

2 digits

border-radius: 20px 5px

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt reiciendis veniam ea possimus fuga, ipsam.

Voluptatum nam hic, quibusdam architecto facere fuga suscipit! Soluta corrupti asperiores, nesciunt in pariatur totam.

Рисунок 36

Три значения оформляют закругления в такой последовательности: сначала для верхнего левого угла, затем сразу для верхнего правого и нижнего левого углов, и последнее значение — для нижнего правого угла.

```
border-radius: 10px 5px 35px;
```

3 digits

border-radius: 10px 5px 35px;

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Incidunt
 reiciendis veniam ea possimus fuga, ipsam.

 Voluptatum nam hic, quibusdam architecto facere fuga suscipit! Soluta
 corrupti asperiores, nesciunt in pariatur totam.

Рисунок 37

Четыре значения позволяют закруглить углы, начиная с верхнего левого угла по часовой стрелке.

4 digits

border-radius: 3em 7px 35px 0;

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Excepturi,
 cupiditate, reiciendis. Esse, ea. Quibusdam, accusantium!

 Minima alias voluptate voluptatibus perspiciatis voluptatem sequi
 itaque nihil dolorum quam amet, reiciendis harum culpa.

4

2

3

Рисунок 38

border-radius: 3em 7px 35px 0;

Как и для других свойств box-модели, border-radius можно назначить для каждого угла в отдельности:

```
border-top-left-radius: 10px;  
border-top-right-radius: 20px;  
border-bottom-right-radius: 10pt;  
border-bottom-left-radius: 20pt;
```

The radius of one corner

`border-top-left-radius: 10px;`

`border-top-right-radius: 20px;`

`border-bottom-right-radius: 10pt;`

`border-bottom-left-radius: 20pt;`

Рисунок 39

Аббревиатуры Emmet:

`btlr, btrr, bbrr, bblr`

Существует еще одна возможность задавать радиус скругления угла — на основе эллипса. В этом случае для любого из вариантов свойства `border-radius` значения пишутся через слэш, где первое значение задает радиус по горизонтали, а второе — по вертикали:

`border-radius: 30px/15px;`

Ellipse corners

`Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium, porro.`

`Dicta nesciunt nam cumque dolorum aspernatur eligendi, consequatur, repellat corporis!`

Рисунок 40

Насколько это вам понадобится, сложно сказать, но использовать вы это можете.

Полезные ссылки

1. <https://developer.mozilla.org/en-US/docs/Web/CSS/border-radius>.
2. https://www.w3schools.com/cssref/css3_pr_border-radius.asp.
3. https://www.w3schools.com/css/css3_borders.asp.
4. <http://htmlbook.ru/css/border-radius>.

Использование свойств блочной модели



Рисунок 41

Сейчас мы рассмотрим один из примеров, в котором свойства box-модели помогут нам сверстать ... модель телефона. В дальнейшем эту верстку можно использовать для создания CSS-анимаций или изменения фоновых картинок с помощью JavaScript — языка сценариев, который позволяет создать интерактивность на веб-страницах. За основу возьмем изображение телефона. Все нюансы изображения мы пока передать не сможем, но нам нужна упрощенная модель.

Посмотреть файл примера можно в файле *box-model-phone.html* в папке examples.

Для создания html-разметки необходимо проанализировать внешний вид телефона. Он состоит из 6 блоков, один из которых является внешним, или родительским, остальные 5 — внутренними, или дочерними.

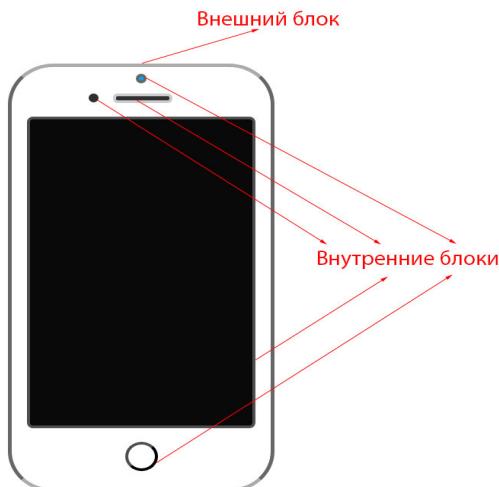


Рисунок 42

Поэтому начальный код в элементе `<body>` будет таким:

```
<div class="phone">
    <div class="camera"></div>
    <div class="circle"></div>
    <div class="dinamic"></div>
    <div class="inner-block"></div>
    <div class="control"></div>
</div>
```

Форматирование элементов начнем с основного. Для него необходимо назначить размеры (ширину `width: 300px` и высоту — `height: 600px`), фоновый цвет (`background-color: black`) и тень (`box-shadow: 0 0 10px 5px black`).

background-color: #fff), рамку (border: 5px solid), внутренние отступы для того, чтобы отодвинуть вложенные элементы (padding: 10px 20px), а также внешние отступы, чтобы разместить телефон по центру браузера с небольшим смещением сверху и снизу (margin: 20px auto).

```
.phone {  
    width: 300px;  
    height: 600px;  
    background-color: #fff;  
    border: 5px solid;  
    border-color: #aaa #676767;  
    border-radius: 60px;  
    padding: 10px 20px;  
    margin: 20px auto;  
}
```



Рисунок 43

Чтобы телефон казался объемней, цвет рамки зададим двумя цветами — border-color: #aaa #676767. Скругления для блока назначим с помощью свойства border-radius: 60px.

В результате получим пустой внешний блок, т.к. все внутренние элементы не имеют содержимого, т.е. текста.

Далее нужно отформатировать элемент с классом *camera*.

Поскольку камера — это небольшой круглый объект с рамкой и голубым фоном, в css-правилах используем такие свойства, как *width*, *height*, *background-color*, *border* и *border-radius*:

```
.camera {  
    width: 6px;  
    height: 6px;  
    border-radius: 50%;  
    border: 4px solid #666;  
    background-color: #4fa3eb;  
    margin: auto;  
    margin-bottom: 15px;  
}
```

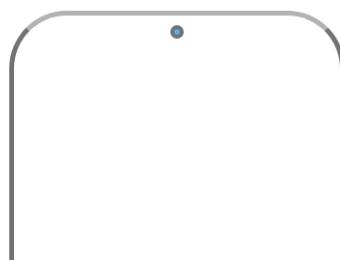


Рисунок 44

Для того чтобы разместить этот элемент по центру, используем свойство *margin: auto*, а для смещения нижних элементов добавим свойство *margin-bottom: 15px*.

Далее нам нужно отформатировать элементы с классами *circle* и *dinamic*. Для элемента *.circle* css-свойства будут

похожи на те, что задавались для `.camera`, т.е. опять нужно задать размеры, фоновый цвет и `border-radius: 50%`.

```
.circle {  
    width: 13px;  
    height: 13px;  
    background-color: #333;  
    border-radius: 50%;  
}
```



Рисунок 45

Для элемента с классом `dinamic` опять-таки используем похожие свойства, но необходимо увеличить ширину по сравнению со свойствами для `.circle`, а `border-radius` уменьшить до `4px`.

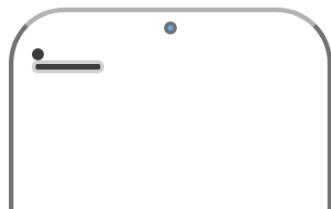


Рисунок 46

```
.dinamic {  
    width: 70px;  
    height: 6px;
```

```

border: 4px solid #ccc;
border-radius: 6px;
background-color: #333;
}

```

Изменения выглядят не совсем так, как хотелось бы. Во-первых, поскольку элементы являются блочными, они разместились друг под другом, а их нужно разместить рядом. Во-вторых, необходимо их выровнять по центру.

Для размещения элементов рядом, используем свойство `float`, речь о котором пойдет ниже. Для `.circle` зададим значение `left`, а для `.dinamic` — значение `right`:

```

.circle { float: left; }
.dinamic { float: right; }

```

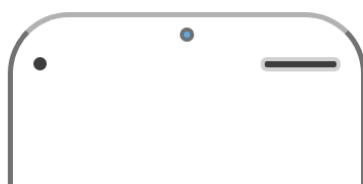


Рисунок 47

За счет использования свойства `float` элементы «разбежались» в разные стороны. Для того чтобы их сдвинуть в центр, необходимо вложить их в один элемент, и назначить для него `width + margin: auto`:

```

<div class="top">
  <div class="circle"></div>
  <div class="dinamic"></div>
</div>

```

```
.top {  
    width: 110px;  
    margin: auto;  
}
```

Размер *width* определялся, исходя из размеров элементов: *13px (width.circle)+8px (2 border.circle)+70px (width.dinamic)*+15-20px на отступ между элементами получается размер в 106 — 111px.

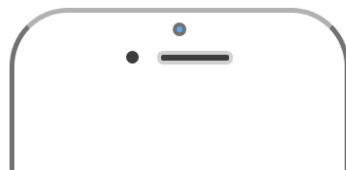


Рисунок 48

Теперь нужно подкорректировать размещение элемента *.dinamic* относительно верхнего элемента *.camera* — они должны быть центрированы друг относительно друга. Поэтому нужно несколько подкорректировать значения свойства *margin*:

```
.top {  
    width: 110px;  
    margin: 10px auto 20px 27%;  
}
```



Рисунок 49

В данном случае значения отступов будут 10px сверху и 20px снизу, слева — 27%, а справа — значение auto позволит браузеру рассчитать значение отступа самостоятельно. Размер отступа в 27% подбирался с помощью Инструментов разработчика.

Теперь необходимо написать css-свойства для центрального внутреннего блока. Правила будут похожи на предыдущие, т.к. нам опять понадобится описать рамку и закругления углов, а также добавить фоновый цвет. Что касается размеров, достаточно будет указать только высоту, т.к. блочный элемент `<div>` с классом *inner-block* займет все доступное пространство по ширине, а отступы мы уже назначили в родительском элементе с классом *phone*.

```
.inner-block {
    border: 4px solid #505050;
    border-radius: 8px;
    background-color: #111;
    height: 75%;
}
```

На скриншоте можно увидеть, что внешний вид не радует глаз, т.к. темный внутренний блок находится под блоком с классом *top*, несмотря на то, что мы задали для *.top* отступ снизу в 20px. Странная ситуация, т.к. с точки зрения назначенных css-свойств блоки должны отделяться друг от друга.

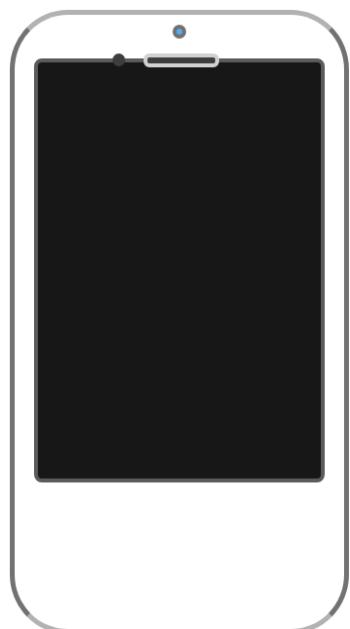


Рисунок 50

С точки же зрения обработки этих свойств браузером ничего странного нет, т.к. внутри элемента с классом *top* мы расположили 2 плавающих элемента с назначенными для них свойством *float*, о котором речь пойдет чуть ниже, и получили так называемое «схлопывание» высоты в родительском элементе. В Инструментах разработчика, которых мы подробно рассмотрим в следующем разделе, видно, что размеры элемента с классом *top* составляют 110x0px, т.е. его высота равна 0, и margin-bottom в 20px недостаточно, чтобы отодвинуть элемент с классом *inner-block* (рис. 51).

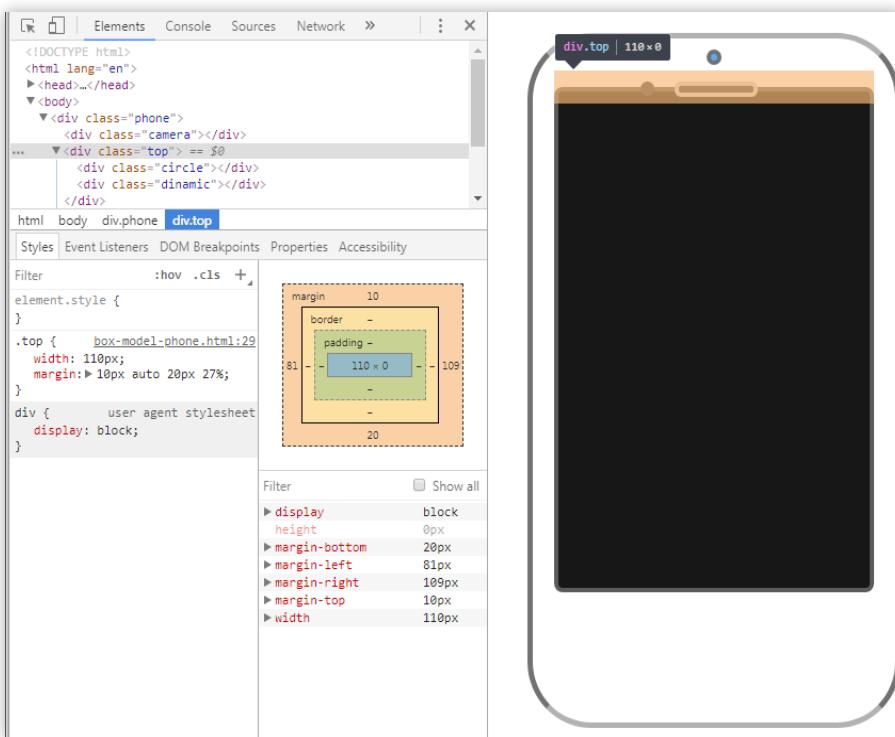


Рисунок 51

Для того чтобы отменить это неприятное для нас «схлопывание», воспользуемся свойством `overflow` со значением `hidden`, как самым простым в данном случае. Правила для `.top` будут теперь выглядеть так:

```
.top {  
    width: 110px;  
    margin: 10px auto 20px 27%;  
    overflow: hidden;  
}
```

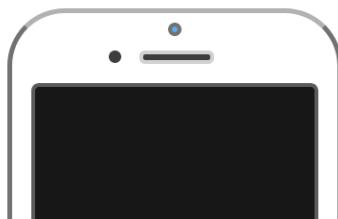


Рисунок 52

Теперь внутренний блок сместился и занимает положенное ему место.

Последний элемент, который нужно отформатировать — это `<div>` с классом `control`. Поскольку это тоже круглый элемент с рамкой, CSS-свойства будут похожи на те, что мы использовали для предыдущих классов:

```
.control {  
    width: 35px;  
    height: 35px;  
    border-radius: 50%;  
    border: 4px outset #505050;  
    margin: 25px auto 0;  
}
```

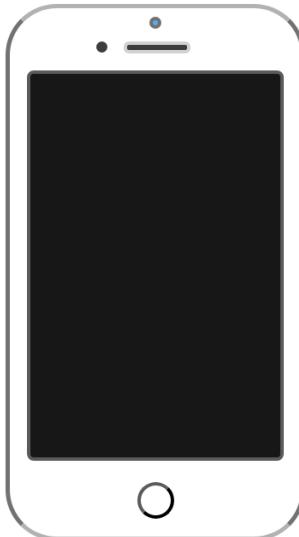


Рисунок 53

Размеры элемента `.control` больше, чем у `.circle` и `.camera`. И для стиля рамки используем значение `outset`, чтобы получить простейший эффект объемности.

Инструменты разработчика, или Инспектор свойств

В процессе верстки страницы или отдельных её блоков у вас наверняка будут появляться сложности, связанные с тем, что какие-то css-свойства, которые вы используете, применяются не так, как вы ожидали или не применяются вовсе. Именно это мы получили при форматировании элементов, имеющих свойство `float`, в примере с телефоном (файл `box-model-phone.html`).

Посмотреть, каковы значения различных свойств элементов и не только, можно, используя Инструменты разработчика, или Инспектор свойств, который на данный момент существует в каждом популярном браузере.

ре, и даже вызывается одинаковым сочетанием клавиш: F12 или Ctrl + Shift + I (в Internet Explorer/Microsoft Edge только F12). Также вы можете открыть Инспектор свойств, выбрав из контекстного меню пункт «Просмотреть код элемента», «Просмотреть код» или «Исследовать элемент» по правому клику на любом элементе на html-странице. Также Инструменты разработчика доступны в меню любого браузера.

Вид Инструментов разработчика в браузере Opera:

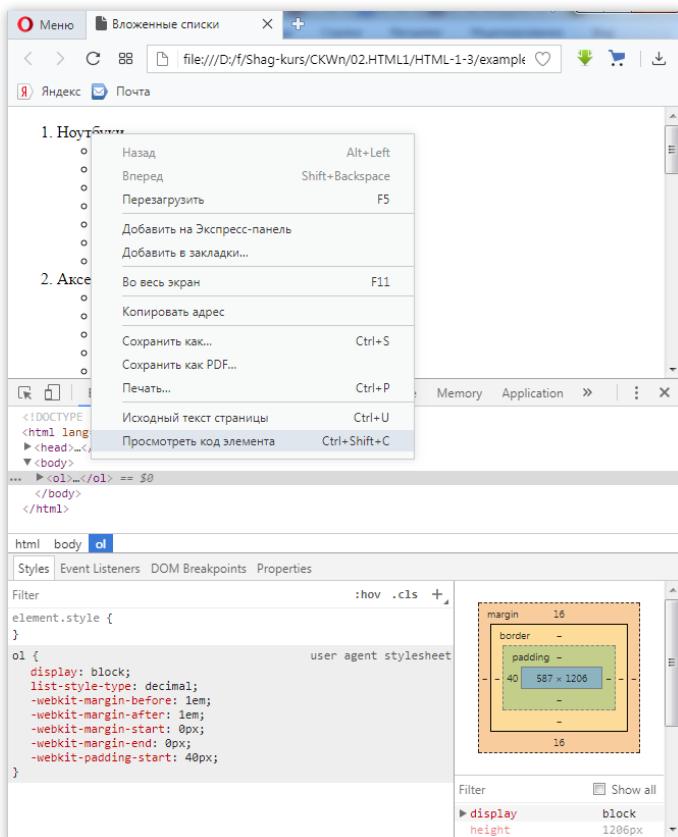


Рисунок 54

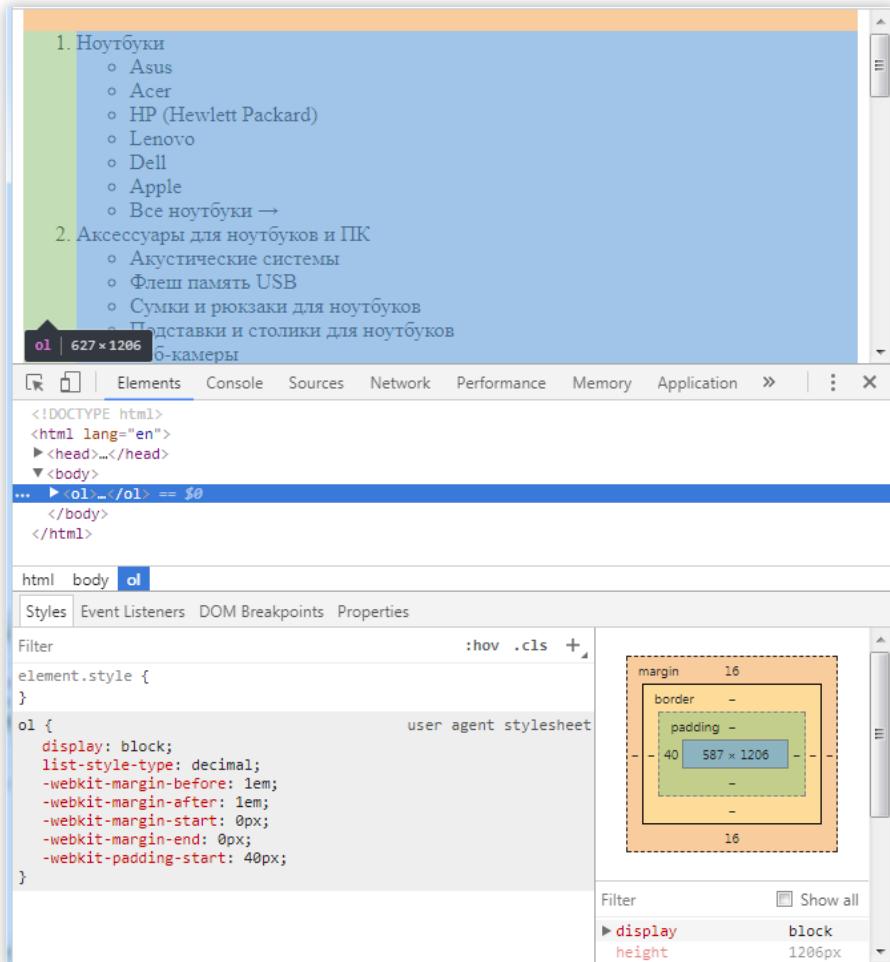


Рисунок 55

Урок 6.1. Блочная модель элементов

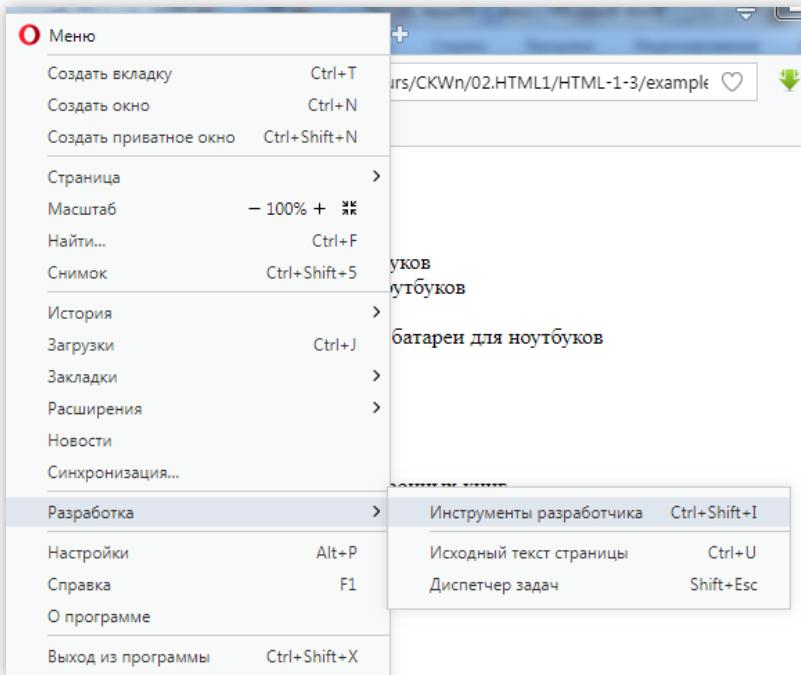


Рисунок 56

Вид Инспектора свойств в браузере Chrome:

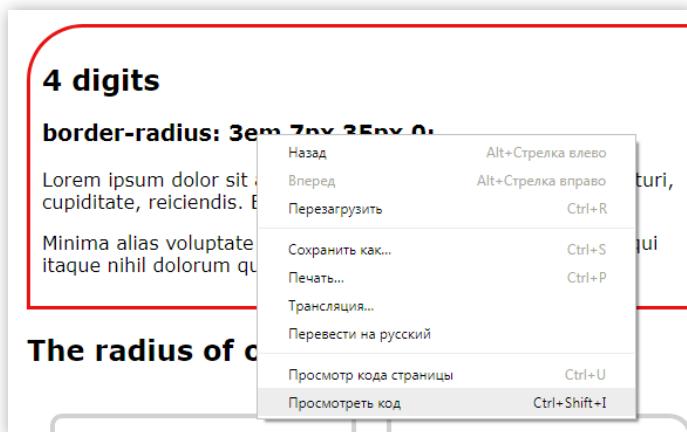


Рисунок 57

Использование свойств блочной модели

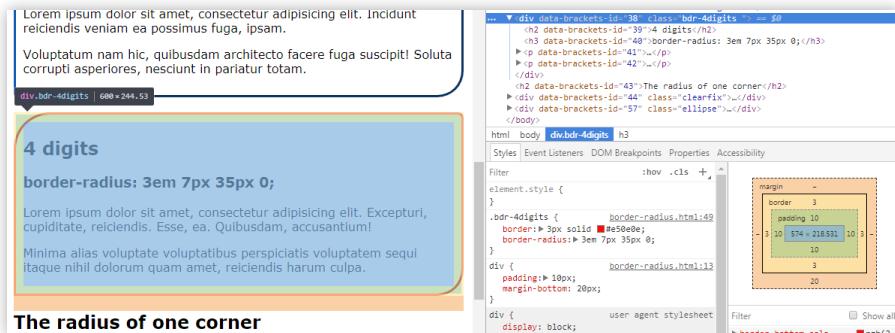


Рисунок 58

Вид Инспектора свойств в браузере Mozilla Firefox

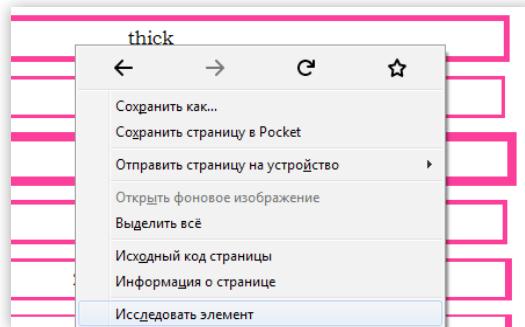


Рисунок 59

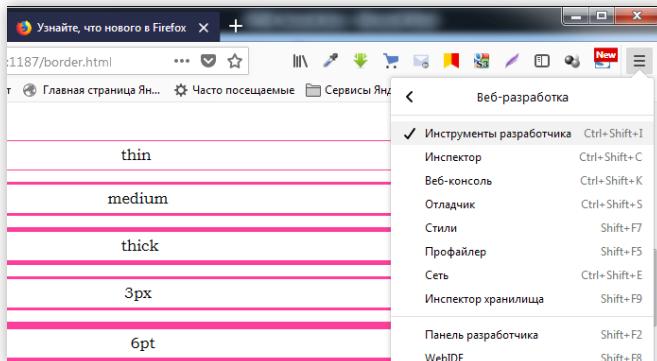


Рисунок 60

Урок 6.1. Блочная модель элементов

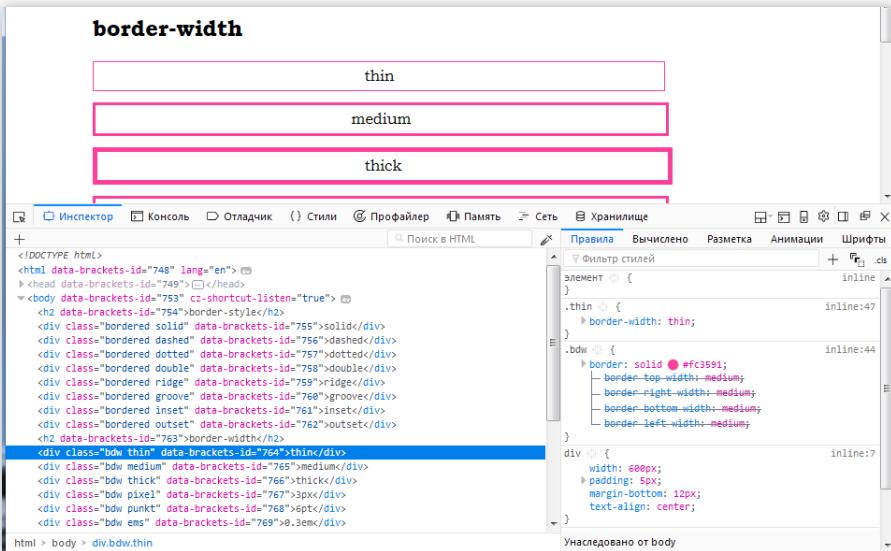


Рисунок 61

Вид Инспектора свойств в браузере Microsoft Edge.

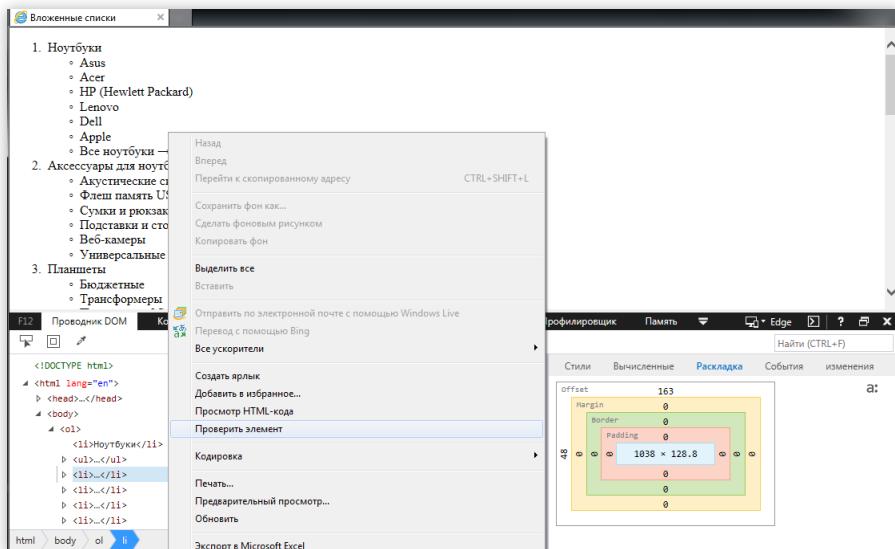


Рисунок 62

В зависимости от браузера и от сделанных вами настроек, Инструменты разработчика могут находиться справа, слева, внизу или в отдельном окне по отношению к основному контенту вашей страницы. Вы можете настроить это самостоятельно, кликнув на кнопке с тремя точками в верхнем левом углу в Chrome:

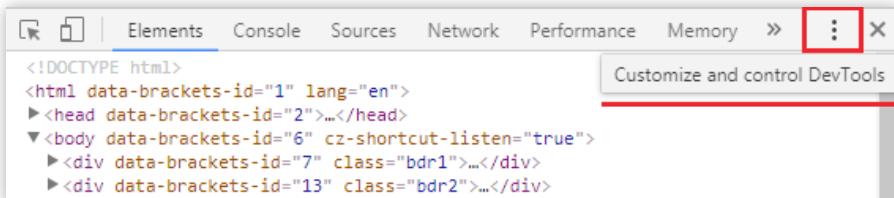


Рисунок 63

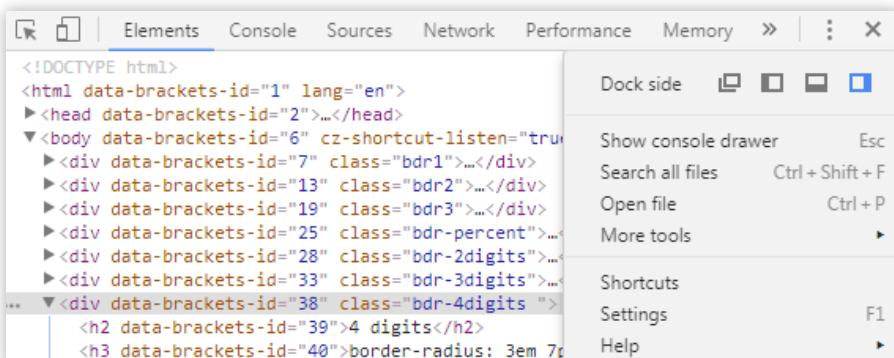


Рисунок 64

... или щелчком на соответствующей кнопке в Mozilla Firefox:

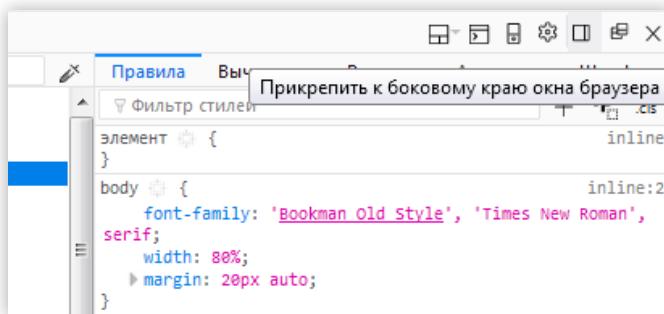


Рисунок 65

Слева в Инспекторе свойств вы увидите структуру html-документа, которую создавали в редакторе кода, а справа или внизу — все стилевые правила, записанные для данной страницы.

Рисунок 66

Когда вы выбираете какой-либо конкретный элемент на странице, то видите не только CSS-свойства, связанные с селектором этого элемента, назначенного ему класса или id, но и подсветку основных свойств его блочной модели в виде разных цветов для отступов. В браузере Chrome цвета подсветки таковы: оранжево-розовая для margin и зеленая для padding, рамка (border) оформлена бежевым, а размеры содержимого элемента (width и height) — голубым.

В других браузерах цвета могут отличаться, но суть подсветки остаётся той же. Благодаря этому цветному выделению можно наглядно посмотреть, сколько пространства занимает тот или иной элемент, и какие свойства можно изменить, чтобы его положение на странице визуально изменилось.

Для того чтобы выделить любой элемент на странице, в Инспекторе свойств предусмотрена стрелка в левом верхнем углу. Нужно сначала щелкнуть на ней, а затем сделать клик по нужному элементу на странице для отображения его свойств. Вызов стрелки также выполняется с помощью сочетания клавиш **Ctrl + Shift + C**.

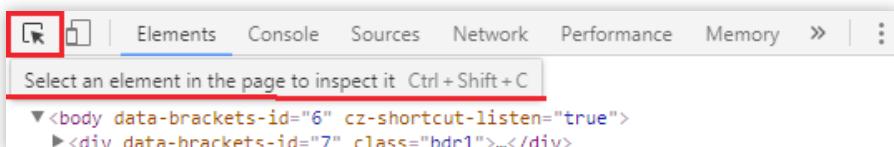


Рисунок 67

Когда вы видите стили элемента, вы можете их редактировать непосредственно в Инструментах разработчика. Кликните на значении нужного свойства и введите

любое другое. Например, можно изменить padding на margin. Кстати, браузер будет подсказывать вам при выборе свойств (рис. 68, 69).



Рисунок 68

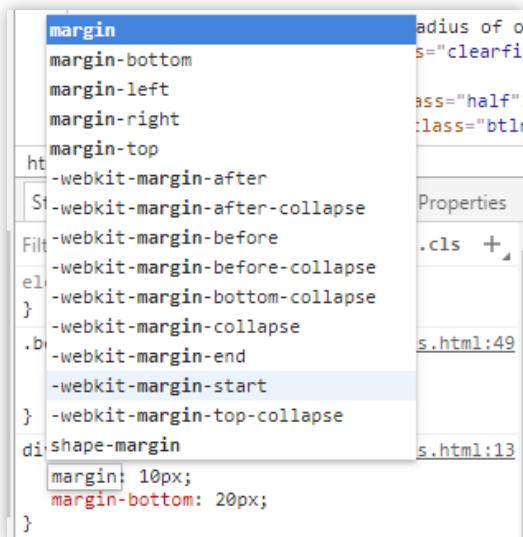


Рисунок 69

Чаще всего вам придется пользоваться стрелками вверх и вниз для изменения цифровых значений. Для этого нужно поставить курсор в поле для ввода значений и уменьшать цифру нажатием стрелки вниз или увеличивать нажатием стрелки вверх. Визуальное изменение внешнего вида блока происходит без всякого сохранения сразу у вас на глазах. Это очень удобно, особенно, если вы должны подобрать значение для того, чтобы вид блока

на странице совпадал с его видом в psd-макете. Дополнительные клавиши позволяют увеличивать или уменьшать значения свойств на 10 единиц (Shift), на 100 единиц (Ctrl) или на 1/10 единицы (Alt). Кстати, стрелки можно заменить прокручиванием колесика мыши. Проверьте, чтобы значение свойства было выделено, иначе будете изменять масштаб отображения страницы в браузере (рис. 70).

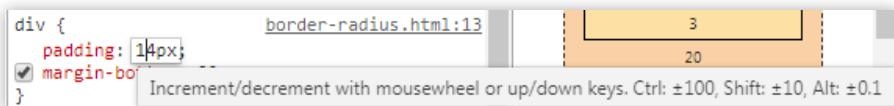


Рисунок 70

Вы также можете добавить новое свойство, щелкнув внутри фигурных скобок или отключить существующее, сняв флажок перед свойством. При добавлении браузер будет предлагать вам варианты свойств, начинающихся с введённых вами символов, а затем и возможные значения этих свойств. Пользуйтесь этими подсказками, это ускоряет написание кода и предотвращает ошибки (рис. 71).

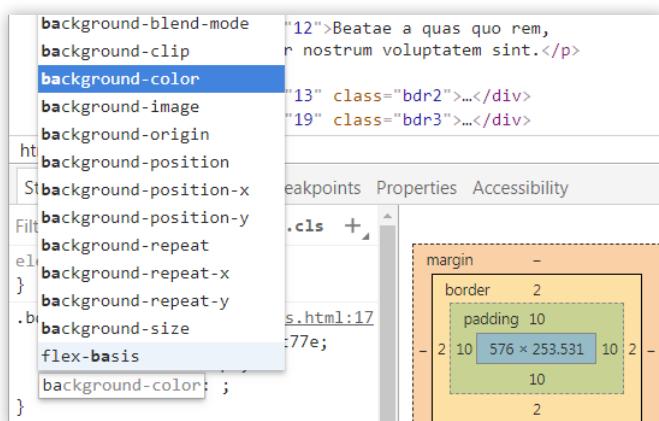


Рисунок 71

Если ошибку вы всё-таки сделали, Инспектор свойств тут же сообщит вам об этом, подсветив ошибочное свойство треугольником с восклицательным знаком и перечеркнув само свойство и значение.

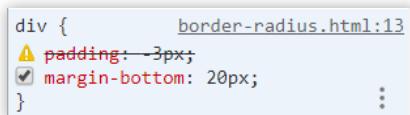


Рисунок 72

Если нужно изменить цвет текста, фона или рамки, вы также можете задействовать Инструменты разработчика: Shift + клик на цвете отобразит цвет в другом формате (rgb, hsl, hex), а обычный клик на цветном прямоугольнике (круге в Firefox) выведет цветовую палитру, в которой вы можете выбрать другой цвет или оттенок (рис. 73, 74).



Рисунок 73

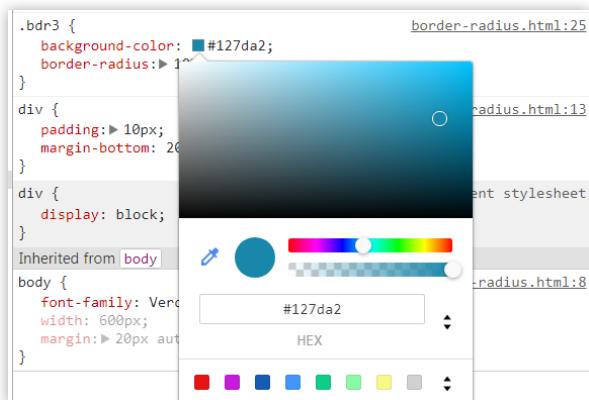


Рисунок 74

Еще одним важным помощником html-кодера является пункт **Computed** (Вычислено), который может располагаться рядом со стилями элемента или быть отдельной вкладкой. Здесь вы наглядно видите структуру блочной модели элемента со всеми размерами, а ниже — отдельные свойства и их значения.

Здесь можно увидеть значения даже тех свойств, которые не указывались явно для данного элемента, а были унаследованы им от родительских элементов. Например, это семейство и размер шрифта (`font-family` и `font-size`) или вычисленные размеры блока (`width` и `height`).

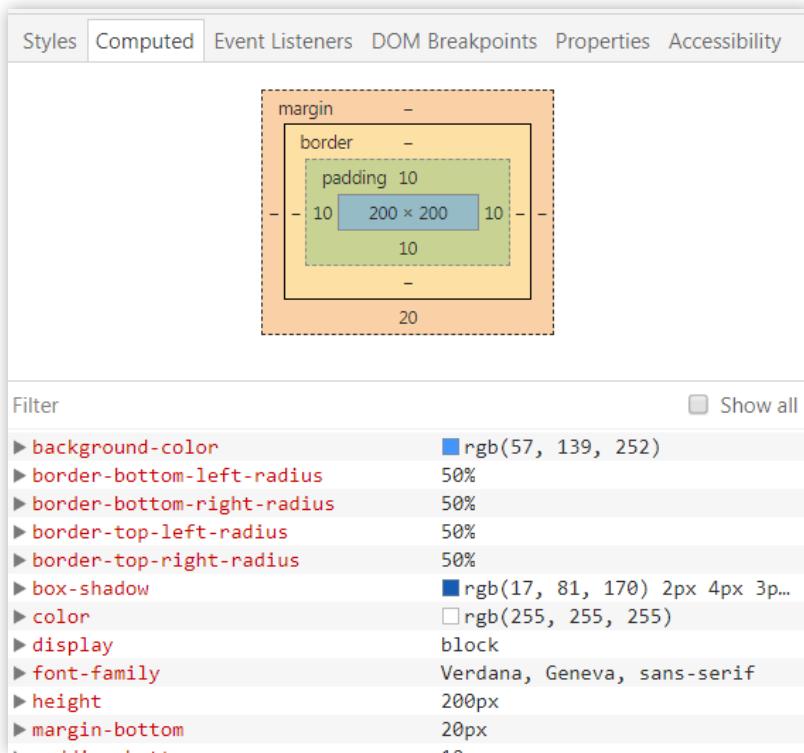


Рисунок 75

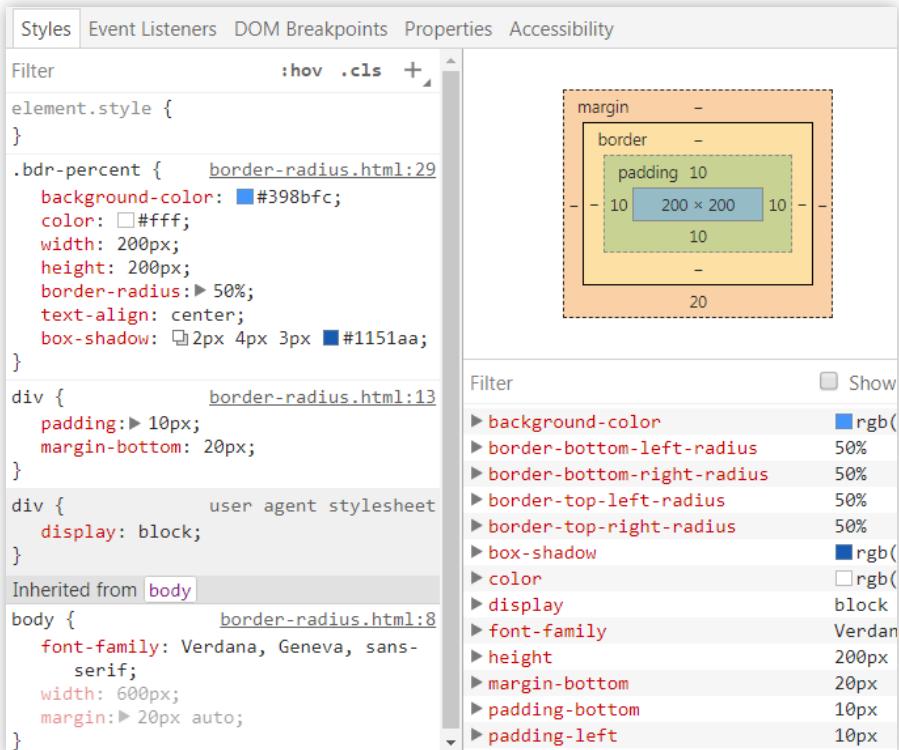


Рисунок 76

Используйте Инструменты разработчика при выполнении домашних заданий и на занятиях в аудитории — это очень удобный и полезный вариант создания и редактирования кода страницы.

Плавающие элементы. Свойство float

Хотелось бы верить, что вы не только читаете этот урок, но и смотрите файлы примеров, например *height.html*, *height-2.html*, *min-height.html*, *overflow.html*. И возможно, уже задались вопросом, что это за свойство *float*, которое периодически попадается в CSS-свойствах неко-

торых классов и уже доставило нам некоторые проблемы при форматировании телефона.

Свойство `float` определяет, с какой стороны будет расположен элемент (слева или справа) от обтекающего его текста.

Если вы активно используете текстовый редактор Microsoft Word, то возможно, применяли обтекание к картинкам, размещая их сбоку от текста. Если же вы когда-либо верстали буклеты или что-либо еще в Adobe InDesign, то наверняка пользовались этим приемом, причем не только для изображений, но и для блоков с текстом.

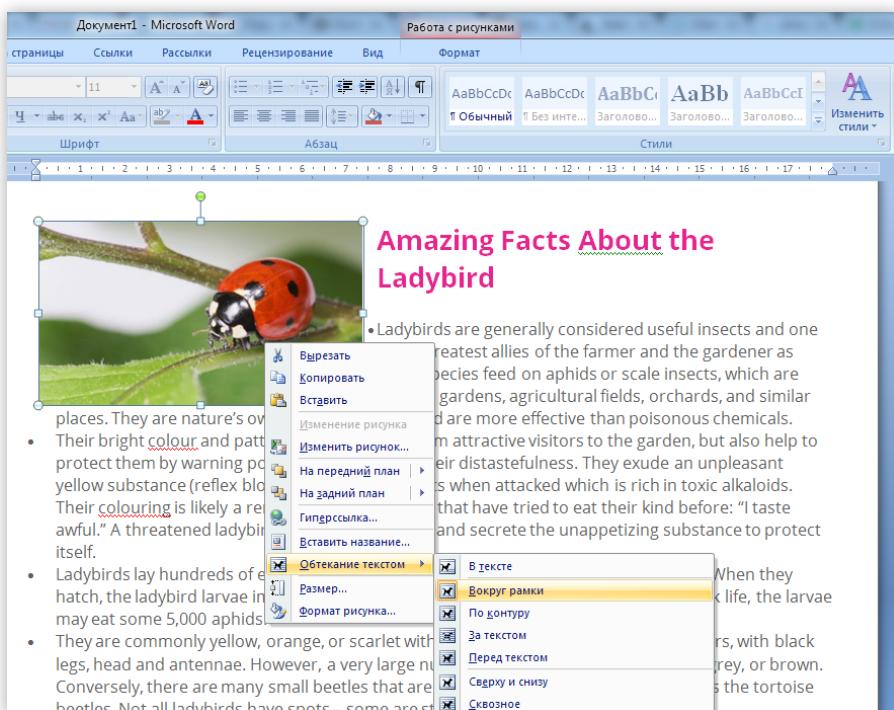


Рисунок 77. Обтекание картинки в Microsoft Word



Рисунок 78. Обтекание картинки в Adobe InDesign

В HTML такие блоки называются плавающими. С точки зрения CSS свойство **float** может принимать такие значения:

```
float: left | right | none | inherit
```

Значение **left** выравнивает элемент слева от обтекающего его текста, **right** выравнивает элемент справа от текста, **none** — обтекание не применяется к элементу (является значением этого свойства по умолчанию), **inherit** — значение наследуется от родительского элемента.

Если свойство применяется к элементу с небольшим количеством текста, то блочный элемент автоматически уменьшается по ширине. В данном примере использован такой код:

```
.letters {  
    float: left;  
    border: 2px dotted #8d89ff;  
    font-size: 2.5em;  
    font-weight: bold;  
    padding: 10px;
```

```
margin-right: 10px;  
margin-bottom: 10px;  
}
```

Lorem
sit amet, consectetur adipisicing elit. Dolore obcaecati a tempore doloribus inventore, suscipit fugiat aut sit, iusto iste, consectetur error voluptatibus dolor nemo nihil culpa rerum similique nam.
Fuga a repellendus qui minima deleniti veritatis sed, temporibus laborum, officia non, repudiandae porro cum. Labore eum distinctio eligendi porro deleniti eius, dolorum quibusdam itaque nisi culpa pariatur vero tempore.

Minima
a voluptate praesentium ut, repellendus fugiat officia magni. Minus eum commodi mollitia nobis, beatae maxime repellat, tempore aliquam alias asperiores totam quae excepturi similique eligendi preferendis ipsam cupiditate autem!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dignissimos pariatur, facilis ex quod itaque, esse nesciunt? Unde voluptate dolorum, corporis expedita enim sit molestias, at sapiente nulla cum commodi adipisci.

Рисунок 79

В файле *float.html* можно посмотреть размеры всех элементов, используя инспектор свойств. На скриншоте видно, что **div** с классом **letters** имеет размеры 161.16x70px (выделен голубой рамкой), а абзацы рядом с ним уже большего размера — 1008x38px, хотя и **div**, и абзацы — блочные элементы, которые по умолчанию занимают все доступное пространство внутри родительского элемента.

Еще на этом скриншоте видно, что абзацы заходят под **div**, хотя визуально текст абзаца находится сбоку от **div-a**.

Lorem
sit amet, consectetur adipisicing elit. Dolore obcaecati a tempore doloribus inventore, suscipit fugiat aut sit, iusto iste, consectetur error voluptatibus dolor nemo nihil culpa rerum similique nam.
Fuga a repellendus qui minima deleniti veritatis sed, temporibus laborum, officia non, repudiandae porro cum. Labore eum distinctio eligendi porro deleniti eius, dolorum quibusdam itaque nisi culpa pariatur vero tempore.

Lorem
p | 1008 x 38
sit amet, consectetur adipisicing elit. Dolore obcaecati a tempore doloribus inventore, suscipit fugiat aut sit, iusto iste, consectetur error voluptatibus dolor nemo nihil culpa rerum similique nam.
Fuga a repellendus qui minima deleniti veritatis sed, temporibus laborum, officia non, repudiandae porro cum. Labore eum distinctio eligendi porro deleniti eius, dolorum quibusdam itaque nisi culpa pariatur vero tempore.

Minima
a voluptate praesentium ut, repellendus fugiat officia magni. Minus eum commodi mollitia nobis, beatae maxime repellat, tempore aliquam alias asperiores totam quae excepturi similique eligendi preferendis ipsam cupiditate autem!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dignissimos pariatur, facilis ex quod itaque, esse

Рисунок 80

То же самое относится к элементам, которым назначено свойство **float:right**:

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Deserunt harum fuga numquam earum cumque eum non ratione omnis, dolor ullam. Debitis corporis itaque quisquam sequi, libero nisi praesentium ipsa aspernatur similique maiores quibusdam illo aliquid dolore. Iste placeat suscipit saepe, mollitia repudiandae eos esse debitis.



Laborum ullam obcaecati voluptatibus dolore repudiandae quam molestias! Iure accusantium incident, numquam! Pariatur ipsa cum optio saepe quisquam quod velit expedita, illo quo est veniam fuga possimus amet, nemo labore quia tempora consequuntur. Facere, amet, ducimus. Ad fugit molestias repudiandae inventore laboriosam at necessitatibus consequatur.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Quisquam et, magnam cupiditate, nobis sunt corporis, vel quae mollitia modi beatae molestias odit iusto nesciunt dolor deserunt. Asperiores libero eos ad ratione dolor sunt illum eum reiciendis fuga blanditiis laboriosam, voluptatum dolorem expedit placeat hic ut.

Et cumque dicta blanditiis, distinctio recusandae similique quasi neque illum fugiat impedit, nulla. Repellendus, doloribus nam praesentium inventore. Dolores, ex! Vel aspernatur nesciunt quia quasi deleniti esse aperiam, atque sed reprehenderit, laborum quibusdam consectetur fugit placeat temporibus labore tempore repellat eum, sunt nemo, illo quod.



Рисунок 81

Если текста в блоке достаточно много, то элемент будет занимать все доступное пространство, вне зависимости от свойства **float**. Если целью использования этого свойства является создание нескольких колонок, то обязательно придется указывать свойство **width**.

В файле *float-columns.html* свойство **float** использовано для построения 4-х колонок, имеющих примерно одинаковое количество текста и ширину. Первый блок колонок имеет следующие стилевые свойства:

```
.block-left {  
    width: 23%;  
    margin: 1%;  
    float: left;  
}
```

Колонки имеют ширину в 23% + 2% на левый и правый отступ (**margin: 1%**), поэтому внутреннее пространство родительского элемента, имеющего класс *container*, делится поровну на 4 колонки ($100\%/4 = 25\%$). Каждый блок имеет свойство **float** со значением **left**, поэтому ко-

колонки располагаются друг за другом в привычном нам порядке слева направо.

Второй блок колонок имеет те же стилевые свойства, но свойство **float** для них установлено, как **right**. Поэтому визуально размещение колонок меняется на противоположное — первый столбец размещается справа, а последний слева.

Float Left Block 1

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Rerum voluptas, distinctio odio error tenetur magnam!

 Repellendus, aspernatur, saepe optio architecto et numquam blanditiis repellat voluptatibus nisi non quasi! Cumque, nostrum!

Float Left Block 2

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Excepturi, ad, dolorum? Quis, repudiandae sed ea!

 Maxime eligendi, at architecto illum, obcaecati veniam iusto dignissimos corporis harum impedit voluptas rem porro.

Float Left Block 3

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptates voluptas, laudantium atque iusto cum alias.

 Tenetur repudiandae ut inventore nemo expedita veritatis alias dolor commodi dolores nulla. Rem, fugit saepe.

Float Left Block 4

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Reprehenderit maxime nobis vero inventore ipsum praesentium!

 Pariatur necessitatibus, maxime ab repellendus, hic omnis ex debitis sequi itaque facilis deleniti recusandae consequatur.

Float Right Block 4

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Reprehenderit maxime nobis vero inventore ipsum praesentium!

 Pariatur necessitatibus, maxime ab repellendus, hic omnis ex debitis sequi itaque facilis deleniti recusandae consequatur.

Float Right Block 3

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptates voluptas, laudantium atque iusto cum alias.

 Tenetur repudiandae ut inventore nemo expedita veritatis alias dolor commodi dolores nulla. Rem, fugit saepe.

Float Right Block 2

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Excepturi, ad, dolorum? Quis, repudiandae sed ea!

 Maxime eligendi, at architecto illum, obcaecati veniam iusto dignissimos corporis harum impedit voluptas rem porro.

Float Right Block 1

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Rerum voluptas, distinctio odio error tenetur magnam!

 Repellendus, aspernatur, saepe optio architecto et numquam blanditiis repellat voluptatibus nisi non quasi! Cumque, nostrum!

Рисунок 82

Этой особенностью пользуются, когда необходимо разместить один блок слева, а второй справа в одном ряду. Например, такой подход можно использовать для блока с логотипом или названием сайта, который обычно размещается в левом верхнем углу, а меню, напротив, — в правом углу страницы. Этот вариант вы также найдете в файле *float-columns.html*.

Floating Columns

[Home](#) [About](#) [Portfolio](#) [Blog](#) [Contacts](#)

Рисунок 83

Полезные ссылки:

1. <https://developer.mozilla.org/en-US/docs/Web/CSS/float>.
2. <https://css-tricks.com/all-about-floats/>.
3. <https://www.smashingmagazine.com/2009/10/the-mystery-of-css-float-property/>.
4. https://www.w3schools.com/css/css_float.asp.
5. <http://htmlbook.ru/css/float>.
6. <https://devionity.com/ru/courses/css-fundamentals/css-float>.
7. <http://html-plus.in.ua/css-svoystvo-float-plavayushchie-yelementy-otmena-obtekaniya/>.

Отмена обтекания

В ряде случаев float-элемент может занимать больше пространства по высоте, чем это необходимо в соответствии с макетом дизайнера или по логике расположения текстовых элементов. Весь текст или другие блоки, которые в html-разметке следуют за плавающим элементом, будут «обтекать» его слева или справа, ломая внешний вид страницы.

В этих ситуациях следует использовать css-свойство clear, которое имеет следующие значения:

```
clear: left | right | both | none | inherit
```

Значение **none** является значением по умолчанию и никак не влияет на обтекание элементов. Значение **left** применяется для отмены обтекания элементов, имеющих свойство **float: left**, **right** — для элементов со свойством **float: right**, а значение **both** отменяет обтекание с обеих сторон, поэтому используется в тех случаях, когда либо

неизвестно, какое было значение у свойства `float`, либо обтекание применялось с двух сторон.

Это свойство можно применять к любому элементу, как блочному, так и строчному, но чаще всего его используют либо для `div`, либо для `br`:

```
.clear { clear: both; }  
<div class="clear"></div>  
<br class="clear">
```

Располагают элемент с классом `clear` в том месте, где нужно отменить обтекание плавающего блока текстом.

Минусом использования этого способа является то, что в разметке появляется лишний элемент, не несущий никакой другой нагрузки, кроме отмены обтекания. Плюсом можно назвать то, что внутри текста статьи это может быть самым простым способом прервать обтекание элементов текстом, особенно при редактировании страниц и записей в системах управления контентом (Content Management System, или CMS).

Рассмотрим практический пример использования такого способа отмены обтекания. На странице «Our Team» необходимо расположить фото нескольких работников компании, их имена, название должности и краткую информацию о каждом из них. Мы пока не разбирали, как добавляются изображения на страницу, поэтому заменим фотографию блоком с надписью «Photo 1». Этот блок имеет свойство `float: left`, поэтому текст обтекает его справа. Как видно из рисунка ниже, обтекание касается не только текста, но и следом идущего блока «Photo 2», т.к. текста в первом блоке меньше, чем

высота нашей воображаемой картинки. Не слишком красиво, не так ли (рис. 84)?

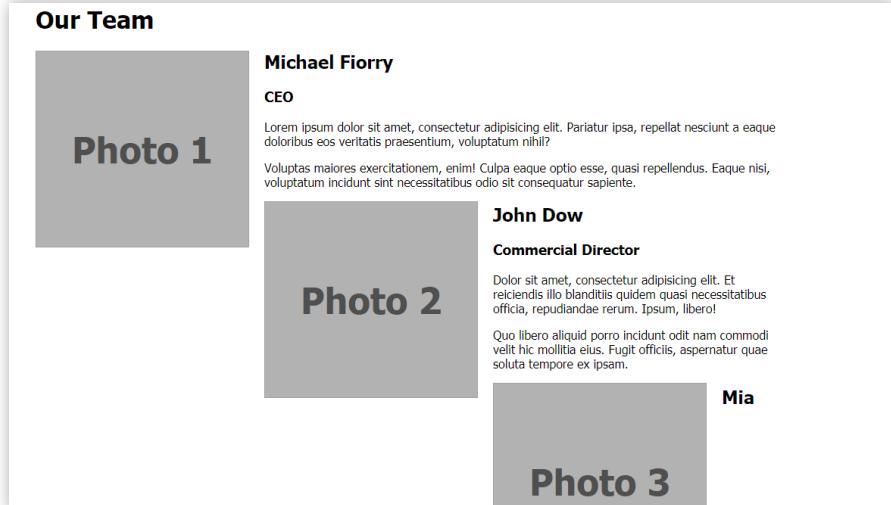


Рисунок 84

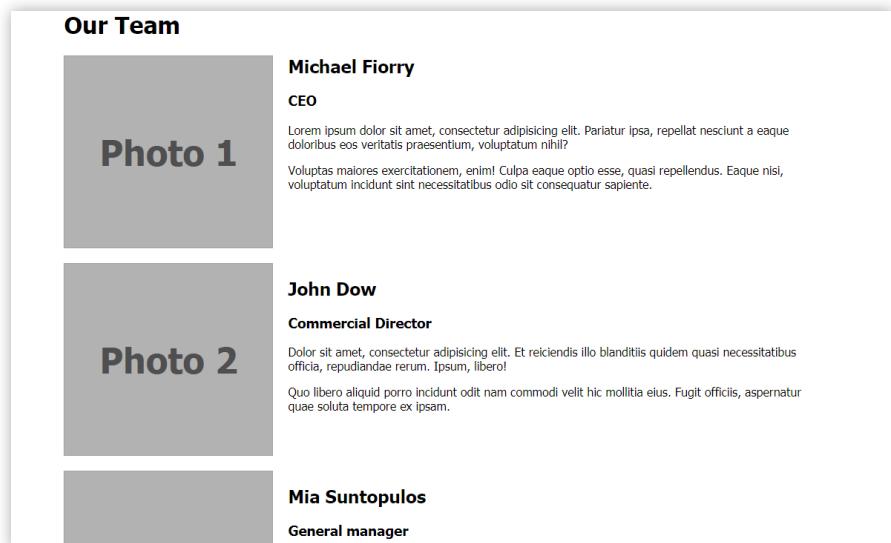


Рисунок 85

В этом случае нам как раз пригодится отмена обтекания, связанная с классом `clear`, описанном выше. Пример вы найдете в файле `clear.html`. Можете применить класс `clear` еще к одному блоку в нем. Результат выглядит намного лучше (рис. 85).

Использование свойства `overflow`

Вторым способом отмены обтекания является использование свойства `overflow` со значениями `hidden` или `auto` для родительского блока-контейнера.

Дело в том, что при расположении внутри какого-либо блочного элемента нескольких плавающих блоков, высота родительского элемента «схлопывается», т.е. превращается в 0. При необходимости задать этому родительскому элементу фоновый цвет или фоновое изображение, вы неизбежно столкнетесь с тем, что ни цвет, ни картинка не отображаются. Связано же такое поведение элемента с тем, что его высота равна 0, и ни цвет, ни изображение невозможна увидеть.

В файле `float-plus-overflow.html` для `div` с классом `wrap` Инспектор свойств показывает размеры 1000x0px, хотя в нем находятся 3 плавающих элемента с классом `block`.



Рисунок 86

Именно из-за плавающих элементов и произошло «схлопывание» блока. CSS-правила для этих классов таковы:

```
.wrap {
    width: 1000px;
    margin: auto;
    background-color: orange;
}

.block {
    width: 30.3%;
    margin: 1.5%;
    float: left;
    background-color: white;
}
```

Заметьте, что у блока с классом *wrap* назначено свойство **background-color: orange**, т.е. оранжевый цвет фона, но мы его не видим на скриншоте вверху.

Именно в этом случае имеет смысл выполнить отмену обтекания внутри родительского элемента, задав ему свойство **overflow**:

```
.wrap {
    width: 1000px;
    margin: auto;
    background-color: orange;
    overflow: hidden;
}
```

И внешний вид сразу изменится (рис. 87).

Инспектор свойств показывает, что теперь размеры родительского блока с классом *wrap* 1000x340.81px, т.е. высота восстановилась (рис. 88).

Use overflow: hidden for parent element

Float Block 1

Quintum bulatore ipsum dolor sit amet, consectetur adipisciing elit. Quis eaque asperiores recusandae labore ipsam ullam eos modi iste tempora quidem illo incident!

Ut sit voluptatem nulla voluptates. Doloribus illo incident eveniet est, recusandae cum ratione. Ad, cum, reprehenderit. Hic vitae dolor, fuga ex saepe harum sit doloremque!

Float Block 2

Lorem ipsum dolor sit amet, consectetur adipisciing elit. Odit similique quo eaque nulla recusandae molestiae explicabo, est blanditiis quisquam aut.

Fugit temporibus nisi sequi fuga ex saepe accusamus, est voluptates sapiente blanditiis, dolores ut repudiandae harum. Veritatis, minus in ea.

Float Block 3

Lorem ipsum dolor sit amet, consectetur adipisciing elit. Laudantium aperiam itaque voluptatem architecto ratione id quos totam, maiores odit nulla.

Repellat, expedita? Illum explicabo sequi sit quis expedita nam accusantium itaque in temporibus harum earum quaerat laborum deserunt, nobis optio.

Рисунок 87

Use overflow: hidden for parent element

Float Block 1

Quintum bulatore ipsum dolor sit amet, consectetur adipisciing elit. Quis eaque asperiores recusandae labore ipsam ullam eos modi iste tempora quidem illo incident!

Ut sit voluptatem nulla voluptates. Doloribus illo incident eveniet est, recusandae cum ratione. Ad, cum, reprehenderit. Hic vitae dolor, fuga ex saepe harum sit doloremque!

Float Block 2

Lorem ipsum dolor sit amet, consectetur adipisciing elit. Odit similique quo eaque nulla recusandae molestiae explicabo, est blanditiis quisquam aut.

Fugit temporibus nisi sequi fuga ex saepe accusamus, est voluptates sapiente blanditiis, dolores ut repudiandae harum. Veritatis, minus in ea.

Float Block 3

Lorem ipsum dolor sit amet, consectetur adipisciing elit. Laudantium aperiam itaque voluptatem architecto ratione id quos totam, maiores odit nulla.

Repellat, expedita? Illum explicabo sequi sit quis expedita nam accusantium itaque in temporibus harum earum quaerat laborum deserunt, nobis optio.

Рисунок 88

Раскомментируйте строку 23 в файле *float-plus-overflow.html*, чтобы добиться аналогичного результата:

```
19 ▼      .wrap {  
20          width: 1000px;  
21          margin: auto;  
22          background-color: orange;  
23          /* overflow: hidden; */  
24      }
```

Рисунок 89

Вернемся к файлу *float-columns.html*. Если закомментировать в нем свойство `overflow: auto` в 25-ой строке для класса `.container`, то внешний вид колонок на странице изменится.

```
22 ▼      .container {  
23          width: 90%;  
24          margin: 20px auto;  
25          /* overflow: auto; */  
26      }
```

Рисунок 90

Поскольку осталось место между названием страницы слева и меню справа, одна из колонок «подплыла» к названию страницы и внешний вид стал значительно хуже.

Floating Columns		Float Left Block 1	Home	About	Portfolio	Blog	Contacts
		<p>Lore ipsum dolor sit amet, consectetur adipisciing elit. Rerum voluptas, distinctio odio error tenetur magnam!</p> <p>Repellendus, aspernatur, saepe optio architecto et numquam blanditiis repellat voluptatibus nisi non quasi! Cumque, nostrum!</p>	Float Left Block 2	Float Left Block 3			
			<p>Lore ipsum dolor sit amet, consectetur adipisciing elit. Excepturi, ad, dolorum? Quis, repudiandae sed ea!</p> <p>Maxime eligendi, at architecto illum, obcaecati veniam iusto dignissimos corporis harum impedit voluptas rem porro.</p>	<p>Lore ipsum dolor sit amet, consectetur adipisciing elit. Voluptates voluptas, laudantium atque iusto cum alias.</p> <p>Tenetur repudiandae ut inventore nemo expedita veritatis alias dolor commodi dolores nulla. Rem, fugit saepe.</p>			

Рисунок 91

Этот способ не требует добавления лишнего элемента разметки, как это было в случае с применением свойства `clear`, но также имеет свои минусы — если внутри такого элемента расположен абсолютно или относительно позиционированный элемент, он может исчезнуть с экрана после применения `overflow: hidden`.

Класс clearfix

Класс `clearfix` был придуман фронтенд-разработчиком **Николасом Галахером** ([Nicolas Gallagher](#)). Это, пожалуй, самый популярный и эффективный на данный момент способ отмены обтекания. Заключается он в том, что для псевдоэлементов `::before` и `::after` селектора класса `clearfix` записываются следующие правила:

```
.clearfix::before,  
.clearfix::after { content: ""; display: table; }  
.clearfix::after { clear: both; }
```

Затем сам класс записывается в качестве атрибута для родительского элемента:

```
<div class="clearfix"> ... </div>
```

В этом случае нет ни дополнительных ненужных элементов разметки, ни скрытия позиционированных элементов. Очистка обтекания назначается после последнего элемента внутри родительского контейнера, т.к. описана в псевдоэлементе `::after`. Правила для `::before` позволяют избежать проблем с схлопывающимися верхними отступами в современных браузерах.

В файле `clearfix.html` размещены несколько классов для построения колонок разной ширины, имеющих свойство `float: left` (`.col-2` и `.col-3`). Родительским элементом для них является `div` с классом `wrapper`. Но высота колонок в верхнем блоке различна, поэтому те колонки, которые размещенные в html-разметке во втором ряду (второй блок с классом `wrapper`), обтекают их справа и внешний вид макета «плывет» (рис. 92).

Урок 6.1. Блочная модель элементов

About Us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Excepturi et facilis, fugit dicta adipisci omnis repudiandae, quidem esse reiciendis, a doloribus praesentium vero id obcaecati alias necessitatibus ullam ex ea!

Quas, velit quibusdam adipisci, quos assumenda numquam totam neque blanditiis nostrum asperiores perferendis delectus, accusamus natus similique maxime? Rem, sunt perferendis cum rerum tempore querat molestiae vero? Impedit iste, nobis.

Obcaecati quasi saepe, blanditiis aliquid fuga nobis. Consectetur saepe necessitatibus esse suscipit quidem omnis paratur enim tempora excepturi labore!

Our Experience

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fuga harum voluptates expedita dolores, laudantium facere accusamus nisi! Cumque excepturi reiciendis voluptate ipsa dolores, neque laborum, porro magni quis exercitatem suscipit.

Quia officia harum esse tenetur consequuntur, eligendi totam inventore reiciendis culpa

Blog Post 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Facere dicta impedit quiquam

Repellat quas nihil ullam, eum ipsa voluptatem nisi saepe nobis explicabo! Modi ab nesciunt dicta sed, explicabo excepturi?

Quae voluptate ullam aspernatur, laboriosam quisquam, sed. Dolorum alia!

Voluptatis sunt, neque corporis error natus! Maxime facere nobis, nisi, ea ratione esse, ex sit dolores officiis eaque totam illa. Suscipit!

[Read More](#)

Blog Post 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Qui sapiente aliquid repellat in explicabo impedit eos a eaque nobis esse fugiat quae non enim unde, adipisci quis doloreremque harum odit.

Commodi, tempore libero quos. Quae neque voluptate rerum libero blanditiis asperiores, error doloribus natus et eligendi commodi veniam laudantium ducimus eaque nobis iusto dicta aliquid! Illum ea numquam, natus qui.

[Read More](#)

Blog Post 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Atque doloribus quidem sit similique veritatis alias molestias quae, fugiat sunt paritaur facere sed debitis molestiae rem voluptate sequi in sapiente vero.

Molestias porro animi, id illo recusandae impedit eius eum itaque magni similique, iusto. Recusandae sunt dicta ipsam quisquam autem eius sint. Ab totam, eos labore ea nihil! Repudiandae, et vel?

[Read More](#)

IMPORTANT INFORMATION

Corem ipsum dolor sit amet, consectetur adipisicing elit. Aut veritatis dolore tempora odit laboriosam, non, corporis tempore, delectus deserunt at porro asperiores totam doloribus nobis.

CHANGES IN THE SCHEDULE

Harum magnam atque, excepturi eveniet accusantium illum fura animi in lura eius sint adiosci

Рисунок 92

```
44      .clearfix::before,  
45 ▼    .clearfix::after {  
46        content: "";  
47        display: table;  
48    }  
49  
50 ▼    .clearfix::after {  
51        clear: both  
52    }  
53  </style>  
54  </head>  
55  
56 ▼ <body>  
57 ▼   <div class="wrapper clearfix">  
58 ▶     <div class="col-2">...</div>  
64 ▶     <div class="col-2">...</div>  
70   </div>  
71 ▼   <div class="wrapper clearfix">  
72 ▶     <div class="col-3">...</div>  
80 ▶     <div class="col-3">...</div>  
86 ▶     <div class="col-3">...</div>  
92   </div>  
93 ▼   <div class="wrapper clearfix">  
94 ▶     <div class="col-3">...</div>  
100 ▶     <div class="col-3">...</div>  
106 ▶     <div class="col-3">...</div>  
112   </div>  
113  </body>
```

Рисунок 93

Достаточно описать стилевые свойства класса `clearfix` в этом документе и добавить сам класс в качестве атрибута для каждого родительского элемента с классом `wrapper` — и внешний вид документа будет иметь строгую структуру (рис. 93, 94).

The screenshot shows a website layout with a header and several content sections. The main content area is enclosed in a `.wrapper` class. Inside, there are two columns: "About Us" and "Our Experience". Each column contains a paragraph of placeholder text. Below these are three "Blog Post" sections, each with its own heading and text. The first blog post has a "Read More" link. The second and third blog posts also have "Read More" links. At the bottom, there are three callout boxes: "IMPORTANT INFORMATION" (blue), "CHANGES IN THE SCHEDULE" (green), and "ATTENTION" (yellow). Each box contains a short paragraph of placeholder text.

About Us

Our Experience

Blog Post 1

Blog Post 2

Blog Post 3

IMPORTANT INFORMATION

CHANGES IN THE SCHEDULE

ATTENTION

Рисунок 94

Попробуйте это сделать самостоятельно.

Примечание: в большинстве современных фреймворков, предназначенных для верстки, например, *Bootstrap* или *Foundation*, этот класс включён в состав основных css-правил. Его название стало широкоупотребимым, поэтому имеет смысл использовать его именно с таким именем.

Ссылки:

1. <https://developer.mozilla.org/en-US/docs/Web/CSS/clear>.
2. https://www.w3schools.com/css/css_float.asp.
3. <http://nicolasgallagher.com/micro-clearfix-hack/>.
4. <https://www.smashingmagazine.com/2007/05/css-float-theory-things-you-should-know/>.
5. <http://www.yuiblog.com/blog/2010/09/27/clearfix-reload-ed-overflowhidden-demystified/>.
6. <http://www.cssmojo.com/the-very-latest-clearfix-reloaded/>.
7. <https://getbootstrap.com/docs/4.0/utilities/clearfix/>.
8. <https://foundation.zurb.com/sites/docs/float-classes.html>.
9. <http://html-plus.in.ua/css-svoystvo-float-plavayushchie-lementy-otmena-obtekaniya/#clear-float>.

Использование свойств блочной модели для верстки html-страницы

На данный момент мы уже знаем ряд свойств, которые позволяют нам сверстать простую html-страницу, правда, без графических элементов, т.е. без картинок. Тем не менее на этой странице будет ряд разделов (*section*) с информационными блоками, размещенными в несколько столбцов, что актуально для современных web-страниц, будет заголовочная часть (*header*) и нижняя часть, или подвал (*footer*).

Файл примера вы найдете в папке examples под именем *box-model-example.html*.

Внешний вид нашей страницы представлен на скриншоте.

Sample Page

with CSS Box model

for students

About Us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id unde cum dolilia quisquam atque molestiae eos, tempore, rerum cum dolores, ullam aperiam rem veritatis totam, nulla hic quam. Est, iste.

Deleniti, aliquam perforendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequil A, aliquid!

Our mission

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Tempore dicta, earum dolorem hic doloremque voluptates?

Sequi libero ipsum autem quos quod earum dolorem ipsa ad officis. Aliquam hic est ipsum!

Odio molestias doloribus, explicabo? Optio, delectus. Adipisci expedita alias libero non repudiandae tempora, omnis veritatis.

Popular Courses

 <p>The Web Developer</p> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eius atque, saepe provident.</p> <p>\$27.99</p>	 <p>JavaScript Developer</p> <p>Cum nemo repellendus labourum, commod odit deserunt minus et sunt. Minus, iste.</p> <p>\$29.99</p>	 <p>FrontEnd Developer</p> <p>Voluptatem corporis doloremque eaque deleniti iusto odio asperiores veritatis expedita repudiandae maiores!</p> <p>\$39.99</p>	 <p>Modern Web Design</p> <p>Error dolor laudantium soluta ex illo labore eius placeat aliquam deserunt, fugit.</p> <p>\$11.99</p>	 <p>CSS Complete Course</p> <p>Assumenda fugit, adipisci similiq veritatis nihil quas quibusdam nisi et temporibus sed!</p> <p>\$24.99</p>
--	--	--	--	--

Our Courses

HTML5/CSS3
JavaScript/Query
PHP/MySQL
CMS/SEO

Our Services

Group training
Individual training
Online training
The latest techniques

Useful Links

Lorem ipsum dolor sit.
Fugit minus vitae tenetur.
Sunt tempora ipsam facere.
Rerum sunt suscipit officia.

© 2018 | Simple Page Company™

All rights reserved.

Рисунок 95

Начнем форматирование с html-разметки header. В нем будет 2 заголовка и абзац:

```
<header>
  <h1>Sample Page</h1>
  <h2>with CSS Box model</h2>
  <p>for students</p>
</header>
```

В css-правилах для header мы запишем правила для цвета фона и размещения текста по центру:

```
header {  
    text-align: center;  
    background-color: #ff6825;  
}
```

Внешний вид header будет таким:



Sample Page

with CSS Box model

for students

Рисунок 96

Можно заметить, что header-у не хватает высоты, которую мы зададим с помощью свойства `height`. Кроме того, отступы для `body`, которые назначены по умолчанию, лучше убрать. Также имеет смысл установить для всего текста какое-либо семейство шрифта без засечек, т.к. именно такой шрифт удобней читать с экрана компьютера или телефона.

```
body {  
    margin: 0;  
    font-family: sans-serif;  
}  
  
header {  
    background-color: #ff6825;  
    text-align: center;  
    height: 150px;  
}
```

Сейчас мы получили интересную ситуацию: header стал больше, слева и справа отступы ушли, но сверху отступ почему-то остался прежним, а не исчез совсем, как ожидалось.



Рисунок 97

В чем же дело? Оказывается, дело во внешних отступах, заданных по умолчанию для заголовков. На скриншоте ниже отступы показаны светло-оранжевым цветом:

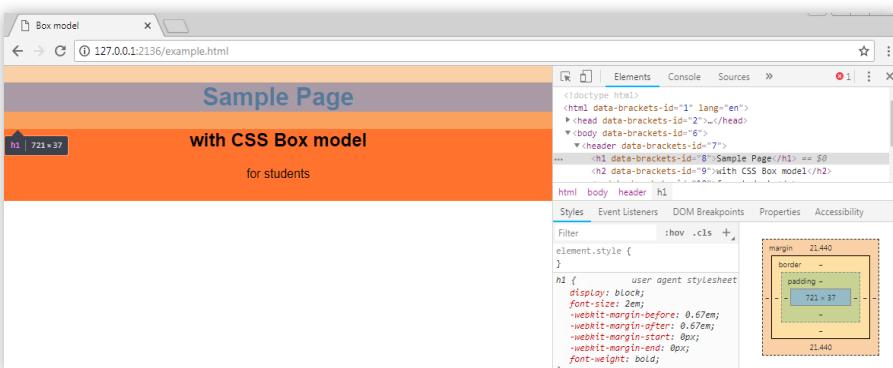


Рисунок 98

Именно эти отступы и отодвигают заголовок вместе с header-ом от верхней границы браузера. Изменим эту ситуацию, обнулив значение `margin` для заголовка первого уровня, т.к. именно он расположен вверху. Заодно увеличим размер шрифта и добавим тень:

```
h1 {  
    margin: 0;  
    font-size: 3.5rem;  
    text-shadow: 2px 2px 2px rgba(0, 0, 0, 0.4);  
}
```

Sample Page

with CSS Box model

for students

Рисунок 99

Теперь явно видно, что в header не хватает отступов сверху и снизу, причем это должны быть внутренние отступы, т.к. на них должна распространяться заливка фоновым цветом. Добавим padding для header только сверху и снизу, т.к. слева и справа они нам не нужны:

```
header {  
    background-color: #ff6825;  
    text-align: center;  
    height: 150px;  
    padding: 8% 0;  
}
```

Sample Page

with CSS Box model

for students

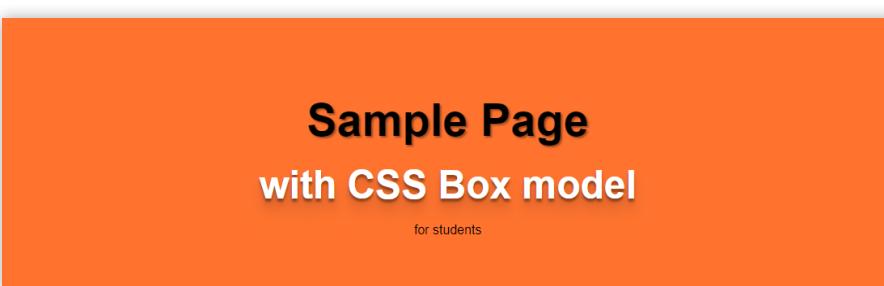
Рисунок 100

Теперь внешний вид header является вполне удовлетворительным (рис. 100).

Далее нам следует заняться заголовком второго уровня, добавить ему немного красоты. Выделим его цветом, размером и тенью и несколько увеличим внешние отступы:

```
h2 {  
    font-size: 3rem;  
    color: #fff;  
    text-shadow: 0px 4px 3px rgba(0, 0, 0, 0.4),  
                0px 8px 13px rgba(0, 0, 0, 0.1),  
                0px 18px 23px rgba(0, 0, 0, 0.1);  
    margin: 1.2rem;  
}
```

Теперь наша заголовочная часть страницы выглядит вполне приемлемо:



Sample Page with CSS Box model

for students

Рисунок 101

Далее мы будем делать html-разметку раздела «About Us». Для этого можно воспользоваться такой аббревиатурой Emmet:

```
section>.half*2>h2{About Us}+p*2>lorem
```

После нажатия на клавишу Tab получим такой html-код:

```
<section>
  <div class="half">
    <h2>About Us</h2>
    <p>Lorem ipsum dolor ... iste.</p>
    <p>Deleniti, aliquam ... A, aliquid!</p>
  </div>

  <div class="half">
    <h2>About Us</h2>
    <p>Lorem ipsum dolor ... vero laudantium.</p>
    <p>Eaque, libero ...Laborum, enim, magni.</p>
  </div>
</section>
```

Внешний вид страницы стал таким:



Рисунок 102

Сразу бросается в глаза, что к заголовкам второго уровня применились такие же стили, что и для `h2` в `header`. Поэтому нам придется несколько изменить стили и селекторы: отдельно напишем правила для `h2` и сделаем контекстный селектор для заголовка в `header`:

```

h2 {
    text-decoration: underline;
    text-decoration-style: dotted;
    font-family: Cambria, serif;
}

header h2 {
    font-size: 3rem;
    color: #fff;
    text-shadow: 0px 4px 3px rgba(0, 0, 0, 0.4),
                  0px 8px 13px rgba(0, 0, 0, 0.1),
                  0px 18px 23px rgba(0, 0, 0, 0.1);
    margin: 1.2rem;
}

```



Рисунок 103

Поскольку для `h2` было добавлено свойство `text-decoration: underline`, то это свойство применилось и для заголовка в `header` (рис. 103).

Мы можем переопределить это свойство, добавив строчку для `header h2`:

```
header h2 {  
    font-size: 3rem;  
    ...  
    text-decoration: none;  
}
```

Теперь все заголовки выглядят нормально.



Рисунок 104

В разметке раздела (`section`) были использованы классы `half`, которые нужны для того, чтобы разместить 2 блока рядом друг с другом, разделив пространство по ширине страницы пополам, т.е. по 50%. Стили будут такими:

```
.half {  
    width: 50%;  
    float: left;  
}
```

Можно увидеть, что 2 блока сейчас действительно находятся рядом, но текст в них «прилипает» к границам соседнего блока.

About Us

About Us

Deleniti, aliquam perferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequi A, aliquid

Deleniti, aliquam perferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequi A, aliquid

Рисунок 105

Исправить это можно, добавив отступы к стилям этого класса:

```
.half {  
    width: 50%;  
    float: left;  
    padding: 1.5%;  
}
```

About Us

About Us

Deleniti, aliquam perferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequi A, aliquid!

About Us

About Us

Eaque, libero. Expedita laborum adipisci dolorum a earum dolore totam doloremque, culpa beatata assumenda dolorum, ipsum paratur sapiente, illum ipsum voluptas laudantium laboriosam deleniti, rerum atque possimus? Laborum, enim, magni.

Рисунок 106

Однако, это привело к тому, что блоки расположились опять друг под другом, т.к. добавление 1.5% увеличило ширину каждого из блоков на 3% (1.5% слева и 1.5%

справа). Для того чтобы расположить блоки рядом, нужно уменьшить ширину колонки в классе `half` на 3%:

```
.half {  
    width: 47%;  
    float: left;  
    padding: 1.5%;  
}
```

Теперь внешний вид колонок вполне приемлем: за счет отступов текст воспринимается намного лучше.

About Us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id unde mollitia quas quisquam atque molestiae eos, tempore, rerum cum dolore, ullam aperiam rem veritatis totam, nulla hic quam. Est, iste.

Deleniti, aliquam perferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequi A, aliquid!

About Us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Neque fugiat, nam veniam reiciendis iste repellat deleniti consectetur. Atque similique, obcaecati minima impedit, aliquam quidem nam ut dolor cum, vero laudantium.

Eaque libero. Expedita laborum adipisci dolorum a earum dolore totam doloremque, culpa beatae assumenda dolore, ipsam pariatur sapiente, illum ipsum voluptas laudantium laboriosam deleniti, rerum atque possimus? Laborum, enim, magni.

Рисунок 107

Для того чтобы тест не дублировался, изменим заголовок второго блока и количество абзацев в нем.

```
<div class="half">  
    <h2>Our mission</h2>  
    <p>Lorem ipsum ...voluptates?</p>  
    <p>Sequi libero ipsum ... est ipsam!</p>  
    <p>Odio molestias ... veritatis.</p>  
</div>
```

В первом блоке добавим еще один `div` с иконкой для проигрывания видео. Пока она никуда не ведет, но в будущем можно привязать к ней ссылку на видео с youtube.



Рисунок 108

сом. Поэтому и внешний вид `div` будем форматировать по аналогии с иконкой youtube. Html-разметка будет состоять из 3-х вложенных друг в друга тегов:

```
<div class="half">
    <h2>About Us</h2>
    <div class="right-block">
        <div class="video" title="Play Video About Us">
            <div class="play"></div>
        </div>
    </div>
    <p>Lorem ... Est, iste.</p>
    ...
</div>
```

Основной элемент с классом *right-block* будет отвечать за размещение элемента относительно текста в первой колонке. Его стилевые правила определяют, что он будет находиться справа от текста (`float: right`), иметь внутренние отступы (`padding`) и внешний отступ только с левой стороны, т.к. другие нам не важны.

```
.right-block {
    float: right;
    padding: 15px;
    margin-left: 10px;
}
```

Пока что мы только «отгородили» некоторое количество места на странице. Поскольку внутренние элементы у нас не имеют контента (текста), то и увидеть мы их не можем.



Рисунок 109

Собственно, контент в этих элементах нам и не нужен, т.к. они предназначены для имитации иконки youtube. Если внимательно на нее посмотреть, можно заметить, что `div` с классом *video* предназначен для формирования красного прямоугольника с закругленными углами, а `div` с классом *play* — для создания белого треугольника.

Для начала зададим для класса *video* ширину, высоту и цвет фона:

```
.video {  
    background-color: red;  
    width: 20px;  
    height: 20px;  
}
```

Сейчас мы получили небольшой квадрат. Его размеры мы будем увеличивать за счет внутренних отступов (*padding*) после того, как сформируем треугольник в середине.

About Us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id unde mollitia quas quisquam atque molestiae eos, tempore, rerum cum dolorem, ullam apriam rem veritatis totam, nulla hic quam. Est, iste.

Deleniti, aliquam perferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequil A, aliquid!

Our mission

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Tempore dicta, earum dolorem hic doloremque voluptates?

Sequi libero ipsum autem quos quod earum dolorem ipsa ad officiis. Aliquam hic est ipsam!

Odio molestias doloribus, explicabo? Optio, delectus. Adipisci expedita alias libero non repudiandae tempora, omnis veritatis.

Рисунок 110

Для того чтобы сделать треугольник, нам понадобится ... свойство **border**. Дело в том, что при больших значениях толщины рамок (свойство **border-width**) становится заметно, что все стороны соединяются в углах в виде скошенных линий.

Мы запишем для класса *play* пока что тестовые цвета рамки, а потом изменим это.

```
.play {
    border: 10px solid;
    border-color: yellow #900 green orange;
}
```

About Us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id unde mollitia quas quisquam atque molestiae eos, tempore, rerum cum dolorem, ullam aperiam rem veritatis totam, nulla hic quam. Est, iste.



Deleniti, aliquam preferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequi! A, aliquid!

Рисунок 111

На скришоте видно, что рамки разных цветов представляют собой треугольники при отсутствии контента. Если мы назначим прозрачный цвет (**transparent**) для большинства сторон рамки, то видна она не будет. Исходя из тестового варианта, можно понять, что треугольник для нашей иконки можно сформировать с помощью левой стороны рамки. Поэтому правила для *div* с классом *play* изменятся так:

```
.play {  
    border: 10px solid transparent;  
    border-left-color: white;  
}
```

About Us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id unde molitiae quas quisquam atque molestiae eos, tempore, rerum cum dolorem, ullam aperiam rem veritatis totam, nulla hic quam. Est, iste.



Deleniti, aliquam preferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequi! A, aliquid!

Рисунок 112

Пока мы получили не иконку youtube, а иконку флагжка. Вернемся к редактированию css-правил для класса *video* и добавим внутренние отступы в 10px.

```
.video {  
    background-color: red;  
    width: 20px;  
    height: 20px;  
    padding: 10px;  
}
```

About Us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id unde molitiae quas quisquam atque molestiae eos, tempore, rerum cum dolorem, ullam aperiam rem veritatis totam, nulla hic quam. Est, iste.



Deleniti, aliquam preferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequi! A, aliquid!

Рисунок 113

Сейчас треугольник переместился ближе к центру, но также стало понятно, что одинаковые отступы нам не подходят. Они должны быть разными, т.к. сама иконка представляет собой прямоугольник, а не квадрат, как сейчас, и треугольник нужно сместить в центр, отступив слева больше, чем справа. Поэтому отступы будут такими:

```
.video {  
    background-color: red;  
    width: 20px;  
    height: 20px;  
    padding: 10px 15px 10px 28px;  
}
```

About Us

Lore ipsum dolor sit amet, consectetur adipisicing elit.
Id unde mollitia quas quisquam atque molestiae eos,
tempore, rerum cum dolore, ullam aperiam rem
veritatis totam, nulla hic quam. Est, iste.



Deleniti, aliquam perferendis iusto fugit, a eveniet dolores maiores
dignissimos ipsa tempore atque voluptate neque animi natus.
Laboriosam, vitae recusandae facere corporis dolores odio voluptatem
placeat at sequi! A, aliquid!

Рисунок 114

Кстати, чтобы подобрать значения отступов, пришлось использовать Инспектор свойств.

Для завершения форматирования нужно добавить закругления углов. Если присмотреться к иконке youtube, можно заметить, что радиус закругления по горизонтали несколько меньше, чем по вертикали. Поэтому воспользуемся 2-мя значениями для свойства border-radius и за-

одно изменим цвет фона для иконки, т.к. он несколько отличается от *red*. Еще добавим для нашей иконки CSS-свойство *cursor: pointer* для того, чтобы при наведении на нее курсор имел вид руки с пальцем, т.к. мы сейчас имитируем ссылку на видео.



Рисунок 115

```
.video {
    background-color: #e02f2f;
    width: 20px;
    height: 20px;
    padding: 10px 15px 10px 28px;
    border-radius: 10px/14px;
    cursor: pointer;
}
```

The screenshot shows a section of a website with a light gray background. On the left, there's a white box containing the text "About Us". Inside this box, there's another white box with the text "Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id unde mollitia quas quisquam atque molestiae eos, tempore, rerum cum dolorem, ullam aperiam rem veritatis totam, nulla hic quam. Est, iste." Below this, there's more text: "Deleniti, aliquam perferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequi! A, aliquid!" To the right of the main content, there's a red play button icon. Further right, there's another white box with the text "Our i" and "Lorem i Tempor". At the bottom right, there's a link labeled "Play Video About Us OFFICIS...".

Рисунок 116

Получилось достаточно похоже. Конечно, проще было вставить картинку, но мы пока не изучали тег для добавления изображений на HTML-страницу. Зато использовали ряд свойств из этого урока и посмотрели, как с помощью *padding* сдвинуть вложенный элемент в центр родительского, а также как сделать треугольник из *border*.

Еще один момент нашего форматирования связан с тем, что на данный момент элементы внутри section занимают все доступное пространство по ширине браузера. Однако на широких экранах это может не слишком хорошо выглядеть. Поэтому обычно эти элементы помещают внутрь контейнера, ширина которого не превышает 1200px, и центрируют относительно окна браузера. Мы используем обрамление элементов с помощью клавиш Ctrl + Shift + A из плагина Emmet и поместим оба элемента с классом *half* в div с классом *container*.

```
<body>
  <header>
    <h1>Sample Page</h1>
    <h2>with CSS Box model</h2>
    <p>for students</p>
  </header>
  <section>
    <div class="container">
      <div class="half">...</div>
      <div class="half">...</div>
    </div>
  </section>
</body>
```

Рисунок 117

Для селектора *.container* зададим максимальную и минимальную ширину в px, для того чтобы на широких экранах наш контент не «расползлся», а на маленьких — не сплющивался, а также назначим ширину в 90%, чтобы на средних экранах справа и слева были отступы. Центрируем контент с помощью свойства *margin: auto*:

```
.container {
  max-width: 1200px;
  width: 90%;
```

```
min-width: 768px;  
margin: auto;  
}
```

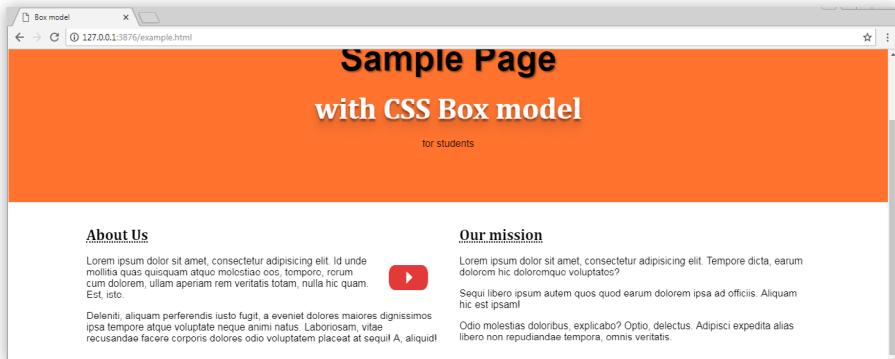


Рисунок 118

Теперь нам необходимо сверстать раздел, в котором будет представлено некоторое количество учебных курсов. Внешний вид будущего раздела представлен на скриншоте.



Рисунок 119

Давайте проанализируем, какие элементы нам будут нужны. Во-первых, мы видим 5 одинаковых блоков с курсами, в которых есть цветная верхняя часть с оранжевым

промо-текстом и заголовком. Во-вторых, ниже идет абзац текста и блок с ценой. Т.е. первоначально мы можем написать такую аббревиатуру Emmet:

```
section>h2{Popular Courses}+.course*5>.top+p+.price
```

Скобка вправо указывает на вложенные элементы. Плюсами показаны элементы, следующие друг за другом, т.е. расположенные внутри одного родительского элемента. В фигурных скобках помещен текст, который потом отобразится внутри тега.

Поскольку у нас есть еще ряд вложенных элементов, аббревиатуру нужно усложнить:

```
section>h2{Popular Courses}+
.course*5>(.top>.promo{Most popular}+
h3{Developer})+(p>lorem12)+.price{19.99}
```

В круглых скобках размещены элементы, представляющие собой группу,ложенную в другие элементы. Еще мы забыли про недавно добавленный элемент с классом *container*, поэтому исправим аббревиатуру еще раз:

```
section>h2{Popular Courses}+
.container>.course*5>(.top>.promo{Most popular}+
h3{Developer})+(p>lorem12)+.price{19.99}
```

Напомню, что ставить пробелы в Emmet-аббревиатурах нельзя — это вызовет ошибку. В конце аббревиатуры нужно поставить курсор и нажать клавишу Tab.

Примечание: если вы хотите освоить написание Emmet-аббревиатур, создайте пустой html-документ и добавляйте в него теги, форматирование и аббревиатуры по мере прочтения этого урока. Само чтение, без практики не даст вам серьезного эффекта запоминания.

После раскрытия аббревиатуры имеем такой внешний вид:

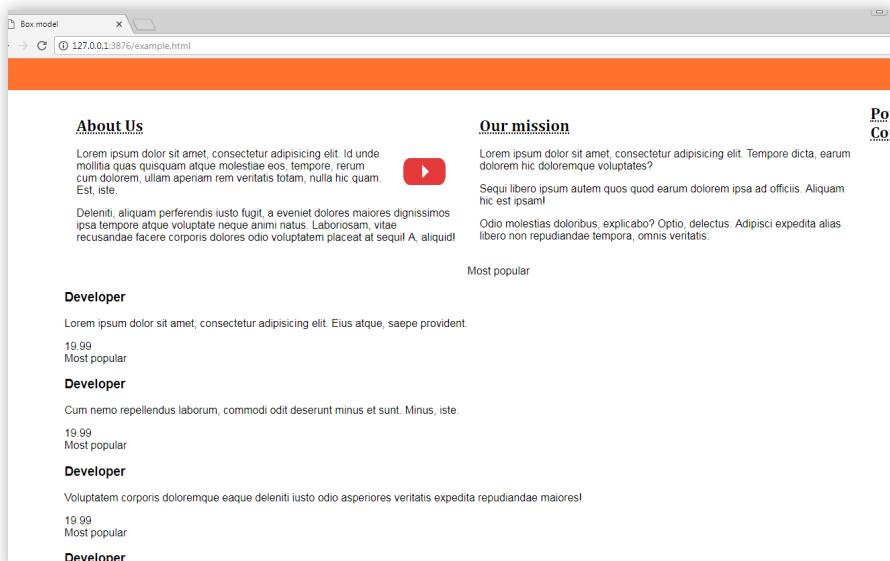


Рисунок 120

Сразу бросается в глаза съехавший вправо заголовок **Popular Courses**. Произошло это потому, что для блоков с классом *half* назначено свойство *float:left*, и заголовок подчиняется этому правилу. Здесь мы как раз сталкиваемся с ситуацией, когда абсолютно необходима отмена обтекания. В этом случае имеет смысл применить пра-

вила, описанные для псевдоэлементов класса *clearfix*. Поскольку отмена обтекания нужна для div-а с классом *container*, мы можем записать именно для него псевдоэлементы с соответствующими свойствами:

```
.container::before,  
.container::after {  
    content: "";  
    display: table;  
}  
  
.container::after {  
    clear: both  
}
```

Заголовок вернулся на то место, которое ему первоначально предназначалось, хотя теперь имеет смысл разместить его по центру.

The screenshot shows a website layout with the following structure:

- Header:** A red header bar containing the text "About Us".
- Content Area:** Contains two columns.
 - Left Column:** Headed by "About Us". It contains placeholder text (Lorem ipsum...) and a small video player icon.
 - Right Column:** Headed by "Our mission". It also contains placeholder text (Lorem ipsum...) and a small video player icon.
- Sidebar:** Headed by "Popular Courses". It lists "Most popular" courses:
 - Developer:** Placeholder text (Lorem ipsum...), 19 99, "Most popular", "Developer".
 - Developer:** Placeholder text (Cum nemo repellendus laborum, commodi edit deserunt minus et sunt. Minus, iste).

Рисунок 121

Поэтому добавим в разметку заголовка атрибут *class="text-center"* и в стилях запишем единственное правило:

```
.text-center {  
    text-align: center;  
}  
...  
<h2 class="text-center">Popular Courses</h2>
```

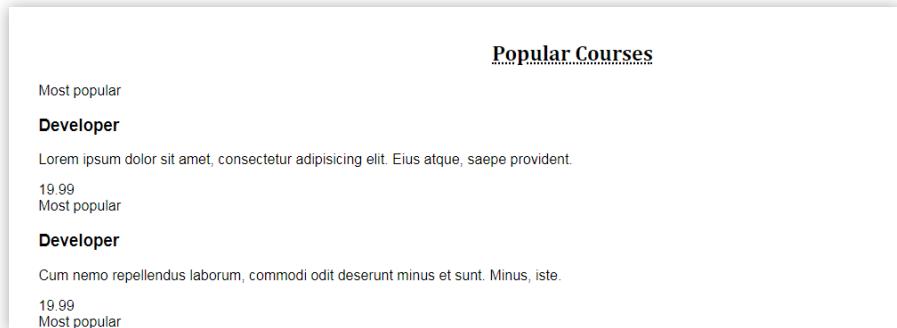


Рисунок 122

Далее нашей задачей будет распределить 5 блоков с курсами в одну линию и несколько поменять текст заголовков и цены по сравнению с Emmet-аббревиатурой. Как известно, пространство родительского контейнера для вложенных в него элементов составляет 100%, каким бы по величине в px оно не было. Поэтому для назначения ширины нам нужно разделить 100% на количество колонок, т.е. на 5. Значит, значение width для каждой из колонок с классом course составит 20%. Плюс нужно назначить обтекание с помощью свойства float:

```
.course {  
    width: 20%;  
    float: left;  
}
```

Получим колонки такого вида:

<u>Popular Courses</u>				
Top sales	New!	Most popular	Super Sale!	Most popular
The Web Developer	JavaScript Developer	FrontEnd Developer	Modern Web Design	CSS Complete Course
Lorem ipsum dolor sit amet, Cum nemo repellendus consectetur adipisciing elit. laborum, commodi odit Eius atque, saepe provident. deserunt minus et sunt. Minus, iste.	Voluptatem corporis doloremque eaque deleniti iusto odio asperiores veritatis expedita repudiandae maiores!	Error dolor laudantium solutaAssumenda fugit, adipisci ex illo labore eius placeat aliquam deserunt, fugit.	11.99	24.99
27.99	29.99	39.99		

Рисунок 123

Сразу же бросается в глаза то, что между колонками не хватает отступов. Зададим свойство *padding*, но сразу вспомним о том, что при форматировании класса *half* нам пришлось уменьшать ширину блока (свойство *width*) на величину *padding**2. Еще желательно добавить внешний отступ снизу, чтобы текст был отодвинут от нижней границы. Поэтому правила для *.course* будут такими:

```
.course {
    width: 17%; /* 20% - 1.5%*2 */
    float: left;
    padding: 1.5%;
    margin-bottom: 20px;
}
```

Результат на скриншоте:

<u>Popular Courses</u>				
Top sales	New!	Most popular	Super Sale!	Most popular
The Web Developer	JavaScript Developer	FrontEnd Developer	Modern Web Design	CSS Complete Course
Lorem ipsum dolor sit amet, consectetur adipisciing elit. Eius atque, saepe provident.	Cum nemo repellendus laborum, commodi odit deserunt minus et sunt. Minus, iste.	Voluptatem corporis doloremque eaque deleniti iusto odio asperiores veritatis expedita repudiandae maiores!	Error dolor laudantium solutaAssumenda fugit, adipisci ex illo labore eius placeat aliquam deserunt, fugit.	Assumenda fugit, adipisci similiqe veritatis nihil quas quibusdam nisi et temporibus sed!
27.99	29.99	39.99	11.99	24.99

Рисунок 124

Давайте отформатируем нижнюю часть каждой из наших колонок — *div* с классом *price*. Текст в нем нужно сделать жирным — за это отвечает свойство *font-weight*, выровнять по правому краю (свойство *text-align*) и добавить перед текстом символ \$ — для этого мы используем псевдокласс *:before* и его обязательное свойство *content*:

```
.price {
    text-align: right;
    font-weight: bold;
}

.price:before {
    content: '$';
}
```

В результате получим:

<u>Popular Courses</u>				
Top sales	New!	Most popular	Super Sale!	Most popular
The Web Developer	JavaScript Developer	FrontEnd Developer	Modern Web Design	CSS Complete Course
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eius atque, saepe provident.	Cum nemo repellendus laborum, commodi odit deserunt minus et sunt. Minus, iste.	Voluptatem corporis doloremque eaque deleniti iusto odio asperiores veritatis expedita repudiandae maiores!	Error dolor laudantium soluta ex illo labore eius, placeat aliquam deserunt, fugit.	Assumenda fugit, adipisci similique veritatis nihil quas quibusdam nisi et temporibus sed!
\$27.99	\$29.99	\$39.99	\$11.99	\$24.99

Рисунок 125

Теперь нам предстоит форматирование верхнего блока с классом *top*. Для него необходимо задать рамку (*border*), выравнивание текста по центру (*text-align*) и цвет фона (*background-color* — используем цвет из *header*) и белый цвет текста (*color*):

```
.top {
    border: 3px double #666;
    background-color: #ff6825;
    color: #fff;
    text-align: center;
}
```



Рисунок 126

Сразу бросаются в глаза отсутствие внутренних отступов (`padding`) и неодинаковая высота элементов в силу разного количества текста в них. Последнее мы можем исправить с помощью свойства `min-height`. Также желательно изменить семейство шрифта (`Cambria, serif`), чтобы шрифты в заголовках второго и третьего уровня у нас были одинаковыми:

```
.top {
    ...
    min-height: 100px;
    padding: 15px;
    font-family: Cambria, serif;
}
```

Получилось намного аккуратнее (рис. 127).



Рисунок 127

Последнее, что нужно будет сделать — это закруглить верхний правый угол (свойство `border-top-right-radius`). Окончательные свойства таковы:

```
.top {  
    border: 3px double #666;  
    border-top-right-radius: 20px;  
    background-color: #ff6825;  
    color: #fff;  
    text-align: center;  
    min-height: 100px;  
    padding: 15px;  
    font-family: Cambria, serif;  
}
```

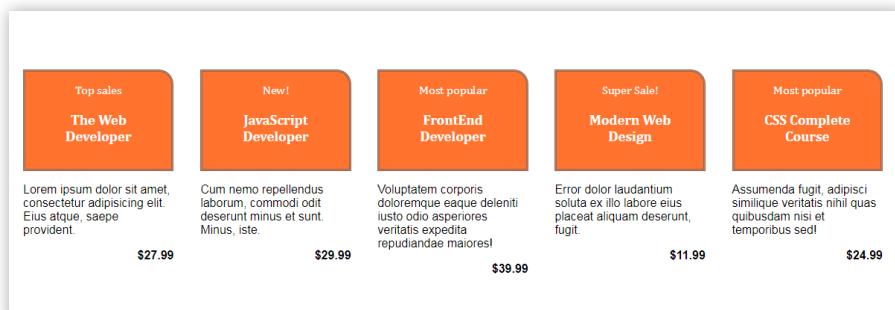


Рисунок 128

Теперь нужно разнообразить цвет фона у разных блоков. Для этого воспользуемся назначением 2-х классов для одного блока (мультиклассы). Первый блок мы не изменяем, а во втором и в последующих блоках добавим в разметку имя класса, связанное с цветом. А в CSS-стилях для каждого цветового класса запишем свойство `background-color`:

```
.red      { background-color: #900;   }
.blue    { background-color: #1b3d89; }
.violet  { background-color: #5a109f; }
.green   { background-color: #18b430; }

...
<div class="top red">
  <div class="promo">New!</div>
  <h3>JavaScript Developer</h3>
</div>

...
<div class="top blue">
  <div class="promo">Most popular</div>
  <h3>FrontEnd Developer</h3>
</div>
... и т.п.
```



Рисунок 129

Цвет фона и названия классов вы можете изменять по своему вкусу, т.к. дизайн-макета у нас нет, но принцип останется таким же. На следующем занятии мы с вами посмотрим, как можно сделать то же самое с помощью псевдоклассов :nth-child(n) или :nth-of-type(n).

Еще одной задачей является форматирование промо-текста, который в разметке представлен div-ом с классом *promo*. Для него необходимо задать желтый цвет фона, выравнивание текста по левому краю (*text-align*), прописные буквы (свойство *text-transform*), уменьшенную по сравнению с родительским контейнером ширину (*width*) и закругления для правого верхнего и нижнего углов:

```
.promo {
    width: 70%;
    background-color: #ffce00;
    text-align: left;
    text-transform: uppercase;
    border-radius: 0 15px 15px 0;
}
```

Посмотрим, что получилось:

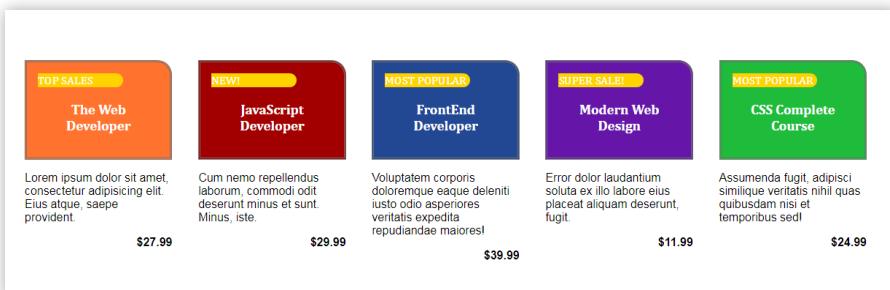


Рисунок 130

Это не совсем то, что хотелось бы получить. Во-первых, промо-текст белого цвета плохо читается на желтом фоне. Во-вторых, шрифт слишком велик — его нужно уменьшить (свойство `font-size`). В-третьих, весь желтый блок сдвинут внутрь цветного прямоугольника с классом `top`, т.к. в последнем назначены внутренние отступы (`padding: 15px;`) — это можно сделать с помощью отрицательных `margin` для класса `promo`. И, в-четвертых, текст влипает в границы блока `.promo`, а это значит, что ему не хватает внутренних отступов (`padding`). Исправим эти недостатки, добавив следующие правила в разметку:

```
.promo {
    color: #222;
    font-size: 10px;
    padding: 5px;
    padding-left: 15px;
    margin-left: -15px;
    margin-top: -5px;
}
```



Рисунок 131

Стало намного лучше. Однако уменьшение шрифта сделало текст малозаметным. Поэтому имеет смысл задать жирное начертание для текста (свойство `font-weight: bold`).

weight). В результате для небольшого по размеру элемента получим внушительное количество свойств:

```
.promo {  
    width: 70%;  
    background-color: #ffce00;  
    text-align: left;  
    text-transform: uppercase;  
    border-radius: 0 15px 15px 0;  
    color: #222;  
    font-size: 10px;  
    font-weight: bold;  
    padding: 5px;  
    padding-left: 15px;  
    margin-left: -15px;  
    margin-top: -5px;  
}
```

Готовый вариант:



Рисунок 132

Осталось сделать разметку и отформатировать подвал (footer), который будет состоять из 2-х частей: верхней со светло-серым фоном и 3-мя колонками и нижней — черной, состоящей из 2-х колонок (рис. 133).

Our Courses HTML5/CSS3 JavaScript/JQuery PHP/MySQL CMS/SEO	Our Services Group training Individual training Online training The latest techniques	Useful Links Lorem ipsum dolor sit. Odit nostrum molestias, tempore! Ea iure voluptatibus nihil. Quaerat perspiciatis non libero?
---	--	--

© 2018 | Simple Page Company ™ All rights reserved.

Рисунок 133

Разметка с помощью аббревиатуры Emmet выглядит следующим образом:

```
footer>(.container>.col*3>h2{Our Courses}+
    ul.unstyled>li*4>lorem4)++
.footer-bottom>.container>.half*2>{&copy; 2018 |
    Simple Page Company &trade;}
```

Получим футер с таким внешним видом и текстом (рис. 134):

Our Courses

- Lorem ipsum dolor sit.
- Fugit minus vitae tenetur.
- Sunt tempora ipsam facere.
- Rerum sunt suscipit officia.

Our Courses

- Lorem ipsum dolor sit.
- Libero sed, earum tempore.
- Molestiae voluptate qui, consequuntur.
- Error ipsa, impedit non!

Our Courses

- Lorem ipsum dolor sit.
- Quo, eius perferendis ad.
- Nam illum, officia nobis?
- Ipsa veritatis est debitis!

© 2018 | Simple Page Company ™ © 2018 | Simple Page Company ™

Рисунок 134

Текст в списках изменим. Также изменим текст во второй колонке в нижней части footer. Результат на скриншоте (рис. 135):

Our Courses

- HTML5/CSS3
- JavaScript/jQuery
- PHP/MySQL
- CMS/SEO

Our Services

- Group training
- Individual training
- Online training
- The latest techniques

Useful Links

- Lorem ipsum dolor sit.
- Fugit minus vitae tenetur.
- Sunt tempora ipsam facere.
- Rerum sunt suscipit officia.

© 2018 | Simple Page Company™

All rights reserved.

Рисунок 135

Для всего подвала, или футера, как его принято называть по английскому варианту наименования, необходимо задать серый цвет фона и темно-серую верхнюю рамку. Для нижней части футера — div-a с классом *footer-bottom* цвет фона будет черным, а цвет текста — белым. Также добавим для него верхнюю рамку того же цвета, что и для footer:

```
footer {  
    background-color: #ccc;  
    border-top: 2px solid #666;  
}  
  
.footer-bottom {  
    background-color: #111;  
    color: #fff;  
    border-top: 2px solid #666;  
}
```

Результат — на скриншоте (рис. 136).

<p> Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eius atque, saepe provident.</p> <p>\$27.99</p>	<p> Cum nemo repellendus laborum, commodi odit deserunt minus et sunt Minus, iste.</p> <p>\$29.99</p>	<p> Voluptatem corporis doloremque eaque deleniti iusto odio asperiores veritatis expedita repudiandae maiores!</p> <p>\$39.99</p>	<p>Error dolor laudantium soluta ex illo labore eius placat aliquam deserunt, fugit.</p> <p>\$11.99</p>	<p> Assumenda fugit, adipisci similique veritatis nihil quas quibusdam nisi et temporibus sed!</p> <p>\$24.99</p>
--	--	---	--	--

Our Courses

- HTML5/CSS3
- JavaScript/jQuery
- PHP/MySQL
- CMS/SEO

Our Services

- Group training
- Individual training
- Online training
- The latest techniques

Useful Links

- Lorem ipsum dolor sit.
- Fugit minus vitae tenetur.
- Sunt tempora ipsam facere.
- Rerum sunt suscipit officia.

© 2018 | Simple Page Company™ All rights reserved.

Рисунок 136

Каждый из трех списков, находящихся в верхней части футера, помещен в *div* с классом *col*. Этот класс предназначен для формирования колонки, которая составляет $\frac{1}{3}$ от ширины родительского контейнера (кстати, им у нас является *div* с классом *container* — это довольно распространенное название класса для такого рода элементов). Поэтому значение ширины для селектора *.col* рассчитывается, как $100\%/3 = 33.33\%$. Кроме того, нам желательно сразу учесть отступы для текста (во всех остальных случаях мы их задавали в виде свойства *padding: 1.5%*). Поэтому сразу уменьшаем ширину на удвоенное значение *padding: 33.33% — 1.5%*2 = 30.33%*. Также добавляем свойство *float: left* для того, чтобы наши колонки разместились рядом:

```
.col {
    width: 30.33%;
    padding: 1.5%;
    float: left;
}
```

Теперь мы имеем 3 колонки в верхней части футера:



Рисунок 137

Поскольку списки в колонках имеют смещение относительно заголовков (`padding-left` в 40px, назначенный в таблице стилей браузера) + ненужные нам маркеры (`list-style-type: disc`), необходимо переопределить эти стили для класса `unstyled`, указанного в нашей разметке для каждого из списков в футере. Таким образом, наши правила никак не повлияют на другие списки, которые можно добавить в другие блоки html-страницы при желании.

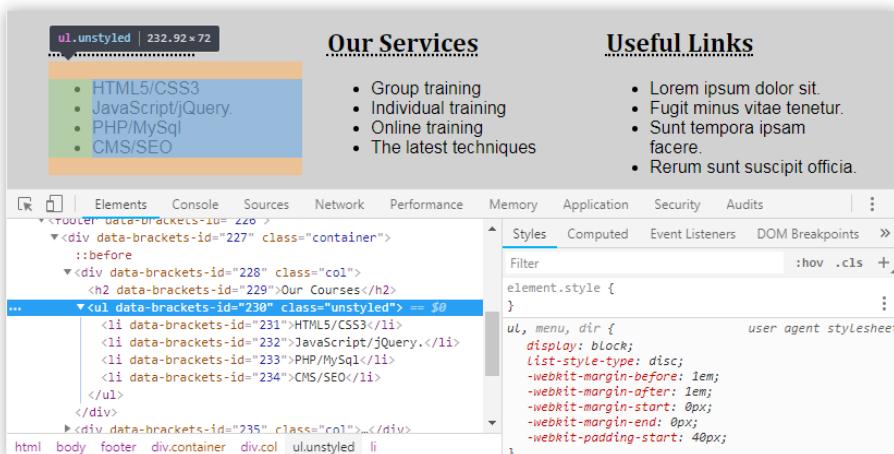


Рисунок 138

Кроме того, хотелось бы увеличить расстояние между элементами списка. Сделаем это с помощью свойства,

отвечающего за высоту строки, или межстрочное расстояние (`line-height`):

```
ul.unstyled {
    list-style-type: none;
    padding: 0;
    line-height: 170%;
}
```

В результате получим следующее форматирование внутри колонок футера:



Рисунок 139

Последнее, что нам осталось сделать — это разместить текст правой колонки в нижней части футера («`All rights reserved`») справа. Поскольку он размещен у нас в блочном элементе `<div class="half">`, для которого ранее уже были заданы CSS-правила, связанные с шириной, отступами и обтеканием текста, имеет смысл не изменять правила класса `half`, а добавить в разметку еще один класс, например `text-right` с простым правилом:

```
.text-right {
    text-align: right;
}
...
<div class="half text-right">All rights reserved.</div>
```

Теперь все в порядке:



Рисунок 140

Мы закончили форматирование футера и всей страницы. На основе тех свойств, которые были рассмотрены за 4 урока, вы уже можете сверстать симпатичную html-страницу, пусть и без графических элементов.

Sample Page
with CSS Box model
for students

About Us
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id unde mollitia quas quisquam atque molestiae eos, tempore, rerum cum dolorum, ullam aperiam rem veritatis totam, nulla hic quam. Est, iste.
Deleniti, aliquam perferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequil A, aliquid!

Our mission
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Tempore dicta, earum dolorem hic doloremque voluptates?
Sequi libero ipsum autem quos quod earum dolorem ipsa ad officiis. Aliquam hic est ipsam!
Odio molestias doloribus, explicabo? Optio, delectus. Adipisci expedita alias libero non repudiandae tempora, omnis veritatis.

Popular Courses

TOP SALES	NEW	MOST POPULAR	SUPER SALES!	MOST POPULAR
The Web Developer	JavaScript Developer	FrontEnd Developer	Modern Web Design	CSS Complete Course
\$27.99	\$29.99	\$39.99	\$11.99	\$24.99

Our Courses
HTML5/CSS3
JavaScript/JQuery
PHP/MySQL
CMS/SEO

Our Services
Group training
Individual training
Online training
The latest techniques

Useful Links
Lorem ipsum dolor sit.
Fugit minus vitae tenetur.
Sunt tempora ipsum facere.
Rerum sunt suscipit officia.

© 2018 | Simple Page Company™ All rights reserved.

Рисунок 141

Для форматирования страницы была использована так называемая «резиновая» верстка, которая предполагает изменение ширины колонок при различных разрешениях, т.к. основана на указании их CSS-свойства width в процентах. Например, при ширине окна браузера в 800px колонки изменят свои размеры в px, но сохранят свое расположение, т.е. не смеются вниз. Не слишком хорошо будут выглядеть колонки в блоке с курсами, но для их изменения уже необходимо использовать медиа-запросы.

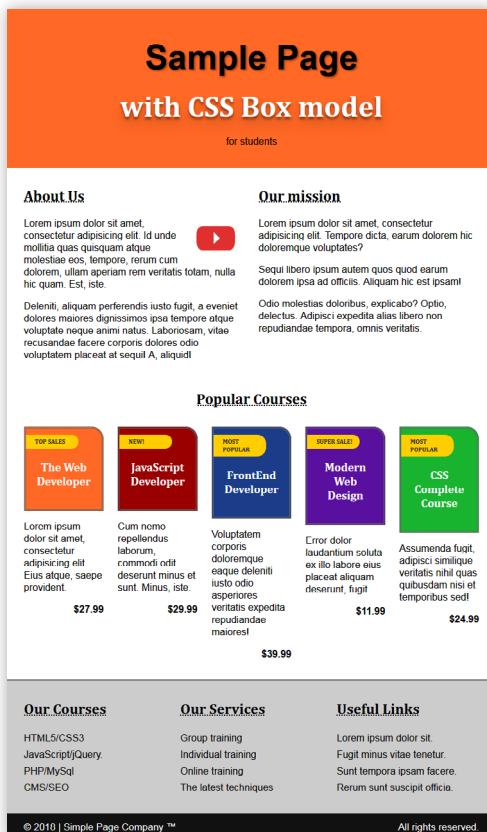


Рисунок 142

На небольших экранах будут наблюдаться проблемы с *header*, т.к. его ширина уменьшится до размера экрана, а текст выйдет за пределы внизу:

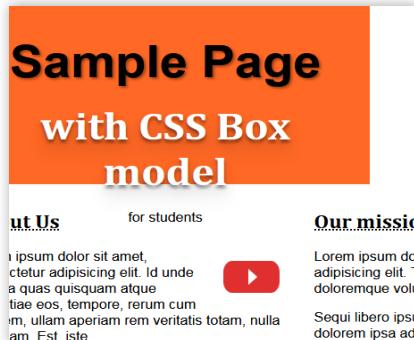


Рисунок 143

Это можно исправить, дописав в css-правила для *header* свойство `min-width: 768px`, т.к. именно такую величину мы указывали в правилах для класса *container*, который определяет размеры всех остальных блоков. Проблема будет решена, однако можно заметить, что верхний и нижний отступы в *header* неодинаковы. Поэтому нужно заменить для него свойство `height` на `min-height`:



Рисунок 144

```
header {
    background-color: #ff6825;
    text-align: center;
    min-width: 768px;
    min-height: 150px;
    padding: 8% 0;
}
```



Рисунок 145

Теперь *header* выглядит вполне приемлемо даже на маленьких экранах, хотя на странице появилась не только вертикальная, но и горизонтальная полоса прокрутки. Убрать ее можно только используя медиа-запросы для перестройки внешнего вида страницы на небольших экранах.

Также хотелось бы отметить важность использования правил из класса *clearfix*, которые в данном примере были записаны для класса *container*, чтобы сократить количество используемых в разметке классов.

Если эти правила закомментировать, то верстка страницы «развалится»:

```
/* .container::before,  
.container::after {  
    content: "";  
    display: table;  
}  
  
.container::after {  
    clear: both  
} */
```

На рисунке 146 видно, что footer потерял цвет фона, его рамка (border-top) сместилась под съехавший заголовок Popular Courses. Колонки футера также расположились в 2 ряда, т.к. первая подчинилась правилу обтекания `float: left`, связанному с колонкой курса «FrontEnd Developer». Нижняя часть футера теперь не видна, т.к. белый текст на белом фоне не читается. *Вывод: при использовании в верстке плавающих элементов, для которых назначено свойство float, отмена обтекания — практически 100% необходимая операция.* Для форматирования страницы в виде колонок лучше всего с этим справляется класс `clearfix` с правилами для псевдоэлементов `:before` и `:after`, т.к. не требует добавлений в разметку никаких лишних элементов. В нашем примере можно было также для класса `container` использовать свойство `overflow` со значением `hidden` или `auto`.

Sample Page with CSS Box model

for students

About Us

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id unde mollitia quas quisquam atque molestiae eos, tempore, rerum cum dolorem, ullam aperiam rem veritatis totam, nulla hic quam. Est, iste.

Deleniti, aliquam perferendis iusto fugit, a eveniet dolores maiores dignissimos ipsa tempore atque voluptate neque animi natus. Laboriosam, vitae recusandae facere corporis dolores odio voluptatem placeat at sequil A, aliquid!

Our mission

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Tempore dicta, earum lorem hic doloremque voluptates?

Sequi libero ipsum autem quod earum dolorem ipsa ad officiis. Aliquam hic est ipsam!

Odio molestias doloribus, explicabo? Optio, delectus. Adipisci expedita alias libero non repudiandae tempora, omnis veritatis.

Popular Courses

<p>The Web Developer <small>TOP SALES</small></p> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Etius atque, saepe provident.</p> <p>\$27.99</p>	<p>JavaScript Developer <small>NEW!</small></p> <p>Cum nemo repellendus laborum, commodi odit deserunt minus et sunt. Minus, iste.</p> <p>\$29.99</p>	<p>FrontEnd Developer <small>MOST POPULAR</small></p> <p>Voluptatem corporis doloremque eaque deleniti iusto odio asperiores veritatis expedita repudiandae maiores!</p> <p>\$39.99</p>	<p>Modern Web Design <small>SUPER SALE!</small></p> <p>Error dolor laudantium soluta ex illo labore eius placeat aliquam deserunt, fugit.</p> <p>\$11.99</p>	<p>CSS Complete Course <small>MOST POPULAR</small></p> <p>Assumenda fugit, adipisci similiqne veritatis nihil quas quibusdam nisi et temporibus sed!</p> <p>\$24.99</p>
--	--	--	---	--

Our Courses

HTML5/CSS3
JavaScript/Query
PHP/MySQL
CMS/SEO

Our Services

Group training
Individual training
Online training
The latest techniques

Useful Links

Lorem ipsum dolor sit.
Fugit minus vitae tenetur.
Sunt tempora ipsum facere.
Rerum sunt suscipit officia.

Рисунок 146

Домашнее задание

Дома вам необходимо будет использовать ряд CSS-свойств, с которыми вы познакомились на этом и на предыдущем занятии, для создания 4-х плавающих блоков с текстом.



Рисунок 147

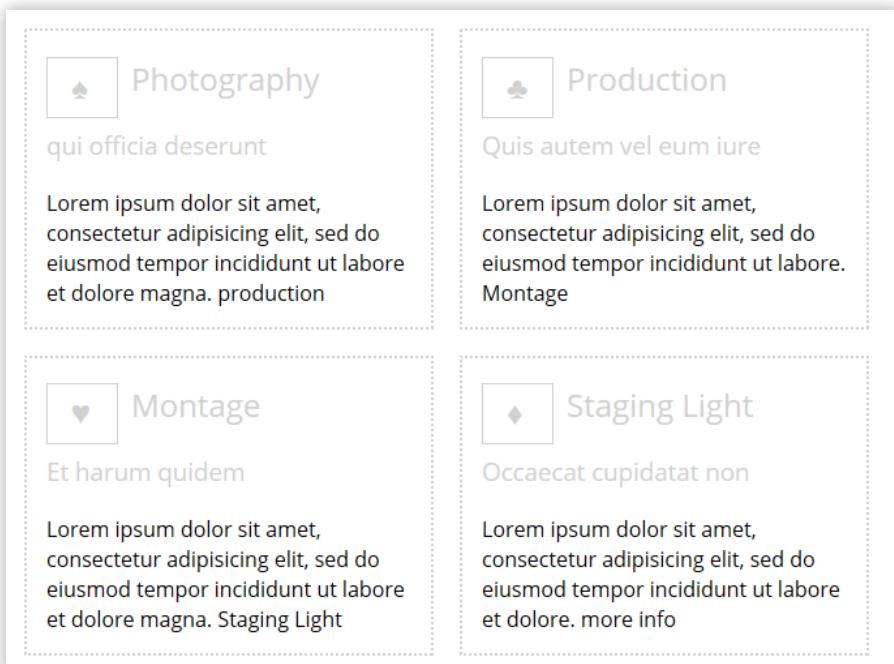


Рисунок 148

Подберите ширину блоков таким образом, чтобы в браузере, открытом на всю ширину экрана, они располагались бы в одну линию, а при уменьшении размера браузера примерно до половины ширины экрана, блоки перестраивались бы в 2 линии, размещаясь друг под другом (рис. 148).

Масти рядом с заголовком нужно записать в виде мнемокода (html-entities, или символьные подстановки типа ♠)

Обратите внимание на семейство шрифта и цвет основного текста и заголовков, а также на наличие внешних и внутренних отступов и рамок.

Текст и изображения с внешним видом вы найдете в папке HW4 этого урока.



Урок 6.1. Блочная модель элементов

© Слуцкая Елена.
© STEP IT Academy, www.itstep.org.

All rights to protected pictures, audio, and video belong to their authors or legal owners.

Fragments of works are used exclusively in illustration purposes to the extent justified by the purpose as part of an educational process and for educational purposes in accordance with Article 1273 Sec. 4 of the Civil Code of the Russian Federation and Articles 21 and 23 of the Law of Ukraine "On Copyright and Related Rights". The extent and method of cited works are in conformity with the standards, do not conflict with a normal exploitation of the work, and do not prejudice the legitimate interests of the authors and rightholders. Cited fragments of works can be replaced with alternative, non-protected analogs, and as such correspond the criteria of fair use.

All rights reserved. Any reproduction, in whole or in part, is prohibited. Agreement of the use of works and their fragments is carried out with the authors and other right owners. Materials from this document can be used only with resource link.

Liability for unauthorized copying and commercial use of materials is defined according to the current legislation of Ukraine.