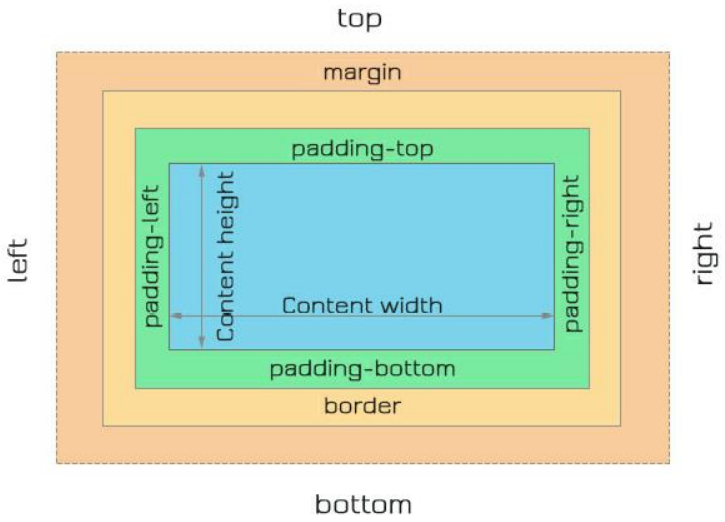
Верстка web-страниц блоками

Модуль 6 (2 пары)

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

- 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 некий контент, т.е. текст, изображения, видео и т.п. Поэтому первым важным свойством для него будет ширина — width.

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 все равно в пикселях относительно размера родительского элемента. Например, если в элементе main (основное содержимое страницы) задать

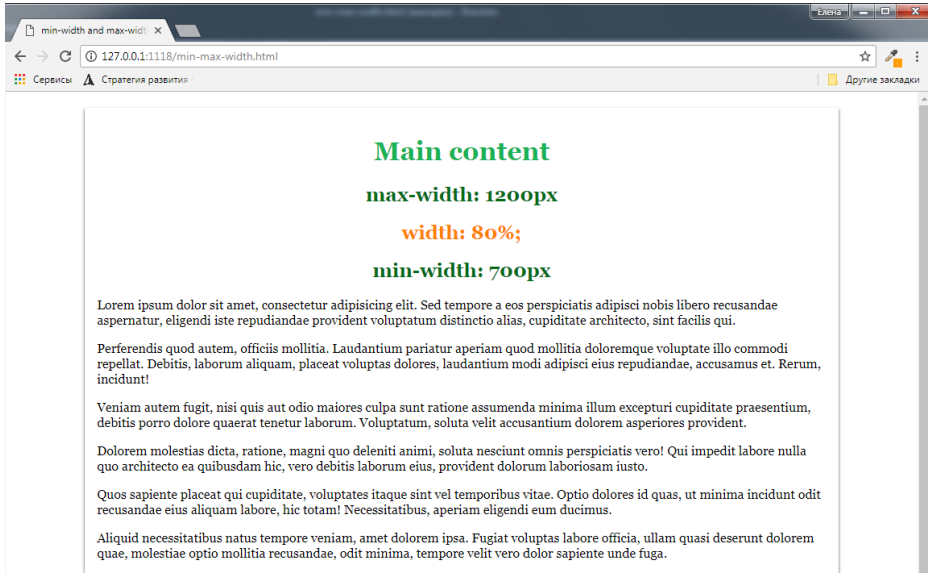
```
main { width: 80%; }
```

- то при ширине окна браузера в 1300px ширина элемента main будет 1040px (1340px*80/100 = 1040px), а при уменьшении окна браузера до 600px ширина этого элемента составит уже 480px (600px*80/100 = 480px).
- 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:

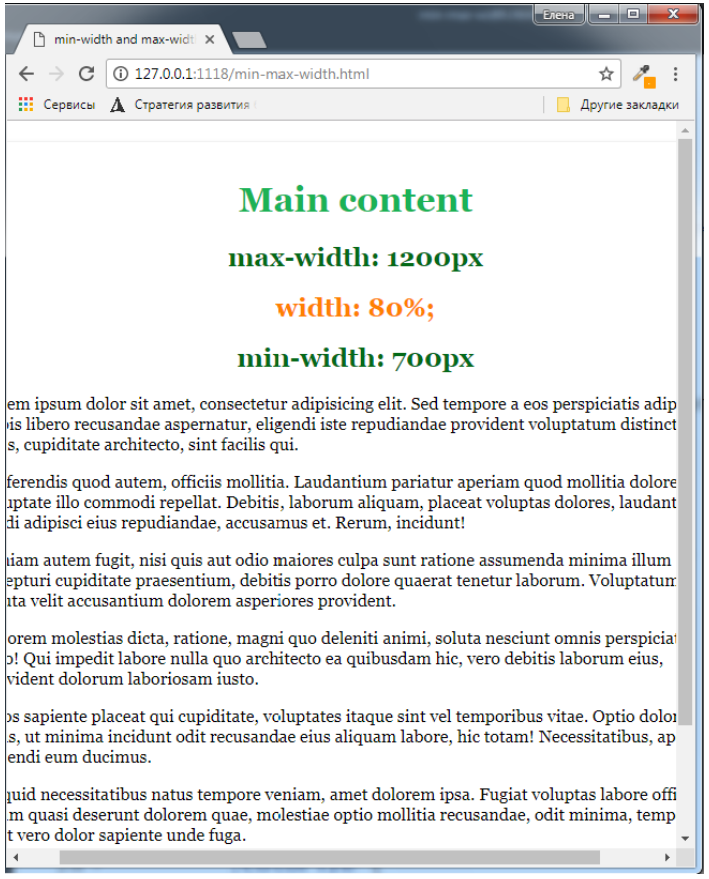
```
max-width: размер в px, pt, em, in, % и др. единицах |
none | inherit min-width: размер в px, pt, em, in, %
и др. единицах | inherit
```

Ширина элемента

- Для min-width значение по умолчанию 0, для maxwidth — none



- 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

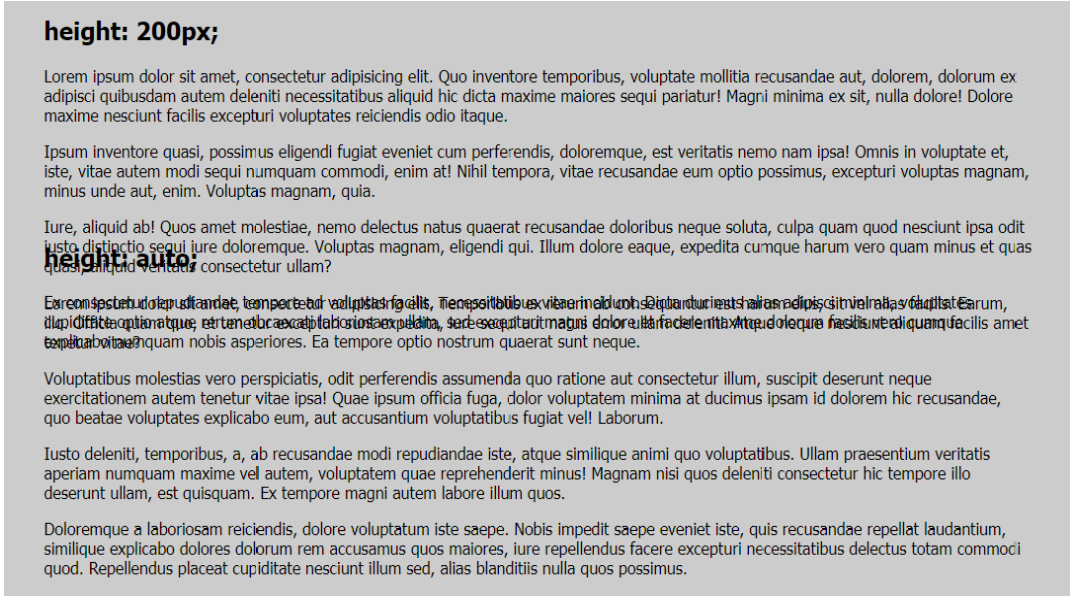


Высота элемента

- 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
```

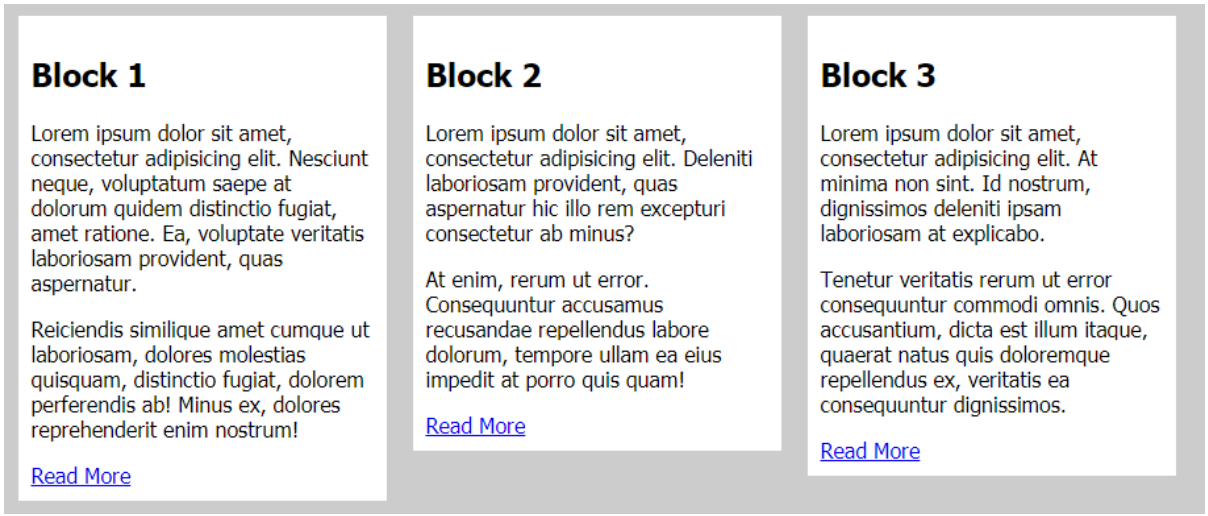
- Для min-height значение по умолчанию 0, для maxheight — none.
- Если значение свойства height будет меньше, чем его содержимое, то содержимое будет выходить за пределы своего контейнера и наползать на текст, идущий следом.
- В примере верхнему элементу div назначен класс fixed, для которого задана высота height: 200px. Текст в этом div не помещается в эти 200px, поэтому накладывается на текст следующего div-а.



Высота элемента

- Второй пример демонстрирует разницу между 2-мя рядом расположенными 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

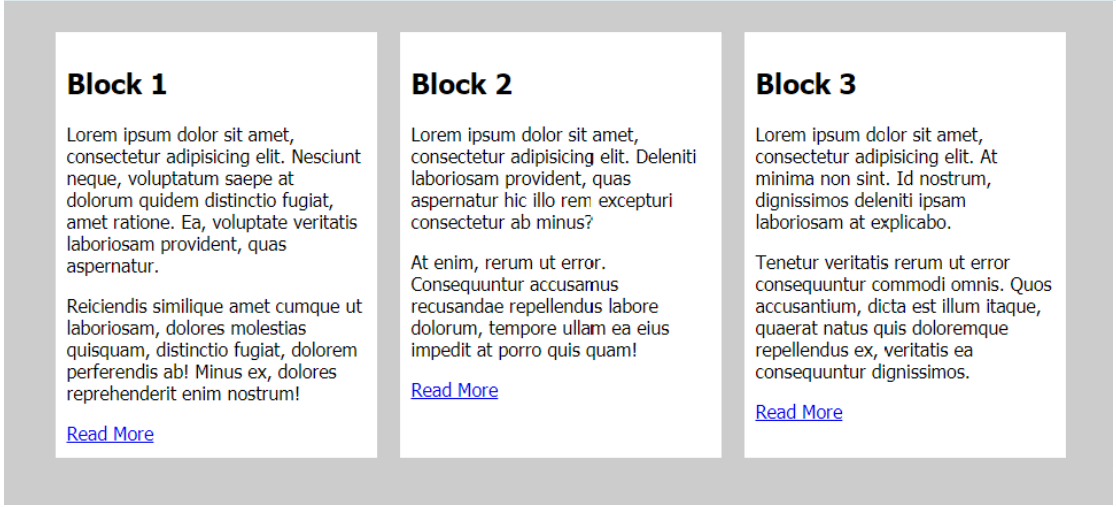
- В этом примере назначение height явно выглядит некрасиво. Это получилось потому, что высота блока имеет слишком маленькую величину. Если указать достаточное значение, явно заданное свойство height может улучшить внешний вид элементов.
- Предположим, у нас есть 3 блока с разным количеством контента в каждом из них



Высота элемента

- 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;
```

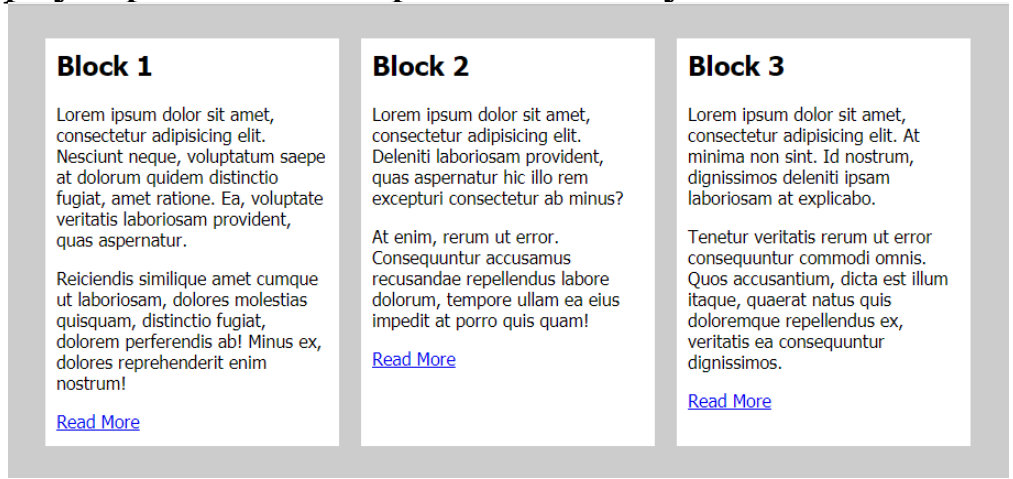


- Рассмотрим еще один вариант. Предположим, мы не можем задать для 3-х рядом расположенных блоков фиксированную ширину в px, как это было в примере (см. скриншот выше). Ширина должна быть в %, т.к. родительский элемент также имеет ширину в % относительно body

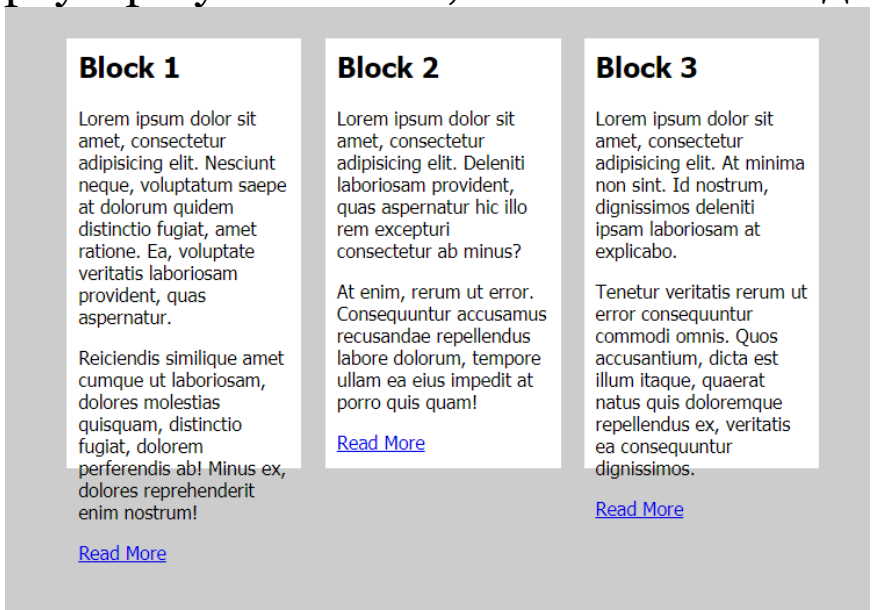
```
.wrap {
    width: 90%;
    margin: 20px auto;
}
.box {
    width: 33.3%;
    float: left;
}
```

Высота элемента

- 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



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



Высота элемента

Здесь нам понадобится свойство min-height:

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

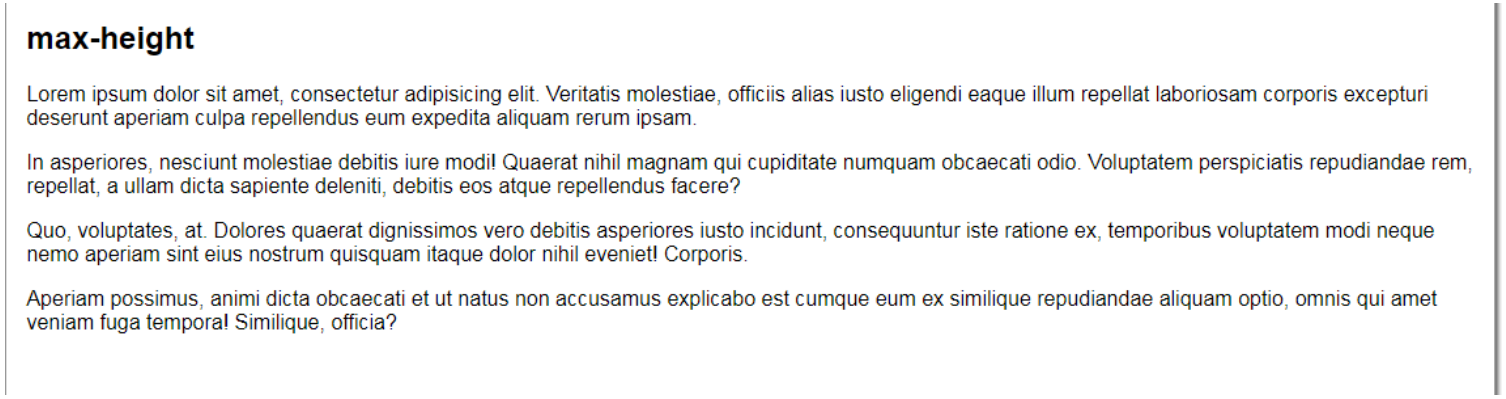
Вы не заметите разницы между height и min-height на широком экране. При уменьшении ширины браузера высота блоков будет интерпретироваться как auto, но не будет меньше указанных 350px. Блоки будут иметь рваные края — это можно исправить только с помощью медиа-запросов. Однако текст будет находиться в пределах блока на белом фоне, а не за его пределами

Высота элемента

- 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 на широком экране свойство height: 320px увеличивает пустое пространство после последнего абзаца



Высота элемента

- 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.

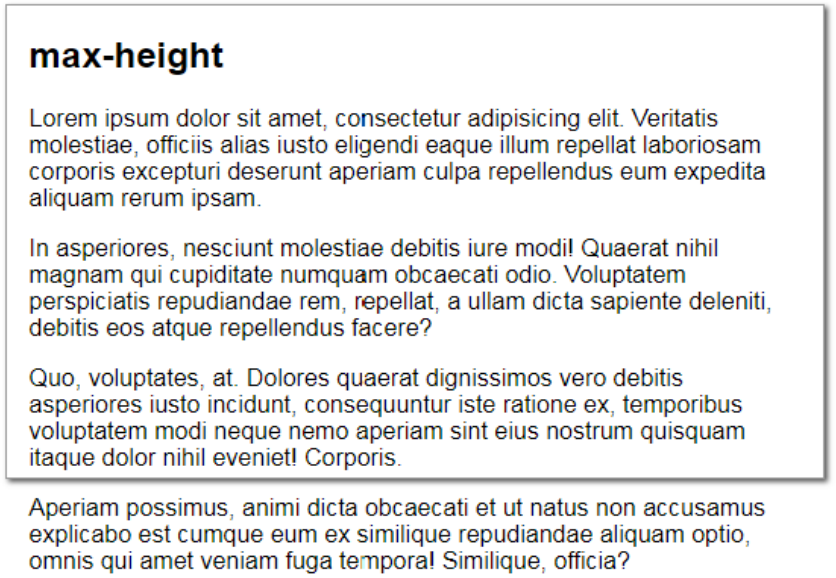
In asperiores, nesciunt molestiae debitis iure modi! Quaerat nihil magnam qui cupiditate numquam obcaecati odio. Voluptatem perspiciatis repudiandae rem, repellat, a ullam dicta sapiente deleniti, debitis eos atque repellendus facere?

Quo, voluptates, at. Dolores quaerat dignissimos vero debitis asperiores iusto incidunt, 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 tempora! Similique, officia?

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

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



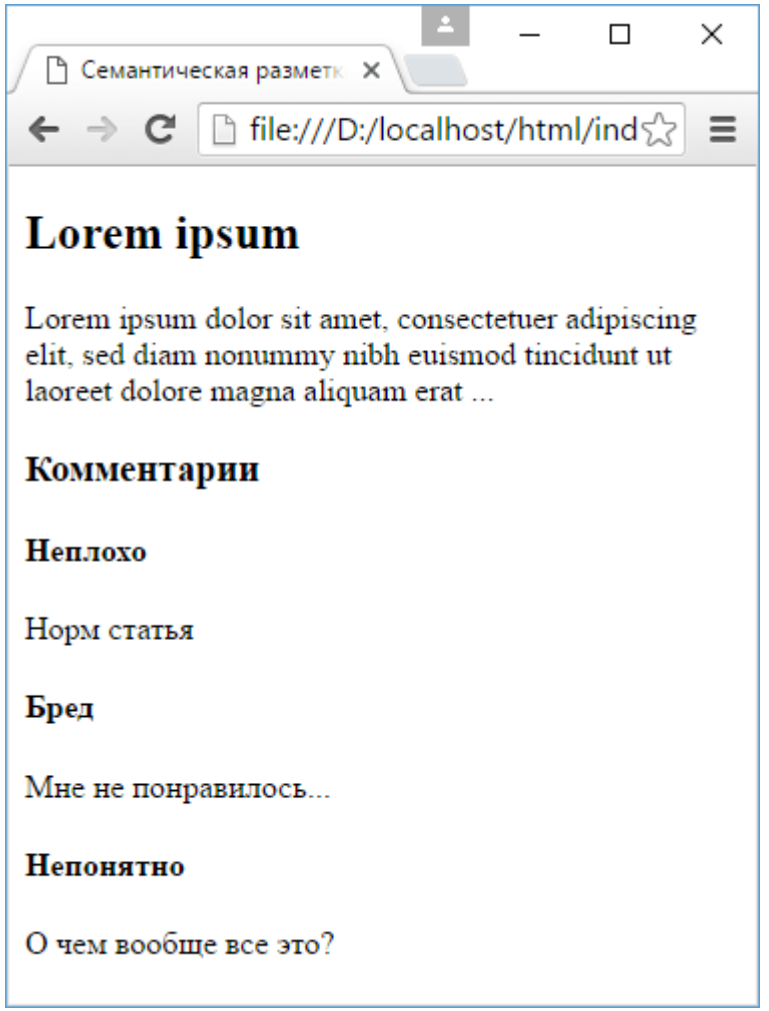
Семантическая структура страницы

Элемент article

- Элемент article представляет целостный блок информации на странице, который может рассматриваться отдельно и использоваться независимо от других блоков. Например, это может быть пост на форуме или статья в блоге, онлайн-журнале, комментарий пользователя.
- Один элемент article может включать несколько элементов article. Например, мы можем заключить в элемент article всю статью в блоге, и этот элемент будет содержать другие элементы article, которые представляют комментарии к этой статье в блоге. То есть статья в блоге может рассматриваться нами как отдельная семантическая единица, и в то же время комментарии также могут рассматривать отдельно вне зависимости от другого содержимого.
- Здесь вся статья может быть помещена в элемент article, и в то же время каждый отдельный комментарий также представляет отдельный элемент article.
- При использовании article надо учитывать, что каждый элемент article должен быть идентифицирован с помощью включения в него заголовка h1-h6.

Элемент article

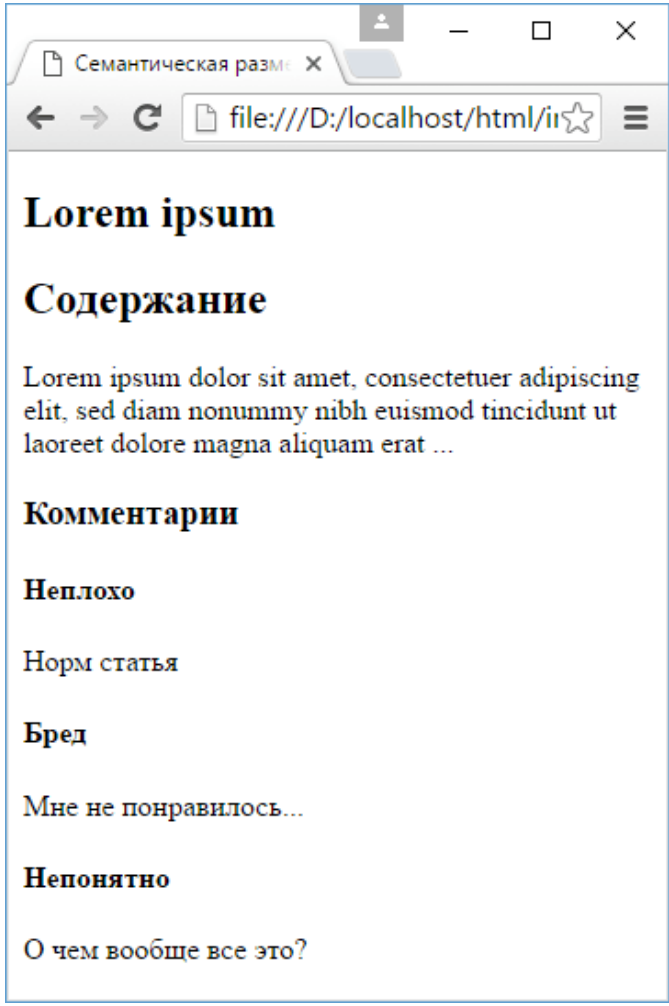
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Семантическая разметка в HTML5</title>
    </head>
    <body>
        <article>
            <h2>Lorem ipsum</h2>
            <div>
                Lorem ipsum dolor sit amet, consectetuer
adipiscing elit, sed diam nonummy nibh
                euismod tincidunt ut laoreet dolore magna
aliquam erat ...
            </div>
            <div>
                <h3>Комментарии</h3>
                <article>
                    <h4>Неплохо</h4>
                    <p>Норм статья</p>
                </article>
                <article>
                    <h4>Бред</h4>
                    <p>Мне не понравилось...</p>
                </article>
                <article>
                    <h4>Непонятно</h4>
                    <p>О чем вообще все это?</p>
                </article>
            </div>
        </article>
    </body>
</html>
```



Элемент section

- Элемент section объединяет связанные между собой куски информации html-документа, выполняя их группировку. Например, section может включать набор вкладок на странице, новости, объединенные по категории и т.д.
- Каждый элемент section должен быть идентифицирован с помощью заголовка h1-h6.
- При этом элемент section может содержать несколько элементов article, выполняя их группировку, так и один элемент article может содержать несколько элементов section.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Семантическая разметка в HTML5</title>
    </head>
    <body>
        <article>
            <h1>Lorem ipsum</h1>
            <section>
                <h2>Содержание</h2>
                <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed
diam nonummy nibh
                euismod tincidunt ut laoreet dolore magna aliquam erat ...</p>
            </section>
            <section>
                <h3>Комментарии</h3>
                <article>
                    <h4>Неплохо</h4>
                    <p>Норм статья</p>
                </article>
                <article>
                    <h4>Бред</h4>
                    <p>Мне не понравилось...</p>
                </article>
                <article>
                    <h4>Непонятно</h4>
                    <p>О чем вообще все это?</p>
                </article>
            </section>
        </article>
    </body>
</html>
```



Элемент nav

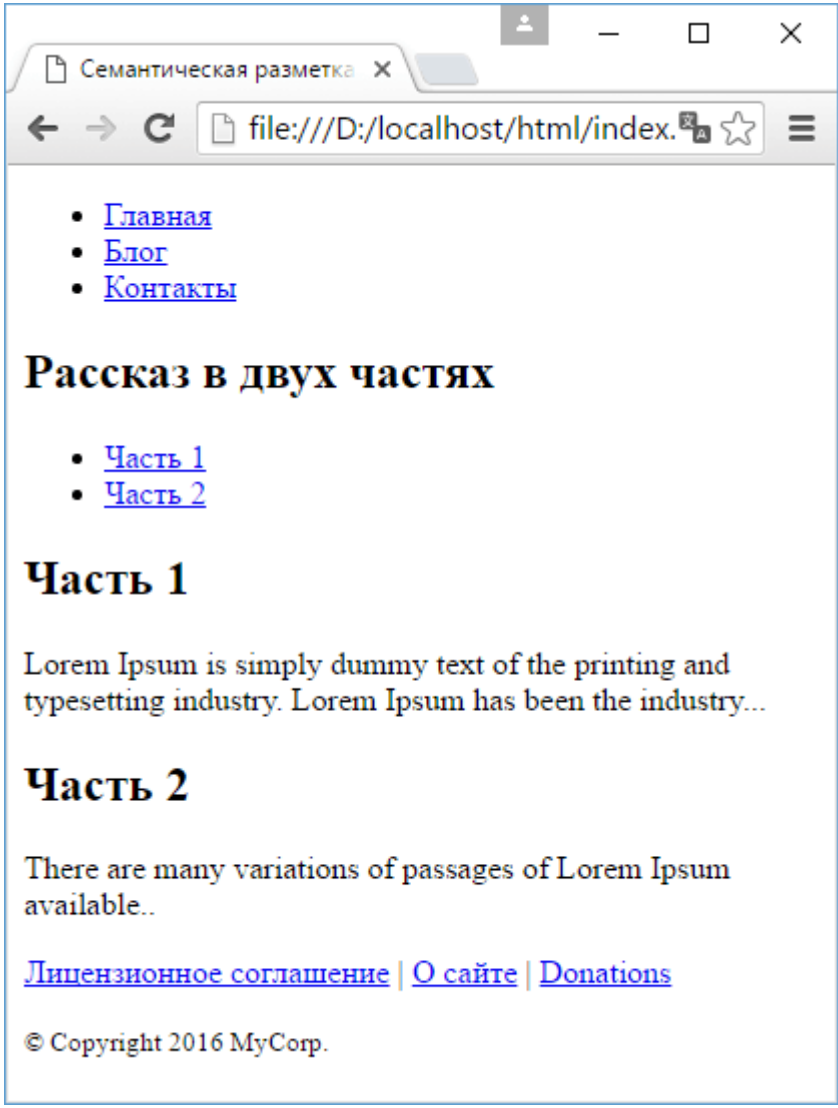
- Элемент nav призван содержать элементы навигации по сайту. Как правило, это ненумерованный список с набором ссылок.
- На одной веб-странице можно использовать несколько элементов nav. Например, один элемент навигации для перехода по страницам на сайте, а другой - для перехода внутри html-документа.
- Не все ссылки обязательно помещать в элемент nav. Например, некоторые ссылки могут не представлять связанного блока навигации, например, ссылка на главную страницу, на лицензионное соглашение по поводу использования сервиса и подобные ссылки, которые часто помещаются внизу страницы. Как правило, их достаточно определить в элементе footer, а элемент nav для них использовать необязательно.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Семантическая разметка в HTML5</title>
    </head>
    <body>
        <nav>
            <ul>
                <li><a href="/">Главная</a></li>
                <li><a href="/blog">Блог</a></li>
                <li><a href="/contacts">Контакты</a></li>
            </ul>
        </nav>
        <article>
            <header>
                <h2>Рассказ в двух частях</h2>
            </header>
            <nav>
                <ul>
                    <li><a href="#part1">Часть 1</a></li>
                    <li><a href="#part2">Часть 2</a></li>
                </ul>
            </nav>
```

Элемент nav

```
<div>
                <section id="part1">
                    <h2>Часть 1</h2>
                    <p>Lorem Ipsum is simply dummy
text of the printing and typesetting industry.
                    Lorem Ipsum has been the
industry...</p>
                </section>
                <section id="part2">
                    <h2>Часть 2</h2>
                    <p>There are many variations of
passages of Lorem Ipsum available..</p>
                </section>
            </div>
            <footer>

            </footer>
        </article>
        <footer>
            <p><a href="/license">Лицензионное
соглашение</a> |
            <a href="/about">О сайте</a> |
            <a href="/donation">Donations</a></p>
            <p><small>© Copyright 2016
MyCorp.</small></p>
        </footer>
    </body>
</html>
```



Элемент header

- Элемент header является как бы вводным элементом, предваряющим основное содержимое. Здесь могут быть заголовки, элементы навигации или какие-либо другие вспомогательные элементы, например, логотип, форма поиска и т.п. Например:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Семантическая разметка в HTML5</title>
    </head>
    <body>
        <header>
            <h1>Онлайн-магазин телефонов</h1>
                <nav>
                    <ul>
                        <li><a href="/apple">Apple</a>
                        <li><a href="/microsoft">Microsoft</a>
                        <li><a href="/samsung">Samsung</a>
                    </ul>
                </nav>
        </header>
        <div>
            Информация о новинках мобильного мира....
        </div>
    </body>
</html>
```

- Элемент header нельзя помещать в такие элементы как address, footer или другой header.

Элемент Footer

- Элемент footer обычно содержит информацию о том, кто автор контента на веб-странице, копирайт, дата публикации, блок ссылок на похожие ресурсы и т.д. Как правило, подобная информация располагается в конце веб-страницы или основного содержимого, однако, footer не имеет четкой привязки к позиции и может использоваться в различных местах веб-страницы.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Семантическая разметка в HTML5</title>
    </head>
    <body>
        <h1>Xiaomi Mi 5</h1>
        <div>
            Xiaomi Mi 5 оснащен восьмиядерным процессором Qualcomm Snapdragon 820.
            Размер внутреннего хранилища - 32 и 64 МБ.
        </div>
        <footer>
            <p><a href="/license">Лицензионное соглашение</a><br/>
            Copyright © 2016. SomeSite.com</p>
        </footer>
    </body>
</html>
```

- Здесь определен футер для всей веб-страницы. В него помещена ссылка на лицензионное соглашение использования сервисом и информация о копирайте.

Элемент Footer

- Футер необязательно должен быть определен для всей страницы. Это может быть и отдельная секция контента:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Семантическая разметка в HTML5</title>
    </head>
    <body>
        <section>
            <h1>Последние статьи</h1>
            <article>
                <h2>Анонс Samsung Galaxy S7</h2>
                <p>Состоялся выход нового флагмана от компании Samsung Galaxy S7.....</p>
                <footer>
                    Дата публикации: <time datetime="2016-03-16T15:16-00:00">16.03.2016 15:16</TIME>
                </footer>
            </article>
            <article>
                <h2>Скидки на Microsoft Lumia 950</h2>
                <p>С 1 марта смартфон Microsoft Lumia 950 стоит на 10 000 рублей дешевле</p>
                <footer>
                    Дата публикации: <time datetime="2016-03-01T14:36-00:00">01.03.2016 14:36</TIME>
                </footer>
            </article>
        </section>
    </body>
</html>
```

- Элемент footer не следует помещать в такие элементы как address, header или другой footer.

Элемент Address

- Элемент address предназначен для отображения контактной информации, которая связана с ближайшим элементом article или body. Нередко данный элемент размещается в футере:

```
<footer>
    <address>
        Контакты для связи <a href="mailto:js@example.com">Том Смит</a>.
    </address>
    <p>© copyright 2016 Example Corp.</p>
</footer>
```

Элемент aside

- Элемент aside представляет содержимое, которое косвенно связано с остальным контентом веб-станицы и которое может рассматриваться независимо от него. Данный элемент можно использовать, например, для сайдбаров, для рекламных блоков, блоков навигационных элементов, различных плагинов типа твиттера или фейсбука и т.д.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Семантическая разметка в HTML5</title>
    </head>
    <body>
        <aside style="float:right; width:200px;">
            <h2>Скидки на Microsoft Lumia 950</h2>
            <p>Только до 31 марта смартфон Microsoft Lumia 950 стоит на 10 000 рублей
дешевле. В
                подарок вы получите бесплатный чупа-чупс. <a
href="buy/id=3">Купить</a></p>
        </aside>
        <article>
            <h2>Релиз Samsung Galaxy S7</h2>
            <p>Состоялся выход нового флагмана от компании Samsung Galaxt S7. Вместе с
новым флагманом компания
            Samsung представила новый шлем виртуальной реальности Gear VR...</p>
        </article>
    </body>
</html>
```

Элемент aside

- Здесь содержимое блока aside довольно косвенно связано с основным контентом из элемента article. Поэтому все это содержимое мы можем поместить в aside.

Элемент main

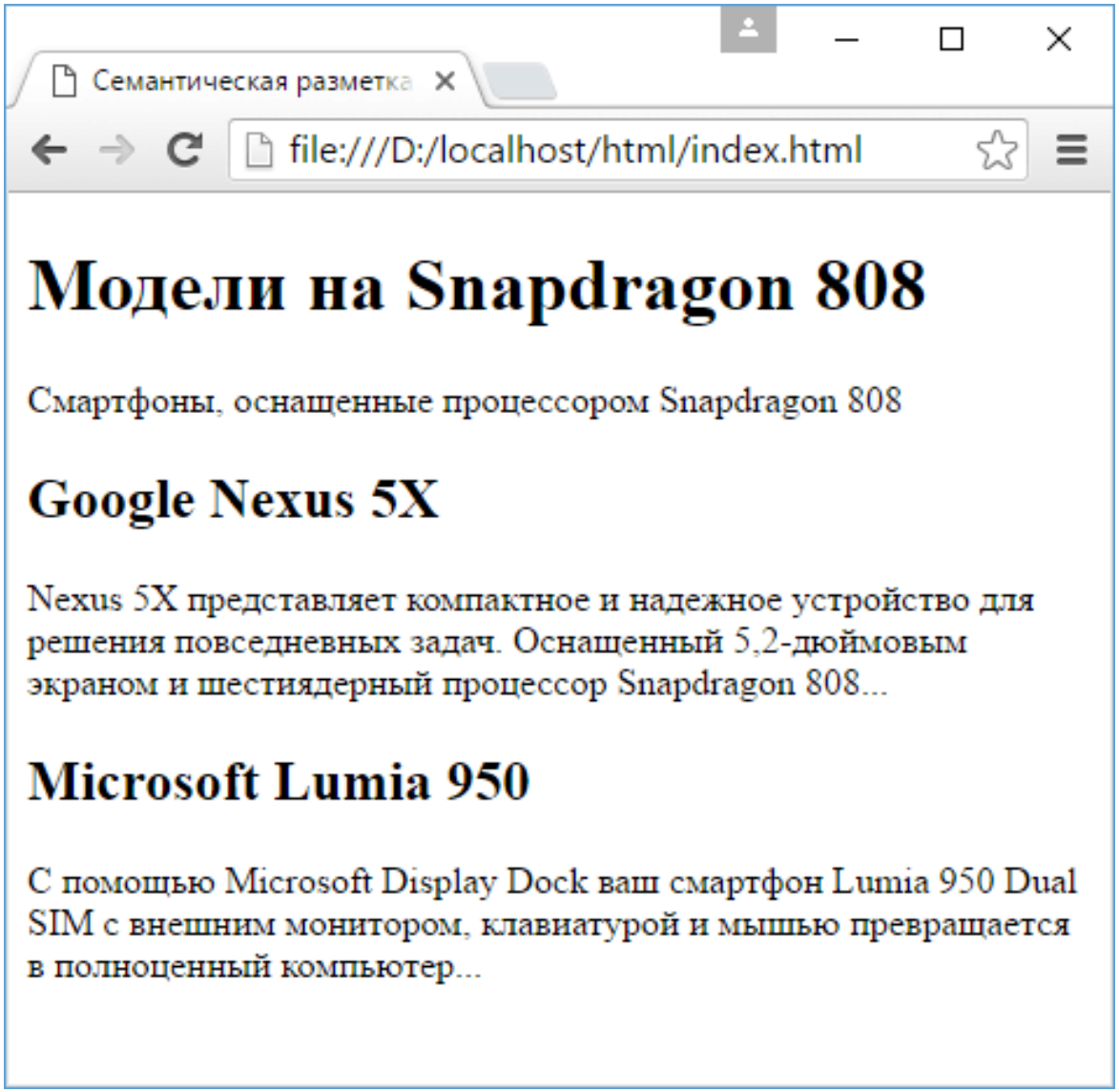
- Элемент main представляет основное содержимое веб-страницы. Он представляет уникальный контент, в который не следует включать повторяющиеся на разных веб-страницах элементы сайдбаров, навигационные ссылки, информацию о копирайте, логотипы и тому подобное.
- Используем элемент main:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Семантическая разметка в HTML5</title>
    </head>
    <body>
        <main>
          <h1>Модели на Snapdragon 808</h1>
          <p>Смартфоны, оснащенные процессором Snapdragon 808</p>

          <article>
            <h2>Google Nexus 5X</h2>
            <p>Nexus 5X представляет компактное и надежное устройство для решения повседневных задач.
            Оснащенный 5,2-дюймовым экраном и шестиядерный процессор Snapdragon 808...</p>
          </article>

          <article>
            <h2>Microsoft Lumia 950</h2>
            <p>С помощью Microsoft Display Dock ваш смартфон Lumia 950 Dual SIM с внешним монитором,
                клавиатурой и мышью превращается в полноценный компьютер...</p>
          </article>
        </main>
    </body>
</html>
```

Элемент main



Элемент main

- Не стоит думать, что абсолютно все содержимое надо обязательно помещать в элемент main. Нет мы также можем использовать вне его другие элементы, например, header и footer:

```
<body>
    <header>
        ..............
    </header>
    <main>
        ..................
    </main>
    <footer>
        ..................
    </footer>
</body>
```

- Однако надо помнить, что элемент main не может быть вложенным в такие элементы, как article, aside, footer, header, nav. Кроме того, на веб-странице допустимо наличие только одного элемента main.
- Также стоит отметить, что на данный момент есть небольшие проблемы с поддержкой этого элемента в браузерах. В частности, IE 11 не поддерживает данный элемент (в остальных браузерах полная поддержка), поэтому в этом случае стоит использовать атрибут роли:

```
<main role="main">
  ...
</main>
```

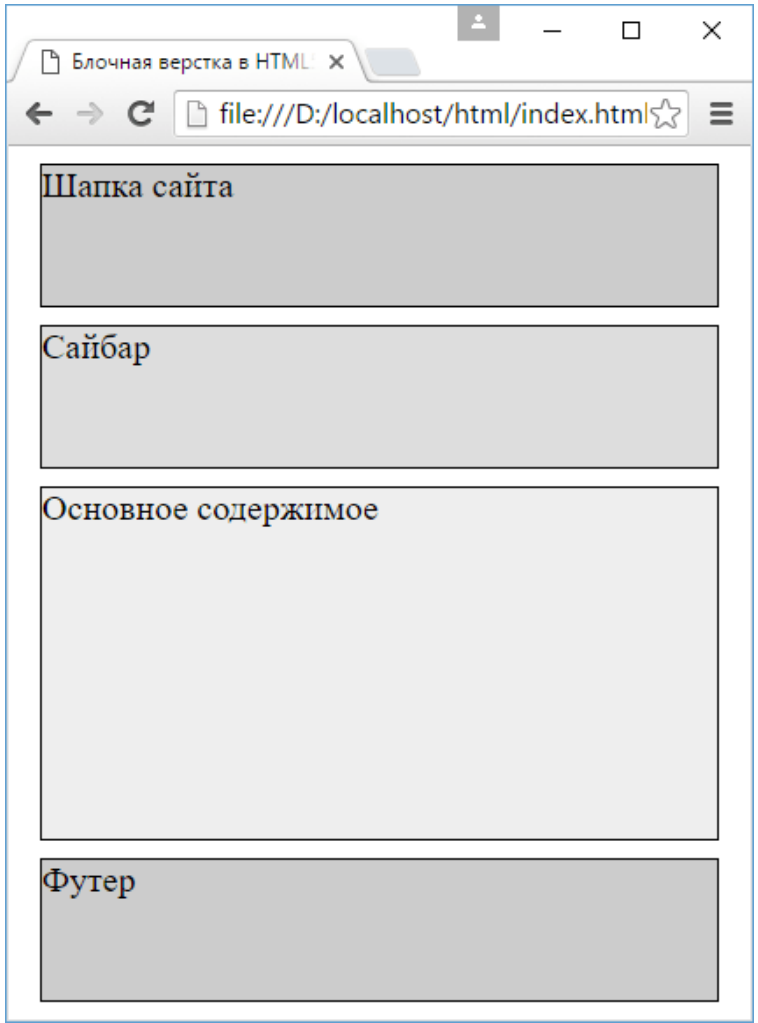
Блочная верстка

Блочная верстка

- Как правило, веб-страница состоит из множества различных элементов, которые могут иметь сложную структуру. Поэтому при создании веб-страницы возникает необходимость нужным образом позиционировать эти элементы, стилизовать их так, чтобы они располагались на странице нужным образом. То есть возникает вопрос создания макета страницы, ее верстки.
- Существуют различные способы, стратегии и виды верстки. Изначально распространенной была верстка на основе таблиц. Так как таблицы позволяет при необходимости очень легко и просто разделить все пространство веб-страницы на строки и столбцы. Строками и столбцами довольно легко управлять, в них легко позиционировать любое содержимое. Именно это и определило популярность табличной верстки.
- Однако табличная верстка создает не самые гибкие по дизайну страницы, что является особенно актуальным аспектом в мире, где нет одного единственного разрешения экрана, за то есть большие экраны на телевизорах, малые экраны на планшетах и фаблетах, очень маленькие экраны на смартфонах и т.д. Все это многообразие экранов табличная верстка оказалась не в состоянии удовлетворить. Поэтому постепенно ей на смену пришла блочная верстка. Блочная верстка - это отосительно условное название способов и приемов верстки, когда в большинстве веб-страниц для разметки используется CSS-свойство float, а основным строительным элементов веб-страниц является элемент , то есть по сути блок. Используя свойство float и элементы div или другие элементы, можно создать структуру страницы из нескольких столбцов, как при табличной верстке, которая будет значительно гибче.
- Ранее в одной из прошлых тем рассматривалось действие свойства float. Теперь используем его для создания двухколоночной веб-страницы. Допустим, вверху и внизу у нас будут стандартно шапка и футер, а в центре - две колонки: колонка с меню или сайдбар и колонка с основным содержимым.

Класс clearfix

- В начале определим все блоки. При работе с элементами, которые используют обтекание и свойство float, важен их порядок. Так, код плавающего элемента, у которого устанавливается свойство float, должен идти перед элементом, который обтекает плавающий элемент. То есть блок сайдбара будет идти до блока основного содержимого:

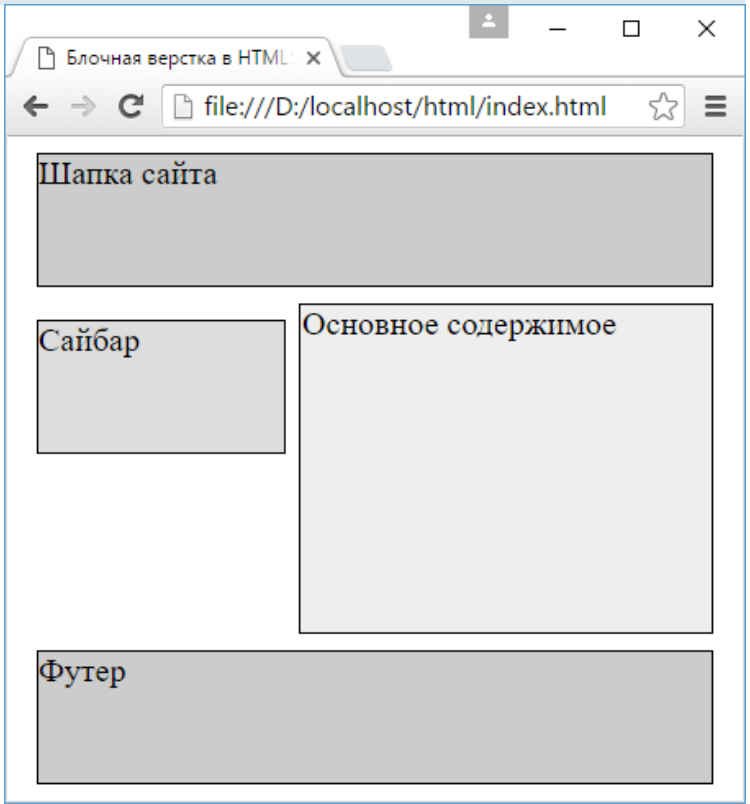



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Блочная верстка в HTML5</title>
        <style>
            div{
                margin: 10px;
                border: 1px solid black;
                font-size: 20px;
                height: 80px;
            }
            #header{
                background-color: #ccc;
            }
            #sidebar{
                background-color: #ddd;
            }
            #main{
                background-color: #eee;
                height: 200px;
            }
            #footer{
                background-color: #ccc;
            }
        </style>
    </head>
    <body>
        <div id="header">Шапка сайта</div>
        <div id="sidebar">Сайбар</div>
        <div id="main">Основное содержимое</div>
        <div id="footer">Футер</div>
    </body>
</html>
```

- Высота, граница и отступы блоков в данном случае добавлены только для красоты, чтобы идентифицировать пространство блока и отделять его от других.
- Далее, чтобы переместить блок сайдбара влево по отношению к блоку основного содержимого и получить эффект обтекания, нам надо указать у блока сайдбара свойство float: left и предпочтительную ширину. Ширина может быть фиксированной, например, 150 px или 8 em. Либо также можно использовать проценты, например, 30% - 30% от ширины контейнера body. С одной стороны, блоками с фиксированной шириной легче управлять, но с другой процентные значения ширины позволяют создавать более гибкие, резиновые блоки, которые изменяют размеры при изменении размеров окна браузера.
- Последним шагом является установка отступа блока с основным содержимым от блока сайдбара. Поскольку при обтекании обтекающий блок может обтекать плавающий элемент и справа и снизу, если плавающий элемент имеет меньшую высоту, то нам надо установить отступ, как минимум равный ширине плавающего элемента. Например, если ширина сайдбара равна 150px, то для блока основного содержимого можно задать отступ в 170px, что позволит создать пустое пространство между двумя блоками.
- При этом не стоит у блока основного содержимого указывать явным образом ширину, так как браузеры расширяют его автоматически, чтобы он занимал все доступное место.

- Итак, принимая во внимание все выше сказанное, изменим стили блоков сайдбара и основного содержимого следующим образом:

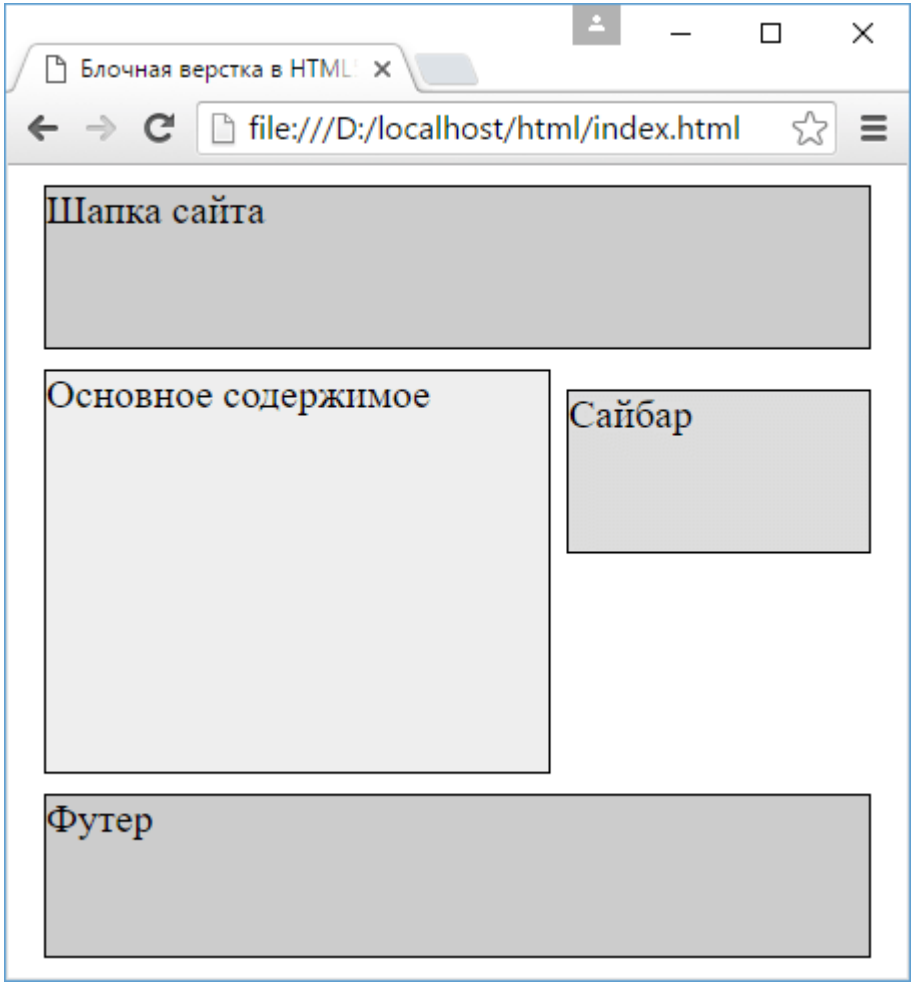
```
#sidebar{
    background-color: #ddd;
    float: left;
    width: 150px;
}
#main{
    background-color: #eee;
    height: 200px;
    margin-left: 170px; /* 150px (ширина сайдбара) + 10px + 10px (2 отступа) */
}
```



- Высота блоков в данном случае указана условно для большей наглядности, в реальности, как правило, высоту будет автоматически устанавливать браузер.

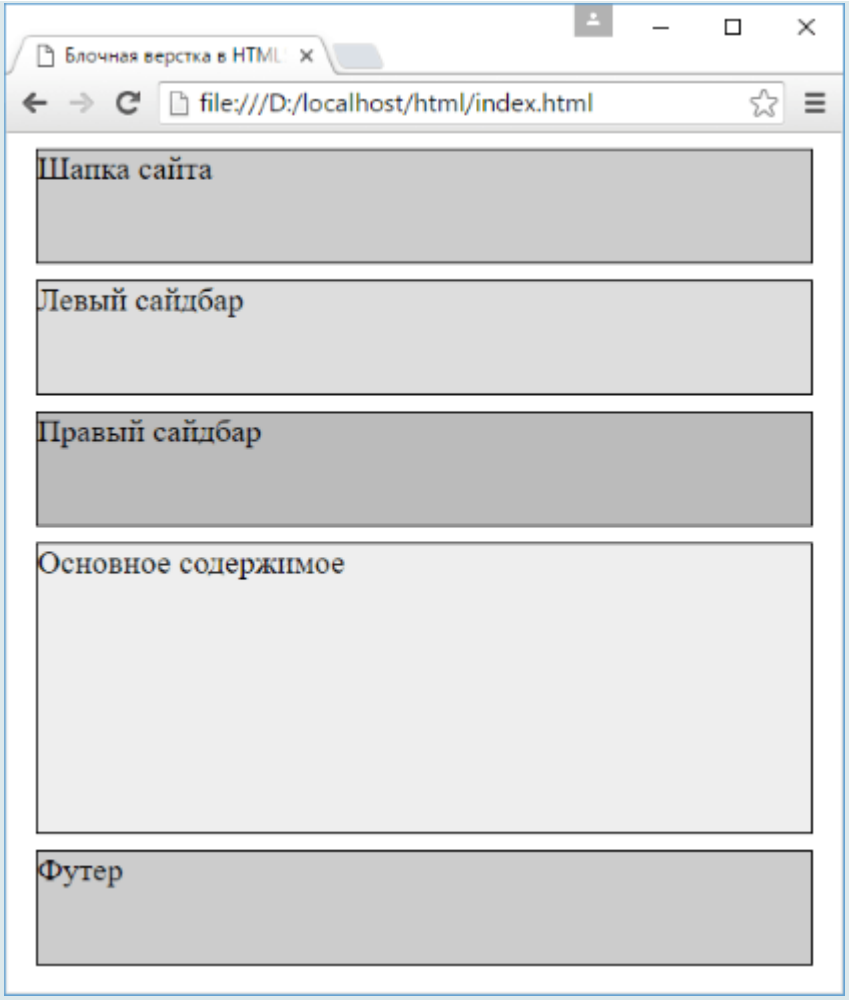
- Создание правого сайдбара будет аналогично, только теперь нам надо установить у сайдбара значение float: right, а у блока основного содержимого - отступ справа:

```
#sidebar{
    background-color: #ddd;
    float: right;
    width: 150px;
}
#main{
    background-color: #eee;
    height: 200px;
    margin-right: 170px;
}
```



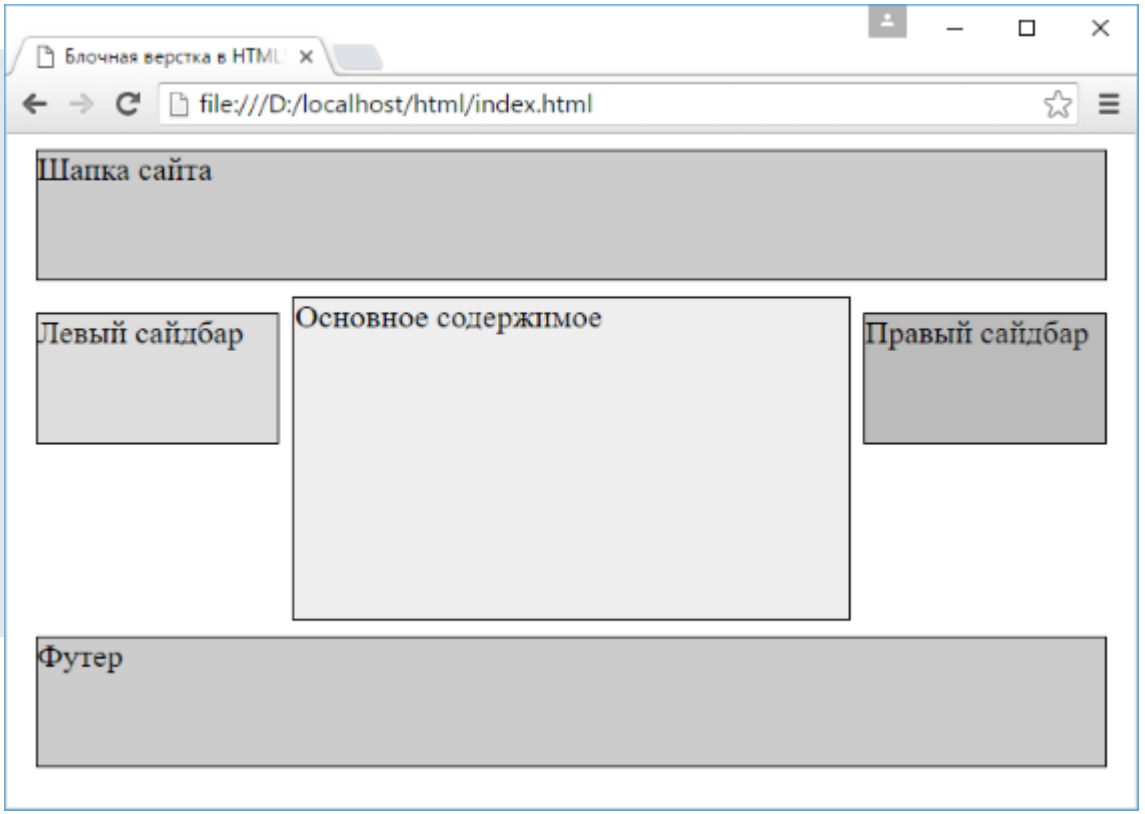
- В прошлой теме было рассмотрено создание страницы с двумя колонками. Подобным образом мы можем добавить на страницу и большее количество колонок и сделать более сложную структуру. Например, добавим на веб-страницу второй сайдбар:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Блочная верстка в HTML5</title>
        <style>
            div{
                margin: 10px;
                border: 1px solid black;
                font-size: 20px;
                height: 80px;
            }
            #header{
                background-color: #ccc;
            }
            #leftSidebar{
                background-color: #ddd;
            }
            #rightSidebar{
                background-color: #bbb;
            }
            #main{
                background-color: #eee;
                height: 200px;
            }
            #footer{
                background-color: #ccc;
            }
        </style>
    </head>
    <body>
        <div id="header">Шапка сайта</div>
        <div id="leftSidebar">Левый сайдбар</div>
        <div id="rightSidebar">Правый сайдбар</div>
        <div id="main">Основное содержимое</div>
        <div id="footer">Футер</div>
    </body>
</html>
```



- Теперь изменим стили обоих сайдбаров и основного блока:

```
#leftSidebar{
    background-color: #ddd;
    float: left;
    width: 150px;
}
#rightSidebar{
    background-color: #bbb;
    float: right;
    width: 150px;
}
#main{
    background-color: #eee;
    height: 200px;
    margin-left: 170px;
    margin-right: 170px;
}
```



- Опять же у обоих плавающих блоков - сайдбаров нам надо установить ширину и свойство float - у одного значение left, а у другого right.

Вложенные плавающие блоки

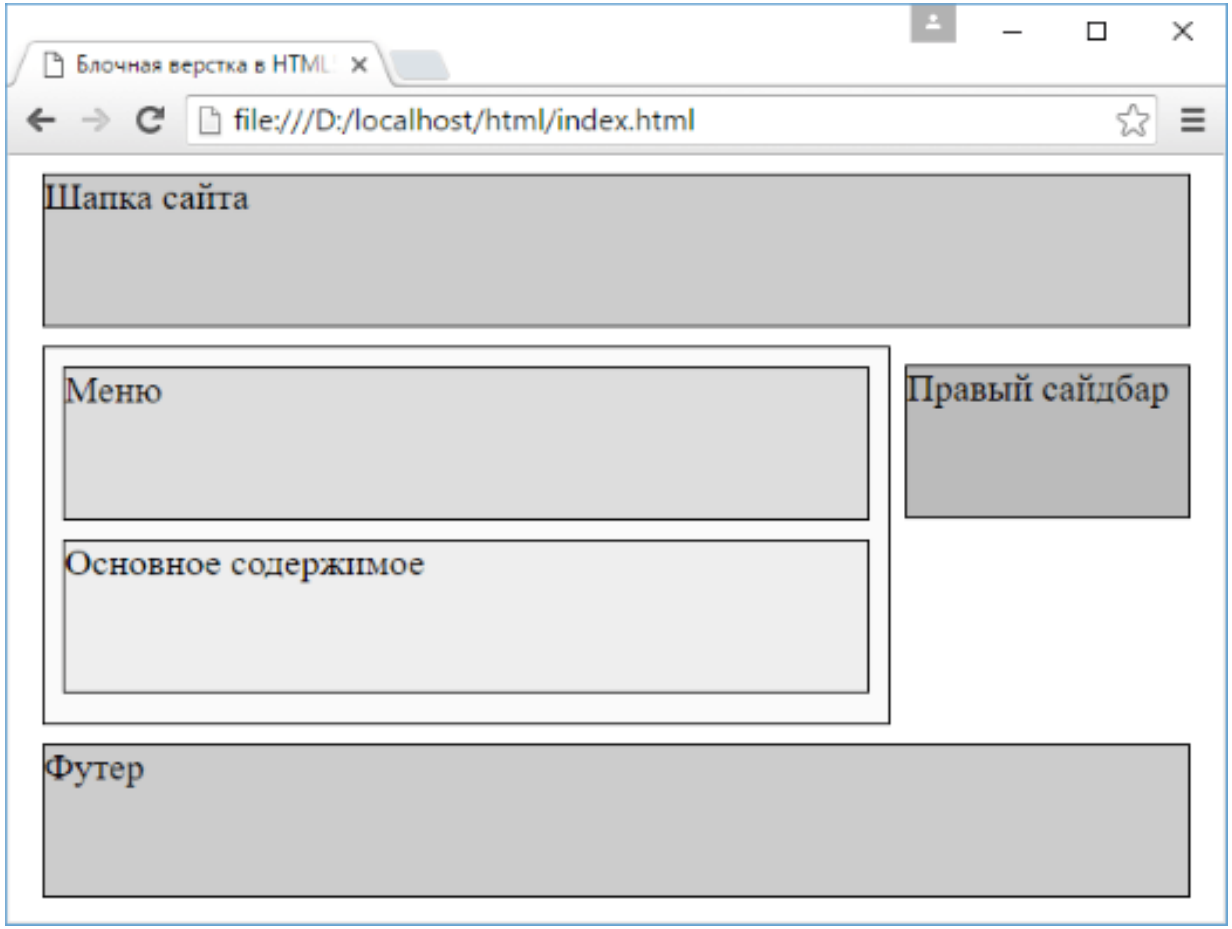
- Нередко встречается ситуация, когда к вложенным в обтекающий блок элементам также применяется обтекание. Например, блок основного содержимого может включать блок собственно содержимого и блок меню. В принципе к таким блокам будут применяться все те же правила, что были рассмотрены ранее.
- Определим сначала последовательно все блоки веб-страницы:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Блочная верстка в HTML5</title>
        <style>
            div{
                margin: 10px;
                border: 1px solid black;
                font-size: 20px;
                height: 80px;
            }
            #header{
                background-color: #ccc;
            }
            #sidebar{
                background-color: #bbb;
                float: right;
                width: 150px;
            }
            #main{
                background-color: #fafafa;
                height: 200px;
                margin-right: 170px;
            }
            #menu{
                background-color: #ddd;
            }
```

```
#content{
                background-color: #eee;
            }
            #footer{
                background-color: #ccc;
            }
        </style>
    </head>
    <body>
        <div id="header">Шапка сайта</div>
        <div id="sidebar">Правый сайдбар</div>
        <div id="main">
            <div id="menu">Меню</div>
            <div id="content">Основное
содержимое</div>
        </div>
        <div id="footer">Футер</div>
    </body>
</html>
```

Вложенные плавающие блоки

- Опять же в главном блоке вложенные блоки идут последовательно: сначала блок меню, а потом блок основного текста.

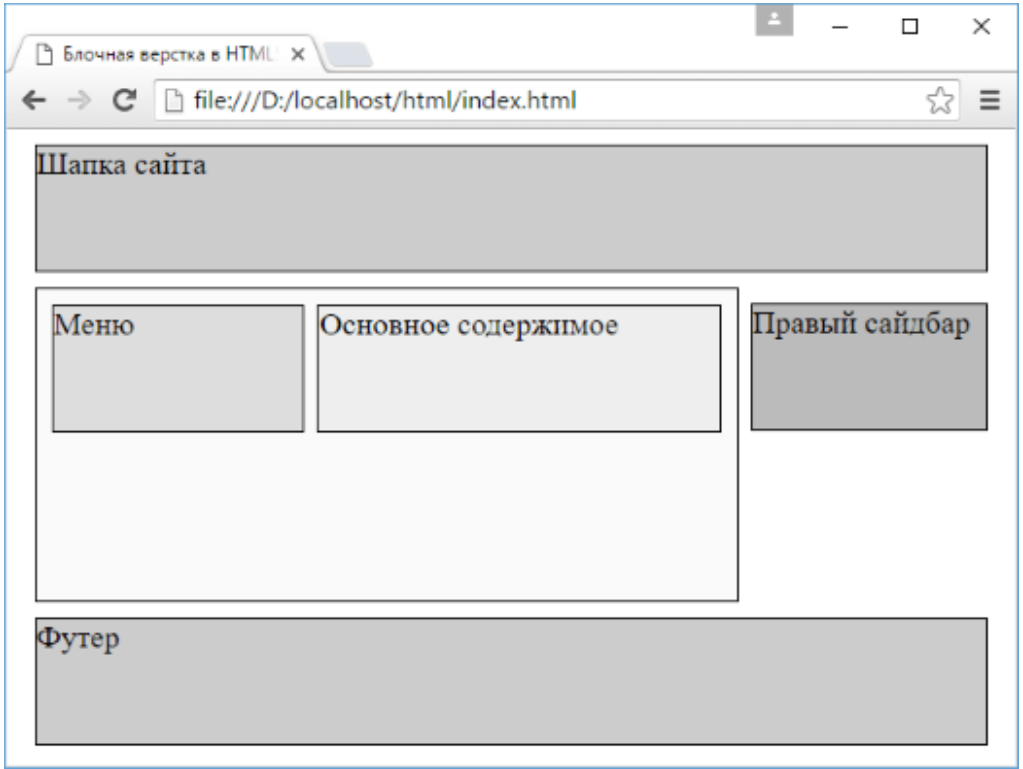


Вложенные плавающие блоки

- Теперь применим обтекание к блоку меню:

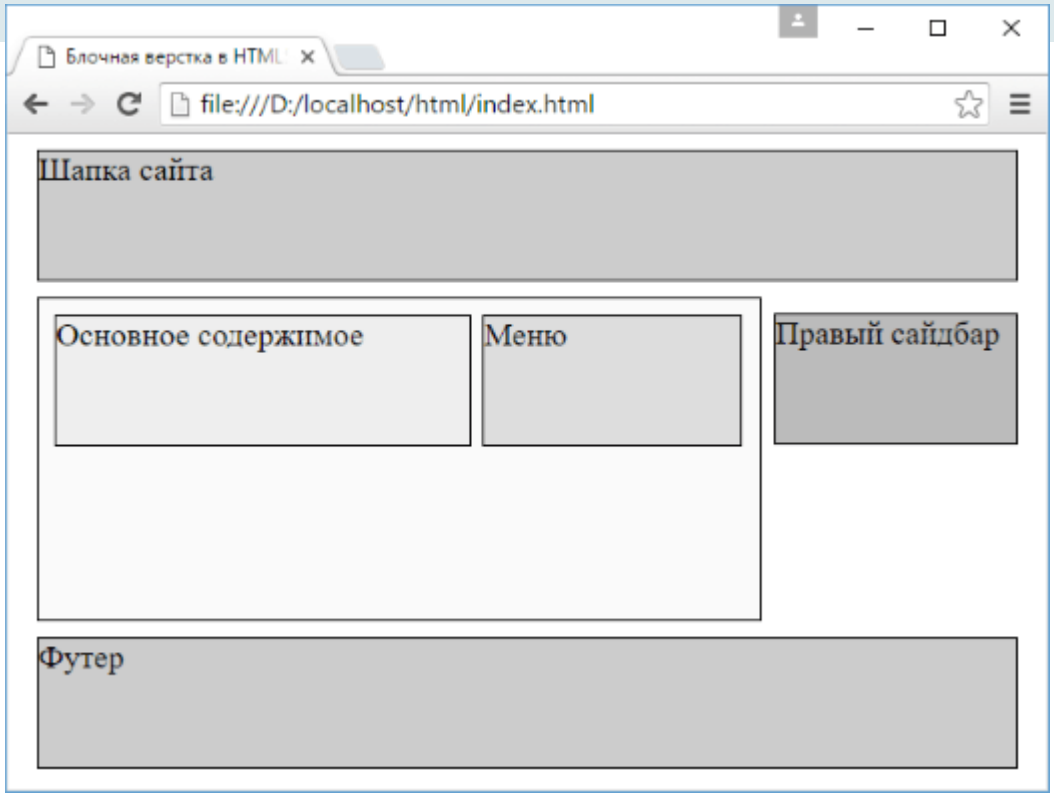
```
#menu{
    background-color: #ddd;
    float: left;
    width: 160px;
}
#content{
    background-color: #eee;
    margin-left: 180px;
}
```

- Опять же у плавающего элемента, коим является блок меню, устанавливаются свойства float и width. А у обтекающего его блока content устанавливается отступ слева.



- Аналогично можно сделать блок меню справа:

```
#menu{
    background-color: #ddd;
    float: right;
    width: 160px;
}
#content{
    background-color: #eee;
    margin-right: 180px;
}
```



Спасибо за внимание.