

Урок 6. Методичка

Блочная модель.
Строим дом.



Содержание

- [Краткое содержание урока 5;](#)
- [Селекторы атрибутов;](#)
- [Наследование стилей;](#)
- [Приоритеты стилей;](#)
- [Дополнительные теги и свойства;](#)
- [Блочные и строчные элементы;](#)
- [Позиционирование;](#)
- [Свойство float;](#)
- [Центрирование сайта.](#)

Краткое содержание урока 5

На пятом уроке мы познакомились с продвинутыми вариантами селекторов, которые позволяют точно выбирать элементы на странице без использования классов, а используя структуру html-документа. Познакомились с псевдоклассами и псевдоэлементами. Выучили новый тег списка.

На этом тема селекторов не заканчивается, и прежде чем перейти к блокам, поговорим о селекторах атрибутов, приоритетах стилей и наследовании.



То, что мы научились выбирать элементы без классов, это не означает, что классы больше не нужны, в таком способе есть большой минус, если изменится структура html-документа, то стили уже не будут корректно работать.

Селекторы атрибутов

Можно обращаться к элементам по их атрибутам. Например, сделаем 3 изображения с разными атрибутами, и выберем 2 и 3 картинку, указав им разные границы:

```
<img src="" alt="img">  
<img src="" alt="img" title="123">  
<img src="" alt="img" title="456">
```

```
img[title] {  
    border: 2px solid red;  
}  
img[title="456"] {  
    border: 2px solid blue;  
}
```



Как видите можно еще точнее выбирать элементы по атрибутам, даже можно выбирать с конкретным значением атрибута.

Наследование стилей

Наследование, это когда стили, относящиеся к одному элементу, применяются и на все его вложенные элементы. Обратите внимание, что свойство color применилось и к абзацу, и к тегу b, т.к. они вложены в body и наследуют его свойства, причем на всех уровнях вложенности:

```
body {  
    color: red;  
}  
  
</style>  
</head>  
  
<body>  
    <p>Абзац <b>текста</b></p>  
    <p>Абзац текста</p>  
</body>
```



Абзац текста

Абзац текста



Наследуются далеко не все свойства, например font наследуется, а border не наследуется, этот механизм нужен для того, чтобы не нужно было прописывать однообразные свойства ВСЕМ элементам на странице.

Приоритеты стилей

А что произойдет, если мы сделаем разные стили одному и тому же элементу, например, абзацу применим один цвет и тут же другой?

```
<style>
  p {color: red;}

  p {color: blue;}
</style>
</head>

<body>
  <p>Абзац</p>
</body>
```

Абзац

```
p {
  color: ■ blue;
}

p {
  color: ■ red;
}
```



Если одному и тому же селектору назначить последовательно разные стили, то работать будет последний, откройте отладчик F12 и посмотрите, что красный цвет зачеркнут, т.к. он переопределяется стилем, который идет ниже.

Приоритеты стилей

А теперь, в предыдущем примере, добавим стиль для body красный. Красный унаследуется для b, но смотрите, все равно b стал голубым, несмотря на то, что мы переопределили его через наследование.

```
<style>
  b {color: blue;}
  body {color: red;}
</style>
</head>
<body>
  <p>Абзац <b>текста</b></p>
  <p>Абзац текста</p>
</body>
```



Абзац текста

Абзац текста



Здесь работает **каскадность** – ряд правил, определяющих какое именно свойство будет применено к элементу. Поэтому css-это каскадные таблицы стилей. В данном случае body не смог переопределить цвет для b, почему же?

Приоритеты стилей

Посмотрим подробнее что выдает отладчик на для стилей тега b:

в итоге применился этот стиль, несмотря на то, что он идет раньше



```
b {  
  color: ■ blue;  
}
```

index3.html:9

user agent стили, это стили, которые задает сам браузер



```
b {  
  font-weight: bold;  
}
```

user agent stylesheet

Inherited from *body*

видим, что стиль переопределился, Inherited означает унаследован от body



```
body {  
  color: ■ red;  
}
```

index3.html:10



Так почему же унаследованный стиль не смог «перебить» заданный нами явно?

Приоритеты стилей

CSS предлагает метод определения приоритетов применения стилей. Он основан на присвоении условных значений каждому типу селекторов:

- селектор тегов имеет значимость (b, div, body) – 1 пункт;
- селектор классов (.menu) – 10 пунктов;
- id-селектор (#main) – 100 пунктов;
- встроенный (inline) стиль – 1000 пунктов!

```
<style>
  b {color: blue;}
  body {color: red;}
</style>
</head>
<body>
  <p>Абзац <b>текста</b></p>
  <p>Абзац текста</p>
</body>
```

Псевдоклассы например :link , рассматриваются как класс и имеют значимость 10, псевдоэлементы, например :first-child обрабатываются как тег и имеют значимость 1 пункт.



Получается что элемент b получает 1 пункт значения веса, и он оказался выше унаследованного!

Приоритеты стилей

Вот несколько примеров подсчета веса, чтобы определить какой стиль победит:

p – просто тег, и вес 1;

.menu – это класс, значит вес 10;

ul.menu – и класс и тег, получаем 11;

#main – идентификатор, значение веса 100;

#main .menu – и класс и идентификатор, вес 110;

a:link – класс и тег, получаем 11;

h1 b – 2 тега, итого 2;

#container #main .menu a:hover – 2 идентификатора + 2 класса + 1 тег = 221.



Получается что, чем точнее мы укажем селектор элемента, тем больше будет приоритет у этого стиля.



Стили могут пересекаться и за счет наследования, вот тут уже такие приоритеты не работают, всегда побеждает тот стиль, который задан явно для элемента!

Дополнительные теги и свойства

Соберем вместе некоторые дополнительные теги свойства, которые нам могут понадобиться в будущем:

:first-letter – псевдоэлемент, позволяет выбрать первую букву для стилизации;

<hr> - тег, выводящий горизонтальную серую полосу;

border-radius: 6px; - позволяет скруглить углы у элемента (6px радиус в этом примере);

background-repeat: repeat-y; - позволяет управлять повторяемостью фона по горизонтали и по вертикали, y-по вертикали repeat-x по горизонтали, no-repeat не дублировать фон вообще;

text-decoration: underline; - оформление текста подчеркнутый, можно еще none – отменить оформление.

**** - парный тег, аналог div, только для строчных элементов;

Блочные и строчные элементы

Исследуем поведение блочных элементов, например `div`, для этого сделаем 3 блока с текстом, внешним и внутренним отступом:

```
<style>
  .block {
    margin: 20px;
    background: silver;
    padding: 10px
  }
</style>
</head>
<body>
  <div class="block">Блочный элемент</div>
  <div class="block">Блочный элемент</div>
  <div class="block">Блочный элемент</div>
</body>
```

Блочный элемент

Блочный элемент

Блочный элемент



Как мы видим, блочные элементы стараются занять всю доступную ширину и располагаются друг под другом, как кирпичики, из которых строится дом.

Блочные и строчные элементы

Заменяем блочный `div` на строчный `span`, и сделаем внешний отступ `0px`:

```
<style>
  .block {
    margin: 0px;
    background: silver;
    padding: 10px
  }
</style>
</head>
<body>
  <span class="block">Строчный элемент</span>
  <span class="block">Строчный элемент</span>
  <span class="block">Строчный элемент</span>
</body>
```

Строчный элемент

Строчный элемент

Строчный элемент



Строчные элементы занимают ширину по своему содержимому, и располагаются по горизонтали в строку. Еще есть особенность, что мы видим небольшой пробел между элементами, чтобы его не было, нужно размещать подряд `` теги в одну строку. Чтобы не было пробела или переноса, который мы и видим.

Блочные и строчные элементы

Свойство `display` как раз отвечает за то, каким будет элемент, для `div` например, по умолчанию:
`display: block;`

Для `span` по умолчанию:
`display: inline;`

Еще одним важным отличием строчных элементов от блочных является то, что для строчных не работают свойства ширины и высоты, размером строчных элементов управляет их содержимое! Не работает также внешний отступ, а внутренний работает.

```
<span>Текст</span>
```

```
span {  
  background: silver;  
  width: 100px;  
}
```

Текст



Получается, что при помощи стилей мы можем переопределять строчные элементы в блочные и наоборот так, как нам нужно. Блочные располагаются сверху вниз, занимая всю ширину, а строчные слева-направо и у строчных нельзя изменить свойствами размер.

Блочные и строчные элементы

Возможно еще одно значение **display: inline-block;**

```
<style>
  ul li {
    display: inline-block;
    width: 100px;
    text-align: center;
  }
  li:hover {background: silver;}
</style>
</head>
<body>
  <ul>
    <li>Пункт 1</li>
    <li>Пункт 2</li>
    <li>Пункт 3</li>
  </ul>
</body>
```

Пункт 1

Пункт 2

Пункт 3



При этом элемент остается строчным, однако для него будут работать свойства ширины и внешнего отступа, часто используется для создания меню. Это так называемый строчно-блочный элемент.

Позиционирование

Свойство CSS **position**: задает режим позиционирования элементов, по умолчанию действует режим:

position: static;

Обозначающее обычное позиционирование (т.е. блочные элементы сверху вниз, строчные слева – направо)

Чтобы задать точно положение элемента нужно использовать режим:

position: absolute;

И задать положение элемента:

top: 100px;

left: 150px;

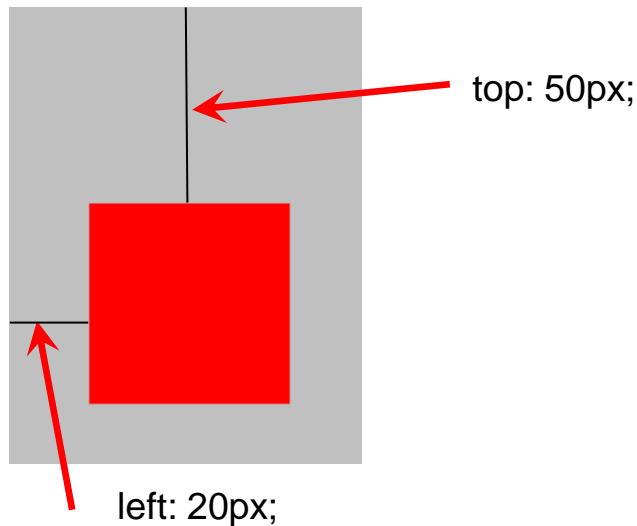


Возможны еще варианты положения **right** – справа и **bottom** – снизу.

Позиционирование

В этом примере отсчет для блока `div` ведется от левого верхнего угла браузера, т.к. блок внутри `body`.

```
<style>
  body {background: silver;}
  div {
    position: absolute;
    top: 50px;
    left: 20px;
    width: 50px;
    height: 50px;
    background: red;
  }
</style>
</head>
<body>
  <div></div>
</body>
```

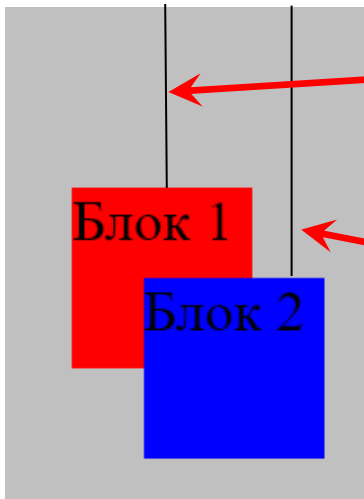


Возможны еще варианты положения `right` – справа и `bottom` – снизу.

Позиционирование

Добавим второй блок рядом с первым и тоже абсолютно его позиционируем в левый верхний угол:

```
<div class="block1">Блок 1</div>  
<div class="block2">Блок 2</div>
```



```
div {width:50px; height: 50px;}  
.block1 {  
  position: absolute;  
  top: 50px;  
  left: 20px;  
  background: red;  
}  
.block2 {  
  position: absolute;  
  top: 75px;  
  left: 40px;  
  background: blue;  
}
```

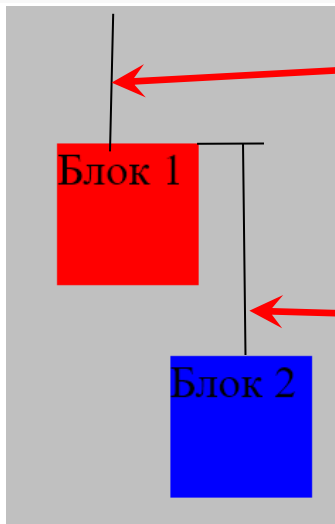


Абсолютное позиционирование можно сравнить с гвоздями, мы как будто прибиваем блок точно в определенное положение жестко. Даже видим наложение блоков друг на друга.

Позиционирование

А теперь вложим один блок в другой:

```
<div class="block1">Блок 1  
  <div class="block2">Блок 2</div>  
</div>
```



```
div {width:50px; height: 50px;}  
.block1 {  
  position: absolute;  
  top: 50px;  
  left: 20px;  
  background: red;  
}  
.block2 {  
  position: absolute;  
  top: 75px;  
  left: 40px;  
  background: blue;  
}
```

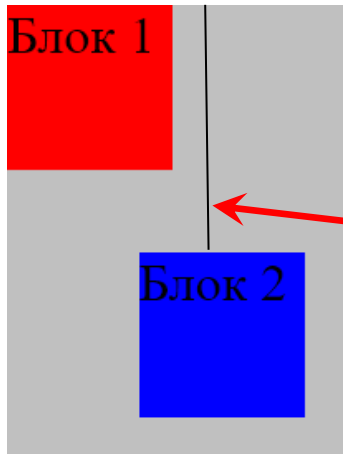


Теперь второй блок позиционируется относительно первого! Получается что absolute не всегда относительно браузера отсчитывается, все зависит от родительского блока. Если он тоже absolute, то отсчет пойдет от него.

Позиционирование

Если вернуть первый блок в обычное состояние, `static`, то позиционирование для него не будет работать, и второй блок позиционируется относительно браузера (`body`):

```
<div class="block1">Блок 1  
  <div class="block2">Блок 2</div>  
</div>
```



```
div {width:50px; height: 50px;}  
.block1 {  
  position: static;  
  top: 50px;  
  left: 20px;  
  background: red;  
}  
.block2 {  
  position: absolute;  
  top: 75px;  
  left: 40px;  
  background: blue;  
}
```

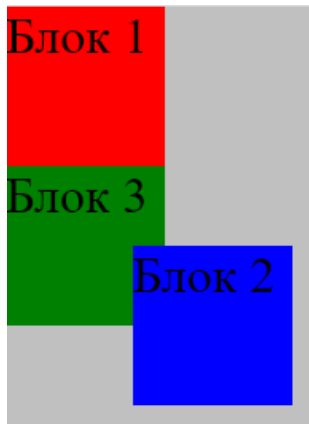


Первый блок позиционируется в 0,0, но мы увидим отступ, т.к. по умолчанию в `body` есть уже отступ, можете его убрать через `margin:0`; для `body`;

Позиционирование

Еще одно значение позиционирования relative – относительное, добавим 3-й блок:

```
<div class="block1">Блок 1  
  <div class="block2">Блок 2</div>  
</div>  
<div class="block3">Блок 3</div>
```



```
div {width:50px; height: 50px;}  
.block1 {  
  position: relative;  
  top: 0px;  
  left: 0px;  
  background: red;  
}  
.block2 {  
  position: absolute;  
  top: 75px;  
  left: 40px;  
  background: blue;  
}  
.block3 {  
  background: green;  
}
```

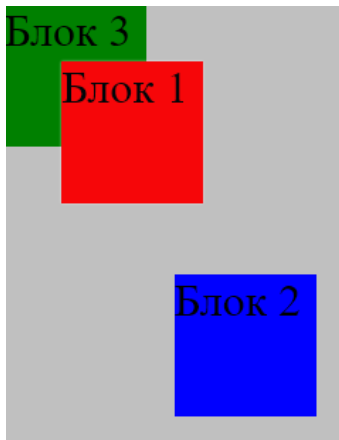


Получается что relative тоже можно позиционировать, но этот блок находится «в потоке» и влияет на положение следующих блоков в потоке.

Позиционирование

Вернем relative на absolute и видим как 3-й блок занял верхнюю позицию.

```
<div class="block1">Блок 1  
  <div class="block2">Блок 2</div>  
</div>  
<div class="block3">Блок 3</div>
```



```
div {width:50px; height: 50px;}  
.block1 {  
  position: absolute;  
  top: 20px;  
  left:20px;  
  background: red;  
}  
.block2 {  
  position: absolute;  
  top: 75px;  
  left: 40px;  
  background: blue;  
}  
.block3 {  
  background: green;  
}
```



1 и 2 блок «выпали из потока», и третий блок их не видит!

Свойство float

Еще одно свойство, которое позволяет изменить поведение элемента, **float**, посмотрим пример:

```
<style>
  img {
    width: 50px;
    height: 50px;
    background: silver;
    float: left;
  }
</style>
</head>
<body>
  <p>
    <img>Lorem ipsum dolor sit amet, consectetur adipiscing
    elit. Modi eius unde aspernatur recusandae consectetur
    accusantium earum odio maxime expedita nam Modi!
  </p>
```



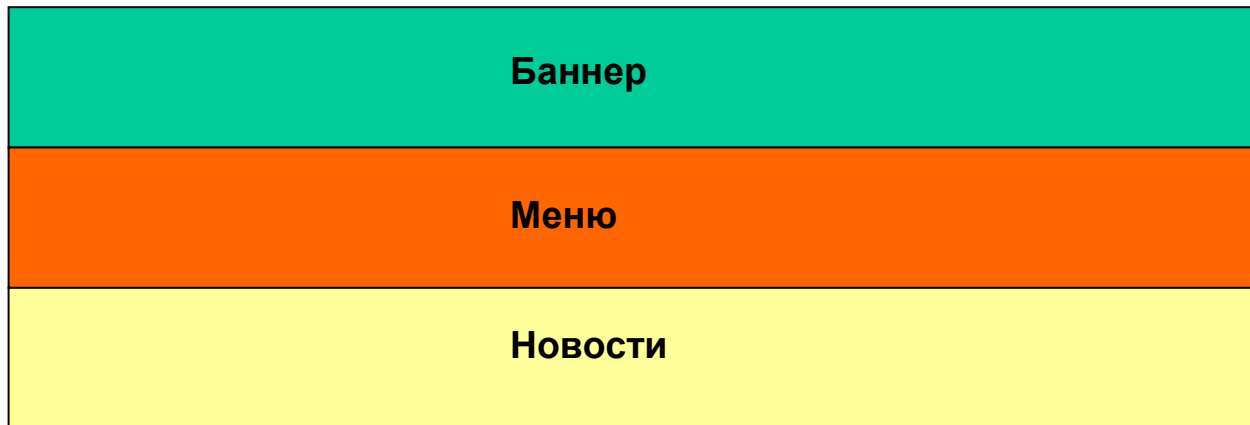
Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Modi eius
unde aspernatur recusandae
consectetur accusantium earum odio maxime
expedita nam Modi!



float: left; делает 2 действия – указывает, что блок прижимается в лево и выпадает из потока, это значит не влияет на другие блоки, влияет только на обтекание его текстом.

Свойство float

Это свойство можно использовать для размещения колонок на сайте. Например составим сайт из блоков:



Здесь просто 3 блока div, которые находятся в «потоке» т.е. «видят» друг друга, и как кирпичики накладываются друг на друга.

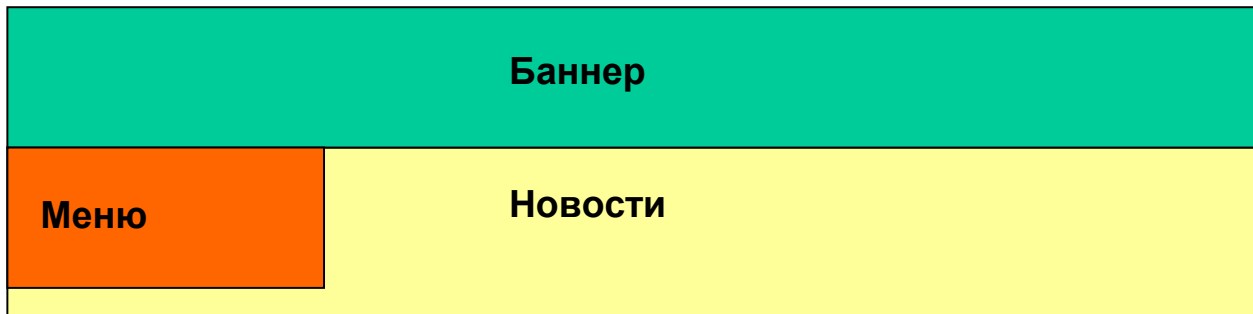
Свойство float

Зададим для меню свойства:

float: left;

width: 150px;

Таким образом текст новости будет обтекать меню, но блок будет занимать место под меню, и нужно освободить его.



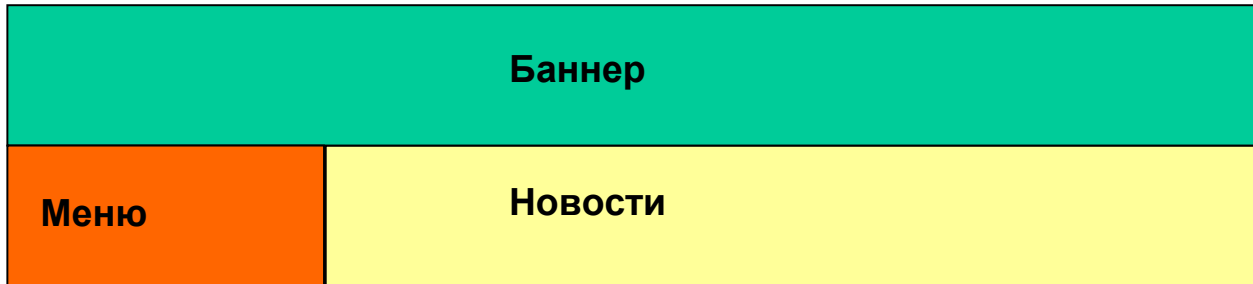
Обратите внимание, блок меню «выпал из потока» и новости прижимаются к баннеру. Точно также «выпадают из потока» абсолютно позиционированные блоки.

Свойство float

Чтобы подравнять новости зададим для новостей свойства:

width: 150px;

margin-left: 150px;



Это один из приемов горизонтального размещения блочных элементов, который можно использовать для простого позиционирования.

Центрирование сайта

Чтобы позиционировать любой блок по центру, можно воспользоваться автоматическими отступами справа и слева.

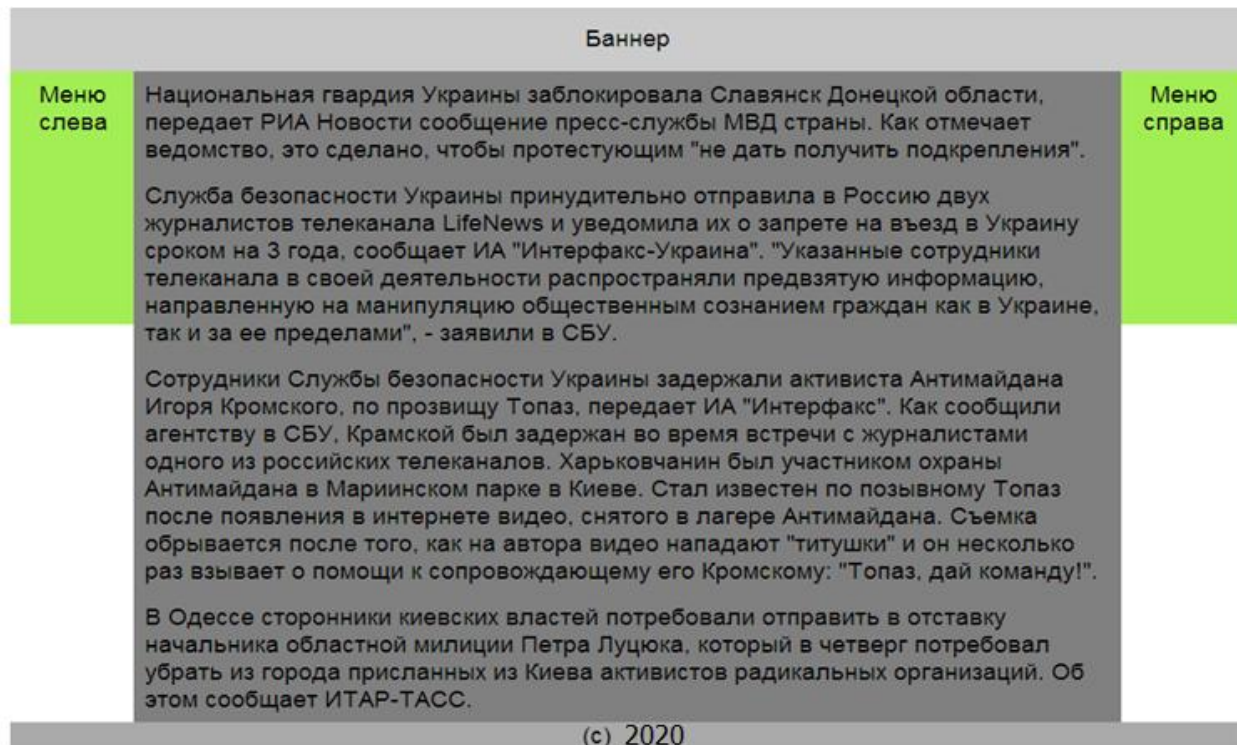
Такой прием часто используется для создания оболочки сайта, в котором собирается остальное содержимое.

Важно, чтобы у оболочки (wrapper) или еще ее называют container, была указана ширина. Иначе центрирование работать не будет.

```
<style>
    .wrapper {
        width: 600px;
        margin: 0 auto;
        background: silver;
    }
</style>
</head>
<body>
    <div class="wrapper">
        Содержимое сайта
    </div>
```

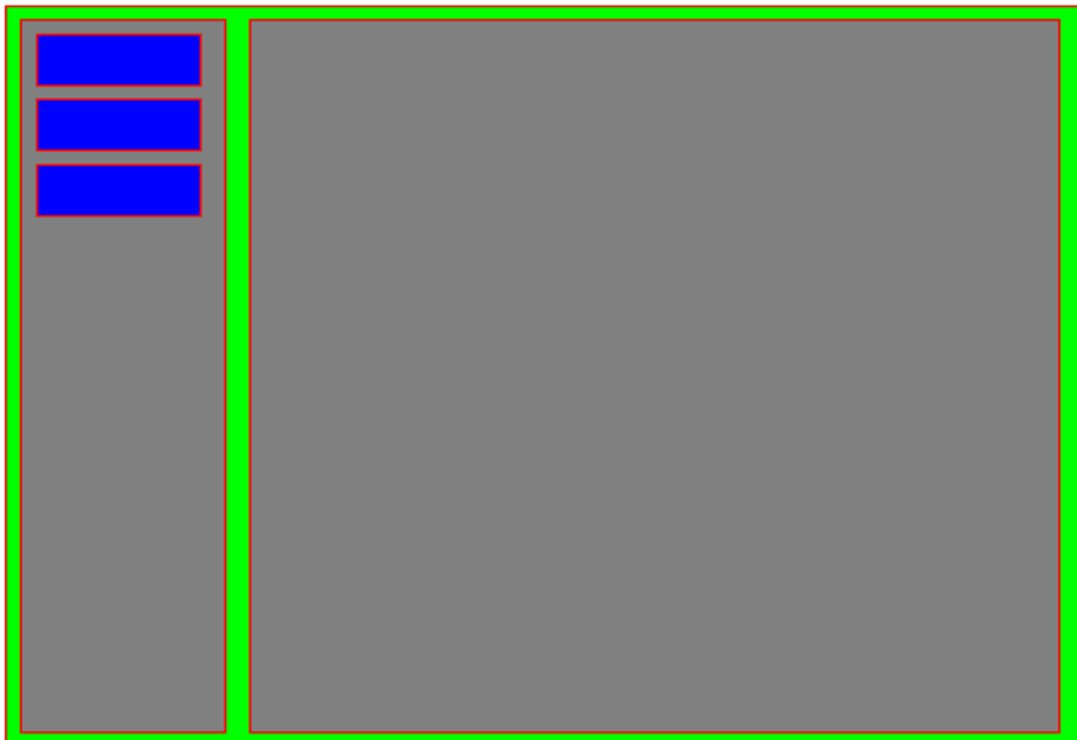
Квест на урок 6

1. Используя float: right; и float: left; соберите следующий макет сайта. Он должен быть обернут в оболочку и позиционирован по центру. Цвета можно использовать свои, главное структура.



Квест на урок 6

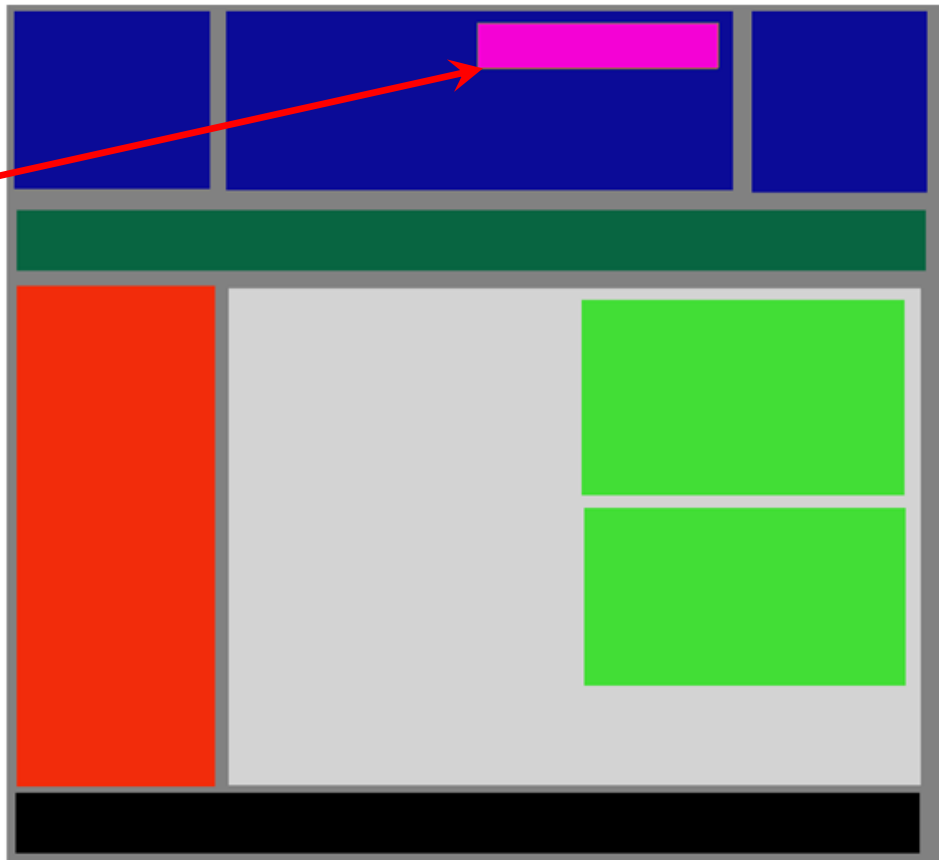
2. Используя полученные знания на уроке, сверстайте следующую структуру. Желательно чтобы не применялось абсолютное позиционирование. А для меню используйте список, сделав его элементы блочными. Цвета можете взять свои, высоту можно указывать через height.



Квест на урок 6

3* Если предыдущее задание покажется простым, сверстайте структуру посложнее.

Здесь уже допустимо абсолютное позиционирование для этого блока. Постарайтесь использовать float для боковых элементов.



Квест на урок 6

4*. Начните верстать следующий макет, все изображения к нему найдете в материалах на сайте. Это задание на 2 урока.

У друзей

друзья группы игры помощь выход

Моя страница

Мои друзья

Мои фотографии

Мои сообщения


Мои группы

Мои заметки

Мои настройки


Акция от НТС

nts.su




Только до 31 мая!

ford.ru



Позвольте себе Ford Focus по специальной цене

Avatar



Avatar

День рождения: 30.09.1975




Родной город: Северск

Семейное положение: Женат

Место работы: ТУСУР


О себе: Хорошо живет на свете пух...

Фотографии




Записи


Оформление стены




Друзья




Юлия Ефимова





Валерия Шпигарь



Grigori Manukyan

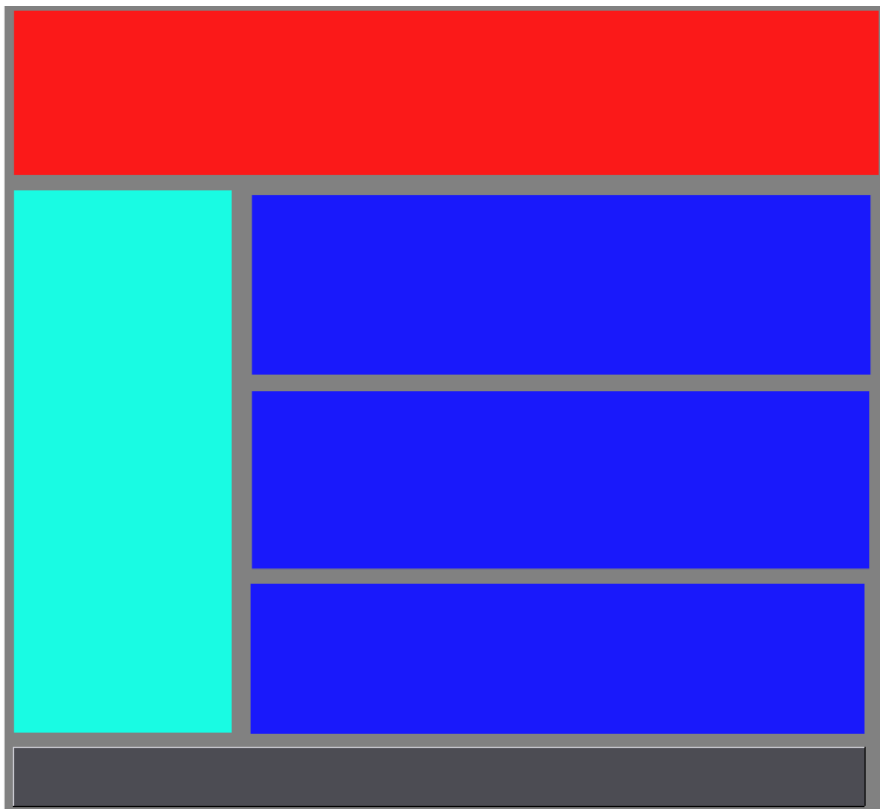




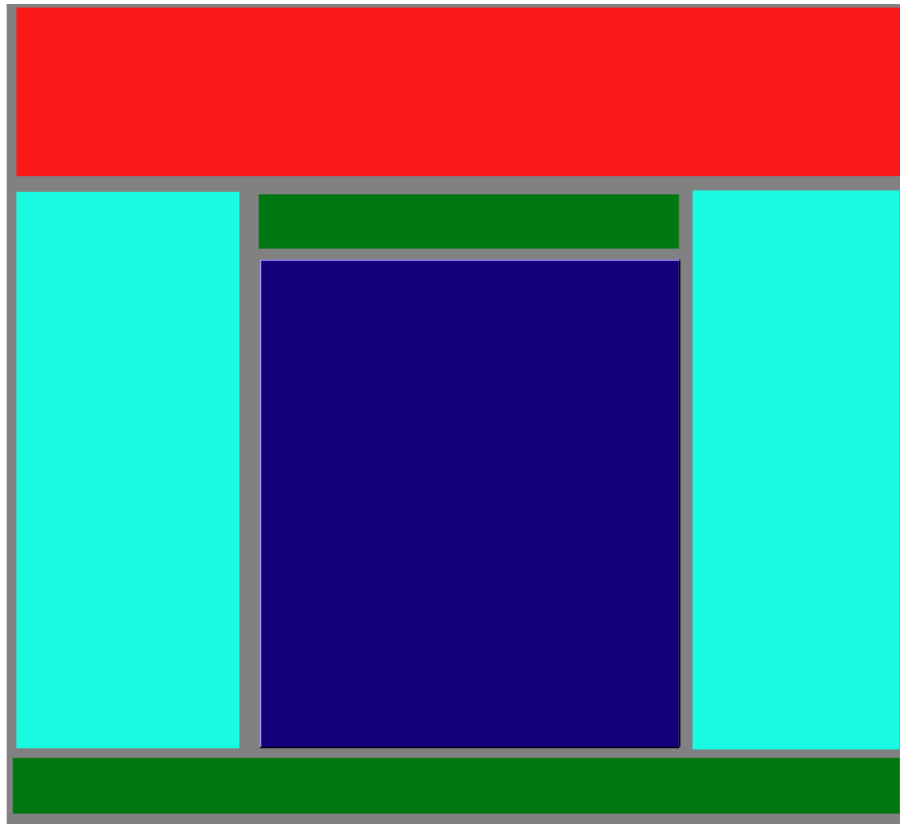


Квест на урок 6

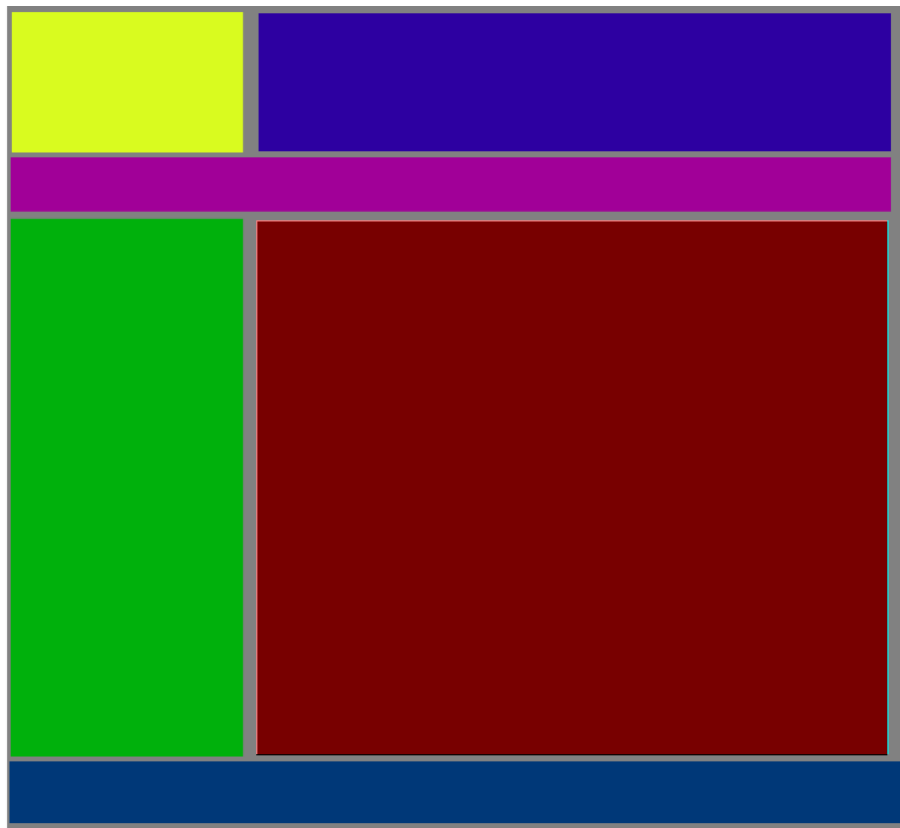
5*. Сверстайте дополнительные макеты на выбор



Квест на урок 6



Квест на урок 6



Квест на урок 6

