БЭМ

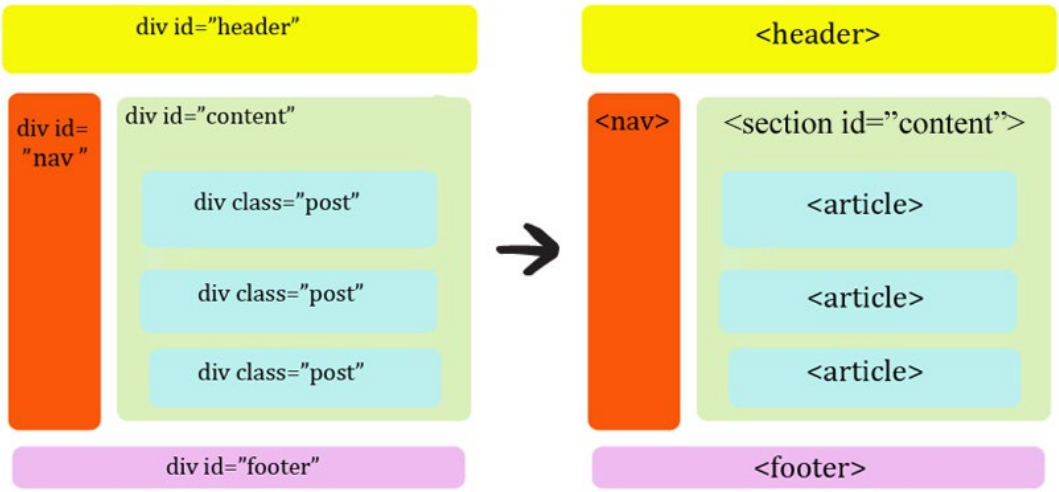
LoftSchool
от мыслителя к создателю

Содержание

1

Структурная разметка HTML5

В HTML5 появилось несколько новых тегов, которые призваны заменить часто применяемые блоки **div**. В отличие от универсального **div**, новые теги несут семантическую нагрузку, и строгое определение для каждого блока его место и роли:



<header> </header>	Определяет область «шапки» сайта с логотипом, первичной навигацией и тд.
<footer> </footer>	Определяет «подвал», колонтитул веб страницы сайта или раздела, в котом обычно размещается дополнительная информация
<section> </section>	Определяет блок, который служит для группировки однотипных объектов, или для разделения текста на разделы
<article> </article>	Определяет автономную часть страницы, это может быть сообщение форума, журнала или газетную статью, запись в блоге и тд.
<nav> </nav>	Определяет область навигации, как правило список ссылок

Помните, что ***семантическая вёрстка*** - это вёрстка с правильным использованием HTML-тегов. С использованием их по назначению, как их и задумывали разработчики языка HTML и веб- стандартов. Например, тег **<p>** — это абзац, и не стоит им размечать строки веб-форм. А **<b>** — это просто выделение текста жирным, а вовсе не заголовок.

2

Viewport и scale

Viewport - это видимая пользователю область веб-страницы. Т.е. это то, что может увидеть пользователь, не прибегая к прокрутке. Размеры этой области опредляются размером экрана устройства.

Метатег **viewport** был разработан компанией Apple для того, чтобы указывать браузерам на то, в каком масштабе необходимо отображать пользователю видимую область веб-страницы. Другими словами **viewport** предназначен для того, чтобы веб-страницы отображались (выглядели) правильно (корректно) на смартфонах, планшетах и других устройствах с высокой плотностью пикселей (>200ppi). Данный метатег предназначен в большой степени для адаптивных сайтов, но с помощью него можно улучшить представления веб-страниц, имеющих фиксированную или гибкую разметку.

Рассмотрим использование метатега **viewport** для адаптивных сайтов.

Включение поддержки тега **<meta> viewport** для адаптивных сайтов осуществляется посредством добавления всего одной строчки в раздел **head** веб-страницы:

```
<meta name="viewport" content="width=device-width,
initial-scale=1">
```

Атрибут **name** предназначен для того чтобы указать браузеру, какую именно информацию о странице хотим ему сообщить. В данном случае эта информация касается viewport. Контент (содержимое) этих сведений указывается в качестве значения атрибута content посредством пар ключ-значение, разделённых между собой запятыми.

Для адаптивного дизайна значения атрибута **content viewport** должно определяться двумя параметрами:

```
width=device-width и initial-scale=1
```

Первый параметр **width=device-width** отвечает за то, чтобы ширина видимой области веб-страницы равнялась CSS ширине устройству (**device-width**). Эта ширина (CSS) - не физическое разрешение экрана. Это некоторая величина, независящая от разрешения экрана. Она предназначена для того, чтобы мобильный адаптивный дизайн сайта отображался на всех устройствах одинаково независимо от их плотности пикселей экрана.

Второй параметр **initial-scale** - устанавливает первоначальный масштаб веб-страницы. Значение 1 означает то, что масштаб равен 100%.

Если сайт не имеет адаптивный дизайн, то его представление на экране смартфона тоже можно улучшить. Можно сделать так чтобы ширина страницы масштабировалась под ширину устройства. Осуществляется это тоже с помощью установления параметру width значения **device-width**. Т.е. для не адаптивных сайтов в раздел head необходимо добавить следующую строчку:

```
<meta name="viewport" content="width=device-width">
```

3

Сброс исходных стилей

Если мы создадим страницу на «голом» HTML без оформления и стилей, браузер все равно отобразит содержание тега **<h1>** крупным и жирным, **<h2>** - чуть меньшим размером, выделит текст в теге **<i>** курсивом, **<u>** сделает подчеркнутым, а **<b>** -жирным.

Произойдёт так потому, что каждый браузер имеет по умолчанию некий набор базовых стилей, которые он применяет к странице по умолчанию. И дело в том, что в разных браузерах эти правила немного отличаются. Лет 10 назад эти отличия были кардинальными, и очень бросались в глаза. Сейчас они минимальны, но все же есть.

Чтобы убрать эти различия, и сделать по умолчанию отображение страницы во всех браузерах одинаковым - используются специальные **.css** файлы: **Eric Meyer's CSS Reset** или **Normalize.css**

Мы рекомендуем использовать **Normalize.css**

Normalize.css обеспечивает для HTML-элементов лучшую кроссбраузерность в стилях по умолчанию. Используя его, можно сэкономить время, которое в случает с CSS Reset было бы потрачено на прописывание сброшенных стилей.

Он исправляет основные баги на мобильных и десктопных устройствах, которые не затрагивает **CSS Reset**. Это включает в себя параметры отображения элементов HTML5, исправление font-size для предварительно отформатированного текста, отображение SVG в IE9, и многие другие баги, связанные с отображаемым в разных браузерах и операционных системах. Файл разбит на относительно независимые участки, каждый из которых прокомментирован, что даёт вам возможность удалить блоки свойств (например, нормализацию форм), если известно, что они никогда не понадобятся на сайте.

Помните, файл сброса/ нормализации стилей – это первое, что должен увидеть браузер! И только после него вы подключаете свои стили.

4

Условные комментарии

Любой текст в коде HTML можно закомментировать и при этом он никак не будет отображаться на веб-странице. Для этого его следует поместить между элементами **<!-- и -->**. Браузер Internet Explorer кроме того поддерживает специальный синтаксис, в задачу которого входит интерпретировать код, если перед нами Internet Explorer. Остальные браузеры при этом видят обычный комментарий и не отображают его.

```
<!--[if IE]>
   Код для браузера Internet Explorer
<![endif]-->
```

Внутри квадратных скобок допустимо использовать следующие ключевые слова:

IE	любая версия браузера Internet Explorer;
IE 6	Internet Explorer 6;
IE 7	Internet Explorer 7;
IE 8	Internet Explorer 8;
IE 9	Internet Explorer 9;
lt	номер версии браузера меньше указанной;
gt	номер версии больше указанной;
lte	номер версии меньше или равен указанной;
gte	номер версии браузера больше или равен указанной

Наиболее часто используемая конструкция, которая будет использоваться вами выглядит так:

```
<!--[if lt IE 8]>
<p class="chromeframe">Похоже, что ты используешь
устаревший браузер, друг. Пожалуйста, <a href="http://
browsehappy.com/">скачай новый браузер абсолютно
бесплатно</a> или <a href="http://www.google.com/
chromeframe/?redirect=true">активируй Google Chrome Frame</
a></p>
<![endif]-->
```

5

БЭМ нотация

БЭМ — аббревиатура от слов Блок, Элемент, Модификатор — методология именования, изобретённая парнями из Яндекса. Это изящный путь именования классов с целью сделать их понятнее и прозрачнее для других разработчиков. Этот способ строг и информативен, что делает БЭМ идеальной системой для групп разработчиков на больших проектах.
В БЭМ-методологии CSS используется для оформления страниц и рассматривается как одна из технологий реализации блока.

Цель БЭМ — рассказать другим разработчикам как можно больше о том, что делает кусок кода, только по названиям классов в разметке. Читая HTML с небольшим количеством классов внутри, вы можете увидеть взаимодействия во всем коде и между его частями; что-то может быть компонентом, что-то дочерним элементом — элементом этого компонента, а что-то может быть измененной копией — модификатором компонента.

Давайте рассмотрим пример формы поиска на сайте:

```
<form class="site-search  full">
    <input type="text" class="field">
    <input type="Submit" value ="Search" class="button">
</form>
```

Это обычные классы, по ним много не скажешь. Да, мы вполне можем так работать, но с БЭМ мы получим:

```
<form class="site-search  site-search_full">
    <input type="text" class="site-search__field">
    <input type="Submit" value ="Search" class="site-
search__button">
</form>
```

Теперь видно, что у нас есть блок названный **.site-search,** у которого

есть вложенный элемент **.site-search__field**. Также наглядно видно, что существует модификации блока **.site-search** имеющая свой собственный класс **.site-search_full**.

В БЭМ не используют селекторы тегов и идентификаторов. Стили блоков и элементов в БЭМ описываются через селекторы классов. Каждый HTML-элемент должен иметь атрибут **class**.

Ключевая идея БЭМ — независимость блоков. Вложенные селекторы увеличивают связность кода и делают его повторное использование невозможным. Методология БЭМ допускает использование таких селекторов, но рекомендует по максимуму их сократить. Например, вложенность уместна, если нужно изменить стили элементов в зависимости от состояния блока или заданной темы.

Важный принцип именования селекторов состоит в том, что имя должно полно и точно описывать сущность, представляемую селектором.
В качестве примера рассмотрим следующие четыре строки CSS-кода:

```
.button {}
.button__icon {}
.button__text {}
.button_theme_islands {}
```

С определенной долей вероятности можно предположить, что мы имеем дело с одним блоком и его HTML-реализация выглядит следующим образом:

```
<button class="button button_theme_islands">
    <span class="button__icon"></span>
    <span class="button__text">...</span>
</button>
```

Сложнее сделать подобное предположение с такой группой селекторов:

```
.button {}
.icon {}
.text {}
.theme_islands {}
```

Имена **icon, text, theme_islands** не так информативны.

Общие правила именования блоков, элементов и модификаторов позволяют:

- сделать имена CSS-селекторов максимально информативными и понятными;
- решить проблему коллизии имен;
- независимо описывать стили блоков и их опциональных элементов.

Модификаторами в БЭМ задают блокам внешний вид, состояние и поведение. Изменение оформления блока производится при помощи установки/снятия модификатора.

Пример
HTML-реализация:

```
<button class="button button_size_s">...</button>
CSS-реализация:

.button {
    font-family: Arial, sans-serif;
    position: relative;
}
```

```
.button_size_s {
    font-size: 13px;
    line-height: 24px;
}

.button_size_m {
    font-size: 15px;
    line-height: 28px;
}
```

Используя модификаторы, мы можем изменять представление блока, точечно переопределяя необходимые для этого CSS-свойства
Можно использовать другую систему именований БЭМ основанную на шаблоне в соответствии с предложениями Николаса Галлахера:

.block {}
.block__element {}
.block--modifier {}
.block — самый высокий уровень абстракции компонента.
.block__element — дочерний элемент **.block** помогающий поддерживать его целостность.
.block--modifier — модификатор, другое состояние или версия **.block**.

Двойные символы используются вместо одиночных из-за того, что класс может быть сам по себе разделён дефисом, например:

```
.site-search{} /* Блок */
.site-search__field{} /* Элемент */
.site-search--full{} /* Модификатор */
```

6

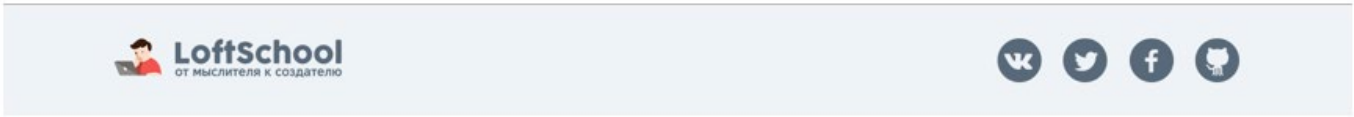
Пример верстки

За основу возьмём тестовое задание нашей школы.

Она поделена на семантические блоки выделенные красным цветом.

Заголовок страницы

Шапка страницы оформляется в тег **header**.

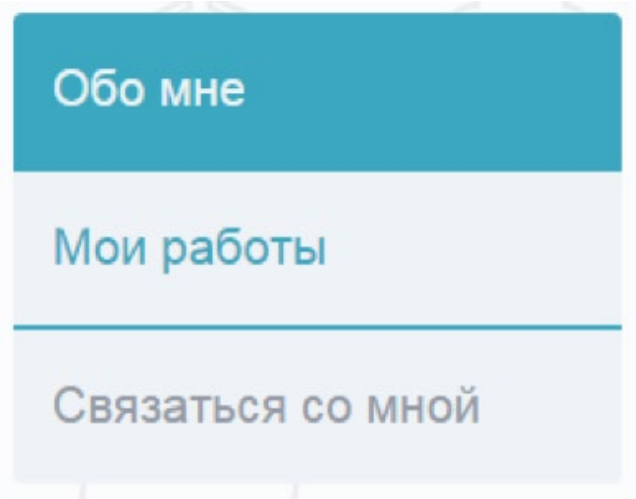



```
<header class="header">
 <div class="container header__container">
  <div class="header__left">
   <a href="/" class="logo">
    <img src="img/logo.png" alt="logo" class="logo__img">
    <h1 class="logo__text">Лого компании Loftschool</h1>
   </a>
```

```
  </div>
  <div class="header__right">
   <ul class="social">
    <li class="social__item"><a href="" class="social__link
social__link--sprite-vk">В контакте</a></li>
    <li class="social__item"><a href="" class="social__link
social__link--sprite-tw">Твиттер</a></li>
    <li class="social__item"><a href="" class="social__link
social__link--sprite-fb">Facebook</a></li>
    <li class="social__item"><a href="" class="social__link
social__link--sprite-git">GitHub</a></li>
   </ul>
  </div>
 </div>
</header>
```

Навигация

Оформление главной навигации по сайту должно заключаться в тег **nav**. Также следует помнить что хорошей практикой считается оформлять навигацию элементами списка.




```
<nav class="navigation">
    <ul class="navigation__menu">
     <li class="navigation__item navigation__item--active">
       <a href="" class="navigation__link">Обо мне</a>
     </li>
     <li class="navigation__item">
       <a href="" class="navigation__link">Мои работы</a>
     </li>
     <li class="navigation__item">
       <a href="" class="navigation__link">Контакты</a>
    </li>
   </ul>
</nav>
```

Контент на странице

Основное содержимое страницы оформляется в тег main. Это может быть одна статья, или несколько превью статей если речь идет о странице блога с несколькими записями. Нельзя помещать сайдбар, хедер страницы, футер или главную навигацию в тег main!

```
<!-- Основное содержимое страниц -->
<main>

...основной контент страницы...

</main>
```

Оформление статьи article

Тег article — служит для обертки статей. В общем этот тег содержит в себе блок контента, который может быть вынут из контекста страницы, и использован отдельно в другом месте. Это может быть статья (полный тескт статьи или превью), пост на форуме, и т.п.

Сайдбар или колонка с виджетами

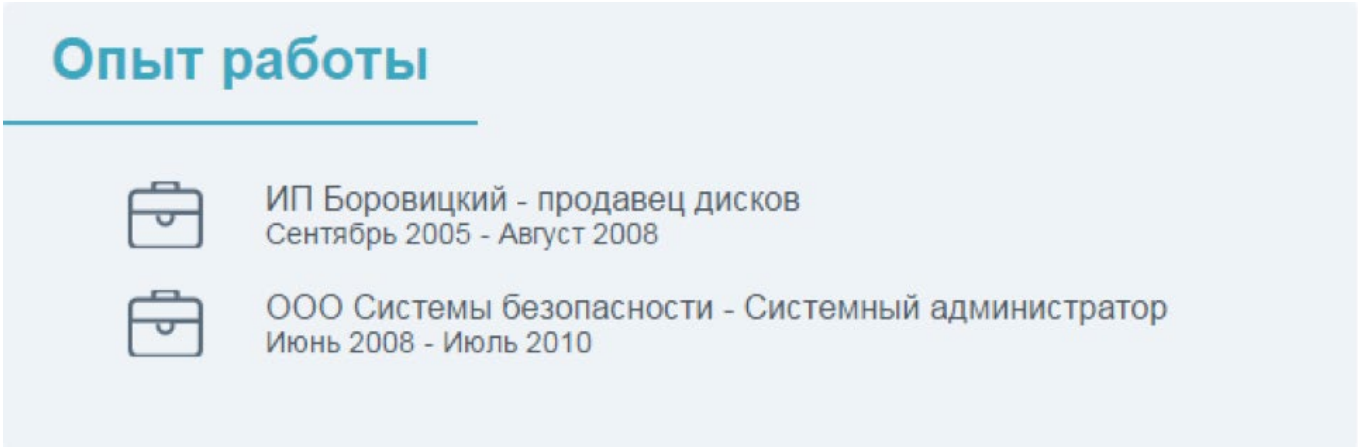
Для каждого отдельного элемента сайдбара используем блок aside. В нашем случае колонка с меню может быть обернута сайдбаром возможно в дальнейшем мы захотим добавить туда какой-то контент или виджет:

```
<aside class="sidebar">
  <nav class="navigation">
    <ul class="navigation__menu">
```

```
      <li class="navigation__item navigation__item--active">
        <a href="" class="navigation__link">Обо мне</a>
      </li>
      <li class="navigation__item">
        <a href="" class="navigation__link">Мои работы</a>
      </li>
      <li class="navigation__item">
        <a href="" class="navigation__link">Контакты</a>
      </li>
    </ul>
  </nav>
</aside>
```

Тег section

Тег section — используется для представления группы или секции тематически связанного контента. Его использование похоже на **article** с главным отличием в том что допускается отсутствие смысла содержимого внутри элемента **<section>** вне контекста самой страницы.



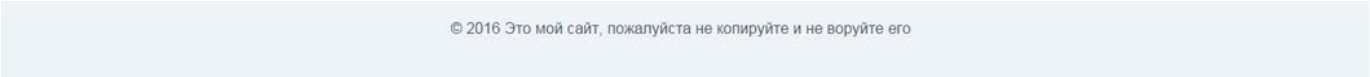
Пример использования тега section в нашем случае:

```
<section class="box">
  <h2 class="box__header">Опыт работы</h2>
  <div class="box__body">
    <ul class="box__list">
      <li class="box__item box__icon--work">
        <div class="box__where">ИП Боровицкий – продавец
дисков</div>
        <div class="box__period">Сентябрь 2005 – Август
2008</div>
      </li>
      <li class="box__item box__icon--work">
        <div class="box__where">ООО Системы безопасности –
Системный администратор</div>
        <div class="box__period">Июнь 2008 – Июль 2010</div>
      </li>
    </ul>
  </div>
</section>
```

Подвал сайта — Footer

Подвал сайта оформляется тегом **<footer>**

```
<footer class="footer">
    <div class="container">
      <div class="footer__text">
        &copy; 2016 Это мой сайт, пожалуйста не копируйте
        и не воруйте его
      </div> </div> </footer>
```



Footer прибитие

Нам элемент с классом wrapper нужен для прижатия футера к низу экрана. Он будет сидеть внизу, даже если страница почти пустая.

Структура HTML такая

```
<div class="wrapper">
   <div class="main-content">
   </div>
</div>
<footer class="page-footer">
</footer>
```

Для тегов **html** и **body** - прописываем такие стили:

```
width: 100%;
height: 100%;
padding: 0px;
margin: 0px;
```

Это нужно для обнуления всяких отступов. Если нужны отступы, их лучше прописать внутренним элементам.

Для прижатия футера нужны такие стили:

```
.wrapper{
   position:relative;
   min-height:100%;
   height:auto!important;
   overflow:hidden;
}
.wrapper:after {
   content: "";
```

```
    height: 80px;
    display: block;
}
.page-footer{
    height: 80px;
    margin-top: -80px;
}
```

Строки: **min-height: 100%; height: auto !important; height: 100%;** Этим мы растягиваем **wrapper** на всю высоту экрана. Футер находится ниже. Далее элементу **.wrapper:after** задаем **padding-bottom: 80px, где 80px** - это высота **footer'a**, а **footer'y** задаем **height: 80px; margin-top: -80px**. Вот и весь секрет. Если хотите сделать футер выше или ниже надо везде заменить **80px** на свое значение.

На самом деле способов прибития футера множество. Вы можете использовать свой.

В контрольном примере, что будет дан в конце методички, используется такой метод.

HTML:

```
<main>Lorem ipsum dolor sit amet, consectetur adipisicing
elit. Voluptates expedita non quam laudantium, voluptatem,
animi dolores. Porro amet provident voluptates pariatur quam
quasi corporis consequatur vel omnis dolore ut nesciunt
natus modi earum, eos nostrum tenetur repellendus tempore
unde aspernatur. Ea, veniam! Et atque laudantium facilis sed
libero, aspernatur minima vitae eaque numquam natus. Eveniet
sint sapiente accusamus voluptas ipsum nihil, voluptatibus!
Impedit amet illo dignissimos fugit itaque laborum,
reprehenderit temporibus libero mollitia soluta facere quos,
expedita nobis nesciunt eius voluptatem quidem aperiam
accusamus aliquam minima quasi tenetur perferendis quas
```

```
quaerat! Fuga assumenda accusamus doloremque rerum atque,
voluptates voluptate nesciunt!</main>
<footer></footer>
```

CSS:

```
html {
    position: relative;
}

html, body {
    min-height: 100%;
}

body{
    background: #ccc;
    margin: 0;
    padding-bottom: 100px;
}

main {
    width: 300px;
    margin: auto;
}

footer {
    height: 100px;
    position: absolute;
    bottom: 0;
    left: 0;
    right: 0;
    background: rgba(0,0,0,.1);
}
```

Здесь футер занимает 100px

6

Ссылки

- Вёрстка сайтов
- Пример сверстаного тестового задания
- CSS по БЭМ
- Что такое viewport?
- Условные комментарии
- Normalize.css
- Box Sizing