

# Разметка



Что такое разметка,  
особенности хорошей  
разметки, стиль кодирования,  
стайлгайды?

Першин Александр Александрович

# Итак, разметка

- Разметка — это создание HTML-структуры проекта.
- На 90% разметка состоит из дивов, секций и спанов с подходящими названиями классов.
- На 10% — из остальных тегов.



# Итак, разметка



## ОТ НОВИЧКА ДО ПРОФЕССИОНАЛА

Вы научитесь создавать современные веб-интерфейсы, работать с живым кодом, использовать новейшие технологии.



## ПРАКТИЧЕСКОЕ ОБУЧЕНИЕ

Минимум скучной теории и максимум практических упражнений, решение реальных задач и настоящие испытания.



## УЧИТЬСЯ ВЕСЕЛО!

Интересные, наглядные и затягивающие курсы, интерактивные интерфейсы, достижения – всё для обучения с удовольствием.

```
45 <div class="ha-container">
46   <div class="ha-layout-1-3">
47     <div class="ha-feature">
48       <div class="ha-feature-icon ha-feature-compass"></div>
49       <div class="ha-feature-title">От новичка<br /> до профессионала</div>
50       <div class="ha-feature-description">Вы научитесь создавать современные веб-интерфейсы,
работать с живым кодом, использовать новейшие технологии.</div>
51     </div>
52   </div>
53   <div class="ha-layout-1-3">
54     <div class="ha-feature">
55       <div class="ha-feature-icon ha-feature-clipboard"></div>
56       <div class="ha-feature-title">Практическое<br /> обучение</div>
57       <div class="ha-feature-description">Минимум скучной теории и максимум практических упражнений,
решение реальных задач и настоящие испытания.</div>
58     </div>
59   </div>
60   <div class="ha-layout-1-3">
61     <div class="ha-feature">
62       <div class="ha-feature-icon ha-feature-colors"></div>
63       <div class="ha-feature-title">Учиться<br /> весело!</div>
64       <div class="ha-feature-description">Интересные, наглядные и затягивающие курсы, интерактивные
интерфейсы, достижения&nbsp;&mdash; всё для обучения с удовольствием.</div>
65     </div>
66   </div>
67 </div>
68 </div>
```



# Исключение

Когда верстается текстовый контент, доля «обычных» тегов возрастает.



# Хорошая разметка

- Простая.
- Понятная.
- Читабельная.
- Модульная (используются пространства имён).



# Простая

```
<form class="modal-form" action="this" method="post">
  <input type="text" placeholder="Логин">
  <input type="password" placeholder="Пароль">
  <div class="clearfix row">
    <div class="save-me to-left">
      <input type="checkbox" id="remember-me">
      <label for="remember-me">Запомните меня</label>
    </div>
    <div class="new-password to-right">
      <a href="#">Я забыл пароль!</a>
    </div>
  </div>
  <button type="submit" class="btn">Войти</button>
</form>
```

```
<form class="login-form" action="/echo" method="post">
  <input type="text" placeholder="Логин">
  <input type="password" placeholder="Пароль">
  <a href="#" class="restore">Я забыл пароль!</a>
  <input type="checkbox" name="remember" id="remember-me">
  <label for="remember-me">Запомните меня</label>
  <button type="submit" class="btn">Войти</button>
</form>
```



В основном, это достигается за счет именования классов:

**.cart**

**.main-menu**

**.catalog-item**

**.nbr-razdela**

**.siniy-treugolnik**

**.idishnik**



Что это за блок или блоки?

```
<div class="slider-content">  
    
</div>  
<div class="button-combination clearfix">  
  <a href="#" class="btn to-left">...</a>  
  <a href="#" class="btn to-right">...</a>  
</div>
```





Что это за блок?

```
<div class="gallery">
  <figure class="gallery-content">
    
  </figure>
  <button class="btn gallery-prev" disabled>...</button>
  <button class="btn gallery-next">...</button>
</div>
```



```
<div class="slider-content">  
    
</div>  
<div class="button-combination clearfix">  
  <a href="#" class="btn to-left">...</a>  
  <a href="#" class="btn to-right">...</a>  
</div>
```

```
<div class="gallery">  
  <figure class="gallery-content">  
      
  </figure>  
  <button class="btn gallery-prev" disabled>...</button>  
  <button class="btn gallery-next">...</button>  
</div>
```

## ФОТОГАЛЕРЕЯ



НАЗАД

ВПЕРЕД



Форматирование кода за счёт отступов и переносов в HTML и CSS.



Какой режим работы?

```
<div class="main-contacts"><h2
class="content-column-title">Контактная
    информация</h2><p>
Barbershop «Borodinski»<br>Адрес: г. Санкт-Петербург, ул. Собаки
Павлова, д. 13<br>Телефон: +7 (495) 666-02-66
</p><p>Время работы:<br>пн – пт: с 10:00
    до 22:00<br>сб – вс: с 10:00 до 19:00</p><a
    href="#" class="btn">Как проехать</a>
<a href="#" class="btn">Обратная связь</a></div>
```



## Какой режим работы?

```
<div class="main-contacts">  
  <h2 class="content-column-title">Контактная информация</h2>  
  <p>  
    Barbershop «Borodinski»<br>  
    Адрес: г. Санкт-Петербург, ул. Собаки Павлова, д. 13<br>  
    Телефон: +7 (495) 666-02-66  
  </p>  
  <p>  
    Время работы:<br>  
    пн — пт: с 10:00 до 22:00<br>  
    сб — вс: с 10:00 до 19:00  
  </p>  
  <a href="#" class="btn">Как проехать</a>  
  <a href="#" class="btn">Обратная связь</a>  
</div>
```



# Используются пространства имён

Первый пример:

HTML:

```
<div class="features-item">  
  <div class="icon"></div>  
  <h2 class="title">Практические занятия</h2>  
  <p>На одной теории далеко не уедешь, поэтому мы обучаем на практике.</p>  
</div>
```

CSS:

```
.features-item {...}  
.features-item .icon {...}  
.features-item .title {...}  
.features-item p {...}
```



# Используются пространства имён

Второй пример:

HTML:

```
<div class="features-item">  
  <div class="features-icon"></div>  
  <h2 class="features-title">Практические занятия</h2>  
  <p>На одной теории далеко не уедешь, поэтому мы обучаем на практике.</p>  
</div>
```

CSS:

```
.features-item {...}  
.features-icon {...}  
.features-title {...}  
.features-item p {...}
```



# Зачем нужны пространства имён?

- Упрощают именование классов.
- Предотвращают нежелательное перемешивание стилей.
- Разбивают код на независимые блоки.
- Ускоряют разработку и упрощают поддержку.





# <ARTICLE>, <SECTION>, <DIV>

- **ARTICLE** — самостоятельный, независимый, отчуждаемый «кусок» содержимого.
- **SECTION** — составная часть чего-либо, может именоваться.
- **DIV** — чисто декоративное назначение, никакой семантики.



# СРЕДСТВА ДЛЯ УХОДА

ГЛАВНАЯ ♦ МАГАЗИН ♦ СРЕДСТВА ДЛЯ УХОДА

## ПРОИЗВОДИТЕЛИ:

- ☒ BAXTER OF CALIFORNIA
- ☐ MR NATTY
- ☒ SUAVECITO
- ☐ MALIN+GOETZ
- ☐ MURRAY'S
- ☒ AMERICAN CREW

## ГРУППЫ ТОВАРОВ:

- ☒ БРИТВЕННЫЕ ПРИНАДЛЕЖНОСТИ
- ☐ СРЕДСТВА ДЛЯ УХОДА
- ☐ АКСЕССУАРЫ



Набор для путешествий  
«BAXTER OF CALIFORNIA»

2 900 РУБ.

КУПИТЬ



Увлажняющий кондиционер  
«BAXTER OF CALIFORNIA»

750 РУБ.

КУПИТЬ



Гель для волос  
«SAUVECITO»

290 РУБ.

КУПИТЬ



Глина для укладки волос  
«AMERICAN CREW»



Гель для волос  
«AMERICAN CREW»



Набор для бритья  
«BAXTER OF CALIFORNIA»



# НАБОР ДЛЯ ПУТЕШЕСТВИИ «BAXTER OF CALIFORNIA»

ГЛАВНАЯ ♦ МАГАЗИН ♦ СРЕДСТВА ДЛЯ УХОДА ♦ НАБОР ДЛЯ ПУТЕШЕСТВИЙ «BAXTER OF CALIFORNIA»



ЕСТЬ В НАЛИЧИИ

АКТИКУЛ: DEXTER-AE

TRAVEL KIT – НЕОБХОДИМЫЙ АКСЕССУАР ВО ВРЕМЯ ЛЮБОГО ПУТЕШЕСТВИЯ. В АККУРАТНОЙ КОЖАНОЙ СУМКЕ НАХОДИТСЯ ВСЕ, ЧТО НУЖНО ДЛЯ БРИТЬЯ И УХОДА ЗА КОЖЕЙ ВО ВРЕМЯ РАБОЧЕЙ ПОЕЗДКИ ИЛИ ОТДЫХА: СРЕДСТВО ДЛЯ УМЫВАНИЯ, УВЛАЖНЯЮЩИЙ КРЕМ, КРЕМ ДЛЯ БРИТЬЯ, КРЕМ ПОСЛЕ БРИТЬЯ, ШАМПУНЬ. НАБОР ТАКЖЕ МОЖЕТ СТАТЬ ОТЛИЧНЫМ ПОДАРКОМ.

2 900 РУБ.

КУПИТЬ

## В НАБОР ВХОДЯТ:

- ♦ СРЕДСТВО ДЛЯ УМЫВАНИЯ (50 МЛ)
- ♦ УВЛАЖНЯЮЩИЙ КРЕМ (50 МЛ)
- ♦ КРЕМ ДЛЯ БРИТЬЯ (50 МЛ)
- ♦ КРЕМ ПОСЛЕ БРИТЬЯ, ШАМПУНЬ (50 МЛ)
- ♦ УДОБНАЯ КОЖАНАЯ КОСМЕТИЧКА



# ИСТИННО МУЖСКАЯ КЛАССИКА

## МЫ ИСПОЛЬЗУЕМ ТОЛЬКО ЛУЧШИЕ СРЕДСТВА:

- ◆ BAXTER OF CALIFORNIA
- ◆ MR NATTY
- ◆ SUAVECITO
- ◆ MALIN+GOETZ

## НЕСКОЛЬКО СЛОВ О НАС:

НАША ПАРИКМАХЕРСКАЯ ЗАНИМАЕТСЯ ИСКЛЮЧИТЕЛЬНО МУЖСКИМИ СТРИЖКАМИ. СТРИЖКА КАЖДОГО КЛИЕНТА ДЛЯ НАС - ЭТО УНИКАЛЬНАЯ И ПРОДУМАННАЯ ДО МЕЛОЧЕЙ РАБОТА. МЫ НЕ РАБОТАЕМ НА КОЛИЧЕСТВО, МЫ ДЕЛАЕМ КАЧЕСТВО.

## ЦЕНЫ НА УСЛУГИ НАШИХ МАСТЕРОВ:

СТРИЖКА	1500 Р.
СТРИЖКА БОРОДЫ	500 Р.
НАКРУТКА УСОВ	350 Р.

НАША МАСТЕРСКАЯ РАСПОЛОЖЕНА В ЦЕНТРЕ ГОРОДА, ПОЭТОМУ СДЕЛАТЬ СТИЛЬНУЮ СТРИЖКУ МОЖНО В ЛЮБОЕ ВРЕМЯ, ДАЖЕ В ОБЕДЕННЫЙ ПЕРЕРЫВ. ЗДЕСЬ ВЫ МОЖЕТЕ ПОГРУЗИТЬСЯ В УДОБНУЮ ДЛЯ ВАС АТМОСФЕРУ, ЧУВСТВОВАТЬ СЕБЯ КОМФОРТНО И СВОБОДНО!



# Споры о форматировании кода

- Разделитель в именах классов:  
`some_class`, `some-class`, `someClass`?
- Пробел или табуляция?
- Размер отступов 2 или 4 пробела? А может быть таб?
- Как сортировать CSS-свойства: по алфавиту или по назначению?



Стиль кодирования — дело личных предпочтений или соглашений внутри команды.

Главное, чтобы стиль кодирования применялся:

- последовательно.
- единообразно.





# Разработчик, у которого есть свой стиль

```
2 <html lang="ru">
3 <head>
4 <meta charset="UTF-8">
5 <title>A+</title>
6 </head>
7 <body>
8 <header class="main-header">
9 <div class="container">
10 <div class="header-top">
11 <div class="logo">
12 
13 </div>
14 <nav class="main-menu">
15 <ul>
16 <li><a href="#">Программа обучения</a></li>
17 <li><a href="#">Выпускники</a></li>
18 <li><a href="#">Как поступить</a></li>
19 <li><a href="#">Контакты</a></li>
20 </ul>
21 </nav>
22 </div>
23 <div class="promo">
24 <div class="promo-title">Учеба еще никогда не была такой интересной!</div>
25 <p>Обучаем готовых к трудоустройству специалистов на теоретических и практических курсах.</p>
26 <a href="#" class="btn btn-green">Подать заявку</a>
27 </div>
28 </div>
29 </header>
30 <section class="features">
31 <div class="container">
32 <div class="features-item">
33 <div class="features-icon features-icon-schedule"></div>
34 <h2 class="features-title">Гибкий график обучения</h2>
35 <p>Современный темп жизни требует гибкого подхода к расписанию.</p>
36 </div>
```



# Человек, овладевший стилями Муравья и Богомола

```
var js, fjs = d.getElementsByTagName(s)[0];
if (d.getElementById(id)) return;
js.src = "/connect.facebook.net/ru_ko/all.js#xfbml=1";
fjs.parentNode.insertBefore(js, fjs);
}(document, 'script', 'facebook-jssdk'));</script>

<table class="frame1 index" cellpadding="0">
  <tr>
    <td class="filler"></td>
    <td id="left_arrow">
      <br>
      <a id="nav_prevPage" href="/?offset=35" title="Показать другие работы" rel="nofollow"></a>
    </td>
    <td id="content">
      <table style="width:100%;">
        <tr>
          <td colspan="3" id="menu">
            <ul>
              <li class='menu_i_aboutgo'><a href='/company/'>Агентство </a></li>
              <li class='menu_services plus'><a href='/services/'>Услуги </a><
              <li class='menu_inner_portfolio'><a href='/portfolio/'>Портфолио
              <li class='menu_contacts'><a href='/contacts/'>Контакты </a></li>
              <li class='menu_order'><a href='/order/'>Заказ </a></li>
              <li class='menu_blog'><a href='/blog/'>Блог </a></li><li class='nimax-top-phone'><span><a href="tel:+78123092320"
class="tel">+7 (812) <strong> 309-23-20</strong></a><a href="tel:+78123855206" class="tel">+7 (812) <strong>
385-52-06</strong></a></span></li></ul>
            </td>
          </tr>
        </table>

        <table id="blocks" >
          <tr>
            <td colspan="2">
              <table class='frame'>
                <tr>
```





# Стайлгайды

Стайлгайд от MDO

<http://mdo.github.io/code-guide/>

Перевод: <http://instanceof.pro/code-guide/>

Стайлгайд от GOOGLE

<http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml>

Idiomatic CSS

<http://github.com/necolas/idiomatic-css/>



# Написание HTML-кода

## Порядок атрибутов

Для удобства чтения HTML-атрибуты должны быть указаны именно в этом порядке:

- `class`
- `id`, `name`
- `data-*`
- `src`, `for`, `type`, `href`
- `title`, `alt`
- `aria-*`, `role`

Классы создают для многократно используемых компонентов верстки, поэтому они идут первыми. Идентификаторы более специфичны и должны использоваться умеренно (например, для закладок на странице), поэтому они следуют вторыми.

```
<a class="..." id="..." data-modal="toggle" href="#">
```

Какая-то ссылка

```
</a>
```

```
<input class="form-control" type="text">
```

```

```

## Логические атрибуты

Логические атрибуты — одни из тех, которые не требуют объявленного значения. XHTML требует от вас задать значение, но в HTML5 нет такого требования.

За подробной информацией обратимся к [разделу о логических атрибутах на WhatWG](#):

```
<input type="text" disabled>
```

```
<input type="checkbox" value="1" checked>
```

```
<select>
```

```
<option value="1" selected>1</option>
```

```
</select>
```



# Порядок CSS-свойств в правилах

## Порядок объявления

Объявления логически связанных свойств должны быть сгруппированы в следующем порядке:

1. Позиционирование
2. Блочная модель
3. Типографика
4. Отображение

Позиционирование следует первым потому, что оно может удалить элемент из нормального потока документа и переопределить блочную модель связанных стилей. Блочная модель идет следующей, так как она диктует размеры и расположение компонента.

Все остальные объявления, выполняющиеся *внутри* компонента или не оказывающие влияния на предыдущие два раздела, следуют в последнюю очередь.

Для ознакомления с полным списком свойств и их порядком обратитесь к [Recess](#).

```
.declaration-order {  
  /* Позиционирование */  
  position: absolute;  
  top: 0;  
  right: 0;  
  bottom: 0;  
  left: 0;  
  z-index: 100;  
  
  /* Блочная модель */  
  display: block;  
  float: right;  
  width: 100px;  
  height: 100px;  
  
  /* Типографика */  
  font: normal 13px "Helvetica Neue", sans-serif;  
  line-height: 1.5;  
  color: #333;  
  text-align: center;  
  
  /* Отображение */  
  background-color: #f5f5f5;  
  border: 1px solid #e5e5e5;  
  border-radius: 3px;  
  
  /* Прочее */  
  opacity: 1;  
}
```



# Не используйте @import

## Не используйте @import

По сравнению с тегом `<link>` правило `@import` медленнее, создает дополнительные запросы и может вызвать иные непредвиденные проблемы. Избегайте это правило и используйте вместо него один из альтернативных подходов:

- Используйте несколько тегов `<link>`
- Компилируйте ваш CSS-код в один файл с помощью препроцессоров, например, Sass или Less
- Склеивайте ваши CSS-файлы с помощью инструментов, которые есть в Rails, Jekyll и других окружениях

Для получения дополнительной информации следует ознакомиться со [статьей Стива Соудерса](#).

```
<!-- Используйте теги link -->
<link rel="stylesheet" href="core.css">

<!-- Избегайте @imports -->
<style>
  @import url("more.css");
</style>
```



# Нет сокращённой записи свойств!

## Сокращённая запись

Старайтесь ограничить использование сокращённых объявлений в тех случаях, когда необходимо явно задать все доступные значения. Наиболее часто злоупотребляют сокращением следующих свойств:

- `padding`
- `margin`
- `font`
- `background`
- `border`
- `border-radius`

Часто нам не нужно устанавливать все значения сокращённой записи свойства. Например, HTML заголовки устанавливают только отступы сверху и снизу, таким образом, в случае необходимости нужно переопределить только эти два значения. Чрезмерное использование сокращённой записи свойств часто приводит к грязному коду с ненужными переопределениями и непреднамеренными побочными эффектами.

На сайте Mozilla Developer Network есть отличная статья о [сокращённой записи свойств](#) для тех кто не знаком с такой формой записи.

```
/* Плохой пример */  
.element {  
  margin: 0 0 10px;  
  background: red;  
  background: url("image.jpg");  
  border-radius: 3px 3px 0 0;  
}
```

```
/* Хороший пример */  
.element {  
  margin-bottom: 10px;  
  background-color: red;  
  background-image: url("image.jpg");  
  border-top-left-radius: 3px;  
  border-top-right-radius: 3px;  
}
```

