**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**«Ярославский государственный педагогический университет**

**им. К.Д. Ушинского»**

**Кафедра теории и методики обучения информатике**

**Направление 44.03.05 Педагогическое образование,   
профили «Информатика и информационные технологии в образовании», «Образование в области математики»**

**КУРСОВАЯ РАБОТА**

**На тему: «Разработка web страниц»**

**Работа выполнена студентом**

Давыденковой Татьяной Александровной

**Научный руководитель**

Кокорева Ирина Евгеньевна

**Ярославль**

**2019\_\_**

**Оглавление**

[Введение 3](#_Toc12672415)

[Глава 1. Верстка web страниц 5](#_Toc12672416)

[1.1. Основные понятия 5](#_Toc12672417)

[1.2. Классификации верстки 8](#_Toc12672418)

[1.3. Препроцессоры 11](#_Toc12672419)

[1.4. Медиа-запросы 14](#_Toc12672420)

[1.5. Семантика 14](#_Toc12672421)

[1.6. Методология 15](#_Toc12672422)

[Глава 2. Реализация проекта 18](#_Toc12672423)

[2.1. Описание проекта 18](#_Toc12672424)

[2.2. Выбор и обоснование языка программирования 19](#_Toc12672425)

[2.3. Выбор стиля, методов и средств программирования 19](#_Toc12672426)

[Заключение 21](#_Toc12672427)

[Список литературы 22](#_Toc12672428)

[Приложение 1. Стили и разметка p-index 23](#_Toc12672429)

[Приложение 2. Стили и разметка p-photo 27](#_Toc12672430)

[Приложение 3. Стили и разметка p-form 35](#_Toc12672431)

# Введение

Разработка программного обеспечения в настоящее время является важной, большой, быстрорастущей и динамично развивающейся отраслью человеческих знаний. Одной из самых востребованных отраслей создания программного обеспечения - это веб программирование. С проектами, созданными в рамках этой отрасли, люди сталкиваются чаще всего в повседневной жизни. Проекты с элементами веб-программирования встречаются: в бизнесе, в рекламе, в образовании и даже в искусстве, если хотите.

Отрасль веб-программирование имеет давнюю историю. В процессе своего развития было создано множество принципов, подходов, парадигм, паттернов, технологий и стеков технологий.

В процессе развития веб-программирования разделилось на множество взаимосвязанных отраслей, в зависимости от назначения программного обеспечения и подхода к его разработке, могут быть выделены, например, серверная часть (backend), клиентская часть (frontend). Клиентская часть, которую мы далее будем рассматривать более подробно, можно условно разделить на UI – юзер интерфейс, верстку и дизайн.

Цель курсовой работы – создание адаптивной верстки для лэндинга отдельного региона. Назначение разрабатываемой верстки: показать пример того уровня, который будущие ученики могут достигнуть в ходе занятий на факультативном курсе по верстке.

В связи с этим были поставлены следующие задачи:

1. Изучить основные понятия в верстке;
2. Изучить различные виды верстки;
3. Изучить различные популярные методы и подходы к верстке;
4. Реализовать существующий дизайн для показания примера на практике.

Программный продукт представляет собой реализованный по макету дизайн веб-страниц, обеспечивающий правильное отображение сайта на различных устройствах, подключённых к Интернету и динамически подстраивающийся под текущие размеры окна браузера в точности как на представленном макете, с учетом пожеланий заказчика.

Особенностью реализации данного проекта стало использование препроцессоров для создания HTML разметки и CSS стилей разметки – Pug и SCSS соответственно, медиа запросов, для реализации адаптивности верстки. А так же реализация SPA (Single page application) для избежания лишних перезагрузок страницы.

# Глава 1. Верстка web страниц

## 1.1. Основные понятия

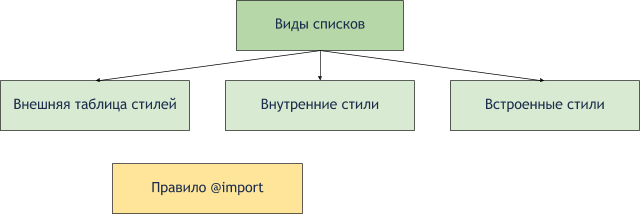
Верстка – это расположение составных элементов на странице документа. В web программировании верстка – это создание основной разметки, задающий структуру и визуальный интерфейс для дальнейшего взаимодействия с пользователем сайта. Как у живого организма есть скелет, мышцы и кожа так и у сайта есть свой скелет с мышцами – html разметка и своя кожа – визуализированный дизайн, описанный с помощью css стилей.

Для того чтобы полноценно разрабатывать верстку сайта необходимо узнать и познакомиться с основными тегами элементов, из которых может состоять страница, например, кнопки, ссылки, текста, картинки и так далее.

HTML-теги — основа HTML разметки. Теги используются для разграничения начала и конца элементов в разметке. Каждый HTML документ состоит из дерева текста и HTML-элементов, которые обозначаются начальным и конечным тегом, внутри которых содержится имя тега. Тэги по своему функционалу делятся на

1. Пустые элементы;
2. Элементы с неформатированным текстом;
3. Элементы, выводящие неформатированный текст;
4. Элементы из другого пространства имён;
5. Обычные элементы.

Помимо создания разметки нам также важно задать стили для уже созданной разметке, чтобы привести разметку к тому виду, который отобразил дизайнер в макете. Для этого надо изучить тему CSS (Cascading Style Sheets) — язык таблиц стилей, который позволяет прикреплять стиль к структурированным документам. Стили которые мы прописываем можно разделить на разные группы. Вот по такой классификации:



**Рисунок 1Классификация списков стилей**

1. Внешняя таблица стилей – представляет собой текстовый файл с расширением .css, в котором находится набор CSS-стилей элементов. Файл создаётся в редакторе кода, так же как и HTML-страница. Внутри файла могут содержатся только стили, без HTML-разметки. Внешняя таблица стилей подключается к веб-странице с помощью тега <link>, расположенного внутри раздела <head></head>;
2. Внутренние стили встраиваются непосредственно в тэг <head></head> и прописываются внутри тега <style></style>. Внутренние стили имеют приоритет над внешними, но уступают встроенным стилям (заданным через атрибут style).;
3. Встроенные стили - Когда мы пишем встроенные стили, мы пишем CSS-код в HTML-файл, непосредственно внутри тега элемента с помощью атрибута style. Такие стили действуют только на тот элемент, для которого они заданы;
4. Правило @import позволяет загружать внешние таблицы стилей. Чтобы директива @import работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами. Правило @import также используется для подключения веб-шрифтов.

Css стили описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Объявление стиля состоит из двух частей: селектора и объявления. Селекторы – это правило по которому определяется к какому компоненту прописаны данные стили.

Таблица 1Основные виды селекторов

|  |  |  |
| --- | --- | --- |
| **Пример** | **Вид селектора** | **Определение** |
| Типовые селекторы | | |
| \* | Универсальный селектор | Соответствует любому HTML-элементу. |
| div h2 | Селектор элемента | Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта. |
| .wrapper .custom-modal\_\_logo | Селектор класса | Селекторы класса позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта. |
| #input1 | Селектор идентификатора | Селектор идентификатора позволяет форматировать один конкретный элемент. Идентификатор id должен быть уникальным и на одной странице может встречаться только один раз. |
| Дочерние селекторы | | |
| h2 i | Селектор потомка | Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера. |
| h2 > i | Дочерний селектор | Дочерний элемент является прямым потомком содержащего его элемента. У одного элемента может быть несколько дочерних элементов, а родительский элемент у каждого элемента может быть только один. |
| Родственные селекторы | | |
| span ~ a | Сестринский селектор | Сестринские отношения возникают между элементами, имеющими общего родителя. |
| span + a | Первый сестринский селектор | Сестринские отношения возникают между элементами, имеющими общего родителя. |
| Селекторы атрибутов | | |
| [attr=value] | Селектор атрибута | Селекторы атрибутов выбирают элементы на основе имени атрибута или значения атрибута:  [атрибут] — все элементы, содержащие указанный атрибут |

Таблица 2Основные виды селекторов ( Продолжение)

|  |  |  |
| --- | --- | --- |
| **Пример** | **Вид селектора** | **Определение** |
| Псевдоклассы | | |
| :active :focus :hover  :not(селектор) | Селектор псевдокласса | Псевдоклассы — это классы, фактически не прикрепленные к HTML-тегам. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу. |
| :nth-child() :nth-last-child() | Селектор структурных псевдоклассов | Структурные псевдоклассы отбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках |
| :nth-of-type() :nth-last-of-type() | Селектор структурных псевдоклассов типа | Указывают на конкретный тип дочернего тега |
| Псевдоэлементы | | |
| :before :after | Селектор псевдоэлемента | Псевдоэлементы используются для добавления содержимого, которое генерируется с помощью свойства content |
| :first-line | Текстовый псевдоэлемент | Выбирает первую строку текста внутри элемента. |
| :first-letter | Текстовый псевдоэлемент | Выбирает первую букву текста внутри элемента. |

Наследование и каскад — два фундаментальных понятия в CSS, которые тесно связаны между собой. Наследование проявляется в том, что элементы наследуют свойства от своего родителя, а каскад - в том, как разные виды таблиц стилей применяются к документу, и как конфликтующие правила переопределяют друг друга.

## 1.2. Классификации верстки

Фиксированная верстка - подход создания страниц сайта, которые:

1. Имеют заданную ширину.
2. Ширина компонентов на странице не изменяется.
3. На мониторах с маленьким разрешением появляется горизонтальная полоса прокрутки.
4. Данный тип верстки не подходит для удобного отображения информации на мобильных устройствах.

Таблица 3Пример использования фиксированной верстки

|  |
| --- |
| body {  width: 1200px;  } |

Особенности:

* Используется редко, чаще всего при разработке десктопных приложений.

Резиновая верстка – подразумевает возможность компонентов сайта менять свои размеры в зависимости от размера окна браузера, растягиваться от и до указанных минимальных и максимальных размеров. ( max-width / min-width, max-height / min-height).

Таблица 4Пример использования резиновой верстки

|  |
| --- |
| body {  width: 90%;  } |

Особенности:

* Отображается «как есть», но масштабируется под размеры дисплея.
* Отдельные мелкие детали можно просмотреть, увеличив масштаб всем знакомым движением двух пальцев.

Адаптивная верстка - позволяет подстраиваться основному контейнеру и любому другому элементу сайта под разрешение экрана, делая возможным менять размер шрифта, расположение объектов, цвет и т. д. Происходит это динамически, например, с использованием медиа-запросов (@media), позволяющих автоматически определять разрешение монитора, тип устройства и подставлять указанные значения в автоматическом режиме. В примере ниже задается ширина div равная 960px для всех устройств, ширина которых < 1200px и 320px для всех устройств, ширина которых < 480px. При этом применяется второе условие, потому что оно идет позже.

Таблица 5Пример использования адаптивной верстки

|  |
| --- |
| @media all and (max-width: 1200px) {  div{  width:960px;  }  }  @media all and (max-width: 480px) {  div{  width:320px;  }  } |

Особенности:

* Сайт имеет всего несколько «вариантов отображения» в зависимости от ширины экрана или типа устройства.
* Если менять размер окна браузера, то сайт будет меняться «рывками», а не так, как мы привыкли видеть на «резиновых сайтах».

Отзывчивая верстка - это объединение резиновой и адаптивной вёрстки. При данном подходе используются как медиа-запросы, так и процентное задание ширины компонентов. Используя данный вид верстки можно с уверенностью сказать, что сайт приспособится к любому устройству.

Ниже задается ширина div равная 50% от размера родительского компонента для всех устройств, ширина которых < 1200px и 100% для всех устройств, ширина которых < 480px.

Таблица 6 Пример использования отзывчивой верстки

|  |
| --- |
| @media all and (max-width: 1200px) {  div{  width:50%;  }  }  @media all and (max-width: 480px) {  div{  width:100%;  }  } |

Особенность:

* Плавные переходы между ключевыми вариантами отображения.
* Если сайт оптимизировался под iPad, то он будет также хорошо смотреться и на Galaxy Tab, и на nVidia Shield, на любом устройстве.Здесь размещается текст пункта 1.2…… Здесь размещается текст пункта 1.2……

## 1.3. Препроцессоры

Препроцессор — это компьютерная программа, принимающая данные на входе и выдающая данные, предназначенные для входа другой программы (например, компилятора). Вот общий принцип работы препроцессора (Рисунок 2).

https://docs.google.com/drawings/u/1/d/sGHNJgM8Ub5oLYJbpvURSFA/image?w=602&h=45&rev=1&ac=1&parent=1KNEV1kIohrZFpFehTXA34UKuvAnPqCJgicQPeQw9-v8

**Рисунок 2 Процесс препроцессинга**

SCSS и Sass — это языки препроцессинга, которые компилируются в CSS. Они делают написание кода проще, предлагая для этого совсем немного рычагов. По отдельности SCSS и Sass имеют одно происхождение, однако технически у них разный синтаксис. Помимо SCSS и SASS существуют другие препроцессоры у которых есть свои преимущества и недостатки. Less, например, не имеет возможности использовать циклы для генерации новых стилей, как в SCSS, но при этом имеет возможность предоставлять доступ до css констант для javascript. Но одной из самых желанных "фич" для CSS является вложенность селекторов.

Таблица 7 Вложенность в SCSS

|  |
| --- |
| #some {  border: 1px solid red;  .some { background: white; }  }  */\* => \*/*  #some {  border: 1px solid red;  }  #some .some {  background: white;  } |

Переменные ― удобная штука и именно в препроцессорах очень легко создать константу и использовать в проекте.

Таблица 8 Создание переменной в SCSS

|  |
| --- |
| $some: red; |

А еще весьма важной частью является возможность поддерживать математические подсчеты прямо в css свойствах

Таблица 9 Реализация математических расчетов в SCSS

|  |
| --- |
| block {  $block\_width: 500px;  padding: 5px;  border: 1px solid black;  width: $block\_width - ( 5px \* 2 ) - ( 1px \* 2 );  } |

Таблица 10 Реализация математических расчетов для определения цвета в SCSS

|  |
| --- |
| .some {  $color: #010203;  color: $color;  border-color: $color - #010101;  &:hover {  color: #010203 \* 2;  }  }  */\* => \*/*  .some {  color: #010203;  border-color: #000102;  }  .some:hover {  color: #020406;  } |

Pug — это шаблонизатор и html-препроцессор, написанный на javascript для node.js. Pug имеет целый ряд преимуществ по сравнению с обычным HTML:

1. Повышение скорости написания html-кода. Синтаксис pug значительно более краток, чем обычный html.
2. Модульность.
3. Шаблонизация и наследование шаблонов
4. Переменные, условия, циклы, миксины. Все это дает возможность создавать настоящие программы для генерации html.
5. Pug чрезвычайно гибок.

Cсинтаксис pug крайне прост:

* Теги как и в html, но без угловых скобок.
* Вложенность определяется отступом — tab или пробелы.
* Классы записываются через точку.
* id записываются через решетку.
* Атрибуты записываются в скобках.
* Текст, который будет содержаться в теге записывается через пробел после обозначения тега.
* Комментарии записываются через // Комментарий в случае однострочного комментария. Для определения блока комментариев после // ставится минус, т.е

//- Блок

комментариев

## 1.4. Медиа-запросы

Медиа-запрос состоит из ключевого слова, описывающего тип устройства и выражения, проверяющего характеристики данного устройства. Медиа-запрос является логическим выражением, которое возвращает истину или ложь.

Таблица 11 Пример медиа-запроса

|  |
| --- |
| @media (max-width: 600px) {  #sidebar {display: none;}  } |

Для управления разметкой в мобильных браузерах используется метатег viewport. Изначально данный тег был представлен разработчиками Apple для браузера Safari на iOS. Мобильные браузеры отображают страницы в виртуальном окне просмотра, которое обычно шире, чем экран устройства. С помощью метатега viewport можно контролировать размер окна просмотра и масштаб.

Страницы, адаптированные для просмотра на разных типах устройств, должны содержать в разделе <head> метатег viewport.

Таблица 12 Метатэг. Стандартное описание

|  |
| --- |
| <meta name="viewport" content="width=device-width, initial-scale=1.0"> |

## 1.5. Семантика

Семантика – это не последнее слово в рассмотрение такой обширной темы как верстка, так как семантика в HTML - предоставление смысла и структуры содержимого с помощью соответствующего элемента, независимо от его стиля или внешнего вида.

Семантика обеспечивает:

* Однозначный смысл для содержимого;
* Доступность;
* Поиск и глобализацию;
* Совместимость;
* Независимость от стилей;
* Легкое управление кодом.

Каждый элемент принадлежит к определенной категории, которая группирует элементы со схожими характеристиками.

Таблица 13Категории элементов в семантике

|  |  |
| --- | --- |
| **Категория** | **Пример** |
| Мета содержимое | <meta charset="utf-8"> |
| Потоковое содержимое | <header>, <footer>, <address>, <main> |
| Секционное содержимое | <nav>, <article>, <section>, <aside> |
| Заголовочное содержимое | <h1><h2><h3><h4><h5><h6> |
| Текстовое содержимое | <a>, <abbr>, <area> |
| Встроенное содержимое | <audio>, <canvas>, <embed>, <iframe>, <img> |
| Интерактивное содержимое | <a>, <audio>, <button>, <details>, <embed>, <iframe> |

## 1.6. Методология

Методология – система принципов и способов организации и построения теоретической и практической деятельности, а также учение об этой системе. Web программирование тоже имеет свои методологии и каждый программист или команда программистов придерживается определенного способа написания кода. Кто-то называет это своим собственным стилем, кто-то вообще не задумывается над этим вопросом, но качество кода получаемого на выходе напрямую зависит от выбора этой самой методологии и того как он придерживался ее правил.

Существует множество различных подходов, но я расскажу про основные 2 методологии, которые пользуются особым спросом – БЭМ и Бутстрап методология.

БЭМ

БЭМ расшифровывается как Блок-Элемент-Модификатор. Методология CSS-архитектуры, созданная в Yandex.

Основная концепция – легкая поддержка проектов со временем и повторное использование компонентов.

Главная стратегия БЭМ состоит в организации CSS-кода в повторно используемые модули с помощью умной системы именования. Все базируется на понятии Блок – цельная логическая единица, отвечающая за одну функцию. Блок может в себе содержать Элементы, которые теряют свой смысл без обозначения Блока. Например, представим, что часы– это отдельный блок, тогда как цифры на циферблате могут быть только элементами этого самого блока. Помимо Элементов у Блока могут быть Модификаторы блока. Рассмотрим на примере с часами. Так как у нас есть блок – часы, то модификатором для этого блока будет указание, что они настенные или электронные. Но и у Элементов блока тоже есть Модификаторы, например, красная цифра на часах.

У БЭМ есть несколько преимуществ:

* Новые разработчики могут быстро понять связь между компонентами в разметке и CSS;
* Методология способствует повышению производительности в команде. Преимущества особенно заметны в крупных проектах;
* Система именования снижает риски коллизий с классами и утечку стилей;
* CSS несильно привязан к разметке в определенном месте на странице;
* CSS становится повторно используемым.

Бутстрап-методология

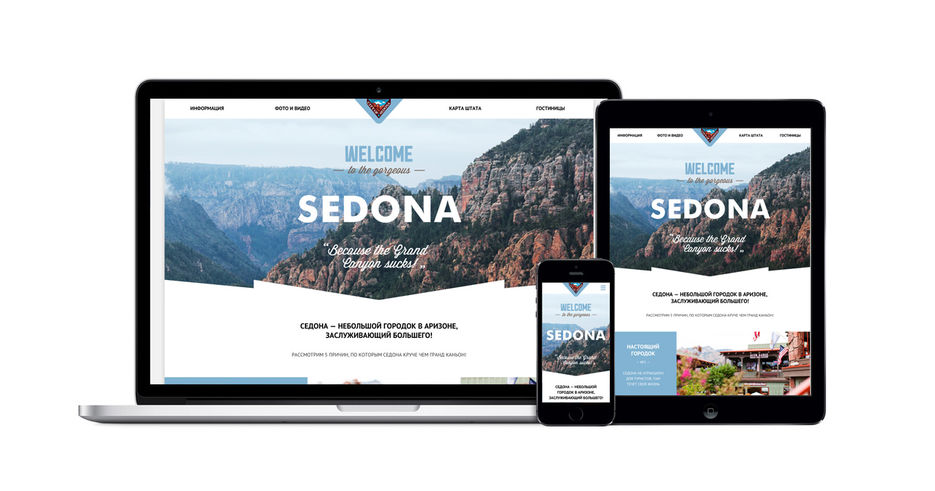
В отличие от БЭМ Бутрстрап методология базируется на уже существующей библиотечке со стилями, и в разметку добавляются классы, отвечающие за определенное поведение. Для того чтобы правильно использовать данную методология, надо отлично понимать, как тот или иной класс будет себя вести, как он повлияет на элемент, то есть хорошо знать библиотеку. Такой подход удобен для разработки прототипов или крупных банковских проектов для поддержания коммерческого стиля.

# Глава 2. Реализация проекта

## 2.1. Описание проекта

Курсовой проект представляет собой реализованную верстку сайта по уже существующему макету. Сайт, как и планировалось, адаптируется относительно размеров экрана у девайса, на котором мы просматриваем курсовой проект. В начале у нас были отдельные макеты для десктропной, планшетной и мобильной версии трех страниц – главная, фото и видео, и форма отзыва – 9 макетов + 1 макет – гайдлист для различных состояний у кнопок и инпутов в формате \*.psd. Просмотр и изучение макетов обеспечивало бесплатное программное обеспечение – Figma.

В проекте была реализована responsive верстка – то есть происходит перерасчет размеров компонентов внутри различных брейк поинтах, благодаря чему сайт будет выглядеть одинаково как на больших, так и на малых десктопных экранах (1920\*1080 и 1280\*675), то же самое относится к планшетной и мобильной версии.



**Рисунок 3 Как должен выглядеть проект на разных устройствах**

Для того чтобы переход между страницами был плавным и анимированным я реализовала SPA (Single Page Application), чтобы избежать повторных перезагрузок всего сайта, а лишь переключать контент в зависимости от того какой пункт меню был выбран.

## 2.2. Выбор и обоснование языка программирования

Была произведена первичная оценка задач проекта и выбор стека технологий для его реализации. Выбор производился на основе анализа сходных проектов, а оценки технологий, которые на анализе распространённости тех или иных инструментов, существующих в данной предметной области.

В итоге был выбран язык разметки – HTML и язык каскадных списковстилей – CSS, а также были использованы препроцессоры Pug иSASS/SCSS. Данные языки используются повсеместно и просты в использовании. Для упрощения описания структуры компонентов и ихстилей были использованы препроцессоры Pug и SСSS, так как они делает процесс написания структуры и стилей для классов и прочих элементов верстки более эффективней, а также позволяет разделить работу при необходимости между несколькими программистами, то есть поддержка модульного подхода для реализации крупного проекта.

Данные инструменты адаптированы для групповой разработки, что было бы актуально для данного проекта и дальнейшего изучения на факультативе для учащихся.

Для того чтобы организовать SPA – необходимо было прописать скрипт для переключения контента. Скрипт был написан на языке программирования – JavaScript с использованием фреймворка JQuery.

## 2.3. Выбор стиля, методов и средств программирования

В ходе разработки программного продукта был выбран модульный подход к разработке. Он является фактически стандартом в отрасли, который позволяет реализовывать работу над крупным проектом команды, а также для внесения новых правок, предлагаемых заказчиком во время разработки.

Помимо выбора модульного подхода была использована методология по БЭМ. Принцип методологии заключается в том, что, описывая класс, мы задаем ему имя исходя из того, что он может быть блоком, содержащим элемент или элементом определенного блока, или модификатором блока или элемента блока. То есть сначала определяется блок, потом его элементы, затем модификаторы. Без блока не будет ни элементов, ни модификаторов.

Такая методология делает код читабельным и понятным для участников команды разработчиков, чтобы можно было обучить учащихся работать как в одиночку, так и в команде.

Так как я реализую проект, как это делается в реальной жизни, то следует позаботиться о системе контроля версии. Для поддержки сохранности всех версий была использована распределённая система управления версиями – Git и программа, служащая графической оболочкой – Sourcetree.

А для того, чтобы организовать сборку всего проекта, состоящего из различных модулей и плагинов, был использован таск-менеджер – Gulp. Он занимается минификацией файлов, тестированием и объединением файлов, а самое главное делает это автоматически, тем самым облегчая работу над проектом.

# Заключение

Таким образом, по результатам проделанного курсового проекта можно сделать следующие выводы:

1. Я изучила основные понятия в верстке, то какой бывает верстка, какие популярные подходы бывают при разработке верстки.
2. Я реализовала существующий дизайн, там самым выполнила еще одну поставленную задачу.
3. В ходе работы над курсовым проектом, я изучила и применила новую тему – реализация SPA.
4. Выяснила зоны роста для дальнейшего изучения и исправления своих практических навыков в данной тематике курсового проекта.

В целом справилась с основными задачами на хорошем уровне.

# Список литературы

1. Tproger. Адаптивная верстка: что это и как использовать [Электронный ресурс] - https://tproger.ru/translations/responsive-web-design-tips/;
2. Webmaster. Вёрстка — это не тупо [Электронный ресурс] - https://webmasters.teamdev.com;
3. Bespoyasov Фронтенд – это не больно [Электронный ресурс] - https://bespoyasov.ru/front-not-pain;
4. Хабр. БЭМ-методология: с чего всё начиналось и зачем это всё нужно [Электронный ресурс] - https://habr.com/ru/company/yandex/blog/276035/;
5. Хабр. SCSS — немного практики, часть I [Электронный ресурс] - https://habr.com/ru/post/140612/
6. Сибирикс. Мобильная версия сайта и адаптивная вёрстка — оно вам надо? [Электронный ресурс] - https://blog.sibirix.ru/2016/06/28/adaptive-vs-mobile/
7. HTML5BOOK. Медиа запросы [Электронный ресурс] - https://html5book.ru/css3-mediazaprosy/#media

# Приложение 1. Стили и разметка p-index

Описанные стили p-index блока на SCSS

.p-index {

// Константы

@mixin mod($mod) {

@include modifier("p-index", $mod) {

@content;

}

}

@mixin item-mod($mod) {

@include modifier("p-index\_\_item", $mod) {

@content;

}

}

display: none;

overflow-x: hidden;

.custom-page\_index &{

display: block;

}

&\_\_info{

display: flex;

flex-direction: column;

align-items: center;

text-align: center;

}

&\_\_info-wrap{

$w: 458px;

margin: auto;

$p: 50px 0 55px;

@include DESKTOP {

width: vw($w);

padding: vw($p);

}

@include TABLET {

width: vw-t($w);

padding: vw-t($p);

}

@include MOBILE {

width: 70vw;

padding: vw-m(44px 0 30px);

}

}

**ПРИЛОЖЕНИЕ 1 (*Продолжение*)**

&\_\_info-list{

display: flex;

flex-direction: row;

flex-wrap: wrap;

@include MOBILE {

flex-direction: column;

}

}

&\_\_info-title{

@extend %H2;

margin: 0;

@include NOT-MOBILE{

animation: fade-in-bottom .3s both ease-in-out 0.1s;

}

}

&\_\_info-subtitle{

$mt:20px;

@include NOT-MOBILE{

animation: fade-in-bottom .3s both ease-in-out 0.3s;

}

@include DESKTOP {

margin-top: vw($mt);

}

@include TABLET {

margin-top: vw-t($mt);

}

@include MOBILE {

margin-top: vw-m($mt);

}

}

&\_\_form{

text-align: center;

@include DESKTOP {

padding: vw(57px 80px 66px);

}

@include TABLET {

padding: vw-t(57px 80px 66px);

}

@include MOBILE {

width: vw-m(280px);

**ПРИЛОЖЕНИЕ 1 (*Продолжение*)**

margin: auto;

padding: vw-m(33px 0 49px);

}

}

&\_\_form-text{

@include DESKTOP {

margin: vw(25px 0 52px);

}

@include TABLET {

margin: vw-t(25px 0 52px);

}

@include MOBILE {

margin: vw-m(10px 0 25px);

}

}

&\_\_map{

width: 100%;

text-align: center;

overflow: hidden;

@include DESKTOP {

padding: vw(57px 0 0);

}

@include TABLET {

padding: vw-t(57px 0 0);

}

@include MOBILE {

padding: vw-m(33px 0 49px);

}

iframe{

$h:350px;

width: 100%;

// height: 100%;

@include DESKTOP {

height: vw($h);

}

@include TABLET {

height: vw-t($h);

}

**ПРИЛОЖЕНИЕ 1 (*Продолжение*)**

@include MOBILE {

height: vw-m($h);

}

}

}

&\_\_map-title{

@extend %FORM-TITLE;

text-align: center;

@include NOT-MOBILE{

animation: fade-in-top .3s 3s ease-in-out both;

}

margin-bottom: vw(25px);

@include TABLET {

margin-bottom: vw-t(25px);

}

@include MOBILE {

margin-bottom: vw-m(25px);

}

}

}

Разметка для p-index блока на Pug

mixin p-index(content, params)

.p-index(id=content.id data-id=content.id)&attributes(attributes)

+p-index\_\_info(content.info)

.p-index\_\_map

.p-index\_\_map-title!=content.form.title

iframe(src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d52385.13565774555!2d-111.8301578595786!3d34.854443815278216!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x872da132f942b00d%3A0x5548c523fa6c8efd!2z0KHQtdC00L7QvdCwLCDQkNGA0LjQt9C-0L3QsCA4NjMzNiwg0KHQqNCQ!5e0!3m2!1sru!2sru!4v1553087302454" frameborder="0" style="border:0" allowfullscreen)

mixin p-index\_\_info(content, params)

.p-index\_\_info&attributes(attributes)

.p-index\_\_info-wrap

h2.p-index\_\_info-title!=content.title

.p-index\_\_info-subtitle!=content.subtitle

.p-index\_\_info-list

+pug-list(content.list, "fact")

# Приложение 2. Стили и разметка p-photo

Описанные стили p-photo блока на SCSS

.p-photo {

// Константы

@mixin mod($mod) {

@include modifier("p-photo", $mod){

@content;

}

}

@mixin item-mod($mod) {

@include modifier("p-photo\_\_item", $mod){

@content;

}

}

@mixin item-hover {

.p-photo\_\_item:hover &{

@include DESKTOP{

@content;

}

}

}

display: none;

user-select: none;

overflow-x: hidden;

.custom-page\_photo &{

display: block;

}

&\_\_info{

text-align: center;

box-sizing: border-box;

@include DESKTOP{

padding: vw(113px 0px 77px);

width: vw(428px);

margin: auto;

}

@include TABLET{

padding: vw-t(76px 0px);

width: vw-t(428px);

margin: auto;

}

@include MOBILE{

padding: vw-m(44px 20px 33px);

**ПРИЛОЖЕНИЕ 2 (*Продолжение*)**

}

}

&\_\_info-title{

@extend %H2;

margin: 0;

@include NOT-MOBILE{

animation: fade-in-bottom .3s both 0.1s;

}

}

&\_\_info-text{

$m:20px;

@include NOT-MOBILE{

animation: fade-in-bottom .3s both 0.2s;

}

@include DESKTOP{

margin-top: vw($m);

}

@include TABLET{

margin-top: vw-t($m);

}

@include MOBILE{

margin-top: vw-m($m);

}

}

&\_\_gallery{

@include NOT-MOBILE{

display: flex;

flex-wrap: wrap;

}

@include TABLET{

padding: vw-t(0 60px);

}

@include MOBILE{

border-bottom: 1px solid #b2b2b2;

}

}

&\_\_item{

position: relative;

@include NOT-MOBILE{

animation: fade-in-left 1s .3s ease-in-out both;

**ПРИЛОЖЕНИЕ 2 (*Продолжение*)**

}

@include DESKTOP{

flex: 1 1 33%;

height: vw(267px);

&\_big{

flex: 1 1 100%;

height: vw(400px);

}

}

@include TABLET{

flex: 0 1 50%;

// height: vw-t(267px);

&\_big{

flex: 1 1 100%;

// height: vw-t(400px);

}

}

@include MOBILE{

padding: vw-m(22px 20px);

box-sizing: border-box;

border-top: 1px solid #b2b2b2;

}

@include NOT-MOBILE{

@for $i from 1 through 4{

&:nth-of-type(#{$i}){

animation: fade-in-left 1s 0.5s\*$i ease-in-out both;

}

}

}

}

&\_\_item-info{

position: relative;

@include DESKTOP{

position: absolute;

left: 0;

right: 0;

bottom: 0;

background: rgba($BLACK,0.3);

color: $WHITE;

padding: vw(21px 22px);

transition: padding .3s, background .3s;

**ПРИЛОЖЕНИЕ 2 (*Продолжение*)**

@include item-hover{

padding-bottom: vw(100px);

background: rgba($BLACK,0.5);

}

}

@include TABLET{

padding: vw-t(21px 22px);

background: $GRAY;

}

}

&\_\_item-title{

font-weight: bold;

}

&\_\_author{

text-transform: none;

}

&\_\_image{

position: relative;

width: 100%;

overflow: hidden;

height: 100%;

@include MOBILE{

height: vw-m(280px);

margin-bottom: vw-m(21px);

}

@include TABLET{

height: vw-t(267px);

@include item-mod(big){

height: vw-t(400px);

}

}

img{

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

@include DESKTOP{

min-height: 100%;

width: 100%;

}

**ПРИЛОЖЕНИЕ 2 (*Продолжение*)**

@include NOT-DESKTOP{

height: 100%;

}

}

}

&\_\_likes{

display: flex;

@include NOT-MOBILE{

position: absolute;

}

@include DESKTOP{

right: vw(22px);

top: vw(33px);

}

@include TABLET{

right: vw-t(22px);

top: vw-t(33px);

}

}

&\_\_likes-icon{

$w: 24px;

$h: 22px;

$m: 9px;

@include MOBILE{

fill: #b2b2b2;

width: vw-m($w);

height: vw-m($h);

margin-right: vw-m($m);

}

@include TABLET{

fill: #b2b2b2;

width: vw-t($w);

height: vw-t($h);

margin-right: vw-t($m);

}

@include DESKTOP{

fill: $WHITE;

width: vw($w);

**ПРИЛОЖЕНИЕ 2 (*Продолжение*)**

height: vw($h);

margin-right: vw($m);

}

}

&\_\_bonus{

display: flex;

flex-direction: column;

align-items: center;

padding: 40px;

text-align: center;

@include DESKTOP{

padding: vw(72px 0 85px);

}

@include TABLET{

padding: vw-t(68px 0 40px);

}

@include MOBILE{

padding: vw-m(39px 0 49px);

}

}

&\_\_bonus-title{

@extend %H2;

margin: 0;

@include NOT-MOBILE{

animation: fade-in-bottom 0.3s 2.3s ease-in-out both;

}

}

&\_\_bonus-text{

$m: 20px 0 76px;

$w: 400px;

@include NOT-MOBILE{

animation: fade-in-bottom 0.3s 2.4s ease-in-out both;

}

@include DESKTOP{

width: vw($w);

margin: vw($m);

}

@include TABLET{

width: vw-t($w);

margin: vw-t($m);

**ПРИЛОЖЕНИЕ 2 (*Продолжение*)**

}

@include MOBILE{

width: vw-m(280px);

margin: vw-m(20px 0 40px);

}

}

&\_\_video{

@include NOT-MOBILE{

animation: fade-in 1s 2.6s ease-in-out both;

}

}

}

Разметка для p-photo блока на Pug

mixin p-photo(content, params)

.p-photo(id=content.id data-id=content.id)&attributes(attributes)

+p-photo\_\_info(content.info)

.p-photo\_\_gallery

+pug-list(content.gallery, "p-photo\_\_item")

+p-photo\_\_bonus(content.bonus)

mixin p-photo\_\_info(content, params)

.p-photo\_\_info&attributes(attributes)

h2.p-photo\_\_info-title!=content.title

.p-photo\_\_info-text!=content.text

mixin p-photo\_\_item(content, params)

.p-photo\_\_item(class=`p-photo\_\_item\_${content.class}`)&attributes(attributes)

.p-photo\_\_image

img(src=content.image)

.p-photo\_\_item-info

.p-photo\_\_item-title!=content.title

.p-photo\_\_author Автор фото #{content.author}

.p-photo\_\_likes

.p-photo\_\_likes-icon

include like.svg

span #{content.likes}

mixin p-photo\_\_bonus(content, params)

.p-photo\_\_bonus&attributes(attributes)

h2.p-photo\_\_bonus-title!=content.title

**ПРИЛОЖЕНИЕ 2 (*Продолжение*)**

.p-photo\_\_bonus-text!=content.text

+video(content.video).p-photo\_\_video

# Приложение 3. Стили и разметка p-form

Описанные стили p-photo блока на SCSS

.p-form {

// Константы

@mixin mod($mod) {

@include modifier("p-form", $mod){

@content;

}

}

@mixin item-mod($mod) {

@include modifier("p-form\_\_item", $mod){

@content;

}

}

display: none;

text-align: center;

overflow-x: hidden;

.custom-page\_feedback &{

display: block;

}

&\_\_info{

@include DESKTOP{

padding: vw(92px 0 60px);

}

@include TABLET{

padding: vw-t(71px 0);

}

@include MOBILE{

padding: vw-m(0 20px 20px);

box-sizing: border-box;

}

}

&\_\_form{

display: flex;

flex-wrap: wrap;

justify-content: space-between;

@include DESKTOP{

padding: vw(0 97px);

}

@include TABLET{

padding: vw-t(0 60px);

}

**ПРИЛОЖЕНИЕ 3 (*Продолжение*)**

@include MOBILE{

flex-direction: column;

}

}

&\_\_title{

@extend %FORM-TITLE;

width: 100%;

$m: 25px 0;

@include NOT-MOBILE{

animation: fade-in-bottom .3s both ease-in-out 0.1s;

}

@include DESKTOP{

margin: vw($m);

}

@include TABLET{

margin: vw-t($m);

}

@include MOBILE{

margin: vw-m(41px 0 30px);

}

}

&\_\_text{

width: 100%;

text-align: center;

@include NOT-MOBILE{

animation: fade-in-bottom .3s both ease-in-out 0.3s;

}

@include MOBILE{

br{

display: none;

}

}

}

&\_\_item{

flex: 0 0 100%;

margin: 0;

box-sizing: border-box;

@include NOT-MOBILE{

@for $i from 1 through 5{

**ПРИЛОЖЕНИЕ 3 (*Продолжение*)**

&:nth-of-type(#{$i}){

animation: fade-in-bottom 1s 0.5s\*$i ease-in-out both;

}

}

}

&\_small{

flex: 0 0 40%;

@include TABLET{

text-align: left;

}

}

&\_full-name{

text-align: left;

@include MOBILE(){

order:-2;

}

}

&\_contacts{

@include MOBILE(){

order:-1;

}

}

@include MOBILE{

padding: vw-m(0 20px);

text-align: center;

border-top: 1px solid #e6e6e6;

}

}

&\_\_item-title{

position: relative;

@extend %H2;

@include MOBILE{

margin: vw-m(20px 0);

}

@include item-mod(big){

@include DESKTOP{

margin: vw(40px 0);

}

@include TABLET{

**ПРИЛОЖЕНИЕ 3 (*Продолжение*)**

margin: vw-t(40px 0);

}

}

@include item-mod(small){

@include DESKTOP{

margin: vw(0 0 50px);

}

@include TABLET{

margin: vw-t(0 0 50px);

}

}

&:before{

content: '';

clear: both;

position: absolute;

top: 0;

bottom: 0;

left: 0;

right: 0;

margin: auto;

display: none;

width: 100%;

height: 1px;

background: #333333;

}

span{

position: relative;

display: block;

margin: auto;

box-sizing: border-box;

background: $WHITE;

width: auto;

padding: 0;

}

@include item-mod(big){

@include DESKTOP{

span{

width: vw(400px);

padding: vw(0 40px);

}

**ПРИЛОЖЕНИЕ 3 (*Продолжение*)**

&:before{

display: block;

}

}

@include TABLET{

span{

width: vw-t(400px);

padding: vw-t(0 40px);

}

&:before{

display: block;

}

}

}

}

&\_\_item-list{

display: flex;

justify-content: space-between;

@include item-mod(small){

flex-direction: column;

text-align: left;

}

@include TABLET{

flex-direction: column;

@include item-mod(attractions){

flex-direction: row;

flex-wrap: wrap;

width: 75%;

margin: auto;

}

}

@include MOBILE{

margin: vw-m(20px 0);

flex-direction: column;

@include item-mod(emotions){

margin: 0;

}

}

}

&\_\_button{

**ПРИЛОЖЕНИЕ 3 (*Продолжение*)**

margin: auto;

@include DESKTOP{

margin: vw(58px 0 66px);

}

@include TABLET{

width: vw-t(324px);

margin: vw-t(51px auto 40px);

}

@include MOBILE{

margin: vw-m(51px auto 40px);

}

}

}

Разметка для p-photo блока на Pug

mixin p-form(content, params)

.p-form(id=content.id data-id=content.id)&attributes(attributes)

.p-form\_\_info

h2.p-form\_\_title!=content.title

.p-form\_\_text!=content.text

+p-form\_\_form(content)

mixin p-form\_\_form(content, params)

form.p-form\_\_form&attributes(attributes)

+pug-list(content.list, "p-form\_\_item")

+button.p-form\_\_button отправить отзыв

mixin p-form\_\_item(content, params)

.p-form\_\_item(class=`p-form\_\_item\_${content.size} p-form\_\_item\_${content.class}`)&attributes(attributes)

.p-form\_\_item-title

span!=content.title

.p-form\_\_item-list

if content.inputs

+pug-list(content.inputs, "input")

if content.textarea

+textarea(content.textarea)

if content.checkbox

+pug-list(content.checkbox, "checkbox")

if content.radio

+pug-list(content.radio, "checkbox")(type="radio" name="radio")