3. Верстка

Наконец можно перейти к верстке. Верстка была выполнена по методологии БЭМ с использованием препроцессора scss.

3.1 HTML

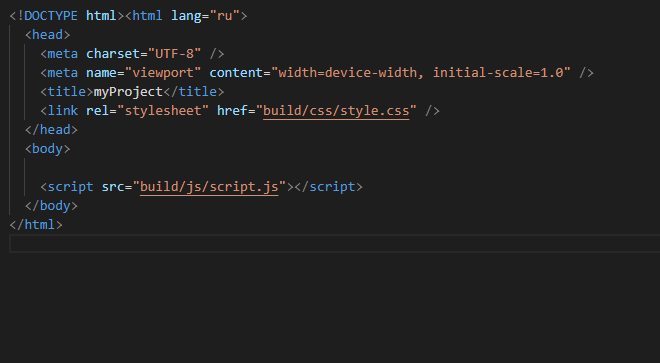
В сборке gulp уже будет создан файл index.html. Открыв его можно увидеть стандартную разметку html документа (рис. 3.1.1).

Рис 3.1.1 Стандартная html разметка

Первым делом необходимо изменить содержимое тега title на название своего проекта.

Так как это готовая сборка gulp, стили и скрипты уже подключены к странице. Стили подключены с помощью тега link в head, а скрипты с помощью тега script в body. Как было сказано ранее, они берутся из папки build куда gulp, предварительно оптимизировав, посылает их из папки src. Не трудно догадаться, что все скрипты и стили мы писать в src.

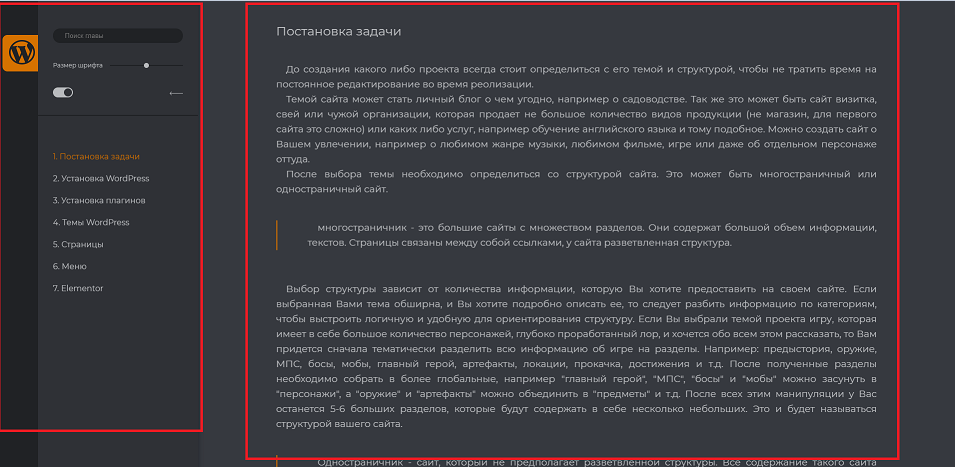
На макете видно, что страницу можно разделит на 2 большие области (рис. 3.1.2). Меню слева и область с текстом с права.

Рис 3.1.2 Разделение на 2 области

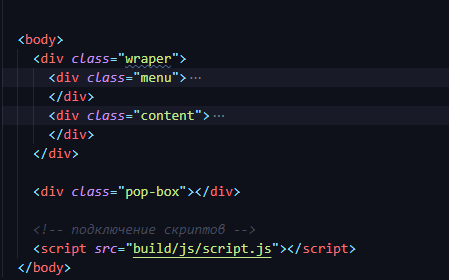
Пусть область с меню будет называться «menu», а область справа «content». Так как они, 2 блока, рассоложены горизонтально, можно использовать flex box. Для того чтобы использовать flex box, необходимо сделать эти блоки дочерними, обернем из в блок «wraper» (рис. 3.1.3).

Рис 3.1.3 Создание родительского блока

Если внимательно посмотреть на макет, можно увидеть, что меню поделена на 2 области: темы; панель с навигацией, поиском, переключателем темы и т.д. Назовём эти области «theme» и «step». «step» можно разделит на дочерние элементы: «step\_\_top», где будут находиться поиск, переключение темы, изменение шрифта и кнопка для сворачивания меню; и «step\_\_chapter», где будет навигация по главам темы. По такому принципу необходимо делить все компоненты на подкоманды, и чтобы не описывать каждый созданный блок, будет представлена финальная разметка меню со всеми созданными дочерними элементами:

<div class="menu">

<div class="holder">

<div class="preloader"><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>

</div>

<div class="theme">

<ul class="theme\_\_content">

<li class="theme\_\_item theme\_\_item--active"><a target="\_self" href="index.html"><svg width="52" height="49" viewBox="0 0 52 49" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M26 0C11.6643 0 0 10.9913 0 24.5C0 38.0087 11.6643 49 26 49C40.3358 49 52 38.0087 52 2…"/>

</svg>

</a>

</li>

<!-- <li class="theme\_\_item">

<a target="\_self" href="2.html">

<svg width="36" height="40" viewBox="0 0 36 40" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M33.4702 14.3078H32.5097V9.6734C32.5097 9…" fill="black"/>

</svg>

</a>

</li> -->

</ul>

</div>

<div class="step">

<div class="step\_\_wraper">

<div class="step\_\_top">

<input placeholder = "Поиск главы" type="text" class="search "></input>

<div class="size-rugulator">

<p>Размер шрифта</p> <input id="range\_size" type="range" value="16" min="16" max="22" step="1" class="size-rugulator\_\_sider" oninput="resizeText()">

</div>

<div class="menu\_\_panel">

<div class="switch"></div>

<div class="btn\_\_hide btn-menu"><svg width="27" height="8" viewBox="0 0 27 8" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M0.646446 3.64645C0.451185 3.84171 0…" fill="#B9BBBE"/>

</svg>

</div>

</div>

</div>

<div class="step\_\_chapter fadeIn"></div>

<div class="arrow-scroll arrow-scroll--hide">

<a href="#0" class="prev">

<svg width="8" height="38" viewBox="0 0 8 38" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M4.35356 0.646446C4.15829 0.451183 …" fill="#202225"/>

</svg>

</a>

<a href="#1" class="next">

<svg width="8" height="38" viewBox="0 0 8 38" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M3.64644 37.3536C3.84171 37.5488 4…" fill="#202225"/>

</svg>

</a>

</div>

</div>

</div>

</div>

Были скрыты настройки svg картинок так как они занимаю слишком много места и не являются обязательными для демонстрации.

Можно заметить блок «preloader». Он нужен для того чтобы не показывать содержимое страницы пока не загрузились все картинки и текст. Находиться он в «menu» так как при смене темы, так же будет меняться содержимое навигации, и чтобы скрыть резкое изменение от пользователя блок должен находиться там. Находясь там он будет закрывать собой все кроме панели с темами.

Еще можно заметить такие классы как «fadeIn». Был подключен Animate.css, это библиотека с готовой анимацией для элементов. Такие классы используется для обозначения, элемента которому нужно применить анимацию. Название такого класса определяет какая именно анимация будет присваивается элементу.

Основная информация содержится в компонента «content». Текст содержит элементы «content--text\_\_item», который разделяет главы. В каждой главе есть заголовок H2, текст p, картинки и иногда выделенный текст (с черточкой слева) «selection-text». У заголовков помимо класса есть так же и id, так называемые якоря. Это нужно для настройки навигации. Картинки необходимо отобразить по-разному: большие изображения должны быть большими, а маленькие маленькими. Для этого есть 3 класса с разными css свойствами: «img\_\_big», «img\_\_middle», «img\_\_small». Вот пример разметки одного раздела:

<div class="content--text\_\_item">

<h2 class="content\_\_title" id="0">Постановка задачи</h2>

<p>До создания какого либо проекта всегда стоит определиться с его темой и структурой, чтобы не тратить время на постоянное редактирование во время реализации. </p>

<p>Темой сайта может стать личный блог о чем угодно, например о садоводстве. Так же это может быть сайт визитка, свей или чужой организации, которая продает не большое количество видов продукции (не магазин, для первого сайта это сложно) или каких либо услуг, например обучение английского языка и тому подобное. Можно создать сайт о Вашем увлечении, например о любимом жанре музыки, любимом фильме, игре или даже об отдельном персонаже оттуда.</p>

<p>После выбора темы необходимо определиться со структурой сайта. Это может быть многостраничный или одностраничный сайт.</p>

<div class="selection-text">

<p>многостраничник - это большие сайты с множеством разделов. Они содержат большой объем информации, текстов. Страницы связаны между собой ссылками, у сайта разветвленная структура.</p>

</div>

<p> Выбор структуры зависит от количества информации, которую Вы хотите предоставить на своем сайте. Если выбранная Вами тема обширна, и Вы хотите подробно описать ее, то следует разбить информацию по категориям, чтобы выстроить логичную и удобную для ориентирования структуру. Если Вы выбрали темой проекта игру, которая имеет в себе большое количество персонажей, глубоко проработанный лор, и хочется обо всем этом рассказать, то Вам придется сначала тематически разделить всю информацию об игре на разделы. Например: предыстория, оружие, МПС, босы, мобы, главный герой, артефакты, локации, прокачка, достижения и т.д. После полученные разделы необходимо собрать в более глобальные, например "главный герой", "МПС", "босы" и "мобы" можно засунуть в "персонажи", а "оружие" и "артефакты" можно объединить в "предметы" и т.д. После всех этим манипуляции у Вас останется 5-6 больших разделов, которые будут содержать в себе несколько небольших. Это и будет называться структурой вашего сайта.</p>

<div class="selection-text">

<p>Одностраничник - сайт, который не предполагает разветвленной структуры. Все содержание такого сайта умещается на одной-единственной странице</p>

</div>

<p>Если же Вы выбрали в качестве темы лишь описание одного персонажа, и это не делиться на большое количество разделов, то вполне можно уместить все на одной странице. При этом не стоит забывать про удобство пользования и все равно разделить информацию на секции.</p>

<p>При составлении многостраничной структуры желательно визуально ее закрепить в виде схемы в тетрадке или в ворде, поверьте во время разработки сайта Вы не раз и не два будете к ней обращаться.</p>

<p>Вот пример создания структуры сайта колледжа: </p>

<div class="img\_\_big">

<img src="build/img/wp/0.1.png" alt="схема сайта колледжа">

</div>

</div>

Можно отметить «pop-box». Это пустой блок, который всегда занимает половину высоты экрана и по сути он не зависит от других блоков, так как абсолютно от позиционирован. Он нужен для того чтобы программа понимала на каком сейчас разделе находится пользователь и, например, подсвечивала нужный пункт в меню навигации.

3.2 CSS

Страница была поделена на 2 компонента, которые должны располагаются горизонтально. Для этого они должны стать flex элементами, это делается следующим образом: родительскому элементу, то есть элементу «wraper», добавляется display: flex, после чего они автоматически встанут горизонтально. Flex очень удобная технология которая появилась не так давно в css. Везде где нужно быстро расположить элементы вертикально или горизонтально в программе используется flex.

Так же не так давно были добавлены переменные в css, которые могут запомнить цвет, шрифт, размер и т.п. Для приложения были выбраны определённые цвета и для каждой была создана переменная. Сделано это для того, чтобы при желании поменять цветовую палитру не было необходимости менять все вручную, персонально для каждого элемента. Вот переменные программы:

:root {

--mainColor: #202225;

--bgMain: #ebebeb;

--bgMenu: #d2d2d2;

--bgTopPanel: #b5b5b5;

--extraColor: #d67200;

--whContainer: 1234px;

--colorShadow: #606265;

transition: all 0.3s;

}

Например, при нажатии на тумблер переключения тем, везде, где цвета указаны через переменные цвета будут меняться на другие.

Браузер - это программа которая имеет свои правила отображения. Поэтому для каждого браузера есть свои префиксы. Gulp позволяет не заморачиваться над каждым, ведь он ставит их автоматически.

Для элементов, которые должны менять свое состояние в течении сессии создаются и описываются 2 класса, например, «menu» и «menu\_\_hide», где в меню прописаны основные свойства, а в «menu\_\_hide» свойство для сворачивания (допустим display:none;). Класс «menu\_\_hide» присваивается элементу, когда надо (в основном через javaScript) тем самым сворачивая этот самый элемент.

Css теперь можно использовать для создания анимации. Уже было сказано об Animate css, который позволяет с анимировать появление или исчезновение элемента. Данная библиотека использует ключевые кадры и сложную смену свойств. Помимо использования таких анимации, css предлагает псевдоэлементы. Одним из них является hover, который определяет свойства при наведении. Используется, например, в меню навигации.

К сожалению, css код довольно громоздкий, что это не позволяет разместить его здесь целиком, но посмотреть его можно в github репозитории «https://github.com/JalsanSarylov/jalsansarylov.github.io/tree/master/srcWebLesson». Вот небольшой пример стилизации блока для выделенного текста:

.selection-text {

padding: 47px 0px;

margin-bottom: 1rem;

& p {

position: relative;

margin-left: 63px;

&::before {

content: "";

height: 100%;

width: 2px;

background-color: $extraColor;

position: absolute;

left: -63px;

top: 0%;

border-radius: 10px;

}

}

}

3.3 JavaScript

Используется для оживления приложения. Важно понимать, что сейчас идет речь о браузерном js а не о js который используется в node.

Для более удобного обращения к элементам DOM, используется библиотека JQuery. Иногда она просто необходима если речь идет о подключении дополнительных библиотек, так как на нем не только написано большинство сайтов, но и большинство js инструментов. Для использования необходимо его подключить самым первым.

Весь js код представляет из себя одну большую функцию которая запускается после погрузки всего содержимого. За это отвечает строчка $(document).Ready(). Важно инициализировать js после загрузки всех элементов, в противном случае скрипт может запуститься раньше, чем нужный DOM элемент успеет прогрузиться, из-за этого могут быть ошибки.

Как было написано выше (в разделе HTML) в программе есть Preloader который не показывает контент пока он не прогрузиться. $(document).Ready() проверяет лишь загрузился ли элемент DOM, а не, например, готовность картинки. Для решения этой задачи отвечает библиотека imagesLoaded (которая требует JQuery). Основная ее задача выполнить код после полной загрузки всех картинок на странице.

По умолчанию Preloader работает, и чтобы выключить его нужно написать функцию, которая будет отвечать за это:

Function offPreloader(){

let preloader = $(".holder");

setTimeout(() => {

preloader.css("opacity","0");

// удаление после прогрузки страницы

setTimeout(() => {

if($(".holder").css('opacity') == "0"){

preloader.remove();

}

}, 500);

}, 500);

}

А затем вызвать ее в нужном месте:

$('#container').imagesLoaded( function() {

offPreloader();

});

Приложение умеет запоминать выбранную тему после перезагрузки, а также размер шрифта и главу на которой остановился пользователь. Для того чтобы хранить эти данные существует кэш браузера. К нему можно обращаться с помощью localStorage. Данные сохраняются даже после перезагрузки компьютера.

Для того чтобы не тратить бумагу попусту, вкратце опишу каждой функции скрипта, а, чтобы подробно изучить их код можно перейти в git репоизитории. Функции:

* offPreloader() - как и было сказано ранее выключает preloader;
* darkThemeOn() – включает темную тему;
* resizeText() – меняет размер шрифта в зависимости от ползунка в меню;
* createNav() – анализирует главы и создает навигацию для страницы;
* markerStep() – окрашивает в особой цвет главу на которой находится пользователь;
* searchStep() – анализирует введённый в input текст и скрывает не совпадающие с ним главы.
* themeSwitchOn() – переключает тему;
* menuHide() – скрывает меню и меняет ширину контейнера;
* lazyScroll() – делает перемотку к якорю более плавной.