Приднестровский государственный университет им. Т.Г. Шевченко
Инженерно-технический институт

РАЗРАБОТКА АДАПТИВНОЙ ВЁРСТКИ ВЕБ-САЙТА ПРИ ПОМОЩИ ТЕХНОЛОГИЙ HTML5 И CSS3

Лабораторный практикум

Разработал:
ст. преподаватель
кафедры ИТиАУПП
Бричаг Д.В.

г. Тирасполь
2021 г.

Лабораторная работа №3

Медиа запросы в CSS. Создание адаптивных элементов для различных диагоналей экранов.

Цель работы: ознакомиться с принципами создания адаптивного макета. Создание элементов, подстраивающихся под ширину макета и соответствующих свойств элемента.

Теоретическая справка

Настало время рассмотреть подробнее виды селекторов в CSS. Селекторы представляют структуру веб-страницы. С их помощью создаются правила для форматирования элементов веб-страницы. Селекторами могут быть элементы, их классы и идентификаторы, а также псевдоклассы и псевдоэлементы.

Универсальный селектор

Соответствует любому HTML-элементу. Например, * {margin: 0;} обнулит внешние отступы для всех элементов сайта. Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом: *:after {CSS-стили}, *:checked {CSS-стили}.

Селектор элемента

Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта. Например, h1 {font-family: Lobster, cursive;} задаст общий стиль форматирования всех заголовков h1.

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. Например, для создания заголовка с классом headline необходимо добавить атрибут class со значением headline в открывающий тег <h1> и задать стиль для указанного класса. Стили, созданные с помощью класса, можно применять к другим элементам, не обязательно данного типа.

Если элемент имеет несколько атрибутов класса, их значения объединяются с пробелами.

```
<h1 class="headline post-title">Заголовок первого уровня</h1>
. headline.post-title {
        text-transform: uppercase;
        color: lightblue;
}
```

Селектор идентификатора

Селектор идентификатора позволяет форматировать один конкретный элемент. Значение id должно быть уникальным, на одной странице может встречаться только один раз и должно содержать хотя бы один символ. Значение не должно содержать пробелов.

Нет никаких других ограничений на то, какую форму может принимать id, в частности, идентификаторы могут состоять только из цифр, начинаться с цифры, начинаться с подчеркивания, состоять только из знаков препинания и т. д.

Уникальный идентификатор элемента может использоваться для различных целей, в частности, как способ ссылки на конкретные части документа с использованием идентификаторов фрагментов, как способ нацеливания на элемент при создании сценариев и как способ стилизации конкретного элемента из CSS.

```
<div id="sidebar"></div>
#sidebar {
width: 300px;
float: left;
}
```

Вёрстка на мобильных дисплеях. Viewport и медиа-запросы

Viewport – это видимая пользователю область веб-страницы. То есть это то, что может увидеть пользователь, не прибегая к прокрутке.

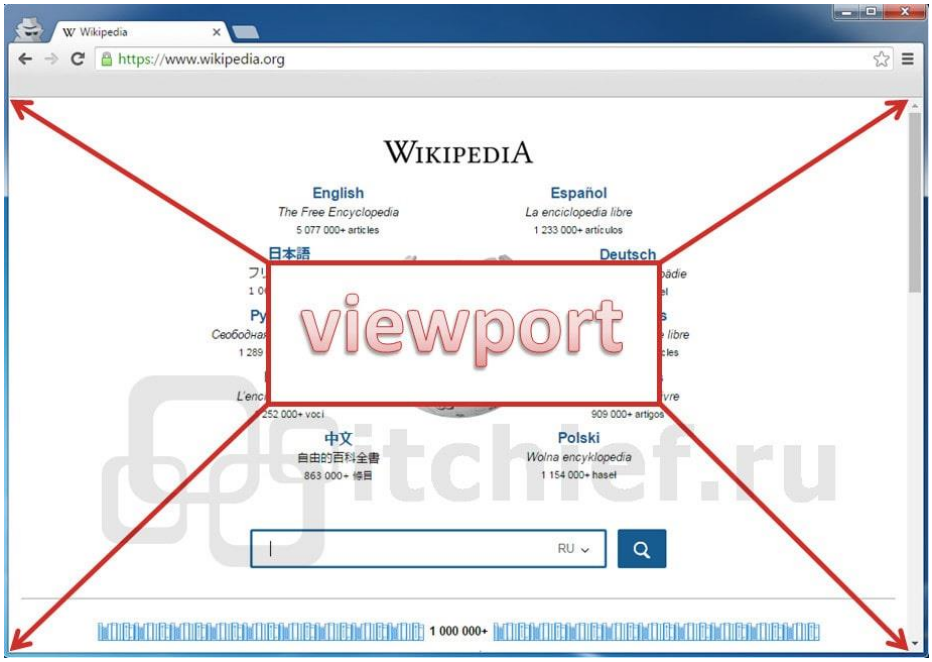

Рис. 23 – Размер viewport на экране монитора

Размеры этой области определяются размером экрана устройства. Самую маленькую область просмотра (viewport) имеют смартфоны, размеры экранов которых колеблются от 4" до 6". А самую большую - мониторы компьютеров, размеры диагоналей которых могут превышать 24".

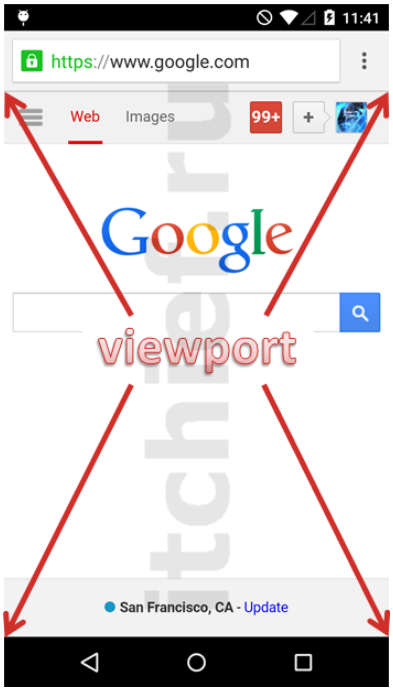

Рис. 24 – Размер viewport на экране смартфона

До появления смартфонов и планшетов, веб-страницы в основном просматривались на экранах компьютерах. Viewport этих экранов хоть и отличался, но не настолько сильно. Для создания сайтов до появления мобильных устройств в основном использовалась фиксированная или резиновая (гибкая) разметка.

После того, как появились смартфоны и планшеты, viewport одних устройств стал сильно отличаться от viewport других устройств. Это привело к тому, что сайты, созданные для компьютеров, стало невозможно или затруднительно просматривать на смартфонах. Выходом из этой ситуации послужило появление адаптивной разметки. Адаптивная – это такая разметка, которую можно настроить под различные размеры экранов. Осуществляется создание адаптивной разметки с помощью медиа запросов, которые появились в спецификации CSS3 и в настоящий момент поддерживаются всеми основными браузерами.

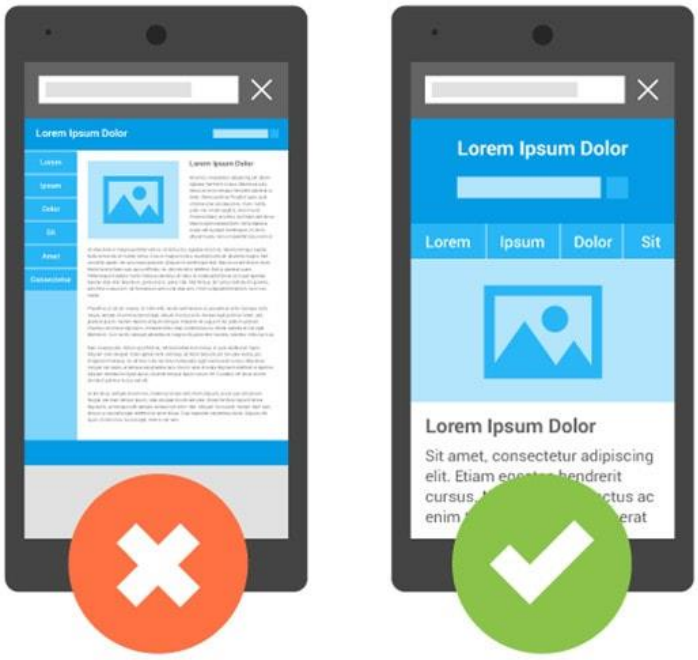

Рис. 25 – Схематическое отображение одного HTML-документа на мобильном дисплее без настроек viewport(слева) и с ними(справа)

Но и у адаптивной разметки появились проблемы после того, как появились смартфоны с высокой плотностью пикселей и, следовательно с высоким разрешением. Чтобы более детально разобраться в этой ситуации рассмотрим следующий пример, в котором сравним 2 устройства.

Рис. 26 – Сравнение дисплеев двух мобильных устройств

Первое устройство – это смартфон Apple iPhone 3 (диагональ 3.5"). Данный телефон не имеет высокую плотность пикселей. У данной модели она составляет 163ppi (меньше 200ppi). Физическое разрешение данного смартфона составляет 320x480. Такое разрешение соответствует диагонали, если его сопоставить с разрешением мониторов настольных устройств (компьютеров). Т.е. на веб-странице этого смартфона, текст, выполненный размером 16px, будет также хорошо читаемым как на мониторе компьютера.

Второе устройство – это смартфон Apple iPhone 4. Он имеет диагональ такую же как у смартфона Apple iPhone 3, т.е. 3.5". Но отличается от него тем, что имеет высокую плотность пикселей (326ppi). Следовательно, более высокое разрешение - 640x960 при тех же размерах экрана. Это приведёт к тому, что тот же

самый текст и остальные объекты веб-страницы будут выглядеть в нём при тех же условиях в 2 раза меньше. Таким образом, текст будет реально выглядеть на 8px. Такая страница будет уже трудночитаемой. Чтобы сделать эту страницу пригодной для чтения, её представление необходимо увеличить в горизонтальном и вертикальном направлении в 2 раза (отмасштабировать).

Назначение метатега viewport

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 meta 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.

Что такое медиа-запрос

В общем случае медиа-запрос состоит из ключевого слова, описывающего тип устройства (необязательный параметр) и выражения, проверяющего характеристики данного устройства. Из всех характеристик чаще всего проверяется ширина устройства width. Медиа-запрос является логическим выражением, которое возвращает истину или ложь.

Медиа-запросы могут быть добавлены следующими способами:

1) С помощью HTML:

<link rel="stylesheet" media="screen and (color)" href="example.css">

2) С помощью правила @import внутри элемента <style> или внешней таблицы стилей:

@import url(color.css) screen and (color);

3) Непосредственно в коде страницы:

```
<style>
@media (max-width: 600px) {
  #sidebar {display: none;}
}
</style>
```

4) Внутри таблицы стилей style.css:

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

Таблица стилей, прикрепленная через тег <link>, будет загружаться вместе с документом, даже если её медиа-запрос вернет ложь.

Современные браузеры представляют из себя в том числе и мощный инструмент для разработки. Одна из функций консоли разработчика браузера является функция эмуляции устройств. С её помощью можно смотреть как ведёт себя вёрстка страницы на разных устройствах и диагоналях.

В браузере Google Chrome или Mozilla Firefox нажмите клавишу F12. Откроется консоль разработчика (в других браузерах, использующих движок Chrome горячая клавиша может быть другой, например, в Opera по умолчанию это консоль вызывается комбинацией Ctrl+Shift+C). Найди и нажмите кнопку активации эмулятора устройств, его иконка указана на рисунке 27 под цифрой 1.

Слева, под цифрой 2 обозначен переключатель устройств. Здесь можно выбрать из готовых шаблонов или произвольные параметры.

Справа, под цифрой 3 вы найдёте меню консоли разработчика. Там быстро можно переключить её расположение в окне браузера по своему вкусу. Это лишь малая часть возможностей консоли, но на данный момент важно уметь пользоваться переключателем устройств.

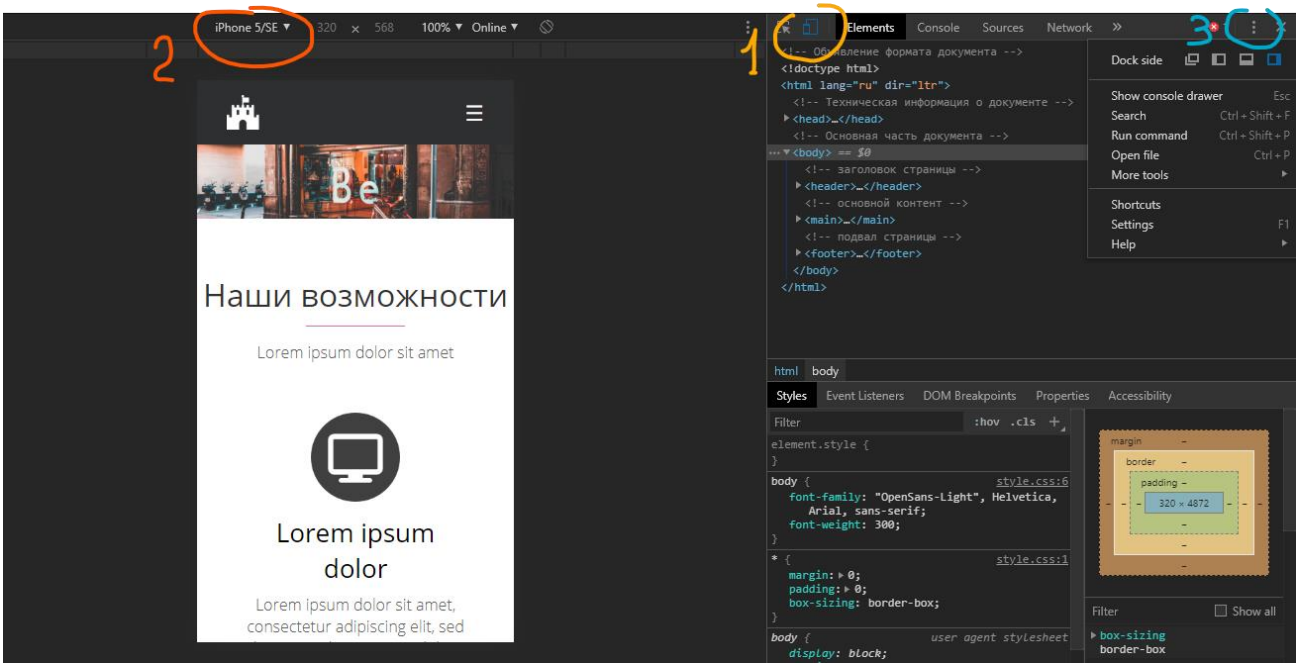

Рис. 27 – Панель разработчика в Google Chrome

CSS-фон

Каждый блок html-элемента имеет фоновый слой, который может быть полностью прозрачным (по умолчанию) или заполнен цветом и/или одним или несколькими изображениями. CSS-свойства фона указывают, какой цвет background-color и изображения background-image использовать, а также их размер, расположение, способ укладки и т.д.

Свойства фона не наследуются, но фон родительского блока будет просвечивать по умолчанию из-за начального значения в background-color: transparent.

Фон не отображается у пустых элементов с нулевой высотой. Отрицательные значения свойства margin не влияют на фон элемента.

Краткая запись свойств фона: свойство background

Свойство background позволяет описать в одном объявлении следующие свойства фона: background-color, background-image, background-position, background-size, background-repeat, background-origin, background-clip и background-attachment. Необязательно указывать все перечисленные свойства, если какое-либо свойство будет пропущено, оно примет значение по-умолчанию.

Если вы указываете в краткой записи фона свойство background-size, то его значения нужно будет записать через слеш /, чтобы отделить его от свойства background-position.

Синтаксис:

background: gold;

background: url("rose.png") repeat-y;

background: border-box red;

background: no-repeat center/80% url("../img/icon.png");

Множественные фоны

Фон блока элемента может иметь несколько слоев в CSS3. Количество слоев определяется количеством значений, разделенных запятыми, указанных в свойстве background-image. Значение none по-прежнему создает слой.

Пример

```
div {
width: 680px;
height: 630px;
background-image:url(https://awesome.com/img/flower_rose.png),
                 url(https://awesome.com/img/flower_rose.png),
                 url(https://awesome.com/img/flower_rose.png);
background-repeat: no-repeat;
background-position: bottom right, center center, top left;
}
```

Первое изображение в списке — это слой, отображаемый ближайший к пользователю, следующий отрисовывается за первым, и так далее. Цвет фона, если он есть, закрашивается под всеми остальными слоями.

Практическая часть

Во второй лабораторной работе мы создали блок услуг, который хорошо отображается на экране настольного ПК или ноутбука. Но лишь половина пользователей посещают страницы с таких устройств. Другая половина, открыв нашу страницу с мобильного устройства, увидит такую разметку как изображено на рисунке 24. Браузер просто отрисует всё в те размеры экрана, что ему доступны. Но прочитать текст на таком масштабе будет невозможно.

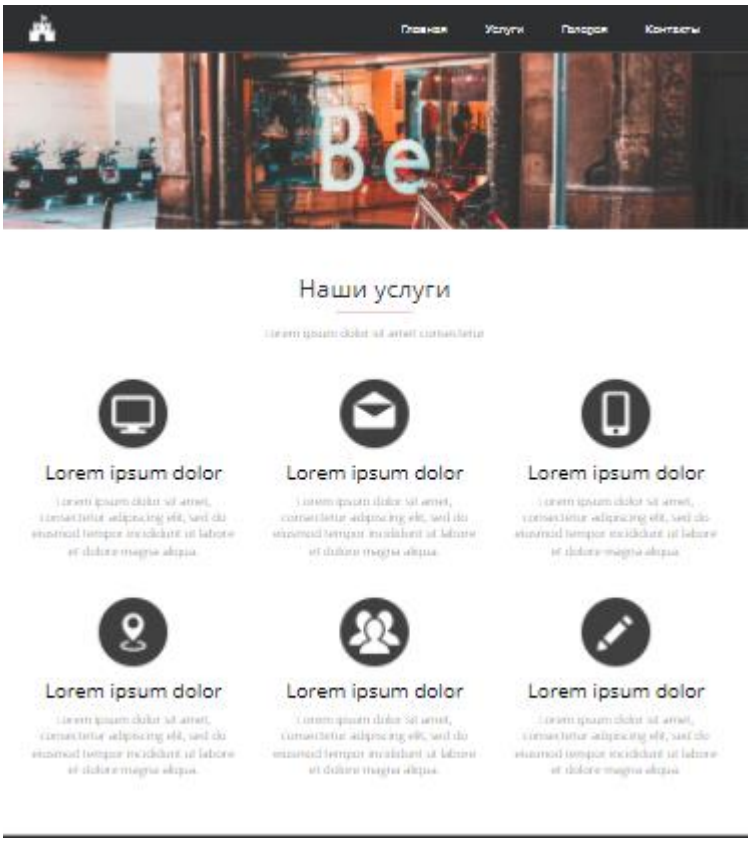

Рис. 28 – Отображение страницы на мобильном устройстве в реальных пропорциях

Наша страница не адаптирована под мобильную вёрстку и это могло быть серьёзной проблемой, но не для нас, ведь мы используем современные возможности верстальщика. Сначала добавим мета-тег в <head> сайта:

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

Теперь можем создавать медиа-запросы в CSS с указанием максимальной ширины дисплея, на которую распространяется правило. На рисунке 29 продемонстрированы стили для диагоналей меньше 992px (большие планшеты) и 768px (обычные планшеты или альбомное положение смартфона):

```
@media screen and (max-width: 992px) {
  .row-item {
    width: 45%;
  }
}

@media screen and (max-width: 768px) {
  .container-row {
    flex-direction: column;
  }
  .row-item {
    width: unset;
  }
}
```

Рис. 29 – Медиазапросы со стилями для блоков

Для диагоналей меньше 992px расположили наши элементы в 2 колонки, каждая шириной в 45%, а для диагонали меньше 768px – изменили направление расположения элементов в флекс-контейнере на колоночное, а ограничение ширины элементов убрали, чтобы они заняли всю ширину своего контейнера.

Если ещё уменьшить ширину экрана, то заметим, что меню страницы уходит за её пределы. На большинстве современных адаптивных сайтов меню скрыто за значком «гамбургера». Поступим аналогично: спрячем пункты меню при помощи медиа-запросов, а раскрытие мобильного меню по клику сделаем при помощи JavaScript в следующих работах.

Добавьте в меню после тега кнопку:

☰

Строка ☰ соответствует значку меню в юникоде стандарта HTML5. В файле стилей скроем эту иконку:

.icon {

```
  display: none;
}
```

А в медиа-запросе для 992px отобразим и увеличим её размер:

```
 .icon {
   display: flex;
   margin: auto 0;
   color: white;
   padding: 5px 10px;
   cursor: pointer;
   border: 1px solid white;
   border-radius: 4px;
   font-size: 20px;  }
```

Отлично, теперь она будет отображаться если ширина экрана меньше 992px. Здесь же для медиа запроса 992px укажем стиль, позволяющий скрыть остальные пункты меню:

```
 nav li {
   display: none;
   padding: 15px 0;
 }
```

В спецификации к классу margin можно узнать, что свойство auto действует только на горизонтальные отступы. А для нашей кнопки мы задали это свойство для вертикальных отступов, и оно сработало. Почему так? Этот трюк работает для элементов внутри контейнера flex, работая подобно justify-content: center и align-items: center вместе.

Результат должен соответствовать рисунку 30, проверьте в консоли разработчика, что вёрстка не выходит за экран.

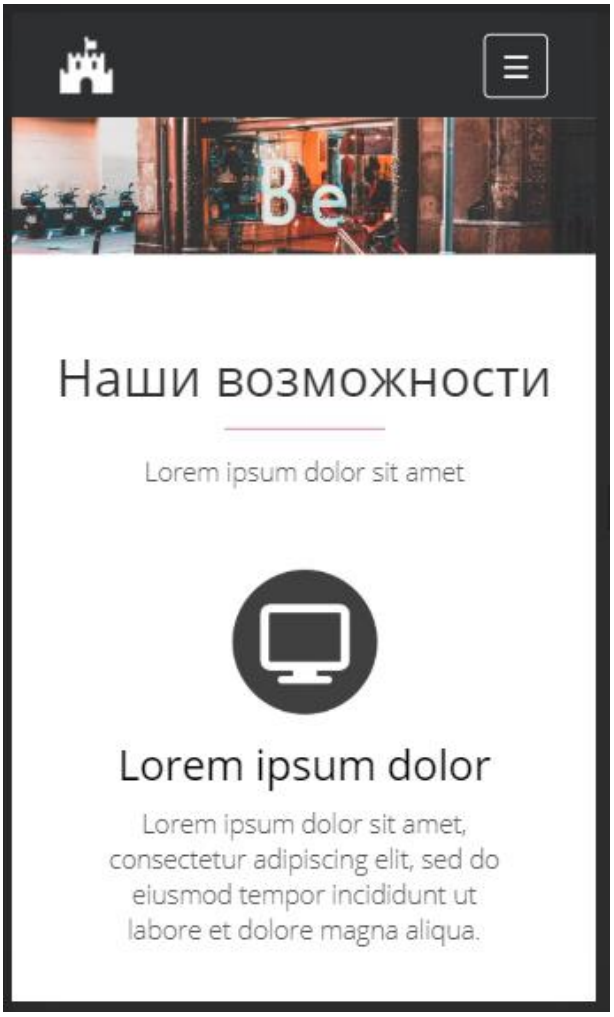

Рис. 30 – Результат работы медиазапросов

Добавим на нашу страницу, по аналогии с предыдущим разделом, ещё один блок с галереей фотографий. В предыдущих работах мы уже познакомились с тегом <img> и создали обложку и логотип для сайта. Но, кроме этого, часто встречается иной подход размещения изображений на странице. В этом случае блоку создают фон изображением. К плюсам такого подхода можно отнести более высокую гибкость настройки фонов, а значит и самого изображения. Также появляется возможность размещать другие элементы поверх изображения, без применения абсолютного позиционирования, которое управляется другими правилами и сложнее поддерживать в адаптивной вёрстке.

Важно отметить минус такого подхода. Поисковым машинам не будет известно о содержании картинки, она не будет отображаться в поисковой выдаче и

не будет анализироваться поисковыми алгоритмами при похожих запросах. Поэтому такой подход нельзя применять на страницах, где картинки несут основную смысловую нагрузку, например, в портфолио фотоателье, в онлайн магазинах и т.п.

Приступим: новую секцию с классом "portfolio" и вставим в него заголовок второго уровня <h2>, разделитель и описание, как для раздела возможностей. На рисунке показано, что заголовок и описание не стилизованы, но у нас стилизован разделитель. Для предыдущего раздела мы создали стили для текста с привязкой к родительскому контейнеру, а не к классу. Исправим это, чтобы не повторяться в будущем и иметь возможность применять повторно классы.

Рис. 31 – Новый заголовок раздела

Добавим заголовку и описанию классы в блоке возможностей и портфолио.

```
41        <section class="services">
42          <h1 class="main-title">Наши возможности</h1>
43          <div class="delimiter"></div>
44          <p class="main-title-description">Lorem ipsum dolor sit amet</p>
45          <!--  внешний блок -->
46 ›        <div class="services-container">
88        </section>
89        <section class="portfolio">
90          <h2 class="main-title">Наши работы</h1>
91          <div class="delimiter"></div>
92          <p class="main-title-description">Lorem ipsum dolor sit amet</p>
93        </section>
```

Рис. 32 – Верстка нового раздела

Обратите внимание на строку 46 скриншота. Блок "services-container" свёрнут, для удобства перемещения по коду. В большинстве редакторов поддерживается такая опция сворачивания контейнеров.

Теперь изменим селектор .services h1 на .main-title и добавим селектор .main-title-description.

```
59  .main-title {
60    text-align: center;
61    font-size: 32px;
62    font-weight: 400;
63    color: #313131;
64  }
65  .delimiter {
66    width: 100px;
67    margin: 10px auto 15px;
68    height: 1px;
69    background: rgba(206, 65, 110, 0.74);
70  }
71  .main-title-description {
72    color: #616161;
73    text-align: center;
74  }
```

Рис. 33 – Стили для заголовков разделов

Теперь мы можем создавать блоки с заголовком и описанием, задавая лишь класс, без необходимости дублировать код.

Приступим к созданию непосредственно галереи. Скопируйте папку portfolio из методических материалов в папку img проекта. Там находится набор изображений различных размеров.

Создадим блок "portfolio-container", в нём 6 одинаковых блоков вида:

```
<div class="portfolio-item" style="background-image: url('img/portfolio/img09.jpg')">
  <div class="description-container">
    <p class="portfolio-title">Lorum dolorum</p>
    <p class="portfolio-description">
      Duis aute irure dolor in reprehenderit in voluptate velit esse
      cillum dolore eu fugiat nulla pariatur.
    </p>
  </div>
</div>
```

Рис. 34 – Наполнение раздела

Текст заголовка и описания значения не имеет. Картинки подберите на ваш выбор. Теперь стилизуем наш блок:

```
.portfolio-container {
  margin: 50px 0; /* отступы сверху и снизу */
  display: flex; /* создание флекс-контейнера */
  flex-wrap: wrap; /* перенос флекс-элементов, если не помещаются в ряд */
}
.portfolio-item {
  background-size: cover; /* заполнение фонового изображения */
  background-position: center; /* центрирование фонового изображения */
  width: 33.33%; /* создание размера элемента */
  height: 250px; /* высота элемента */
  display: flex; /* создание флекс-контейнера */
  align-items: center; /* выравнивание флекс-контейнера по вертикали */
}
.description-container {
  padding: 20px; /* равные поля элемента */
  display: flex; /* создание флекс-контейнера */
  flex-direction: column; /* изменение направления дочерних элементов */
  justify-content: center; /* выравнивание дочерних элементов по горизонтали в центре родительского контейнера */
```

text-align: center; /* выравнивание текста */

opacity: 0; /* прозрачность (от 0 до 1) */

height: 0; /* высота */

background: #2d2f31; /* фоновый цвет */

transition: all 250ms ease-in-out; }/* эффект перехода между двумя состояниями элемента */

Если обратить внимание, мы задали тёмный фон каждому блоку, но установили прозрачность 0, что соответствует полной прозрачности элемента. Теперь если мы можем создать интересный эффект появления текста, при наведении на блок регулируя прозрачность.

Важно понимать, что именно прозрачность даёт возможность создавать эффекты появления/исчезновения. Меняя свойство display: block/none плавного перехода добиться нельзя. Это свойство отвечает за метод отображения контейнера, где display: none – это отсутствие элемента в потоке документа. С opacity: 0 элемент не исчезает из основного потока, а лишь становится прозрачным, но он по-прежнему влияет на другие элементы и его можно выделить. Если необходимо убрать элемент – используйте display: none

Итак, для создания плавного перехода добавьте следующий стиль:

```
.portfolio-item:hover .description-container {
  opacity: 0.8;
  height: 100%; }
```

И, дополнительно, поправим текст в контейнерах:

```
.portfolio-title,
.portfolio-description {
  color: white; }
.portfolio-title {
  font-weight: 400;
  font-size: 22px;
```

margin-bottom: 5px; }

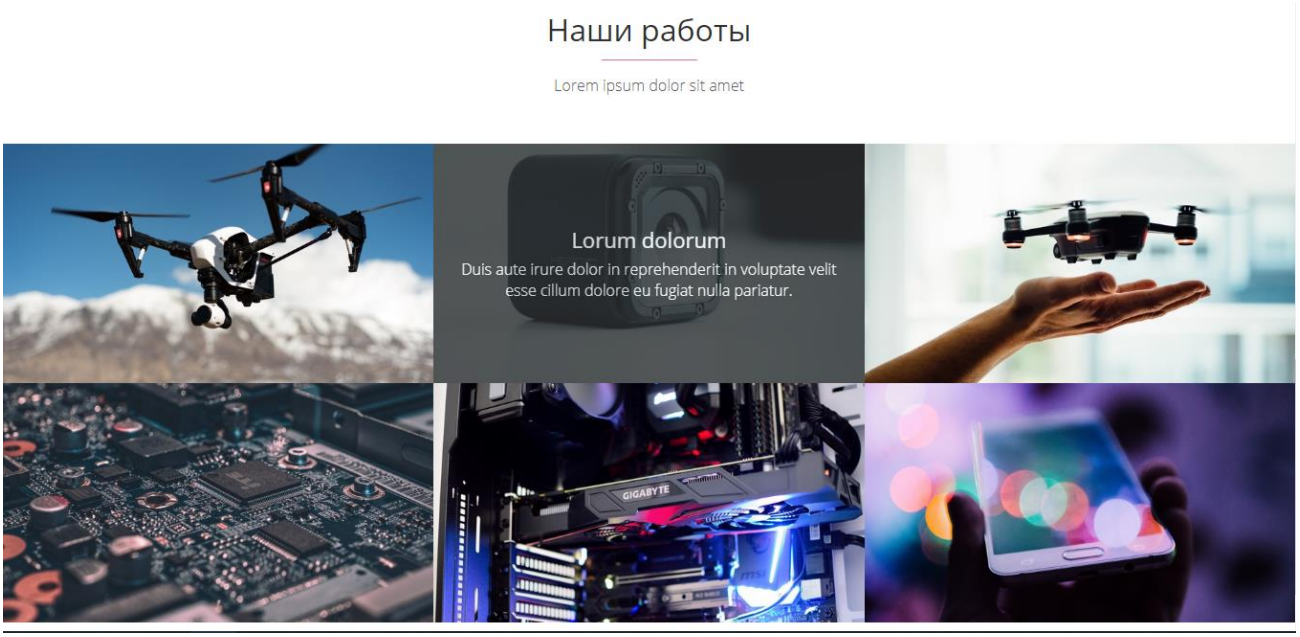

Рис. 35 – Внешний вид нового раздела

Для настройки адаптивности надо добавить лишь два стиля, для медиазапроса 992px:

```
.portfolio-item {
  width: 50%;
}
```

И добавить ещё один медиазапрос для портретной ориентации смартфонов:

```
@media screen and (max-width: 576px) {
  .portfolio-item {    width: 100%;   }
}
```

Обратите внимание, никаких настроек изображения больше не нужно, хотя сами изображения разных размеров. Фоновое изображение заполняет блок, а лишнее не отображается. Фоновые изображения позволяют создавать элементы оформления с большой гибкостью и широко применяются там, где не нужен акцент на самом изображении.

На самом деле фоновые изображения занимают важное место в оформлении страниц. Их используют как иконки, фоны, разделители, заполнители и многое другое. Рассмотрим ещё один пример, простого, но эффектного блока, с применением картинки в качестве фона.

Добавим перед секцией с портфолио ещё одну секцию с классом overlay и типовым заголовком с описанием, а под текстом добавим кнопку.

```
85      <section class="overlay">
86        <h2 class="main-title">Подробности</h2>
87        <div class="delimiter"></div>
88        <p class="main-title-description">Lorem ipsum dolor sit amet</p>
89        <button class="common-button" type="button" name="button">Узнать</button>
90      </section>
```

Рис. 36 – Разметка секции

Применим следующие стили для секции и кнопки, чтобы изменить её стандартный вид:

```
143  .overlay {
144    margin: 80px 0;
145    padding: 80px 30px;
146    background: #2d2f31 url('../img/02.jpg') no-repeat;
147    background-size: cover;
       text-align: center;
   }
150  .common-button {
151    border: 1px solid #2d2f31;
152    background-color: transparent;
153    padding: 15px 25px;
154    border-radius: 4px;
155    color: #2d2f31;
156    font-size: 20px;
157    cursor: pointer;
158    margin-top: 30px;
159    transition: all 250ms ease;
160  }
161  .common-button:hover {
162    color: #fff;
163    background-color: #2d2f31;
```

Рис. 37 – Стили для секции с фоном

У нас должен получится такой результат:

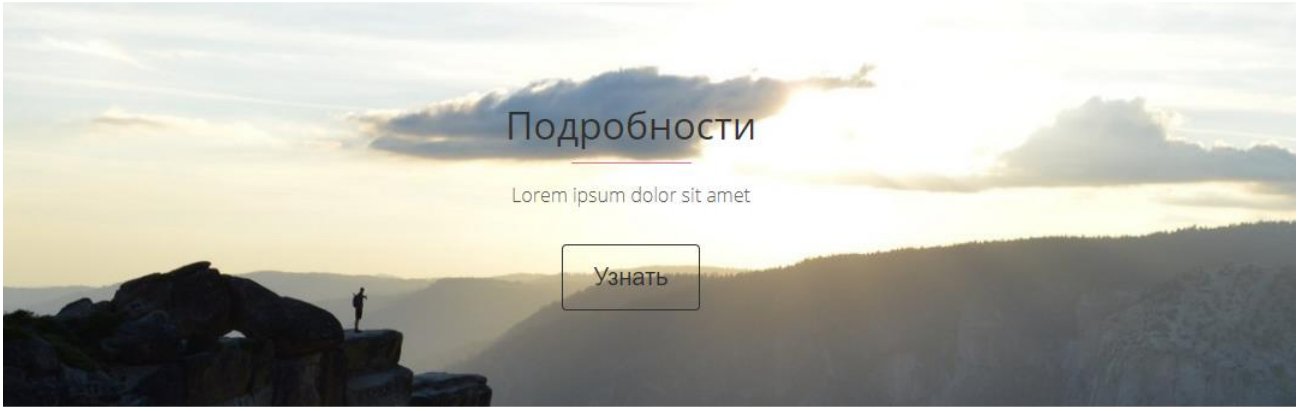

Рис. 38 – Внешний вид секции с фоном

Мы уже делали обложку на всю ширину страницы. Однако, здесь, в отличии от обложки, мы можем, во-первых, добавить контент поверх без дополнительных усилий, просто включая новые элементы в секцию. А во-вторых, сделать красивый эффект статичной подложки. Добавьте к классу **overlay** стиль **background-attachment: fixed**; вернитесь к странице и прокрутите колёсиком мыши новый блок.

Как вы заметили, теперь картинка будто «прибита» к окну браузера, а остальная страница прокручивается. Но, скорее всего, в некоторых частях изображения текст плохо читается. Можно, конечно, подобрать другую картинку. Но можно сделать контраст и стилями.

Прежде всего инвертируем цвета текста и кнопки в секции. Добавим к классу overlay класс black-back. А для его потомков создадим следующие стили:

```
.black-back .common-button {
  border-color: rgba(206, 65, 110, 0.74);
  color: #ce416e;
}
.black-back .common-button:hover {
  color: #fff;
  background-color: rgba(206, 65, 110, 0.74);
}
.black-back .main-title {
  color: #fff;
}
.black-back .main-title-description {
  color: #fff;
```

Рис. 39 – Стили, инвертирующие цвета вложенных элементов

Так благодаря общему родительскому классу мы заменяем цветовую схему всего блока, не создавая новых классов, а лишь переназначая нужные нам свойства. Теперь текст выделяется лучше, однако в светлых местах тоже не хватает контрастности.

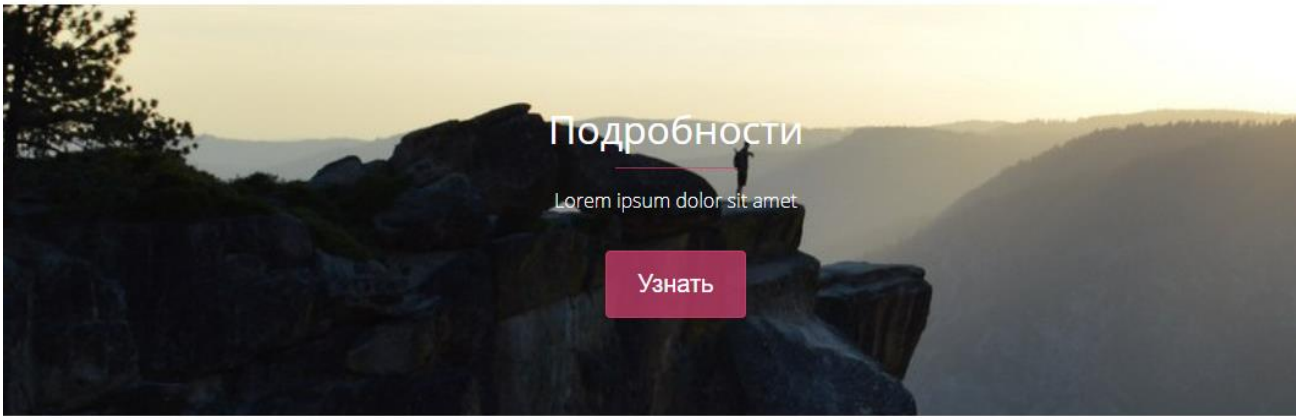

Рис. 40 – Внешний вид секции после инверсии

Добавим фон-наложение. Это тоже популярный приём стилизации, когда вместо затемнения рисунка используют возможности CSS. Тут нам потребуется вспомогательный псевдоэлемент **:before**, который накладывают поверх фонового изображения с некоторым затемнением.

```
.overlay {
  margin: 80px 0;
  padding: 80px 30px;
  background: #2d2f31 url('../img/02.jpg') no-repeat;
  background-size: cover;
  text-align: center;
  background-attachment: fixed;
  position: relative;
  z-index: 1;
}
.overlay::before {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: #081828;
    content: "";
    transition: all .4s ease;
    opacity: .7;
    z-index: -1;
}
```

Рис. 41 – Стили псевдоэлемента

Для создания псевдоэлемента обязательно необходим стиль **content: “”**; даже пустой. Без него браузер проигнорирует этот псевдоэлемент и не отрисует его на странице. Также обратите внимание, что к классу overlay добавлены два свойства: position и z-index.

На рисунке 42 проверяем как отображается новая секция на мобильном дисплее. Текст остаётся удобным для чтения в любой части фона. По желанию можно настроить фоновое изображение, чтобы оно отображалось по центру или по правому краю при помощи стиля background-position.

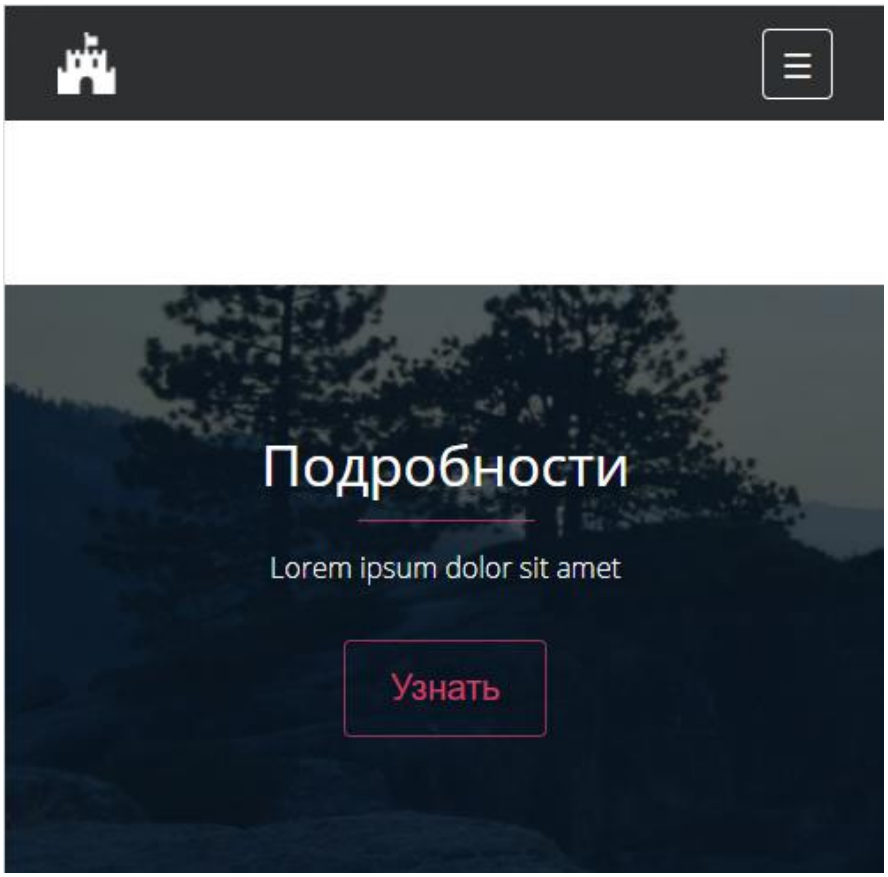

Рис. 42 – Внешний вид секции с затемнённым фоном

Теперь, когда страница сайта выглядит хорошо, на всех диагоналях, мы добавим ещё несколько элементов, например, контактную форму и подвал сайта. А ещё неплохо было бы добавить ещё несколько интересных блоков. Сделаем это в следующей лабораторной работе.

Задание на контроль

1. Повторить пример их практической части.

2. Для страницы примера должен быть стилизовано меню, шапка и разделы «Услуги», «Подробнее» и «Портфолио».

3. Объяснить, как были добавлены изображения к блоку с портфолио.

4. Добавить стили, срабатывающие по наведению на определенный блок.

5. Добавить медиазапросы для разных диагоналей экрана.

6. Объяснить, как создаётся фиксированное фоновое изображение.

7. Показать, как при помощи инструментов разработчика браузера тестировать внешний вид веб страницы для разных диагоналей.

8. Лабораторная работа считается защищенной, если студент показал в окне браузера страницу из практической части со стилизованным меню и ответил правильно на все заданные контрольные вопросы (следующая лабораторная работа не подлежит защите до тех пор, пока не будет защищена предыдущая).