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 №2

Подключение внешних стилей. Правила оформления стилей в стандарте CSS3.

Цель работы: ознакомиться со структурой файла стилей. Создать стили для дерева тегов при помощи классов и атрибутов.

Теоретическая справка

CSS (Cascading Style Sheets) — язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам. Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL. Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

Объявление стиля состоит из двух частей: селектора и объявления. Объявление состоит из двух частей: имя свойства (например, color) и значение свойства (grey). Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются форматирующие команды — свойства и их значения.

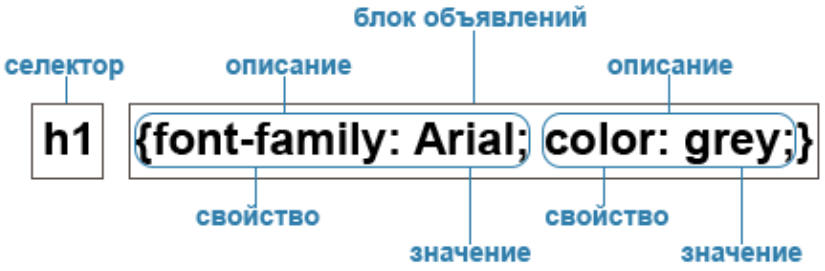

Рис. 10 – Структура объявления

Хотя приведенный пример пытается влиять только на пару свойств, необходимых для рендеринга HTML-документа, он сам по себе квалифицируется как таблица стилей. В сочетании с другими таблицами стилей (одна фундаментальная особенность CSS заключается в том, что таблицы стилей объединяются), правило будет определять окончательное представление документа.

Виды каскадных таблиц стилей и их специфика

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>. Такие стили работают для всех страниц сайта.

К каждой веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько тегов <link>, указав в атрибуте тега media назначение данной таблицы стилей. rel="stylesheet" указывает тип ссылки (ссылка на таблицу стилей). Так были добавлены стили подключения шрифтов в предыдущей лабораторной работе.

```
<head>
   <link rel="stylesheet" href="css/style.css">
   <link rel="stylesheet" href="css/assets.css" media="all">
</head>
```

Атрибут type="text/css" не является обязательным по стандарту HTML5, поэтому его можно не указывать. Если атрибут отсутствует, по умолчанию используется значение type="text/css".

Внутренние стили встраиваются в раздел <head></head> HTML-документа и определяются внутри тега <style></style>. Внутренние стили имеют приоритет над внешними, но уступают встроенным стилям (заданным через атрибут

style). В первой лабораторной работе стиль меню и текста был изменён при помощи внутренних стилей.

Когда мы пишем **встроенные стили**, мы пишем CSS-код в HTML-файл, непосредственно внутри тега элемента с помощью атрибута style:

```
<p style="font-weight: bold; color: orange;">
    Жирный оранжевый текст
</p>
```

Такие стили действуют только на тот элемент, для которого они заданы.

Правило @import позволяет загружать внешние таблицы стилей. Чтобы директива @import работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами:

```
<style>
    @import url(mobile.css);
    p {
        font-size: 0.9em;
        color: grey;
    }
</style>
```

Правило @import также используется для подключения веб-шрифтов. Так, например, можно было подключить в первой работе внешний шрифт через таблицу стилей:

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
```

Практическая часть

В первой лабораторной работе мы добавили простое меню при помощи тега <ul>, формирующего ненумерованный список (для создания нумерованного списка используется тег <ol>). Тег <li> (общий для обоих видов списков) создаёт

строку списка. Мы поместили его в тег <nav> – это семантический тег, в который обычно помещают меню сайта. По правде говоря, мы могли бы вместо него написать обычный тег используемый для обозначения простого блока, ничего бы не изменилось. Но стандарт HTML5 рекомендует использовать новые теги для разметки основных разделов сайта.

Но наше меню не очень похоже на то, что можно увидеть в сети интернет. Давайте это исправлять.

Добавим к нашему проекту в папку css новый файл с тем же расширением, например, style.css. Файл с таким названием уже был подключен, когда создавали документ в предыдущей работе. Убедитесь, что путь и название совпадают.

Так как мы больше не будем писать встроенные стили, их используют крайне редко, когда применение внешних стилей ограничено или невозможно, перенесём их в файл style.css. А из файла index.html удалим тег <style> вместе с содержимым, как это показано на рисунке 11. Сохраните изменения и обновите страницу, если всё было сделано верно, то ничего не должно было измениться.

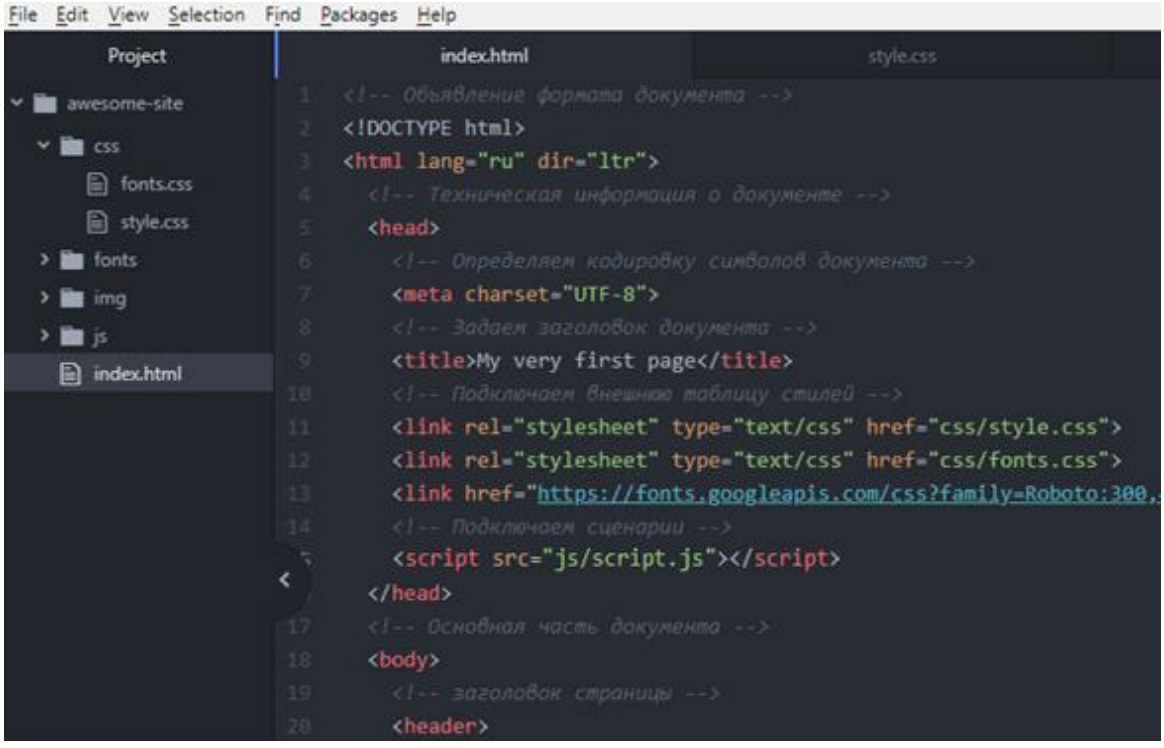

Рис. 11 – Подключение основного файла таблицы стилей

Добавим ещё стилей нашему заголовку и меню.

```
header {  background: #2d2f31; } // изменение цвета фона
header nav {
  display: flex;
  justify-content: space-between; // выравнивание дочерних элементов по горизонтали с одинаковыми отступами между ними
  max-width: 960px; // ограничение максимальной ширины блока  margin: auto;
  margin: auto; // внешние отступы блока
  padding: 0 30px; // внутренняя забивка блока
}
```

Тегу <ul> поменяем насыщенность шрифта и добавим несколько стилей:

```
header ul {
  font-family: "Roboto", "OpenSans-Light", Helvetica, Arial, sans-serif;
  font-weight: 300; // насыщенность текста
  display: flex; // создание контейнера flexbox
  justify-content: flex-end; // выравнивание дочерних элементов по горизонтали к правому краю блока
  align-items: center; // выравнивание дочерних элементов по вертикали по центру
  flex: 1; // установка единой ширины для блока
}
```

А также добавим стили для тега <li>:

```
header li {
  display: inline-block; // создание блока с двойственным поведением
  padding: 15px 0; // внутренний отступ блока
  color: #ffffff; // цвет текста
}
```

Здесь нужно пояснить, что стили margin и padding создают внешние и внутренние отступы блока соответственно. На рисунке 12 показано как эти стили влияют на блок разметки. С каждой стороны блока можно задавать произвольную величину отступа.

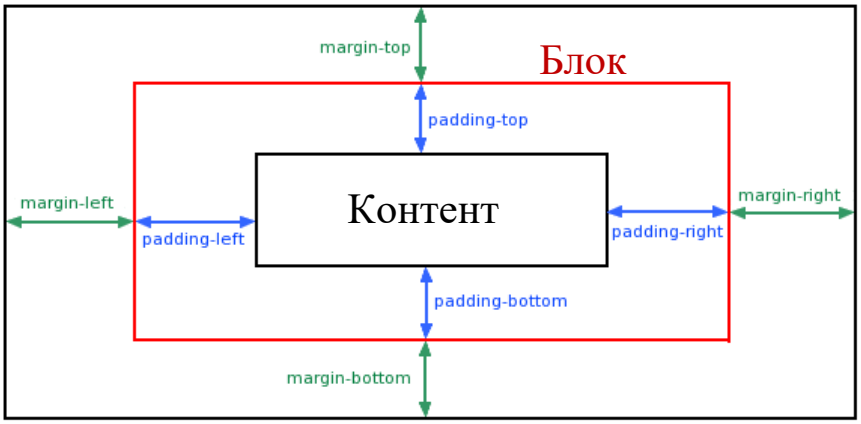

Рис. 12 – Отступы блока

Внешние и внутренние отступы для разных сторон задаются с помощью свойств margin-top, margin-right, margin-bottom, margin-left и padding-top, padding-right, padding-bottom, padding-left.

Но эти стили поддерживают короткие записи:

1) margin/padding: 10px; // Одинаковые отступы со всех сторон.
2) margin/padding: 5px 10px; // Отступы сверху и снизу 5px, справа и слева 10px.
3) margin/padding: 5px 10px 15px; // Отступ сверху 5px, слева и справа 10px, снизу 15px.
4) margin/padding: 5px 10px 15px 20px; // Разные отступы со всех сторон, в порядке верхний, правый, нижний, левый.

Свойство margin поддерживает положительные и отрицательные значения, а также auto для блочных элементов по горизонтали. Свойство padding поддерживает только положительные значения. Запись margin: auto в нашем примере

означает, что блоку будут заданы одинаковые внешние отступы слева и справа (сверху и снизу просто не применятся), то есть он установится по центру.

Теперь наше меню стало принимать вид настоящего (рисунок 13). Но остались отступы до краёв окна браузера. Как упоминалось в первой лабораторной работе у браузеров есть свой набор стилей по умолчанию. Он необходим, чтобы старые документы в сети, не имеющие стилей, оставались удобными для просмотра и чтения.

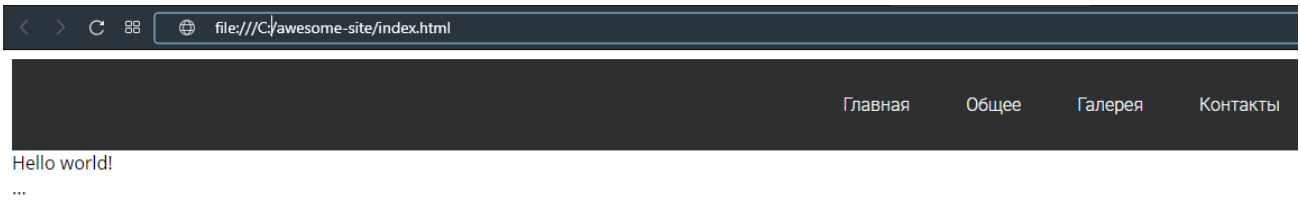

Рис. 13 – Внешние отступы меню

Добавим в style.css правило для обнуления отступов при помощи селектора звёздочка (*). Он применит записанные в него правила для всех элементов html документа. На рисунке 14 скриншот стилей из файла style.css.

Добавим рядом с меню логотип сайта. Разместим изображение при помощи тега <img>. Адрес файла с картинкой задаётся через атрибут src. Данный тег не нужно закрывать. Он поддерживает атрибуты width и height, задающие размеры изображения, но для этих целей стандартом HTML5 рекомендуется использовать CSS стили. А вот атрибут alt наоборот считается обязательным. Его значением необходимо установить текст, описывающий что изображено на картинке. Он будет показан, если картинка не загрузится или пользователь отключит отображение изображений на странице. Также устройства для чтения текста со страницы прочитают этот текст. Стандартом допускается пустой атрибут alt.

<img src="img/fort-icon.png" alt="Логотип сайта">

Саму картинку можете использовать из приложения к лабораторной работе или подобрать по своему усмотрению.

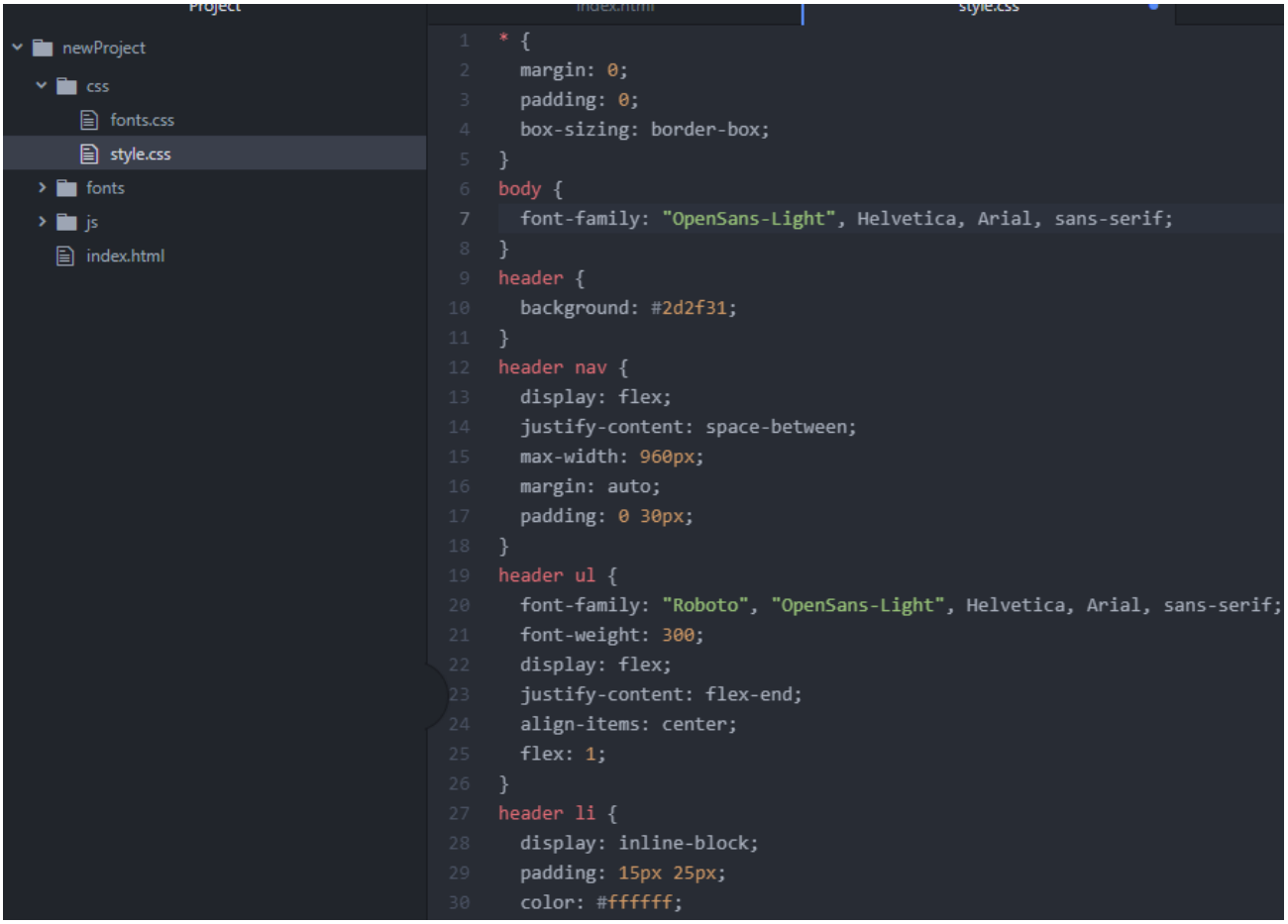

Рис. 14 – Результат создания меню

Картинку логотипа поместим в папку img проекта. Зададим ему стиль:

```
header nav img {
  width: 35px;
  padding: 15px 0;
}
```

Если высота изображению не была задана, то она будет высчитана автоматически, пропорционально оригинальной картинки. В противном случае картинка может исказится. Теперь меню сайта похоже на настоящее (рисунок 15), но разделы меню по-прежнему лишь текст и неактивны.

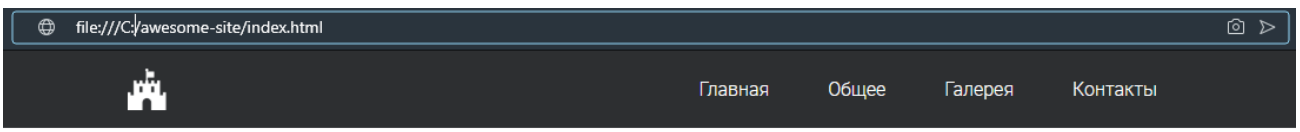

Рис. 15 – Меню с логотипом

Давайте это исправим, обернём текст в пунктах меню в тег Главная, где в атрибуте href указывается адрес ссылки (знак решётки – стандартная заглушка, пока некуда ссылаться), title создаёт подсказку при наведении на ссылку, например куда ведёт ссылка или будет ли она открыта в новой вкладке (чтобы ссылка открывалась в новой вкладке нужно добавить атрибут target="_blank").

Добавим стили:

```
header nav a {
  color: #ffffff;
  text-decoration: none; // отключение системного подчёркивания ссылок
  border-bottom: 1px solid transparent; // прозрачная граница под ссылкой
  transition: border 250ms ease-in-out; // эффект перехода между двумя состояниями элемента
}
header nav a:hover {
  border-bottom: 1px solid #ffffff; // белая граница под ссылкой
}
```

Здесь псевдокласс :hover отвечает за поведение элемента, когда курсор находится над ним. Стиль transition создаёт эффект перехода из состояния, когда курсор вне элемента, и когда над ним. В данном случае в первом состоянии ссылка с прозрачной границей, а при наведении на неё – с белой. Благодаря переходу, создаётся задержка, и граница окрашивается плавно. Если бы мы не создали стиль границы в начальном состоянии, плавный эффект не сработал бы.

Следующим шагом добавим под меню обложку страницы. В дальнейшем на её месте мы создадим слайдер, но пока обойдёмся простой картинкой. Создайте ещё один тег <section> перед секциями с заголовками. А внутри новой секции напишите тег <img>. Только теперь к тегу <img> добавим класс, чтобы обращаться и стилизовать только эту картинку, а не все изображения на странице.

Создадим класс, например, "top-image" и зададим адрес картинки. Картинку необходимо также поместить в папку img из дополнительных материалов к работе или скачать из сети. В папке **backgrounds** предоставлены несколько картинок в большом разрешении и с лицензией на свободное использование. Зададим этому классу простой стиль на всю ширину страницы:

.top-image { width: 100%; }

```
14    <body>
15      <!-- заголовок страницы -->
16      <header>
17        <!-- меню страницы -->
18        <nav>
19          <img src="img/fort-icon.png" alt="Логотип сайта">
20          <ul>
21            <li><a href="#" title="На главную">Главная</a></li>
22            <li><a href="#" title="Об услугах">Услуги</a></li>
23            <li><a href="#" title="Фотогалерея">Галерея</a></li>
24            <li><a href="#" title="Контактные данные">Контакты</a></li>
25          </ul>
26        </nav>
27      </header>
28      <!-- основной контент сайта -->
29      <main>
30        <section>
31          <img class="top-image" src="img/01.jpg" alt="Обложка страницы">
32        </section>
33        <section>
34          <h1>Самый большой заголовок</h1>
35          <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
36        </section>
```

Рис. 16 – Код меню с обложкой

Картинка будет подстраиваться под окно браузера вне зависимости от его размера. Код разметки изображен на рисунке 16. Тег aside можно удалить, он нам пока не потребуется.

На рисунке 17 изображено меню с обложкой страницы.

Рис. 17 – Внешний вид меню с обложкой

Добавим на страницу нашего сайта секцию с перечислением и описанием услуг. Такие секции часто можно увидеть на различных продающих страницах (landing page). Она может содержать информацию о типах услуг, опциях продукта, перечислением достижений или особенностях товара и т.д.

Для этого удалим тег <aside> и его содержимое, так как мы не будем пока создавать боковую панель. А второй секции зададим класс "services". Внутри заголовка первого уровня <h1> зададим осмысленный текст, например «Наши услуги», за ним разместим пустой тег с классом "delimeter". Параграф с тегом <p> и произвольным текстом можем оставить без изменений.

Добавим стили:

```
.services {
  text-align: center;
  margin: 50px 0;
}
.services h1 {
  font-size: 32px;
  font-weight: 300;
  color: #313131;
}
```

```
.delimiter {
  width: 100px;
  margin: 10px auto 15px;
  height: 1px;
  background: rgba(206, 65, 110, 0.74);
}
.services p {
  color: #616161;
}
```

Примерный результат применённых стилей изображён на рисунке 18.

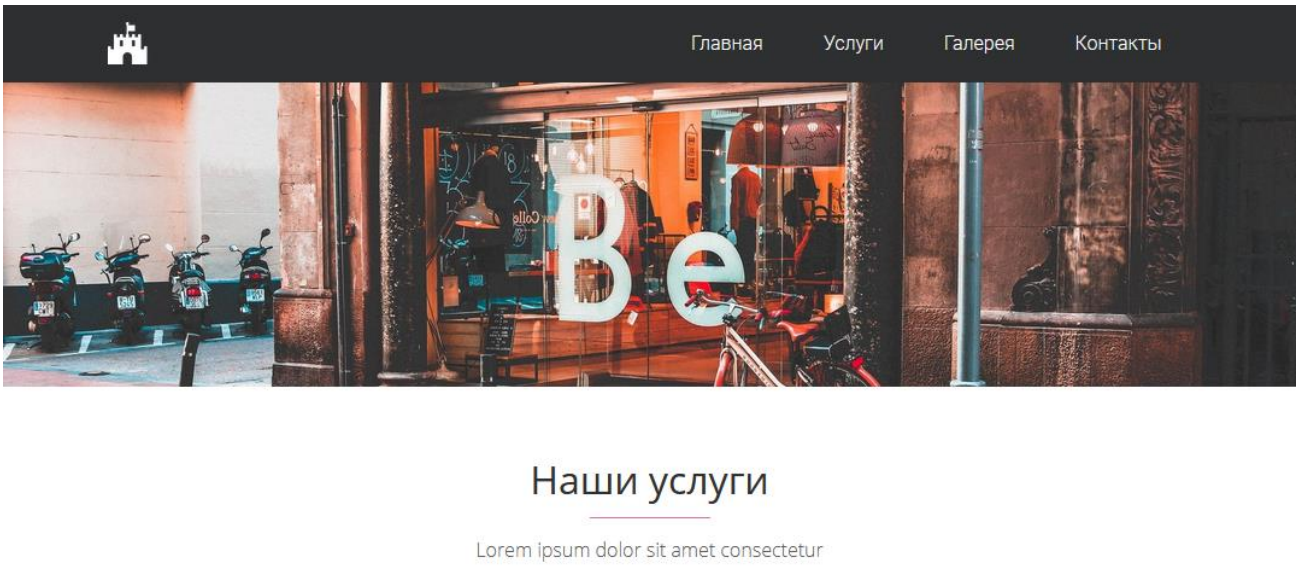

Рис. 18 – Заголовок второй секции

Теперь добавим сам блок услуг. Он будет состоять из внешнего блока, внутреннего блока-обёртки и шести блоков с описанием. На рисунке 19 код блока с наполнением из текста для наглядности. Его также можно набрать быстрым действием среды разработки или скопировать произвольный текст из сети.

```
    </section>
    <section class="services">
      <h1>Наши возможности</h1>
      <div class="delimiter"></div>
      <p>Lorem ipsum dolor sit amet</p>
      <!--  внешний блок -->
      <div class="services-container">
        <!-- обёртка -->
        <div class="container-row">
          <!-- элемент -->
          <div class="row-item">
            <img class="item-img" src="img/icon-1.png" alt="">
            <h2>Lorem ipsum dolor</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
              eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
          </div>
        </div>
      </div>
    </section>
```

Рис. 19 – Заполнение раздела с текстом-заполнителем

Скопируйте блок с классом "row-item" ещё 5 раз (всего 6) внутри блока "container-row". Также как и раньше скопируйте в папку img иконки из методических материалов из папки icons с названиями icon-1.png и далее. А для каждой картинки блока возможностей измените путь на разные иконки.

Если мы проверим как выглядят наши элементы, то увидим, что они выстроились в колонну друг под другом. Добавим стилей для блока-контейнера, а также для каждого отдельного блока:

```
.services-container {
  margin: 25px 0;
  display: flex;
  justify-content: center;
}
.container-row {
  text-align: center;
  max-width: 960px;
  margin: 0 auto;
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  padding: 0 30px;
}
.row-item {
  width: 30%;
  margin: 25px 0;
}
.row-item img {
  width: 90px;
  opacity: 0.75;
}
.row-item h2 {
  font-weight: 400;
  font-size: 26px;
  margin: 8px 0;
```

Рис. 20 – Стили для блоков

После сохранения у вас должен получиться результат, как показано на рисунке 21: блоки выстроились в один ряд.

Наши услуги

Lorem ipsum dolor sit amet consectetur

Lorem ipsum dolor	Lorem ipsum dolor	Lorem ipsum dolor	Lorem ipsum dolor	Lorem ipsum dolor	Lorem ipsum dolor
Lorem ipsum dolor sit amet, consectetur	Lorem ipsum dolor sit amet, consectetur	Lorem ipsum dolor sit amet, consectetur	Lorem ipsum dolor sit amet, consectetur	Lorem ipsum dolor sit amet, consectetur	Lorem ipsum dolor sit amet, consectetur

Рис. 21 – Заполнение секции блоками

Теперь чтобы эти блоки смотрелись красиво лучше, чтобы они были расположены в два ряда. Так текст будет лучше смотреться.

Для этого достаточно добавить всего один стиль – flex-wrap: wrap; для селектора .container-row, который добавит правило переноса флекс-элементов внутри флекс-контейнера. Теперь элементы красиво выстроились в два ряда.

Рис. 22 – Блоки с информацией в два ряда

Уже лучше, но для мобильных дисплеев выглядеть будет не так хорошо. Исправим это в следующей лабораторной работе, а также создадим адаптивное меню для мобильных разрешений экранов и познакомимся с медиа-запросами.

Задание на контроль

1. Повторить пример их практической части.

2. Для страницы примера должен быть стилизовано меню и шапка.

3. Объяснить, как были добавлены изображения к проекту, где они подключаются.

4. Добавить ссылки для меню страницы.

5. Объяснить, как были удалены стили браузера по умолчанию.

6. Объяснить, как была создана секция «Услуги» и отображение блоков в две строки.

7. 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).