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 №6

Создание мультистраничного проекта. Подключение внешних картографических сервисов. Разработка индивидуальных пользовательских интерфейсов

Цель работы: ознакомиться с подключением внешних карт и размещением их на HTML странице. Изменение проекта под новый круг задач. Создание дополнительных страниц с различными темами содержимого, а также научиться разрабатывать уникальные элементы пользовательского интерфейса.

Теоретическая справка

Лэндинги и обычные сайты

Перед руководителями интернет-проектов, часто встаёт вопрос какой сайт разрабатывать: продающую страницу (лендинг) или обычный сайт (мультистраничный). Лэндинги, или одностраничные сайты, появились сравнительно недавно и быстро набрали популярность. Однако не всегда выбор в пользу лэндинга является оптимальным для проекта, и бывает так, что люди делают такой выбор, следуя моде. Ответить на вопрос что лучше, можно, разобравшись в плюсах и минусах каждого решения.

Стоит сразу определиться в терминологии. Есть многостраничные и одностраничные сайты — с ними всё понятно. Лэндинги (от англ. «landing page») в общем случае — это посадочная страница, которая соптимизирована под определенные цели. Например, получить максимальное число конверсий с посетителей определенной тематики. Лэндинги могут быть как на многостраничных сайтах, так и на одностраничных. Но очень часто под лэндингами понимаются именно одностраничные сайты. Для удобства под лэндингами мы также будем иметь в виду одностраничные сайты.

Почему появились лэндинги?

До появления контекстной рекламы и других систем, которые позволяют получить релевантный трафик, лэндинги были не актуальны. Одностраничные

сайты сильно проигрывают многостраничным в возможностях поискового продвижения (SEO). Во времена, когда для получения трафика использовалось только SEO, ни у кого не было мысли, делать заранее проигрышный вариант.

Второй момент — это оптимизация конверсии. Раньше при поиске товара или услуги пользователь попадал на страницу с большим количеством текста, где сложно было найти телефон, условия покупки, преимущества, цены и другую полезную и интересную информацию. Со временем интернет маркетологи, изучая статистику поведения пользователей на странице, стали создавать более удобные в плане предоставления информации страницы. Появились разнообразные интерфейсы сайтов, информацию стали разделять тематическими блоками, стали использовать информативные иконки, формы обратной связи. Пользователям стало удобнее ориентироваться и получать нужную информацию, оставлять заявку.

Также было отмечено, что удобнее пользоваться скролингом, чем переключать разные страницы, запоминая много информации при принятии решения.

Стоимость создания лэндинга меньше, чем многостраничного сайта. Времени на создание одностраничника также требуется меньше, что очень важно, в наше быстрое время.

Все это привело к появлению лэндингов в том виде, в котором мы сейчас их наблюдаем в большом количестве.

Что хорошего в многостраничных сайтах?

Несмотря на преимущества одностраничников, есть то, в чем многостраничные сайты имеют преимущества. Во-первых, возможности SEO. Тут ничего не поменялось. Многостраничники лучше приспособлены для поискового продвижения и, с точки зрения поисковых систем Яндекс и Гугл, выглядят намного привлекательнее одностраничных. SEO в отличие от контекста и других систем работает не сразу. Нужно время, иногда месяцы, чтобы сайт раскрутился. В результате, средняя стоимость перехода и конверсии из естественного поиска будут дешевле, чем из контекста.

Вернемся к положительным моментам использования многостраничного сайта. Если вы создаете интернет-магазин с большим ассортиментом, как, например, магазин электроники «Связной», то разместить всё на одной странице не получится. Разместить весь каталог товаров с подробным описанием каждой позиции, отзывами и другой полезной информацией можно только на многостраничном сайте.

Кроме того, есть консервативные люди, для которых многостраничный сайт привычнее, и вызывает больше доверия. Действительно, все корпоративные сайты крупных и средних компаний — это полноценные, многостраничные сайты. Невозможно представить официальный сайт «Газпрома» или «Сбербанка» в виде одностраничника.

Преимущества многостраничного сайта

1. Хорошо пригоден для СЕО
2. Можно разместить много товаров, услуг
3. Больше подходит для корпоративного сайта солидной компании
4. Преимущества одностраничного сайта
5. Подходит для теста ниши, продвижения определенного товара, услуги, мероприятия
6. Стоимость ниже, чем у многостраничного сайта
7. Сроки изготовления — меньше

Многостраничный сайт — это проект на перспективу. Если вы точно уверены, что будете работать более одного года, или у вас на сайте представлено много товаров, услуг и различной информации, то вам следует сделать выбор в пользу многостраничного сайта.

Если же ваш проект может компактно разместиться на одной странице, при этом оставаясь достаточно информативным, лэндинг будет отличным вариантом для вас. Также одностраничник подойдет тем, кому необходимо протестировать нишу.

Вы также можете использовать оба варианта. Корпоративный сайт использовать для представления товаров и услуг, а лэндинги — для проведения акций, распродаж. В любом случае, тип сайта будет определяться целями проекта. Независимо от типа сайта, каждая его страница должна решать определенную задачу и быть полноценной посадочной страницей.

Свойство z-index в CSS

Любые позиционированные элементы на веб-странице могут накладываться друг на друга в определенном порядке, имитируя тем самым третье измерение, перпендикулярное экрану. Каждый элемент может находиться как ниже, так и выше других объектов веб-страницы, их размещением по z-оси и управляет z-index. Это свойство работает только для элементов, у которых значение position задано как absolute, fixed или relative.

В качестве значения используются целые числа (положительные, отрицательные и ноль). Чем больше значение, тем выше находится элемент по сравнению с теми элементами, у которых оно меньше. При равном значении z-index, на переднем плане находится тот элемент, который в коде HTML описан ниже. Хотя спецификация и разрешает использовать отрицательные значения z-index, но такие элементы не отображаются в браузере Firefox до версии 2.0 включительно.

Кроме числовых значений применяется auto — порядок элементов в этом случае строится автоматически, исходя из их положения в коде HTML и принадлежности к родителю, поскольку дочерние элементы имеют тот же номер, что их родительский элемент. Значение inherit указывает, что оно наследуется у родителя.

Рис. 87 – Позиционирование элементов по умолчанию

Z-index и контекст наложения

Вышеописанная способность элементов изменять порядок наложения друг на друга с помощью свойства z-index работает лишь в том случае, если эти элементы существуют в одном контексте наложения. Что это значит?

Контекст наложения (англ. stacking context) — это концепция трехмерного размещения HTML-элементов по оси Z, расположенной перпендикулярно экрану. Контекст наложения может быть сформирован любым элементом, который соответствует хотя бы одному из следующих условий:

- Элемент является корневым, т. е. существует в корневом контексте наложения. Любой элемент веб-страницы является таковым, если только он не присутствует в локальном контексте наложения (в том, который создается любым из способов ниже).
- Абсолютно позиционированный (position: absolute) либо относительно позиционированный (position: relative) элемент с любым значением z-index, кроме auto.

- Элемент со свойством position: fixed и любым значением z-index.
- Элемент со свойством display: flex либо display: inline-flex и любым значением z-index, кроме auto.
- Элемент со свойством opacity и значением менее 1.
- Элемент с любым значением свойства transform, кроме none.
- Элемент с любым значением свойства mix-blend-mode, кроме normal.
- Элемент с любым значением свойства filter, кроме none.

Итак, если соблюдать один из вышеперечисленных пунктов (применить к элементу позиционирование и z-index либо свойство opacity со значением меньше единицы и т. п.), то формируется новый контекст наложения. Внутри контекста наложения дочерние элементы можно перемещать по оси Z в соответствии с обычными правилами.

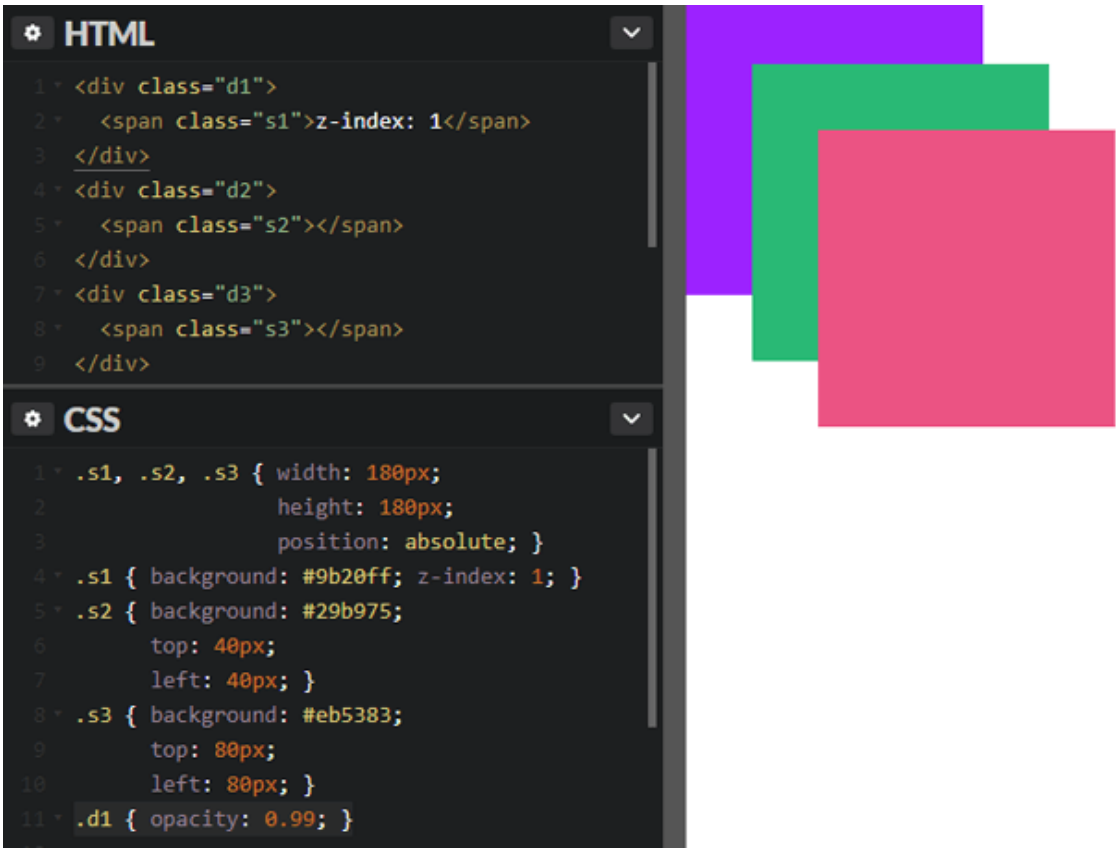


Рис. 88 – Влияние свойства opacity на действие z-index

Контекст наложения может являться частью другого контекста наложения, тем самым создавая своеобразную иерархию контекстов. Если внутри родителя дочерний элемент создает свой собственный контекст наложения, то значения z-index будут учтены в пределах родителя. Элементы, которые не создают свой контекст наложения, используют родительский контекст наложения.

Один контекст наложения является полностью независимым от соседнего контекста. Это означает, что вы не можете, к примеру, наложить дочерний элемент А из одного контекста поверх дочернего элемента Б из другого контекста, если родитель элемента А находится ниже родителя элемента Б (подразумевается, что эти родители являются создателями разных контекстов).

На рисунке 88 показан пример того, как родительский элемент .d1 создает новый контекст наложения при добавлении к нему свойства opacity: 0.99, после чего дочерний элемент .s1 вновь становится нижним слоем, несмотря на свой z-index.

Это происходит потому, что теперь свойство z-index элемента .s1 работает в пределах контекста наложения своего родителя .d1, тогда как другие два блока пока имеют корневой контекст наложения. Каким же образом снова разместить фиолетовый блок выше других, учитывая свойство прозрачности? Для этого необходимо позиционировать все блоки , после чего можно будет установить для них нужный порядок через z-index.

В примитивном варианте свойство z-index работает просто: чем больше значение, тем выше находится элемент (слой). Но стоит только столкнуться с разными контекстами наложения (группами), как всё становится намного сложнее, и начинает казаться, что z-index не работает. Рекомендуем дополнительно попрактиковаться в данной теме: создайте различные контексты наложения, используя список выше, и понаблюдайте за тем, как ведут себя элементы с z-index в этих контекстах.

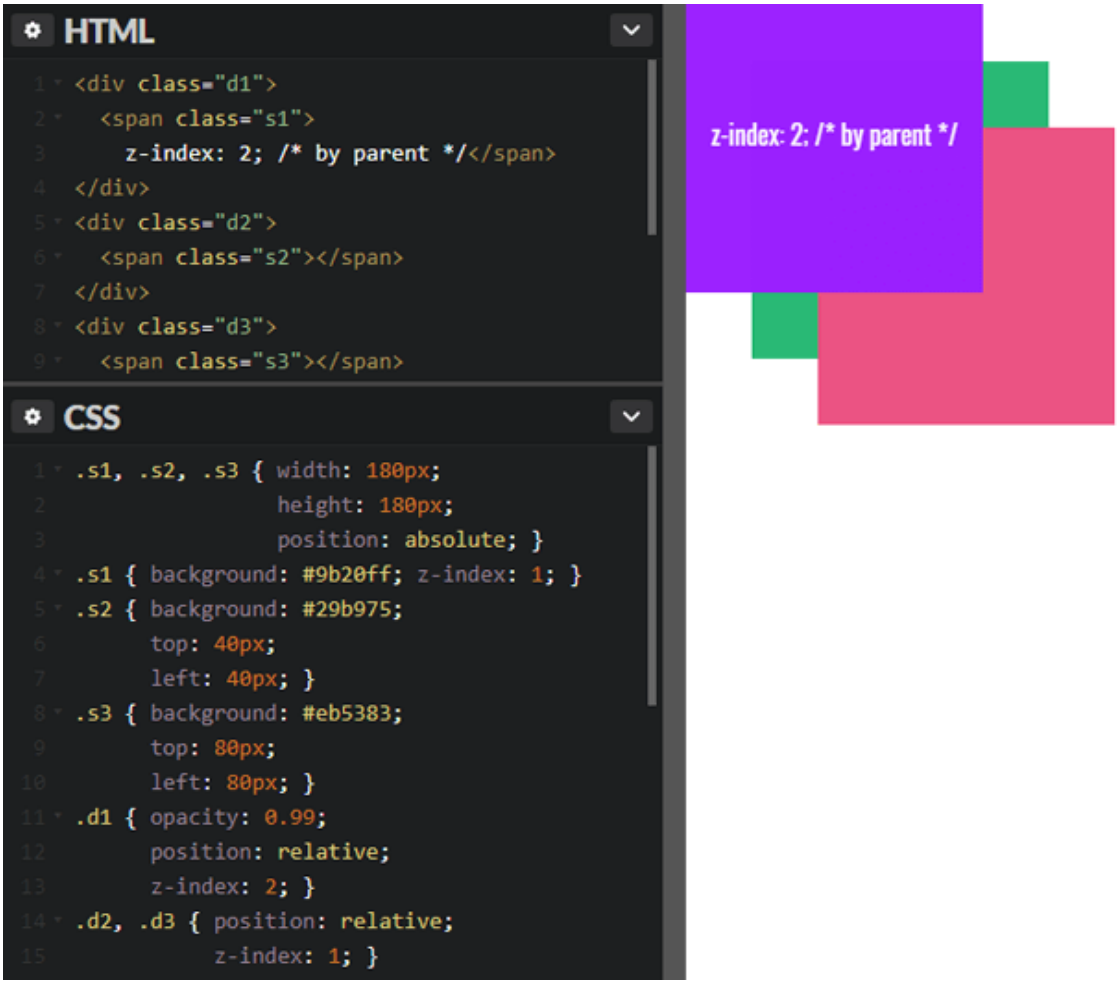


Рис. 89 – Блок снова поверх остальных благодаря z-index

Практическая часть

Продолжим наполнять наш сайт контентом. Можно разместить ещё несколько информативных блоков на этой странице. Тогда получится классическая продающая страница (landing page). Но на таких страницах с длинной прокруткой принято размещать презентационные материалы товара или услуги.

Курс лабораторного практикума позволяет создать более многостраничный веб-сайт. Добавим к сайту новую страницу «Контакты». Для этого откроем каталог нашего проекта и создадим новый файл contacts.html.

Следующим шагом необходимо изменить меню для того, чтобы переходить к новой странице, а не прокручивать текущую к разделу. Путь для атрибута href тега <a> укажем contacts.html и удалим атрибут data-link не нужен, так как мы будем открывать новую страницу.

```
23          <img src="img/fort-icon.png" alt="Логотип сайта">
24          <ul>
25            <li><a href="index.html" class="menu-element" data-link="top" title="На главную">Главная</a></li>
26            <li><a href="#services" class="menu-element" data-link="services" title="Об услугах">Услуги</a></li>
27            <li><a href="#portfolio" class="menu-element" data-link="portfolio" title="Фотогалерея">Галерея</a></li>
28            <li><a href="contacts.html" class="menu-element" title="Контактные данные">Контакты</a></li>
29          </ul>
30          <div id="hamburger" class="icon">
31            &#9776;
32          </div>
```

Рис. 90 – Исправления в меню

Обратите внимание, что для главной мы изменили ссылку, но не убрали атрибут data-link. Ниже станет понятно, зачем нам это нужно. Внесите аналогичные изменения в меню подвала.

Рис. 91 – Меню подвала

После обновления страницы пробуем кликать по ссылкам меню. Однако ничего не происходит, новая страница не открывается. Почему? Всё дело в нашем скрипте, который плавно прокручивает страницу.

По клику на ссылку меню, срабатывает метод preventDefault(), который закрывает стандартное поведение элемента. В нашем случае, это переход на другую страницу. Хорошо, давайте исправим скрипт, чтобы он не мешал работе нашего меню и ссылки вели себя так, как ожидает пользователь.

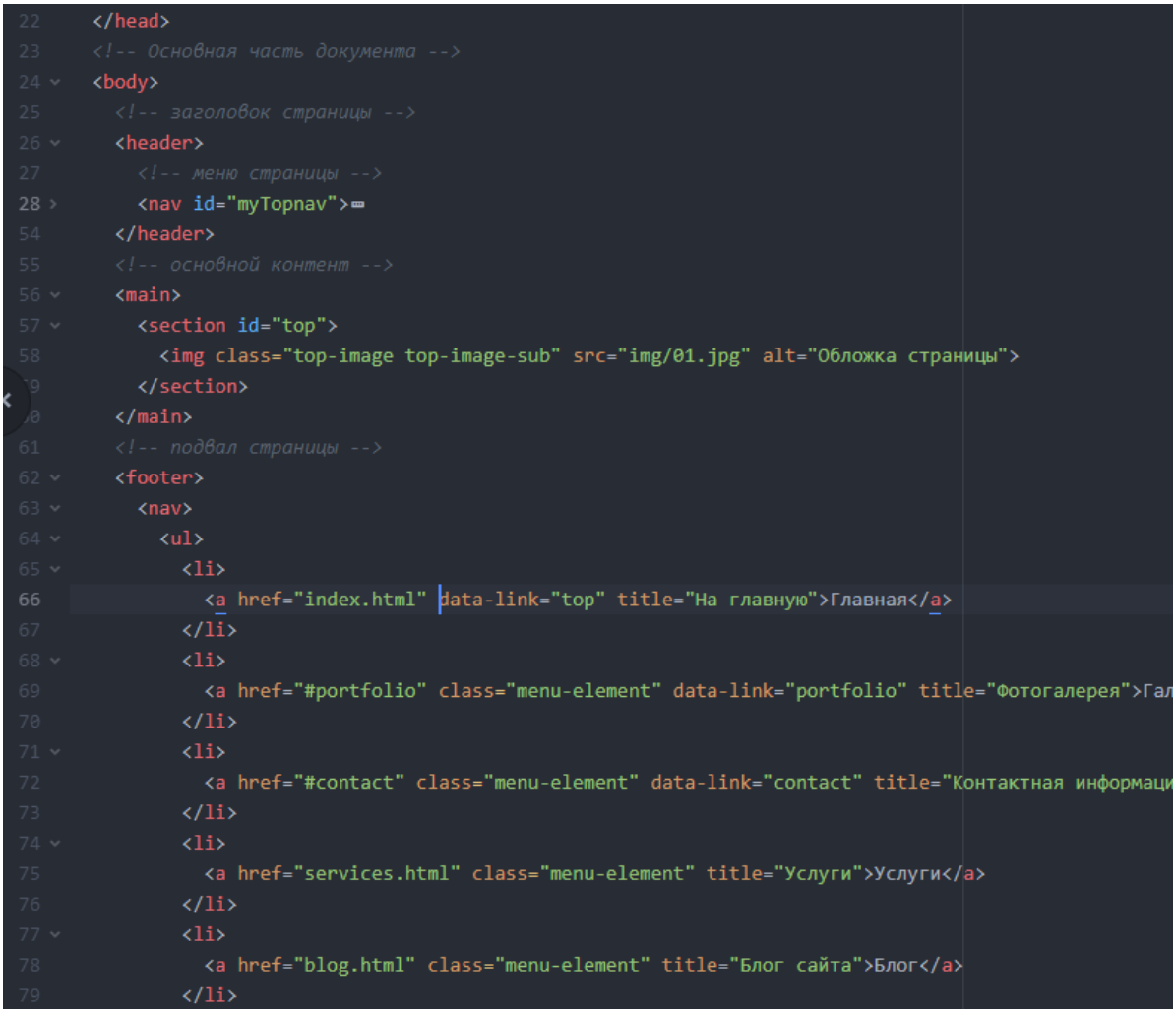
Переместим получение ссылки из атрибута до метода preventDefault() и поставим условие: выполнить дальнейший код только в том случае, если есть elementLink. Для краткости в ES6 пишем просто if (elementLink) – в таком случае код не будет выполняться, если elementLink равен null, undefined или пустой.

```
const menuList = document.querySelectorAll('.menu-element');
// проходим цциклом по найденым элементам
menuList.forEach(function(element) {
  // каждому элементу создаём обработчик события "клик мыши"
  element.addEventListener('click', function(event) {
    // считываем атрибут data у элемента, по которому был произведён щелчок
    const elementLink = element.dataset.link;

    // проверяем, есть ли альтернативная ссылка и тогда выполняем дальше
    if (elementLink) {
      // прерываем стандартное поведение ссылки
      event.preventDefault();
      // плавно прокручиваем страницу к искомому блоку
      document.getElementById(elementLink).scrollIntoView({ behavior: 'smooth'});
    }
    // иначе ничего не делаем
  });
});
```

Рис. 92 – Условие работы прокрутки по клику

Сохраним, обновим и проверяем. Теперь по клику на ссылку с блогом открывается пустая страница, а старые ссылки работают как раньше. Для этого мы оставили атрибут для первой ссылки. Несмотря на то, что там указан путь на страницу index.html страница не обновляется, а только прокручивается к заголовку.



```
</head>
<!-- Основная часть документа -->
<body>
  <!-- заголовок страницы -->
  <header>
    <!-- меню страницы -->
    <nav id="myTopnav">
  </header>
  <!-- основной контент -->
  <main>
    <section id="top">
      <img class="top-image top-image-sub" src="img/01.jpg" alt="Обложка страницы">
    </section>
  </main>
  <!-- подвал страницы -->
  <footer>
    <nav>
      <ul>
        <li>
          <a href="index.html" data-link="top" title="На главную">Главная</a>
        </li>
        <li>
          <a href="#portfolio" class="menu-element" data-link="portfolio" title="Фотогалерея">Гал
        </li>
        <li>
          <a href="#contact" class="menu-element" data-link="contact" title="Контактная информаци
        </li>
        <li>
          <a href="services.html" class="menu-element" title="Услуги">Услуги</a>
        </li>
        <li>
          <a href="blog.html" class="menu-element" title="Блог сайта">Блог</a>
        </li>
```

Рис. 93 – Сегмент страницы contacts.html

Наполним теперь новые страницы. Для этого просто скопируем весь HTML код из файла index.html в страницу contacts.html. Удалим всё лишнее, что нам не нужно, а именно всё содержимое тега main кроме секции с id="top".

В меню этой страницы **удалим атрибут data-link** для ссылки на главную страницу. Как думаете, зачем?

Рис. 94 – Внешний вид страницы contacts.html

На новой странице обложка такая же большая, как на главной. Давайте её слегка модифицируем. Добавим класс top-image-sub к тегу с обложкой и классом "top-image".

В стилях для этого класса пропишем свойства как на скриншоте

```
53 .top-image {
54   width: 100%;
55 }
56
57 .top-image-sub {
58   height: 150px;
59   object-fit: cover;
60 }
```

Рис. 95 – Модификация обложки

Стиль object-fit с свойством cover задаёт картинке такое же поведение, как свойство background-position: cover, который мы использовали ранее. Но этот стиль для картинок появился относительно недавно. К счастью, сегодня у этого стиля хорошая поддержка браузерами (около 97% пользователей), и его уже можно смело использовать в своих проектах.

Принимать решение о том, можно ли использовать какое-то свойство или нет помогает сайт https://caniuse.com/ Он даёт краткую справку о искомой технологии и выдаёт результат по количеству пользователей в сети, у кого тот или иной класс или стиль будет работать.

На рисунке 96 страница сайта и оценка стиля object-fit

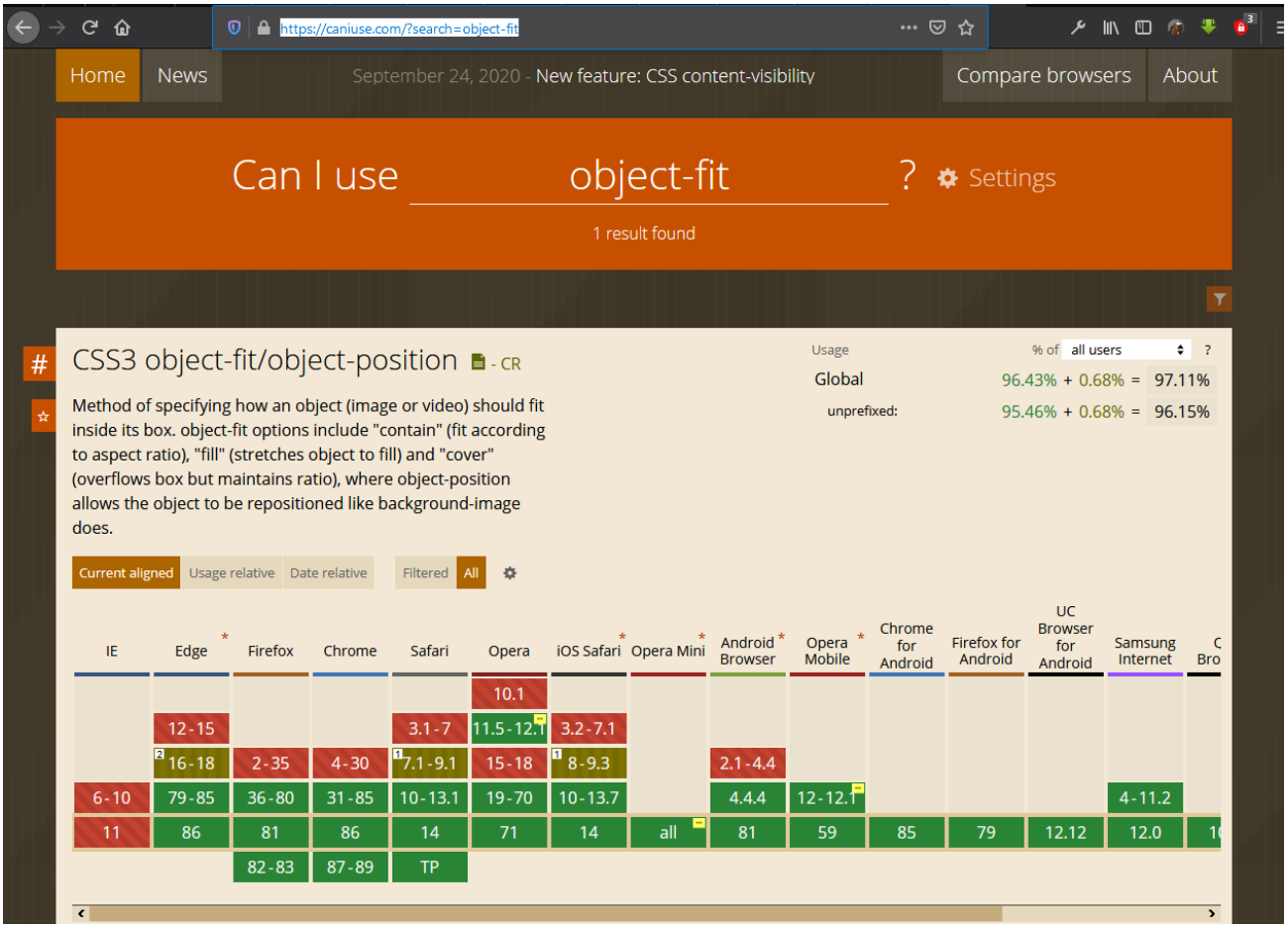


Рис. 96 – Пользовательский интерфейс сервиса caniuse.com

Обновим нашу страницу блога. Если ваш браузер поддерживает этот стиль, то результат должен быть похож на тот, что на скриншоте ниже.

Наполним страницу контакты. В ней будет четыре секции:

1. Краткая информация о владельце сайта. Тут могут быть контактные данные или небольшое описание компании или человека.

2. Секция достижений.

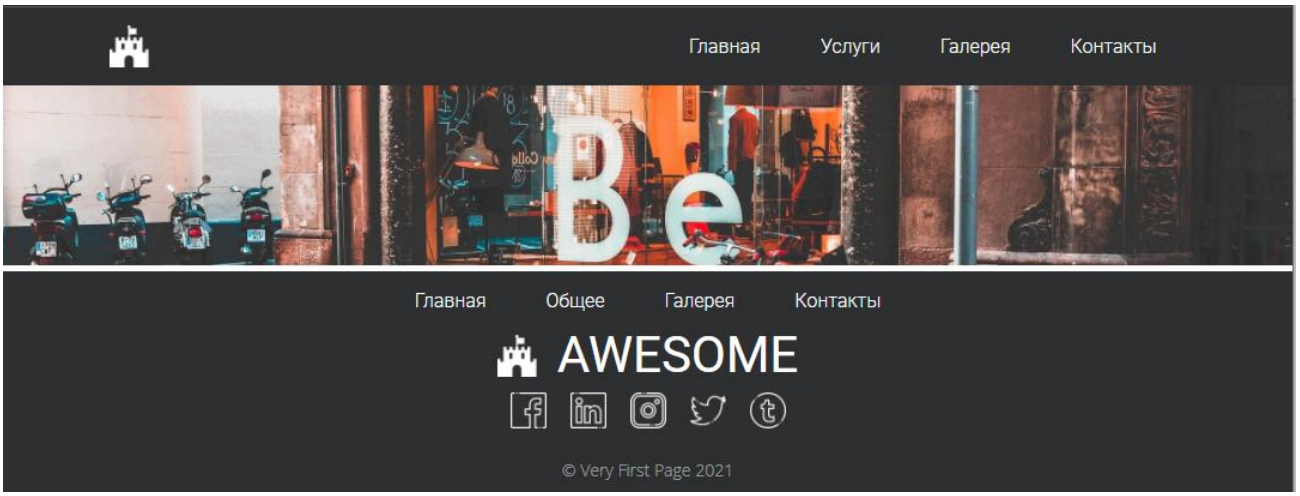


Рис. 97 – Измененная обложка страницы

3. Блок с картой и координатами, где пользователь сможет увидеть, где именно расположен физический адрес.

4. Секция с отзывами из предыдущей работы, скопированная на текущей странице.

Итак, добавим разметку для первой секции. Она будет состоять из двух колонок. В левой части разместим изображение, а в правой части небольшое описание и контактные данные. Эти колонки будут одинаковыми по ширине. Контактные данные также будут размещены в 2 колонки. Но здесь колонка для описания типа данных уже, а вторая, для самих данных – шире.

Чтобы регулировать ширину блоков внутри контейнера flex можем воспользоваться короткой записью свойства flex: 1 для более узкой колонки и flex: 3 для более широкой.

```
.about-contact span:first-child {                    style.css:399
   flex: ▼ 1;
    flex-grow: 1;
    flex-shrink: 1;
    flex-basis: 0%;
}
```

Рис. 98 – Развёртка свойства flex: 1 в панели разработчика браузера

Как видно на скриншоте выше, это сокращение трёх свойств дочернего flex-блока. Где flex-grow – это то, как блок будет увеличиваться, flex-shrink – как сжиматься и flex-basis – какой размер будет занимать по умолчанию.

```
<section class="about-us">
  <div class="about-container">
    <div class="about-row">
      <div class="about-column">
        <div class="content-left">
          <img src="img/portfolio/img09.jpg" alt="О нас">
        </div>
      </div>
      <div class="about-column">
        <div class="content-right">
          <h2>Наша компания</h2>
          <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do ei
          ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostru
          laboris nisi aliquip ex ea commodo consequat.</p>
          <h3>Контакты для связи</h3>
          <div class="about-contact">
            <span>Адрес:</span>
            <span>г. Тирасполь, ул. Свердлова, д. 10</span>
          </div>
          <div class="about-contact">
            <span>Телефон:</span>
            <span>+ (373) 55-200-624</span>
          </div>
          <div class="about-contact">
            <span>E-mail:</span>
            <span>hello@awesome.ws</span>
          </div>
          </div>
        </div>
      </div>
    </div>
  </section>
```

Рис. 99 – Вёрстка и наполнение верхней секции страницы contacts

Рассмотрим стили на рисунке 100. Там для колонок с контактной информацией заданы стили flex: 1 и flex: 3. Таким образом, их размер будет относится как

1 к 3. Браузер сам отрисовал их в таком отношении, выделив нужное пространство из полной ширины их родительского контейнера.

```
/* О НАС */
.about-us {
  color: #454546;
}
.about-container {
  max-width: 960px;
  margin: 80px auto;
}
.about-row {
  display: flex;
  align-items: center;
}
.about-column {
  width: 50%;
  padding: 0 15px;
}
.about-column img {
  width: 100%;
}
```

```
.about-us h2,
.about-us h3,
.about-us p {
  margin-bottom: 15px;
}
.about-contact {
  display: flex;
}
.about-contact span {
  margin-bottom: 7px;
}
.about-contact span:first-child {
  flex: 1;
}
.about-contact span:last-child {
  color: #ce416e;
  font-weight: 600;
  flex: 3;
}
```

Рис. 100 – Стили для секции контактных данных

Это наглядно отображено на внешнем виде конечного результата.

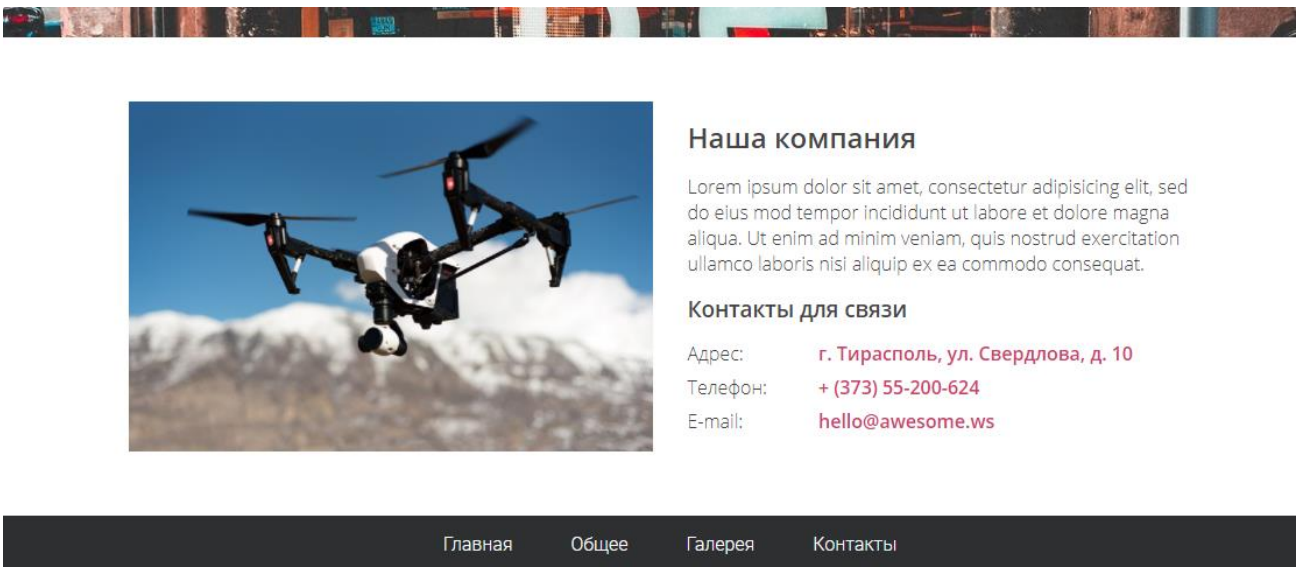


Рис. 101 – Внешний вид верхней секции

Таким образом можно задавать относительную ширину для различного количества столбцов. Так как браузер будет рассчитывать их ширину от родителя, то вложенные элементы не будут выходить за границы.

Для мобильного отображения нам достаточно будет для медиа-запроса **max-width: 992px** ограничить максимальную ширину верхнего блока и повернуть направление дочерних flex-элементов в колонку, при помощи свойства flex-direction: column;

```
456  .about-container {
457    max-width: 450px;
458  }
459  .about-row {
460    flex-direction: column;
461  }
462  .about-column {
463    width: 100%;
464  }
465  .content-left {
466    margin-bottom: 25px;
467  }
```

Рис. 102 – Мобильные стили

С этим набором стилей запись хорошо смотрится и на более узких диагоналях, как видно на рисунке 103. Разбавим нашу страницу живым элементом «Наших достижений». Такие блоки тоже можно часто встретить на страницах сайтов с описанием услуг или сервисов.

Здесь кратко описывают цифры статистики сайта, сервиса или другие данные, которые подчёркивают положительные качества компании. Вёрстка основного блока и колонок похоже на то, что мы делали в разделе выше. Только колонок будет 4, а их ширина соответственно будет равна 25%. На рисунках 104 и 105 приведены стили и код HTML разметки.

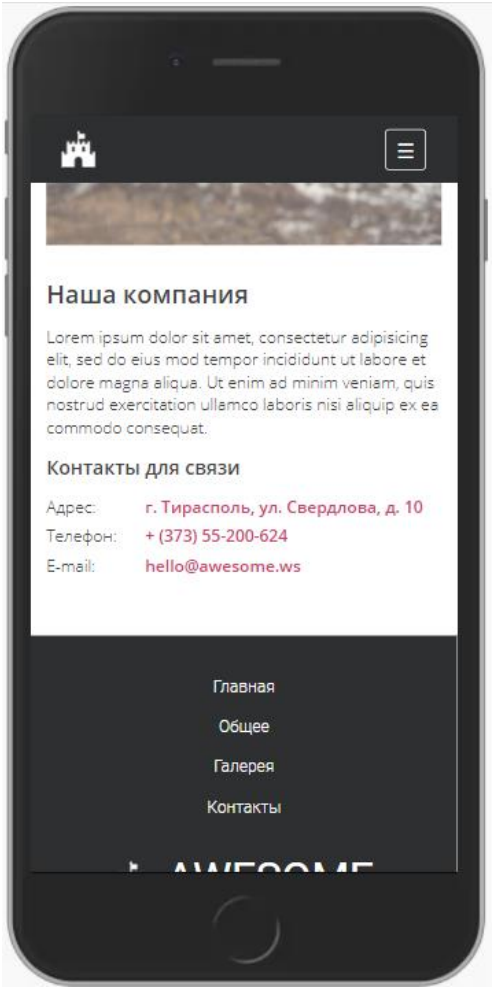


Рис. 103 – Внешний вид записи на мобильной диагонали

```
.achievement-container {
  max-width: 960px;
  margin: 80px auto;
}
.achievement-row {
  display: flex;
  align-items: center;
  padding: 70px 0;
}
.achievement-column {
  width: 25%;
  padding: 0 15px;
}
```

```
.single-achievement {
  text-align: center;
}
.single-achievement p {
  font-size: 18px;
}
.counter {
  font-size: 40px;
  font-weight: 600;
}
.countup {
  display: inline-block;
}
```

Рис. 104 – Стили для секции «наши достижения»

```
<section class="our-achievement">
  <div class="achievement-container">
    <div class="achievement-row">
      <div class="achievement-column">
        <div class="single-achievement">
          <h3 class="counter">
            <span id="secondo1" class="countup" data-end="1250">1250 </span>+
          </h3>
          <p>Видов товаров</p>
        </div>
      </div>
      <div class="achievement-column">
        <div class="single-achievement">
          <h3 class="counter">
            <span id="secondo2" class="countup" data-end="350">350 </span>+
          </h3>
          <p>Заказов ежемесячно</p>
        </div>
      </div>
      <div class="achievement-column">
        <div class="single-achievement">
          <h3 class="counter">
            <span id="secondo3" class="countup" data-end="2500">2500 </span>+
          </h3>
          <p>Довольных клиентов</p>
        </div>
      </div>
      <div class="achievement-column">
        <div class="single-achievement">
          <h3 class="counter">
            <span id="secondo3" class="countup" data-end="250">250 </span>+
          </h3>
          <p>Новых позиций</p>
        </div>
      </div>
    </div>
  </div>
</section>
```

Рис. 105 – Секция «наши достижения» на странице «Контакты»

Также не забываем про адаптивность и создаём стили для ширины экранов меньше 992px. Аналогично предыдущей секции развернём счётчик достижений

в столбец и немного добавим отступов между элементами, а размер текста с числами немного уменьшим.

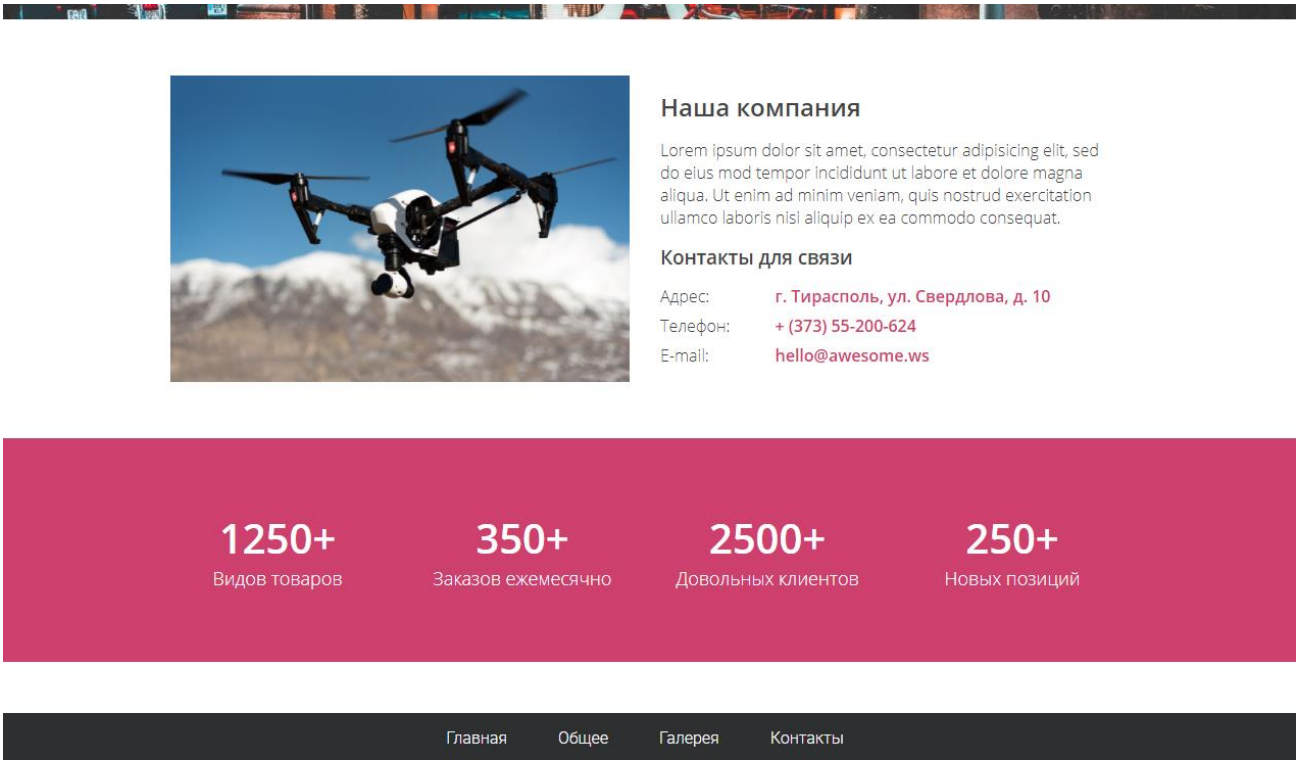


Рис. 106 – Внешний вид секции «Наши достижения»

```
.achievement-container {
  max-width: 450px;
}
.achievement-row {
  flex-direction: column;
  padding: 40px 0;
}
.achievement-column {
  width: 100%;
}
.single-achievement {
  margin: 20px 0;
}
.counter {
  font-size: 32px;
}
```

Рис. 107 – Стили секции «Наши достижения» для диагоналей меньше 992px

Далее добавим интерактивности скучным статичным числам. Добавим при помощи JavaScript небольшую анимацию возрастания значений от нуля до заданного значения. В head страницы добавим после подключения файла popup.js новый файл скриптов counter.js и создадим такой файл в соответствующей папке.

```
// ожидаем загрузки страницы
$(document).ready(() => {
  // проходим циклом по каждому элементу с классом .countup
  $('.countup').each(function() {
    // передаём переменной текущий элемент
    // записываем в переменную конечное значение счётчика из атрибута data-end
    const that = $(this),
        countTo = that.attr('data-end');

    // создаём jQuery функцию animate()
    // которой передаём параметры
    // в первом параметре:
    // countNum - значение, на котором будет остановка счётчика
    $({ countNum: 0 }).animate({
      countNum: countTo
    },
    // второй параметр: объект со значениями
    // 1. длительности анимации,
    // 2. плавности анимации,
    // 3. функции, что должно выполняться на каждом шаге
    // 4. функции, что должно выполнится при завершении
    {
      duration: 8000,
      easing: 'linear',
      step: function() {
        that.text(Math.floor(this.countNum));
      },
      complete: function() {
        that.text(this.countNum);
      }
    });
  });
});
```

Рис. 108 – Код счётчика

Мы могли бы создать код на чистом JavaScript, однако у jQuery есть огромный ассортимент инструментов, значительно ускоряющих разработку. В данном случае использован метод https://api.jquery.com/animate/

Область его применения достаточно широка. Однако, нам важно, что интервал заполнения от 0 до конечного числа будет одинаковым. То есть значение 350 и 5000 будут достигнуты в один момент времени, а не в разное.

Получение данных, обращение к элементам и просчёт шага роста метод выполняет сам, нам необходимо лишь указать нужные параметры.

Часто на страницах «контакты» или «о нас» можно увидеть карту с отметкой физического адреса организации или офиса. Как правило это подключены картографические плагины от внешних сервисов, например, Google Maps или Яндекс.Карты.

Воспользуемся картами от Яндекс по нескольким причинам:

- у Яндекс.Карт документация на русском языке;
- бесплатное подключение;
- лучше наполнены адреса для Приднестровья;
- процесс регистрации в сервисах Google слегка усложнился и потребует под рукой данные кредитной карты, несмотря на бесплатное пользование сервисом. Но платёжные данные теперь являются обязательным условием пользования сервисом.

Конечно, без сети интернет плагин работать не будет. Поэтому для начала перейдём по ссылке https://yandex.ru/map-constructor/

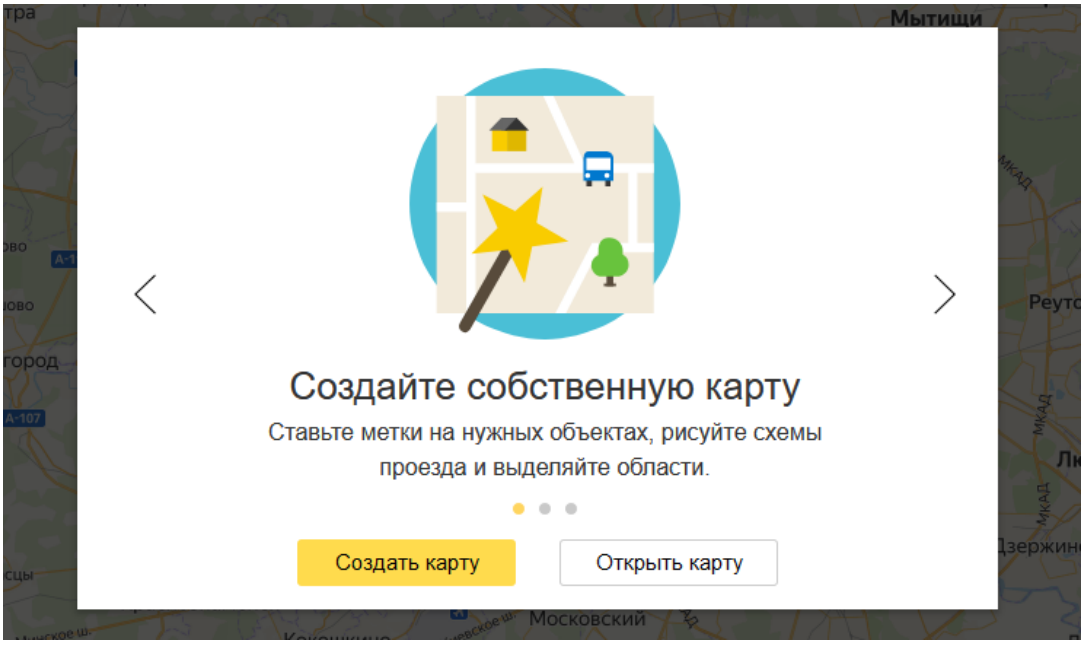


Рис. 109 – Главная страница конструктора карт

В окне приветствия выбираем «Создать карту». Система попросит нас авторизоваться. Авторизуемся под своей учётной записью, или регистрируем новый ящик. Я зарегистрировал новый для демонстрации шагов.

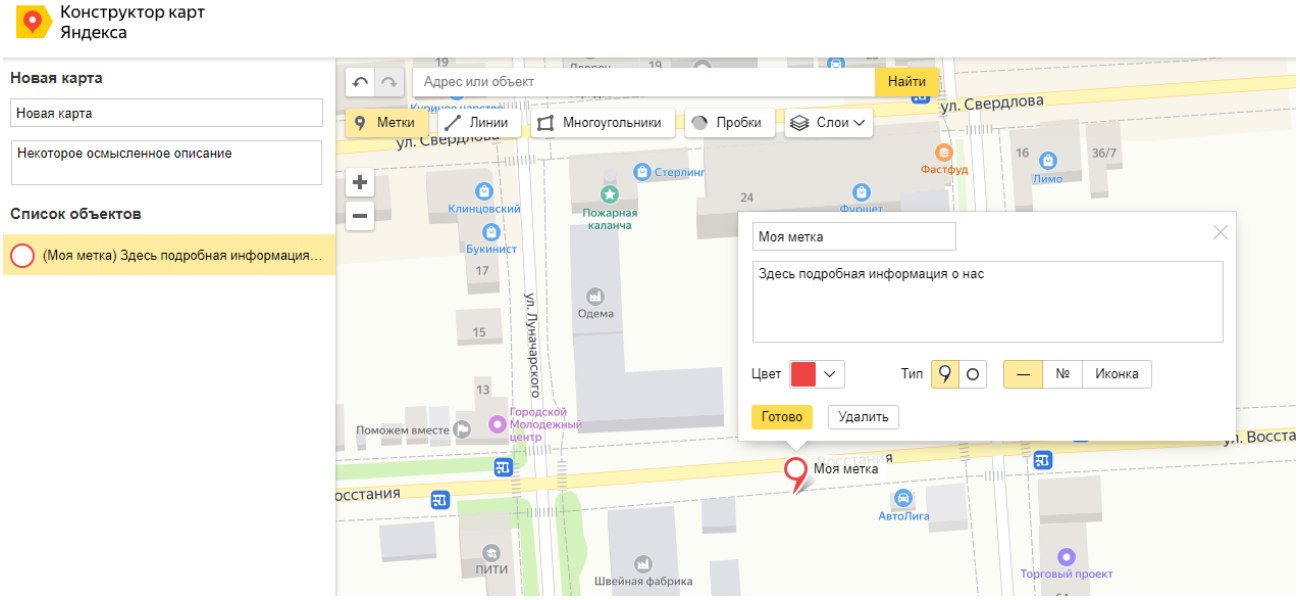


Рис. 110 – Создание проекта Карты

Здесь заполняем поля с названием карты, можем добавить описание, а также добавить метки кликом по значку «Метки» в верхнем ревом углу карты.

Перемещаем метку, настраиваем её внешний вид, а затем жмём «Сохранить и продолжить».

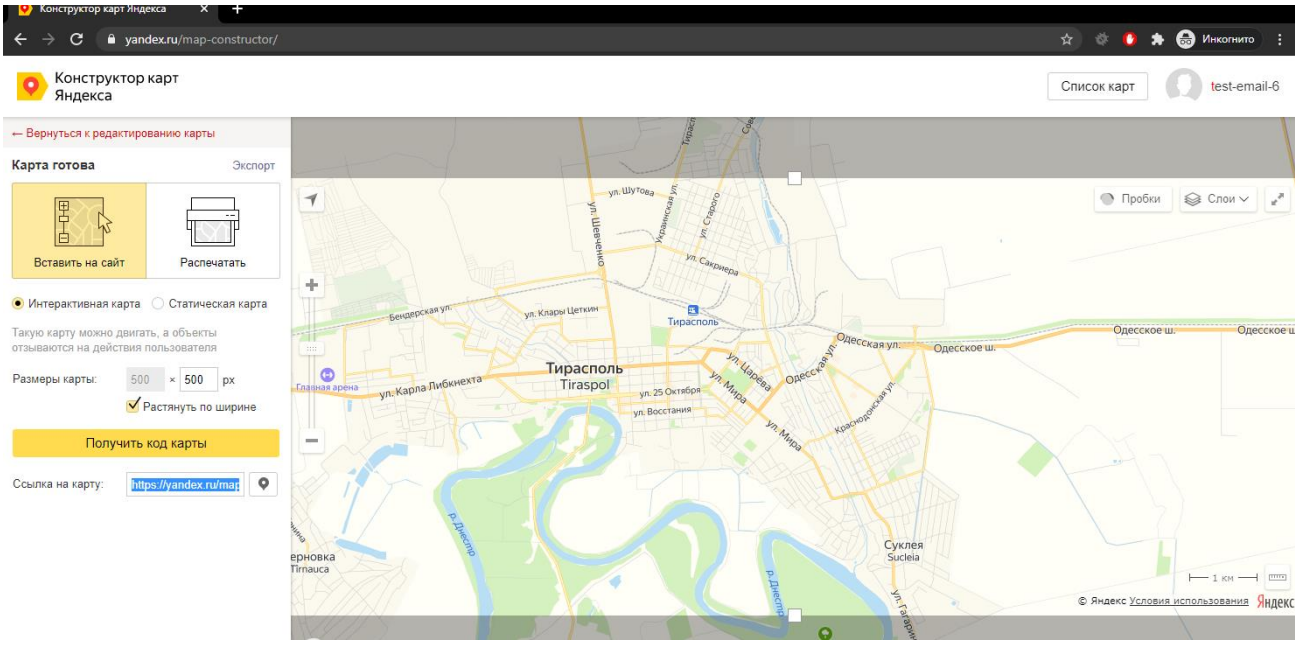


Рис. 111 – Настройки карты

На следующем этапе находим на карте нужный город. Колёсиком мыши устанавливаем в окне предпросмотра масштаб. В настройках оставляем активной интерактивную карту, установим галочку «Растянуть по ширине», а высоту установим 500px. Отлично, жмём «Получить код карты».

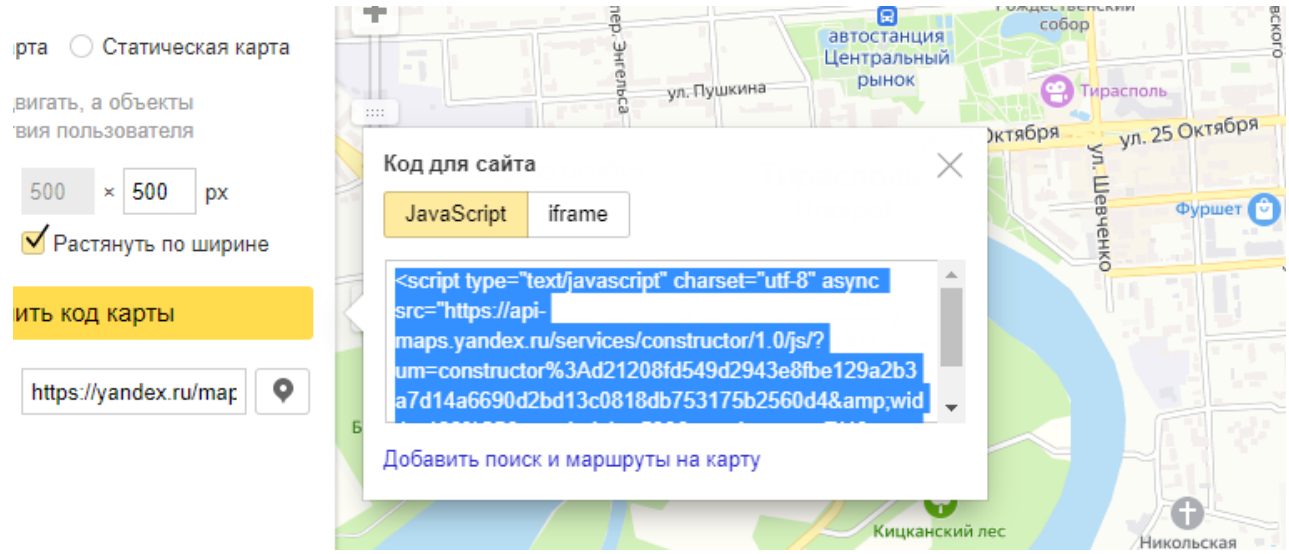


Рис. 112 – Получение кода карты

Теперь скопируем данный код в страницу contacts.html. Создадим секцию с id=”map” и в ней расположим стандартный заголовок секции как на странице index.html и код карты.

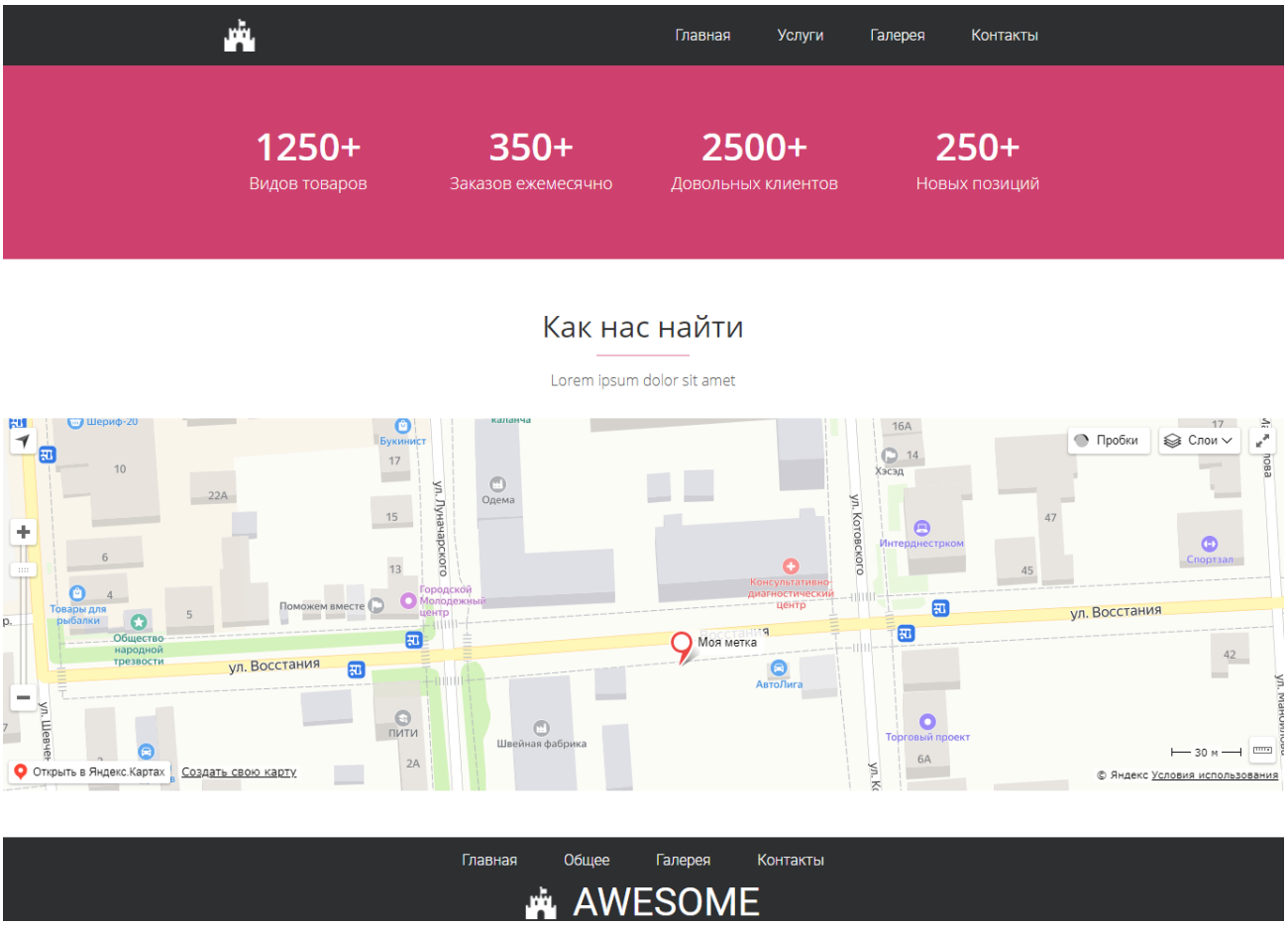


Рис. 113 – Отображение карты на странице сайта

Чуть поправим стили, добавим для карты отступ снизу и до подписи сверху. Можно карту поместить в отдельный тег, для дополнительных стилей, или просто кастомизировать описание для секции .map

```
.map {
  margin: 50px 0;
}
.map .main-title-description {
  margin-bottom: 30px;
}
```

Рис. 114 – Отступ для блока карты

Наконец скопируем секцию с отзывами со страницы index.html. К слову, мы не проверяли в лабораторной работе №5 мобильную вёрстку этого блока. Воспользуемся случаем и поправим стили, чтобы карусель выглядела хорошо, на всех диагоналях.

```
484     .testimonials {
485       margin: 80px 0;
486     }
487     .testimonials-container {
488       max-width: 760px;
489     }
490     .testimonials-card {
491       width: 350px;
492     }
```

Рис. 115 – Стили для медиа-запроса 992px

```
531   .testimonials-container {
532       max-width: 550px;
533     }
534   .testimonials-card {
535       width: 250px;
536     }
```

Рис. 116 – Стили для медиа-запроса 768px

```
560     .testimonials-container {
561       max-width: 100%;
562       padding: 0 15px;
563     }
564     .testimonials-inner {
565       transform: translateX(0px) !important;
566       flex-direction: column;
567     }
568     .testimonials-card {
569       width: auto;
570     }
571     .testimonials-controls {
572       display: none;
573     }
```

Рис. 117 – Стили для медиа-запроса 576px

Обратите внимание на атрибут !important. Он создаёт высший приоритет стилю transform переводя его в нулевое состояние. Анимация на мобильных телефонах не будет работать, а отзывы будут расположены в столбец.

Задание на контроль

1. Повторить пример их практической части.

2. Изменить главное меню.

3. Отрегулировать работу скрипта с учётом изменений.

4. Добавить новую страницу.

5. Создать новый файл скриптов и настроить счётчик.

6. Подключить плагин карт, воспользовавшись внешним картографическим сервисом.

7. Перенести и адаптировать секцию «Отзывы»

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).