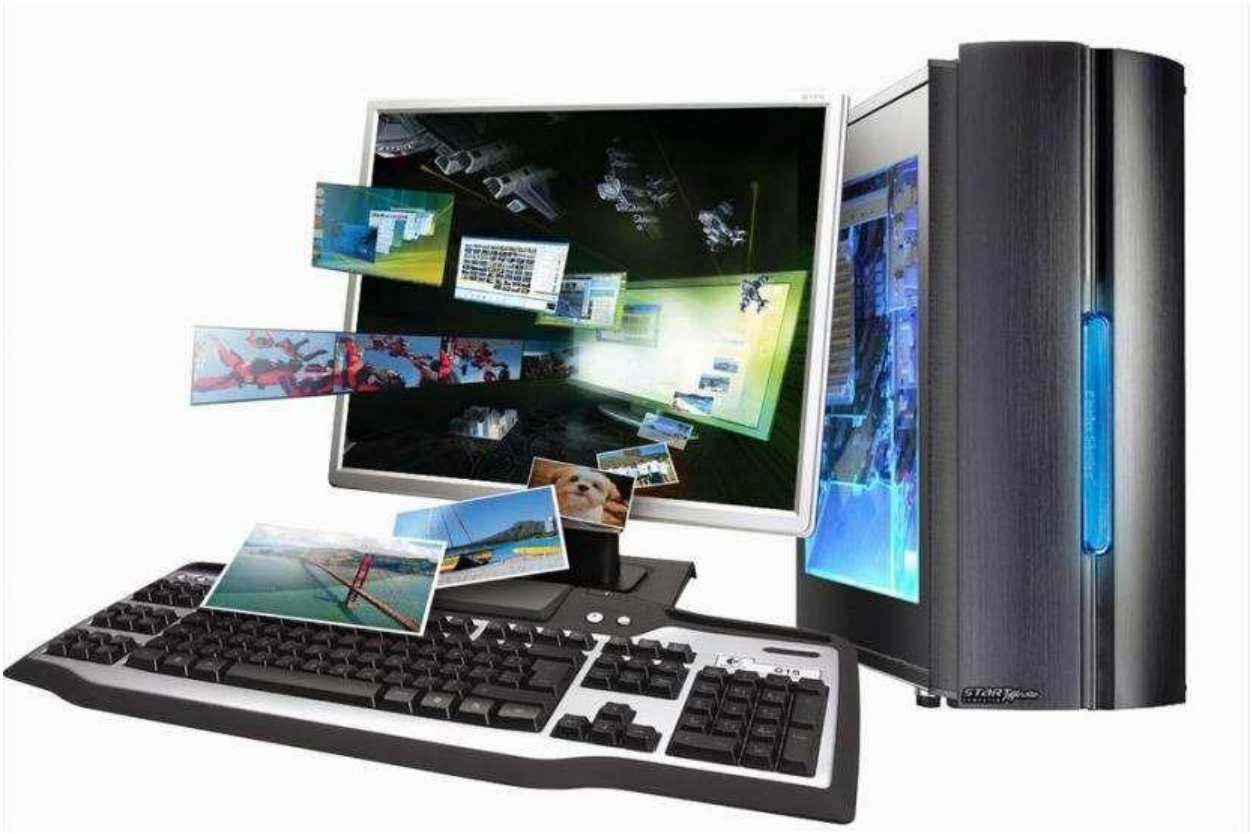
Министерство сельского хозяйства Российской Федерации Федеральное государственное бюджетное образовательное учреждениевысшего образования
Казанский государственный энергетический университет
Кафедра информационных технологий и интеллектуальных систем

Разработка мобильного web-приложения

Методические указания для выполнения лабораторных работ



Казань, 2024

Лабораторная работа
Разработка мобильного web-приложения

Цель работы:

1. Создание адаптивного дизайна для мобильных устройств с использованием CSS и медиазапросов.
2. Работа с мобильными сенсорами и возможностями устройства через JavaScript.
3. Тестирование и оптимизация производительности мобильного web-приложения.

Краткие теоретические сведения

Перед началом создания страничек необходимо подготовить HTML для работы.

В самом начале нужно убедиться в том, что подключён метатег "viewport" со значением "width=device-width", который определяет возможность работы отзывчивости веб-странички. Далее подключаются дополнительные стили, такие какFont Awesome и Normalize.

При работе с Bootstrap всё вышеупомянутое остаётся в силе, а также необходимо подключить и сам фреймворк. Информацию о том, как подключать и использовать фреймворк, можно получить на его официальном сайте (https://getbootstrap.com/). Подключить фреймворк можно двумя способами: скачать библиотеки Bootstrap, распаковать их в папку, где находятся HTML файлы и подключить их к вашему HTML оттуда, либо воспользоваться возможностью подключения к библиотекам Bootstrap посредством ссылок, без необходимости скачивать всю библиотеку. ПодключатьNormalize.ccs, причины использования которого были описаны автором ранее, нет необходимости при работе с Bootstrap, так как сам Normalize, уже, встроен в фреймворк.

```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <!-- My styles -->
    <link rel="stylesheet" href="css/my_styles.css">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-
    MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">
    <!-- Fonts Awesome -->
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/css/all.css" integrity="sha384-
    B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlXeVIU" crossorigin="anonymous">

    <title></title>
  </head>
  <body>
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
    q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-
    ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-
    ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></script>
  </body>
</html>
```

Рисунок 1. Подключение Bootstrap и дополнительных стилей.

После подключения всех необходимых библиотек и файлов можно начинать создавать сами странички. Как правило, странички создаются сверху вниз, поэтому сначала была создана навигационная панель.

```
<nav class="navbar fixed-top navbar-dark navbar-expand-md">
  <a class="navbar-brand ml-3" href="#">Winter Sports Club</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
  aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse mr-3" id="navbarNav">
    <ul class="navbar-nav ml-auto">
      <li class="nav-item active">
        <a class="nav-link" href="index_bt.html">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#about">About</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#services">Services</a>
      </li>
    </ul>
  </div>
</nav>
```

Рисунок 2. Навигационная панель при помощи Bootstrap.

Навигационная панель в Bootstrap создаётся при помощи класса navbar. Класс navbar- expand-md отвечает за отзывчивое "складывание" элементов при определённой контрольной точке, в данном случает, это 768 пикселей. Класс navbar-toggler отвечает за отображение элементов, которые будут "складываться" при изменении размера экрана. Класс

navbar-toggler-icon отображает, так называемое, "меню гамбургер". А в классе navbar-collapse прописываются элементы, которые будут отображаться при "сложившейся" навигационной панели. А также есть класс navbar-brand, который используется специально для логотипов или названия бренда.

```
<div class="nav">
  <div class="nav-header">
    <div class="nav-title">
      Winter Sports Club
    </div>
  </div>
  <div class="nav-btn">
    <label for="nav-check">
      <span></span>
      <span></span>
      <span></span>
    </label>
  </div>
  <input type="checkbox" id="nav-check">
  <div class="nav-links">
    <a href="index.html">Home</a>
    <a href="#about">About</a>
    <a href="#services">Services</a>
  </div>
</div>
```

Рисунок 3. Навигационная панель при помощи HMTL+CSS.

Создание навигационной панели при помощи только HTML и CSS выглядит намногоиначе. Все классы и элементы прописываются вручную, в HTML прописывается основное содержание навигационной панели, а её вид и работоспособность прописываются в CSS. Для правильного отображения и работы элементов, необходимо ссылаться к каждому классу напрямую и прописывать его поведение и вид.

```
.nav {
  height: 50px;
  width: 100%;
  background-color: #02547D;
  position: fixed;
}
.nav > .nav-header {
  display: inline;
}
.nav > .nav-header > .nav-title {
  display: inline-block;
  font-size: 22px;
  color: #fff;
  padding: 10px 10px 10px 10px;
}
.nav > .nav-btn {
  display: none;
}
```

Рисунок 4. CSS код, отвечающий за то, как выглядит навигационная панель

При помощи этих настроек создаётся общий вид навигационной панели, но для того,что- бы сделать панель отзывчивой, необходимо использовать медиа-запросы.

```
@media (max-width:768px) {
  .nav > .nav-btn {
    display: inline-block;
    position: absolute;
    right: 0px;
    top: 0px;
  }
  .nav > .nav-btn > label {
    display: inline-block;
    width: 50px;
    height: 50px;
    padding: 13px;
  }
  .nav > .nav-btn > label:hover {
    background-color: rgba(0, 0, 0, 0.3);
  }
  .nav > .nav-btn > label > span {
    display: block;
    width: 25px;
    height: 10px;
    border-top: 2px solid #eee;
  }
```

Рисунок 5. Фрагмент кода медиа-запроса, отвечающего за отзывчивостьнавигационной панели

С помощью медиа-запросов панель реагирует на изменения размеров экрана иотображается в соответствии с правилами прописанными в медиа-запросе.

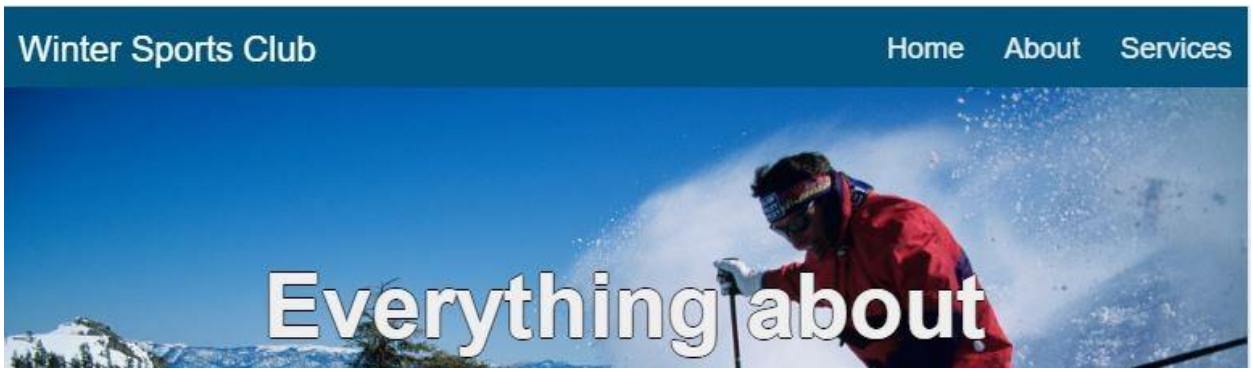


Рисунок 6. Навигационная панель на HTML+CSS

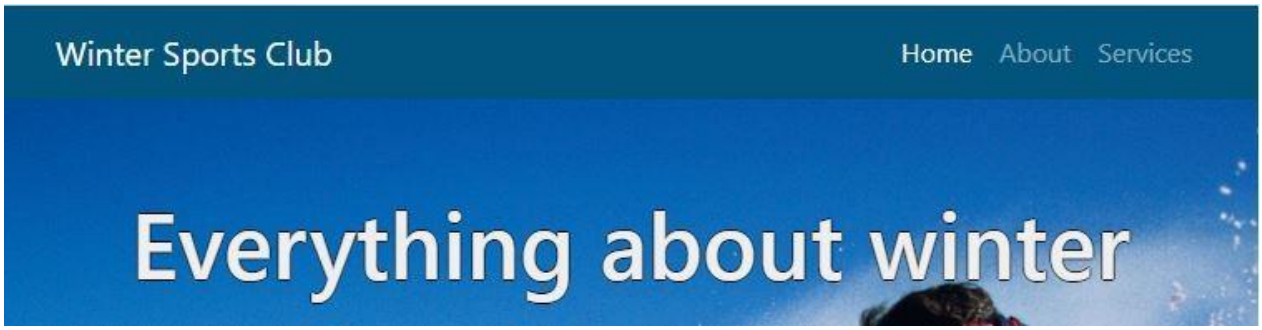


Рисунок 7. Навигационная панель на Bootstrap

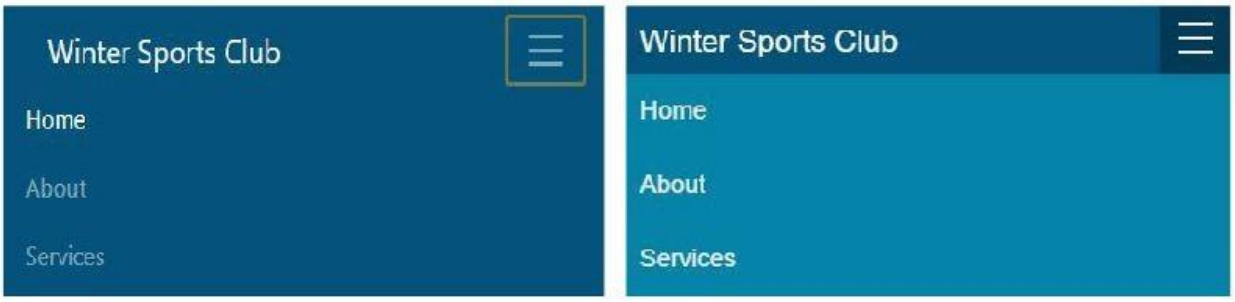


Рисунок 8. Навигационные панели на маленьких экранах. Слева панель Bootstrap,справа HTML и CSS

Одним из основных преимуществ при работе с Bootstrap является его огромнаябиблиотека различных классов, форм и компонентов, многие из которых созданы для применения, часто используемых, элементов в дизайне.

Одним из таких компонентов является компонент jumbotron. Jumbotron - это лёгкий и гибкий компонент, созданный специально для того, чтобы продемонстрировать ключевые моменты на вашем сайте, такие как CTA (Call to action), маркетинговые сообщения и т.д.

```
<div class="jumbotron jumbotron-fluid mb-0">
  <div class="container">
    <h1>Sample text</h1>
    <pSample text</p>
        <a class="btn btn-primary btn-lg" href="#" role="button">Join our club!</a>
  </div>
</div>
```

Рисунок 9. Компонент Jumbotron

Элементы внутри jumbotron автоматически масштабируются, сам jumbotron полностью отзывчивый и не нуждается в применении сетки или

дополнительных изменений в CSS.

В HTML же всё происходит написанием обычных классов и заполнением их содержимым, стилизация и отзывчивость происходит посредством CSS и медиа-запросов.

```
<div class="container" id="showcase-container">
        <h1>Sample text</h1>
        <p>Sample text</p>
        <button type="submit" class="button_1"></button>
</div>
```

```
#showcase-container {
        padding-top: 45px;
}
#showcase h1{
        margin-top: 100px;
        font-size: 55px;
        margin-bottom: 10px;
}
#showcase p{
        font-size: 23px;
}
```

Рисунок 10. HTML код слева, правила CSS справа

Ещё одним хорошим компонентом Bootstrap является компонент Cards.

Компонент cards представляет собой гибкий и расширяемый контейнер содержимого с различными вариантами и опциями. В него можно поместить картинки, заголовки, текст, кнопки, он автоматически масштабируется под размеры своего родительского элемента. С помощью вспомогательных классов можно изменять размеры иповедение карточек, не прибегая к дополнительным настройкам в CSS. К примеру, класс card-deck, с помощью которого можно создать несколько карточек с одинаковыми размерами и отступами, которые будут отображаться в ряд иавтоматически адаптироваться под размеры экрана.

```
<div class="card-deck">
  <div class="card">
    <img class="card-img-top" src".../100px200/" alt="Card image cap">
    <div class="card-body">
      <h5 class="card-title">Card title</h5>
      <p class="card-text">text</p>
    </div>
  </div>
```

Рисунок 11. Фрагмент кода компонента Card с применением класса card-deck.

Компонент Card полностью отзывчивый и не нуждается в

применении сетки или дополнительных изменений в CSS.

Без фреймворка для достижения похожего результата необходимо создать класс (box), в котором будет находиться содержимое, а в CSS прописать стили, чтобы придать классу вид похожий на тот, что используется классами cards в Bootstrap.

```
<div class="box">
        <img src="img/snowboard3.png" alt="snow board img">
        <h3>Snowboarding</h3>
        <p>sample text</p>
</div>
<div class="box">
        <img src="img/skis1.png" alt="skies img">
        <h3>Skiing</h3>
        <p>sample text</p>
</div>
```

Рисунок 12. HTML код класса box

Чтобы достичь похожего эффекта работы класса card из bootstrap, необходимо классу box в CSS применить определённые параметры. Параметр float: left; который разместит все элементы в одну строку (по умолчанию все элементы расположены в столбик), а для достижения отзывчивости элементов им был задан параметр width со значение 30%. А для корректного отображения на маленьких экранах был создан медиа-запрос, в котором параметр float: left; был отменён, а параметру width было задано значение 100%.

```
.box {
        align-content: center;
        float: left;
        text-align: center;
        width: 30%;
        padding: 10px;
        padding-bottom: 60px;
}

.box img{
        max-width: 100%;
}

@media(max-width: 768px){
        box{
              float: none;
              text-align: center;
              width: 100%;
        }
}
```

Рисунок 13. Парамеры класса box в CSS.

Всё основное содержимое в Bootstrap размещается в классах .container или .container- fluid. Класс container имеет фиксированную ширину, а класс .container-fluid располагает элементы по всей ширине экрана. Использование этих классов не всегда обязательно, к примеру, компонент card-deck не требует применения классов

.container для того, чтобы достичь необходимой работоспособности. Если же содержимое, необходимое для отображения, не имеет определённых классов, таких как jumbotron для достижения своей работоспособности, то тогда используются классы .container или .container-fluid с применениями правил сетки для достижения желаемой отзывчивости и адаптивности элементов.

```
<section id="newsletter">
        <div class="container-fluid">
                <div class="row">
                        <div class="col-lg-4">
                                <h2>Subscribe to our news letter</h2>
                        </div>
                        <div class="col-lg-8">
                                <div class="col-md-10 mx-auto mt-1">
            <form>
              <div class="form-row">
                <div class="col-lg-6 offset-lg-3 col-md-9 mb-2 mb-md-0">
                  <input type="email" class="form-control form-control-lg mx-auto" placeholder="Enter your email...">
                </div>
                <div class="col-lg-3 col-md-3">
                  <button type="submit" class="btn btn-block btn-lg w-90 mx-auto">Subscribe</button>
                </div>
              </div>
            </form>
                        </div>
                        </div>
                </div>
        </div>
</section>
```

Рисунок 14. Фрагмент кода раздела newsletter с применением класса .container-fluid исетки bootstrap

В данном примере используется класс .container-fluid для того, чтобы расположить содержимое по всей ширине экрана. Далее идёт класс .row, отвечающий за то, что всеэлементы внутри него будут располагаться в ряд. Классы .col-lg-4 и .col-lg-8 отвечаютза то, сколько колонок будет занимать каждый из элементов на больших экранах, и по такому же принципу прописываются классы, отвечающие за отображение элементов на меньших экранах, разница применения которых была описана автором ранее.

Так же у bootstrap есть и вспомогательные классы, созданные специально для более точечной настройки сетки. Такие классы как .mx-auto - горизонтальное выравниваниеэлементов, класс .mt-1 - отступ сверху, и большое количество других классов, помогающих, более, точно, настроить отображение колонок в сетке bootstrap без необходимости прописывать дополнительные правила этим классам в CSS.

И также как и все другие классы и формы использованные ранее, классы сетки полностью отзывчивые и не требуют дополнительной настройки их поведения черезCSS и медиа-запросов.

Без использования фреймворка основное содержимое располагается похожим образом, создаётся раздел с классами

контейнерами внутри которых и располагается всё основное содержание. Однако в отличие от фреймворка эти классы необходимо настраивать посредством CCS и медиа-запросов для создания им подобающего вида и отзывчивости элементов.

```
<section id="newsletter">
        <div class="container">
                <h1>Subscribe to our news letter</h1>
                        <form>
                                <input type="email" placeholder="Enter Email...">
                                <button type="submit" class="button_1">Subscribe</button>
                        </form>
        </div>
</section>
```

Рисунок 15. HTML код раздела newsletter без фреймворка

Чтобы достичь похожего результата с фреймворком, содержимое классов настраивается в CSS и медиа-запросах для придания им соответствующего вида и отзывчивости.

```
#newsletter {
        padding: 15px;
        color: #ffffff;
        background:#0284A8;
}
#newsletter h1{
        float: left;
}
#newsletter form{
        float: right;
        margin-top: 15px;
}
#newsletter input[type="email"]{
        padding:4px;
        height:38px;
        width: 250px;
}
```

```
@media(max-width: 768px){
        #newsletter h1,
        #newsletter form{
                float: none;
                text-align: center;
                width: 100%;
        }
        #newsletter button, .quote button{
                display: block;
                width: 100%;
        }
        #newsletter form input[type="email"], .quote input, .quote textarea{
                width: 100%;
                margin-bottom: 5px;
        }
}
```

Рисунок 16. CSS и медиа-запросы для стилизации и отзывчивости

Как и ранее, для достижения отзывчивости используются параметры float: left и float:right, располагающие элементы в один ряд, а параметр width задаёт им соответствующую ширину. Для достижения отзывчивости и корректного отображения на маленьких экранах применяются медиа-запросы, отменяющие или меняющие значения этих параметров.

Всё остальное содержимое на страничках было выполнено в похожем порядке, с применением ранее использованных классов, таких как card и сетки при работе с фреймворком, и написанием похожих классов

и настройкой их поведения через CSS и медиа-запросов при работе без фреймворка.

Сравнение вёрстки с фреймворком и без.

Необходимый уровень знаний и умений при вёрстке с помощью фреймворка сильно отличалось от тех знаний и умений, которые требовались для того, чтобы достичь похожего результата посредством "чистого" HTML и CSS.

При работе с bootstrap было необходимо изучить различные классы с которыми пришлось работать. Работа этих классов чётко прописана в документации и всё, что требовалось знать - это какой класс и когда применять. Основное удобство заключалось в том, что в библиотеке фреймворка этим классам было прописано определённое поведение, и чётко знать правила CSS и медиа-запросы не требовалось.При работе без фреймворка была выявлена чёткая необходимость детальных знаний правил CSS и медиа-запросов, нужно было точно знать, какое правило применять и вкакой ситуации, как оно будет работать и реагировать при определённых его параметрах.

Основным преимуществом работы с фреймворком были его компоненты и система сетки. Сетка чётко определяла правила отступов, расположения элементов, то как содержимое будет реагировать на изменения размеров экрана, а компоненты упрощали работу с часто используемыми элементами дизайна и вёрстки. И всё это было доступно посредством настроенных классов фреймворка, без необходимости использования CSS или медиа-запросов. Хотя настроить работу этих классов и

компонентов можно и самому, переписав их стандартное поведение, прописанное в библиотеке фреймворка с помощью собственных стилей CSS, такие действия не рекомендовались, да и были довольно проблематичным процессом.

Без фреймворка работа проходила похожим способом. Были

созданы различные классы и потом настроены с помощью CSS и медиа-запросов. Таким образом, эти классы становились, своего рода, компонентами, написанными вручную и в отличии от bootstrap, гарантировать правильную работу класса или компонента мог только сам верстальщик.

Немаловажной частью всего процесса, также, являлась популярность или распространённость того или иного способа вёрстки. Bootstrap, будучи одним из самых популярных и распространённых фреймворков, имеет под собой огромную базу информации и любителей. Благодаря этому, при поиске необходимой информации о работе какого то класса или элемента, достаточно, просто, найти интересующую информацию или помощь, так как у фреймворка все компоненты, классы и т.д. имеют чёткие правила поведения, что позволяет быстрее и удобнее сузить радиус поиска проблемы.

В случае же с вёрсткой, посредством "чистого" HTML+CSS, всё гораздо проблематичнее. Если у фреймворка установлены чёткие правила как действовать, то при вёрстке на HTML+CSS таких правил нет. У каждого верстальщика свой стиль написания кода, что делает процесс поиска ответа на выявленную проблему гораздо сложнее. Приходится учитывать большое количество нюансов, как написан HTML, какие правила прописаны в CSS и т.д.

Задание 1.

Зарегистрироваться на курс «Разработка Android-приложений для мобильных устройств» на открытой платформе образования https://openedu.ru/course/ITMOUniversity/ANDROID/

В дельнейшем необходимо выполнять задания курса и изучать теоретический материал в рамках самостоятельной работы.

Задание 2.

Изучить возможности следующих он-лайн сервисов:

- https://proto.io/
- https://marvelapp.com/
- https://www.fluidui.com/

Задание 3. Ознакомтесь со способами создания приложений под ОС Android в среде AndroidStudio. Вам предстоит создать простое приложение по типу списка покупок (контактов, задач итп). Можно просматривать весь список, добавлять новый элемент списка, можно удалять и, если получится, редактировать.Приблизительный план работы будет в самом конце.

Поскольку в задачах подобного рода принципиально не терять данные между запусками приложения, мы будем использовать базу данных sqlite, инструменты для работы с которой уже есть в составе androidsdk.

Создание проекта в AndroidStudioFile → New → NewProject

Затем выбираете имя приложения и жмете next. Создание андроид проекта. Выбор версии SDK

Выбираем девайс, на котором мы будем запускать наше приложение. Далее Next

→ Next → Finish. Должен запуститься эмулятор Android с пустым проектом. Теперь, когда все готово к работе, можно приступать. Приложение будет состоять из одного экрана, в котором будет список объектов. Кнопки в левом нижнем углу (уже создана) и всплывающем окне, которое будет появляться при нажатии на кнопку.

Задание 4

Цель- изучение функционала класса Activity

Activity - это класс, который «отвечает за 1 экран». То есть за все элементы интерфейса, расположенные на нем и за их работу (обработка нажатий на кнопки итп). У нас будет только одно Activity, которое за нас создал AndroidStudio. За логику работы Activity отвечает java-класс (в нашем случае MainActivity.java) и его лейаут (activity_main.xml).

Лейаут это XML файл, в котором описывается интерфейс экрана. После того, как был создан новый экземпляр Activity, у него вызывается метод

onCreate, в котором, как правило, происходит первоначальная настройка экрана, такая как инициализация свойств, подгрузка данных и т. п.

Лейауты также можно создавать самостоятельно.

Чтобы создать свой лейаут нужно в папке res создать новый файл лейаута и задать ему имя.

Создание лейаута

Мы можем перетащить нужные элементы интерфейса с Palette (также как мы делали с listview). Нам понадобятся элементы TextView. Их можно настроить во вкладке Properties, ему необходимо проставить идентификатор (также как с listview). Также со свойствам можно работать через редактор xml.

ListView

Для отображения списка объектов можно воспользоваться элементов интерфейса ListView, который представляет собой последовательно (сверху вниз) отрисованные «ячейки» со своим содержимым. Чтобы добавить listVIew на наше активити, можно воспользоваться вкладкой Palette → Containers → ListView.

ViewAllProperties открывает доступ ко всем свойствам объекта.

Задание 5

Цель – изучить назначение класса Adapter

Adapter служит источником данных для ListView. Адаптер имеет базовый класс BaseAdapter от которого нужно унаследовать, чтобы создать свой. Адаптер должен возвратить количество объектов, которые мы хотим отобразить в лист вью а также возвращать сконфигурированный вид для каждого объекта.

Планработы

1. Создатьпроект.
2. ПеренестиListViewна main_activity.xml
3. Создать адаптер для ListView
4. Создать механизм создания записи в БД

Задание 6

Цель – изучить особенности создания мобильных приложений в Visual Studio. Используя VisualStudio, можно создавать приложения для устройств Android, iOS и Windows. Поддерживается создание приложений с помощью C# и .NET Framework, HTMLи JavaScript или C++. Существует возможность совместного использования кода, строк, изображений, а в некоторых случаях даже пользовательского интерфейса.

Задание: создайте кроссплатформенное мобильное приложение калькулятор.

Задание №7

Цель – научиться применять отдельные виды тестирования мобильных приложенийЗадание:

Изучите правила работы с веб-сервисом для управления процессом тестированияSitechco - https://sitechco.ru/.

Разработайте чек-лист для тестирования мобильного приложение Создайте функциональные тесты для проверки мобильного приложения Результаты тестирования сохраните в отчет