Лабораторная работа
Создание одностраничного приложения на React

Задание. Создать одностраничное приложение «Учись на программиста в ИМКТ!».

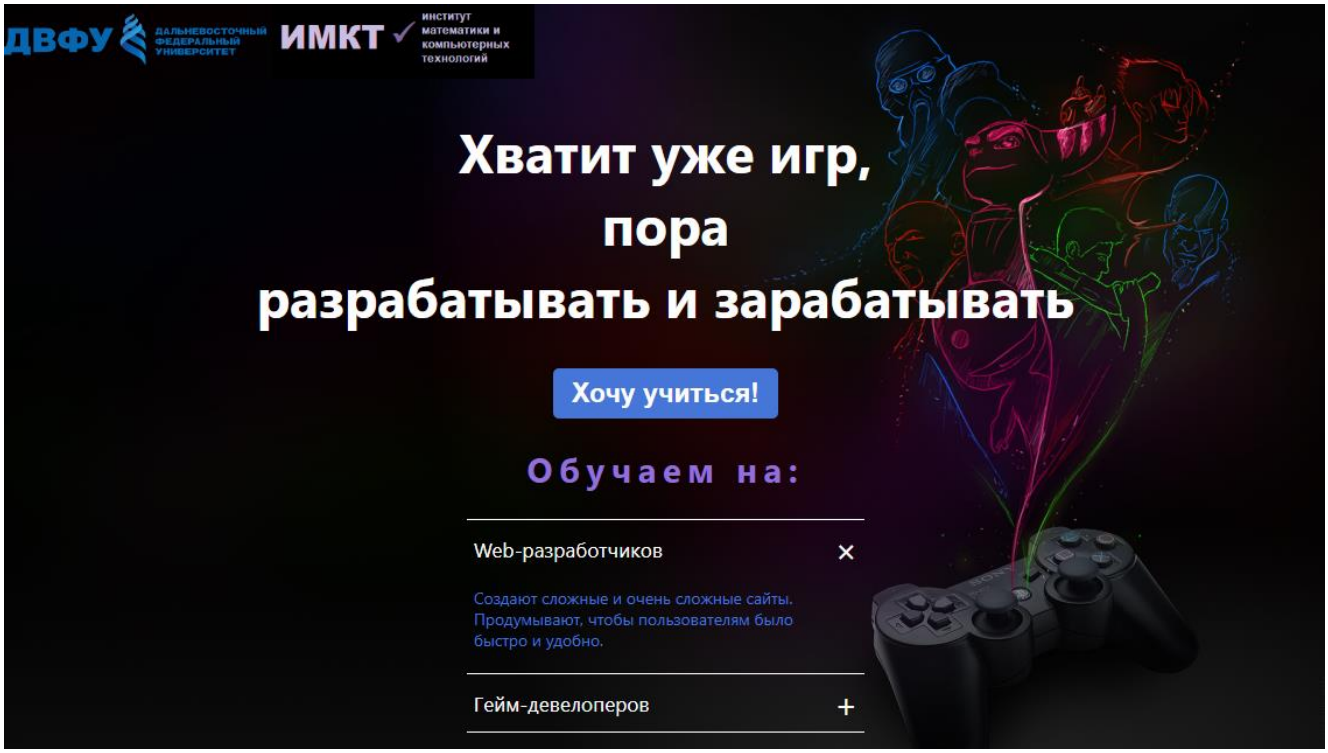
Последовательность шагов проектирования

На кого ориентировано? Данное приложение ориентировано на старших школьников, которые выбирают, куда им пойти учиться после окончания школы.

Какую проблему решает? Помогает сделать выбор будущей профессии. Приложение должно показать абитуриенту, кем можно работать после окончания университета и на какие программы для этого можно выбрать.

Какие материалы нужно разместить на странице:
- логотипы ДВФУ и ИМКТ;
- слоган, привлекающий внимание «Хватит игр, пора разрабатывать и зарабатывать»;
- кнопку «Хочу учиться!»;
- раздел «Обучаем на:» с перечнем будущих профессий и расшифровкой этих профессий;
- раздел с программами подготовки «Выбирай программу:» с перечнем программ и их краткой характеристикой.

Как должна выглядеть страница, ее шаблон:



Что может делать пользователь на странице?
- по клику на строку с профессией должна показываться информация об этой профессии, при этом знак «+» должен меняться на «х»;

- при повтором клике на строку с профессией – подробная информация должна скрываться, а знак «x» меняться на «+»;

- по клику на кнопку «Хочу учиться!» можно перейти на раздел с программами подготовки, при этом кнопка должна «реагировать» на наведение мыши и клике на нее;

- действие пользователя в разделе «Выбирай программу» и структуру раздела придумать самостоятельно.

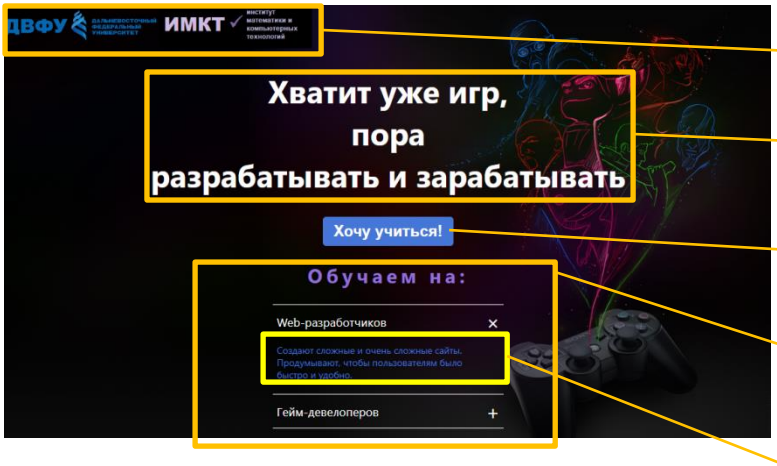
Разработка одностраничного приложения

1. Создать проект, например, с помощью **Create React App**

```
npx create-react-app my-app
cd my-app
npm start
```

В браузере автоматически откроется страница с приложением.

2. На основе шаблона страницы выделить компоненты, которые будут включены в проект:



Head – список логотипов

Tagline– слоган

Button – кнопка

Profession – блок с профессиями и их описанием

ProfItem – описание одной профессии

3. Подготовить данные и изображения, которые будут использоваться на странице.

Изображения размещаем следующим образом:

- те, которые указываются в CSS (например, фон страницы) – в папку **src** проекта;

- остальные – в папку **public** (можно и **src**, но тогда их нужно импортировать, например, **import Logo from './logo.png'**).

Списки и другие данные размещаем в **index.js** (папка **src**):

```
const listProf= [
    {prof: "Web-разработчиков",
     discr: `Создают сложные и очень сложные сайты. Продумывают, чтобы
             пользователям было быстро и удобно.`},
    {prof: "Гейм-девелоперов",
     discr: `Создают видеоигры. Погружают всех нас в новые миры.`},
    {prof: "AI/ML-специалистов",
     discr: `Используют в деле искусственный интеллект и машинное
             обучение. Фактами и прогнозами делает бизнесу хорошо.`},
    {prof: "Аналитиков данных",
```

```
    discr: `С помощью чисел решают, куда двигаться компаниям. Помогают
            бизнесу получать еще больше денег.`},
   {prof: "Мобильных разработчиков",
    discr: `Создают мобильные приложения, которые найдут тебя везде.
            Умещают на маленьких экранах максимальный функционал.`}
]
```

4. В файле **index.css** будем размещать стили страницы. Установим фон, белый цвет текста и выравнивание текста по центру (добавить к уже существующим стилям):

```
body {
    background: url("back_1.png") no-repeat black;
    background-position: top right;
    text-align: center;
    color: white;
}
```

5. Создаем контент страницы в виде компонента и ренедерим его. В этот компонент постепенно будем добавлять JSX разрабатываемых компонентов:

```
function Content() {

  return(
    <>
     /* Здесь будут компоненты */
    </>
  )
}
root.render(<Content />)
```

В результате в браузере должна отобразиться пустая страница с фоном.

6. Создаем компонент **Head**, проверяем его работу в браузере, создаем для него стили. Так как количество логотипов может меняться, занесем их в список.

JavaScript код:

```
const listImg = ["logo_dvfu.png", "logo_imct.png"]

/*Передаем в качестве пропса список с рисунками*/
function Head(props) {

  const logoImages =listImg.map((img, index) =>
            <img key={index} src={img} />
       );
  return(
    <div className="head">
      {logoImages}
    </div>
  )
}
```

CSS код:

```
.head {
    display: grid;
    grid-template-columns: 1fr 1fr 3fr;
    align-items: center;
    column-gap: 0.5vw;
}
```

```
.head img{
    width: 100%;
}
```

Включаем созданный компонент в **Content**, в качестве пропса передаем список с логотипами.

```
function Content() {

  return(
    <>
      <Head listImg={listImg} />
    </>
  )
}
```

В результате на странице должны отобразиться два логотипа в верхнем левом углу.

7. Создаем компонент **Tagline**, проверяем его работу в браузере, создаем для него стили. В CSS размер текста указываем в гибких единицах, в зависимости от ширины экрана.

JavaScript код:

```
function Tagline() {

  return(
    <h1>
      Хватит уже игр, <br/> пора <br/> разрабатывать и зарабатывать
    </h1>
  )
}
```

CSS код:

```
h1 {
    font-size: 4vw;
}
```

Включаем созданный компонент в **Content**, в результате слоган должен отобразиться на странице.

8. Создаем компонент **Button**, в качестве пропса передаем надпись на кнопке, проверяем работу компонента в браузере, создаем для него стили.

JavaScript код:

```
function Button(props) {

  return (
    <input className="button" type="button" value={props.val} />
  )
}
```

CSS код:

```
.button {
    border: 0;
    border-radius: 5px;
    background: #4676D7;
    color: #ffffff;
    padding: 8px 16px;
    font-size: 2vw;
    cursor: pointer;
    font-weight:bold;
}
```

```
.button:hover {
    background: #1d49aa;
}

.button:focus {
    outline: none;
    box-shadow: 0 0 0 4px #cbd6ee;
}
```

Включаем созданный компонент в **Content**, в результате на странице появится кнопка.

9. Создаем компонент **Profession**, который будет отображать список профессий и их описание. Этот компонент будет включать еще один компонент **ProfItem** для описания одной профессии. Также для этого компонента будет реализована динамика.

   Шаг 1. В компонент сначала включаем только названия профессий, в качестве пропсов передаем текст заголовок и массив **listProf**. Создаем заголовок компонента и список <ul>, в качестве элементов которого будут выступать названия профессий.

   JavaScript код:

```
function Professions (props) {

  const listProf = props.list.map((item, index) =>
            <li key={index}> {item.prof}  </li>
        );
  return (
    <div className="prof">
      <h2>{props.title} </h2>
      <ul>
        {listProf}
      </ul>
    </div>
  )
}
```

   CSS код:

```
.prof{
    width: 30%;
    margin: auto;
}

.prof ul {
   padding: 0;
   font-size: 1.5vw;
}
```

   Включаем созданный компонент в **Content:**

```
<Professions title="Обучаем на:" list={listProf} />
```

   В результате на странице появится список с профессиями.

   Шаг 2. Создаем компонент ProfItem. В него в качестве пропсов передаем название профессии и ее описание. Добавим в строку с названием профессии знак «+». Название профессии и знак «+» поместим в span.

   JavaScript код:

```
function ProfItem(props) {

  return(
```

```
      <li>
        <span className="left">{props.prof}</span>
        <span className="right">+</span>
        <p> {props.discr}</p>
      </li>
    )
}
```

CSS код:

```
.prof li {
    display: grid;
    grid-template-columns: 6fr 1fr;
    justify-content: space-between;
    align-items: center;
    border-top: solid white 0.15vw;
    padding: 0.5vw;
    list-style-type: none;
    cursor: pointer;
   }

span.left {
   text-align: left;
}

span.right {
    text-align: right;
    font-size: 2.5vw;
}

.prof li p {
    color: #4676D7;
    font-size: 1.2vw;
    text-align: left;
}
```

Включаем созданный компонент в **Professions** (вместо строки, выделенной серым):

```
<ProfItem key={index} prof={item.prof} discr={item.discr} />
```

В результате на странице появится список с профессиями и их описанием.

Шаг 3. Создаем динамику компонента **ProfItem**. По клику на строку с названием профессии ее описание должно показываться, если оно скрыто, и открываться, если оно отображено на экране. Для этого

- описываем состояние **isOpen**, которое будет иметь значение **false**, если описание скрыто, и **true**, если открыто. В начальный момент времени – описание скрыто;
- создаем хук, в котором описываем состояние и функцию обработки этого состояния:

```
const [isOpen, setOpenClose] = React.useState(false);
```

- создаем функцию, которая будет менять состояние:

```
const press = () => {
  setOpenClose(!isOpen);
}
```

- вызываем эту функцию по клику на <**li**>;

- показываем описание профессии, если состояние **isOpen** установлено в **true**.

JavaScript код:

```
function ProfItem(props) {

  const [isOpen, setOpenClose] = React.useState(false);
  const press = () => {
    setOpenClose(!isOpen);
  }
  return(
      <li onClick={press}>
        <span className="left">{props.prof}</span>
        <span className="right">+</span>
        {isOpen &&
          <p> {props.discr}</p>
        }
      </li>
    )
```

В результате на странице будет открываться/скрываться описание профессии по клику на **<li>**.

Самостоятельные задания

1. В компоненте **ProfItem** выводить «+», если описание профессии скрыто, и «×», если открыто.

2. Создать (информацию взять либо с официального сайта ДВФУ, либо с https://russky.digital/, добавить описание 3-4 направлений подготовки с КРАТКОЙ информацией):

- раздел для вывода информации о направлениях подготовки;
- по клику на кнопку «Хочу учиться» перейти на этот раздел страницы.