Лабораторная работа 2
Основы React

Содрежание:

1. Задание 1: Изучить карточку профиля пользователя
2. Задание 2: Создать многоразовые карточки сотрудников
3. Задание 3: Отобразить список онлайн-курсов.
4. Вопросы к защите лабораторной работы

1. Задание: изучить карточку профиля пользователя

Изучите простую карточку профиля пользователя с помощью JSX. Карточка профиля включает в себя:

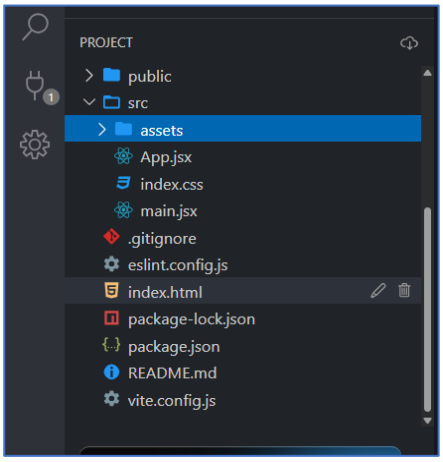
1. Изображение (**img** элемент).
   - Атрибут **src** должен быть равен переменной **avatarUrl**;
   - Атрибут **alt** должен быть равен строке **User Avatar**;
   - Атрибут **width** должен быть равен **120**.
2. Заголовок (**h2** элемент). Его текстовое содержимое должно быть **Alex Johnson.**
3. Абзац (**p** элемент). Его текстовое содержимое должно быть **Frontend Developer.**
4. Ещё один абзац (**p** элемент). Его текстовое содержимое должно быть **Loves React and building UI projects.**

Код решения по ссылке: https://stackblitz.com/github/olreyplay/1dcaf86a-11aa-492e-8e1d-06e055479aa9-challenges/tree/main/challenge-01-solution-build-a-user-profile-card?file=src%2FApp.jsx

StackBlitz — это онлайн-среда разработки (IDE), которая работает прямо в браузере. В вашем случае это **React-проект**, загруженный из GitHub. StackBlitz умеет импортировать публичные GitHub-репозитории и запускать их прямо в браузере.

То есть вам **не нужно устанавливать VS Code, Node.js и React на компьютер**, чтобы выполнять такие учебные задания. StackBlitz запускает среду разработки в браузере, в проектах на WebContainers Node.js работает внутри браузера. **StackBlitz** можно воспринимать как онлайн-аналог: среда VS Code с Node.js и браузер для просмотра результата.

Рисунок связи файлов в React.



1. index.html – основа страницы. В нём находится HTML-контейнер, например:

```
...
<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
 </body>
...
```

React будет помещать приложение именно внутрь этого элемента.

2. main.jsx – запускает React-приложение. Обычно там находится что-то вроде:

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
 <StrictMode>
  <App />
 </StrictMode>,
)
```

То есть **main.jsx** передает команду React: «Найди элемент **root** в **index.html** и отобрази внутри него компонент **App**».

3. App.jsx — само приложение. В App.jsx находятся компоненты и логика приложения:

```
const avatarUrl =
  "https://codefinity-content-media-v2.s3.eu-west-1.amazonaws.com/courses/1dcaf86a-11aa-492e-8e1d-06e055479aa9/9ea7774a-16ac-4122-9a8b-a9a9d105e7c5/boy.png";

// Change the code below this line.
const App = () => {
  return (
    <>
      <img src={avatarUrl} alt="User Avatar" width={120} />
      <h2>Alex Johnson</h2>
      <p>Frontend Developer</p>
```

```
      <p>Loves React and building UI projects</p>
    </>
  );
};
// Change the code above this line.

export default App;
```

Поэтому index.html – это HTML-страница, main.jsx – точка входа React, а App.jsx – главный компонент приложения (рис. 1).

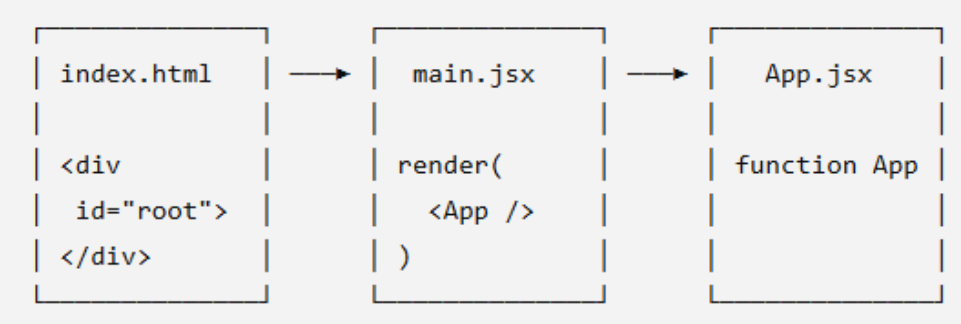


Рисунок 1 – Общая схема взаимосвязи файлов в React-приложении

Задание2. Создать многоразовые карточки сотрудников

Предвартельно необходимо Изучить многоразовый **UserCard** компонент, отображающий информацию о пользователе с помощью свойств (props). **UserCard** компонент должен включать в себя:

1. Заголовок (**h2** элемент). Его текстовое содержимое должно отображать **name** свойство `display`.
2. Абзац (**p** элемент). Его текстовое содержимое должно отображать **job** свойство.

Компонент **App** уже отображает несколько **UserCard** компонентов с разными свойствами.

Код решения по ссылке: https://stackblitz.com/github/olreyplay/1dcaf86a-11aa-492e-8e1d-06e055479aa9-challenges/tree/main/challenge-02-solution?file=src%2Findex.css,src%2FApp.jsx

Задание**:** Создайте React-компонент EmployeeCard, который получает через props информацию о сотруднике и отображает её на странице. Компонент принимает следующие свойства:

- name — имя сотрудника;
- job — должность;
- department — отдел;
- experience — стаж работы.

В компоненте App создайте **три карточки сотрудников** с разными данными.

Задание 3: Отобразить список онлайн-курсов.

Предварительно изучить учебный пример. Выводится список фильмов, используя **map()** метод. Компонент **MovieList** включать в себя:

1. Элемент **ul.**

2. Для отображения lікаждого фильма в moviesмассиве используйте метод **“map()”.**

3. Каждый **li** элемент должен:

- Отобразить название фильма.
- Добавьте уникальное keyсвойство, равное **movie.id.**

Компонент **App** уже передает **moviesData** массив другому **MovieList** компоненту.

Код решения по ссылке: https://stackblitz.com/github/olreyplay/1dcaf86a-11aa-492e-8e1d-06e055479aa9-challenges/tree/main/challenge-04-solution-render-a-movie-list?file=src%2FApp.jsx

```
const moviesData = [
  { id: "m1", title: "Interstellar" },
  { id: "m2", title: "Inception" },
  { id: "m3", title: "The Batman" },
  { id: "m4", title: "Dune" },
];

// Change the code below this line.
const MovieList = (props) => {
  return (
    <ul>
      {props.movies.map((movie) => (
        <li key={movie.id}>{movie.title}</li>
      ))}
    </ul>
  );
};
// Change the code above this line.

const App = () => {
  return (
    <>
      <MovieList movies={moviesData} />
    </>
  );
};

export default App;
```

В этом полном файле становится хорошо видно, как данные проходят от **moviesData → App → MovieList → map() → **. Разберём весь код по шагам.

1. Создаём данные. **moviesData** — это массив из 4 объектов. У каждого фильма есть два свойства:

-id — уникальный идентификатор;

- title — название фильма.
-

2. Создаём компонент **MovieList** - React-компонент.
   props — объект, через который компонент получает данные от родителя.

3. Что передаётся в **props**? Смотрим на **App**:
   Вот эта строка: **<MovieList movies={moviesData} />** означает “Передай компоненту MovieList значение moviesData под именем movies”.
   Получается:

```
moviesData
   ↓
<MovieList movies={moviesData} />
   ↓
props.movies
```

4. **return**. Компонент возвращает HTML-список **<ul>**.

5. **props.movies.map().** Здесь запускается инструкция “Возьми массив props.movies и пройдись по каждому фильму”. Например, в первый раз:
   movie = { id: "m1", title: "Interstellar" }

6. Создаём список **<li>**. Для каждого фильма выполняется:
   **<li key={movie.id}>{movie.title}
**

key нужен **React**, чтобы понимать, какой именно элемент списка изменился, добавился или удалился. При этом key **не выводится пользователю**.

То есть для фрагмента: <li key="m1">Interstellar, на странице будет просто: Interstellar.

Можно даже представить это как таблицу

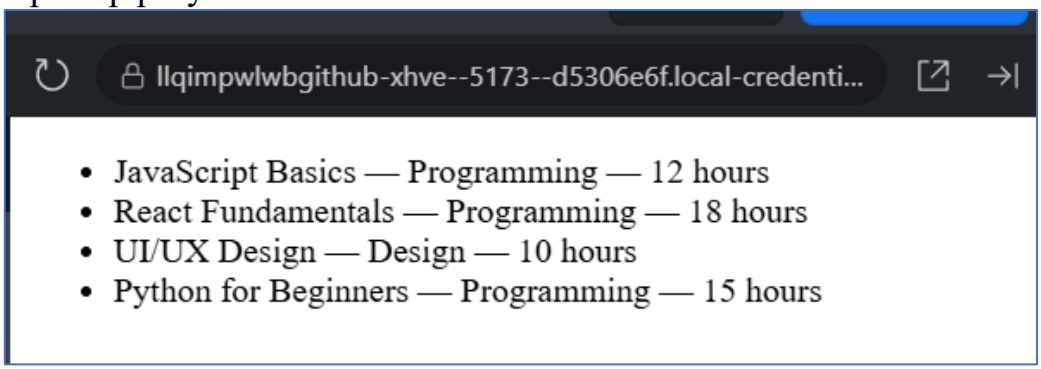
Шаг	Значение
moviesData	массив фильмов
movies={moviesData}	передаём массив в компонент
props	объект с переданными данными
props.movies	получаем массив
map()	перебираем массив
movie	текущий фильм
movie.id	уникальный ID фильма
movie.title	название фильма
	HTML-элемент для отображения

Задание. Создайте React-приложение, которое отображает список онлайн-курсов. В компоненте App уже подготовлен массив:

```
const coursesData = [
  { id: 1, title: "JavaScript Basics", category: "Programming", duration: 12 },
  { id: 2, title: "React Fundamentals", category: "Programming", duration: 18 },
  { id: 3, title: "UI/UX Design", category: "Design", duration: 10 },
  { id: 4, title: "Python for Beginners", category: "Programming", duration: 15 }
];
```

Требования. Создайте компонент **CourseList**, который получает массив **courses** через **props**. Компонент должен:

1. Отображать каждый курс с помощью метода **map().**
2. Для каждого курса создавать отдельный элемент списка.
3. Использовать **course.id** в качестве значения **key**.
4. Отображать:
   - название курса;
   - категорию;
   - продолжительность.
5. Использовать отдельный компонент **CourseItem** для отображения одного курса.
6. Передавать данные курса из **CourseList** в **CourseItem** через props.
7. Пример результата



4. Вопросы к защите лабораторной работы

1. Что такое props в React и для чего они используются в данном приложении?
2. Что находится в props.courses в компоненте CourseList?
3. Для чего используется метод map() в данном коде?
4. Что означает переменная course в выражении props.courses.map((course) => ...)?
5. Для чего используется key={course.id}?
6. Чем отличаются coursesData, props.courses и course?
7. Что произойдёт, если вместо {course.title} написать {course.category}?
8. Почему в App используется <CourseList courses={coursesData} />, а внутри CourseList обращаемся к props.courses?

9. Как изменить код, чтобы для каждого курса отображались название, категория и продолжительность на отдельных строках?
10. Что произойдёт, если добавить новый объект в массив coursesData? Нужно ли изменять компонент CourseList?