Лабораторная работа 2

**Загрузка информационных данных о авиа турах на React.**

***Знакомство с хуком useEffect. Запрос на сервер с помощью метода fetch и обработка данных. Конструкция try….catch. Передача состояний вниз и вверх по дереву компонентов.***

Ход работы

1. Скачать папку setup с исходными файлами и установить все зависимости через терминал редактора кода :

npm install

2. Запустите подготовленный проект:

npm run start



3. В шаблоне главного компонента App.js находиться переменная url для указания ссылки на данные расположенные на сервере, которые будут использоваться для рендера карточек с турами. Проверьте ссылку в браузере на работоспособность: https://course-api.com/react-tours-project



Если ссылка не рабочая, используйте локальные данные из файла data.js

4. Создадим две переменные состояния loading и tours в App.js. Первая будет принимать логическое значении в зависимости от того, идет загрузка данных с сервера или нет. А вторая будет содержать полученные данные с сервера. Изначально назначим ей пустой массив.

Также реализуем условие на проверку стейта loading. И в зависимости от того, какое значение он принимает, будем рендерить либо компонент Loading, либо Tours.

import React, { useState, useEffect } from "react";

import Loading from "./Loading";

import Tours from "./Tours";

// ATTENTION!!!!!!!!!!

// Check URL

const url = "https://course-api.com/react-tours-project";

function App() {

  const [loading, setloading] = useState(true);

  const [tours, setTours] = useState([]);

  if (loading) {

    return (

      <main>

        <Loading />

      </main>

    );

  }

  return (

    <main>

      <Tours />

    </main>

  );

}

export default App;

5. Создадим асинхронную функцию для получения данных с сервера и запустим эту функцию только один раз при первом рендере главного компонента App.js с использованием хука useEffect:

import React, { useState, useEffect } from "react";

import Loading from "./Loading";

import Tours from "./Tours";

// ATTENTION!!!!!!!!!!

// Check URL

const url = "https://course-api.com/react-tours-project";

function App() {

  const [loading, setloading] = useState(true);

  const [tours, setTours] = useState([]);

  const fetchTours = async () => {

    setloading(true);

    const response = await fetch(url);

    const tours = await response.json();

    console.log(tours);

  };

  useEffect(() => {

    fetchTours();

  }, []);

  if (loading) {

    return (

      <main>

        <Loading />

      </main>

    );

  }

  return (

    <main>

      <Tours />

    </main>

  );

}

export default App;

6. При работе с асинхронностью могут возникать ошибки, например, при ошибке в адресе сервера или отсутствии интернета при работе приложения. Одним из способов обработки таких ошибок является использования конструкции try….catch:

  const fetchTours = async () => {

    setloading(true);

    try {

      const response = await fetch(url);

      const tours = await response.json();

      console.log(tours);

      setloading(false);

      setTours(tours);

    } catch (error) {

      setloading(false);

      console.log(error);

    }

  };

В качестве свойства компонента Tours передадим стейт tours

  return (

    <main>

      <Tours tours={tours} />

    </main>

  );

7. Далее реализуем компонент Tours

import React from "react";

import Tour from "./Tour";

const Tours = ({ tours }) => {

  return (

    <section>

      <div className="title">

        <h2>Our tours</h2>

        <div className="underline"></div>

      </div>

      <div>

        {tours.map((tour) => {

          return <Tour key={tour.id} {...tour} />;

        })}

      </div>

    </section>

  );

};

export default Tours;



8. Далее настроим карточки туров в компоненте Tour:

import React, { useState } from "react";

const Tour = ({ id, image, info, price, name }) => {

  return (

    <article className="single-tour">

      <img src={image} alt={name} />

      <footer>

        <div className="tour-info">

          <h4>{name}</h4>

          <h4 className="tour-price">{price} $</h4>

        </div>

        <p>{info}</p>

      </footer>

      <button className="delete-btn">not interested</button>

    </article>

  );

};

export default Tour;



9. Реализуем в этом компоненте возможность скрывать длинное описание туров по ссылке.



import React, { useState } from "react";

const Tour = ({ id, image, info, price, name }) => {

  const [readMore, setReadMore] = useState(false);

  return (

    <article className="single-tour">

      <img src={image} alt={name} />

      <footer>

        <div className="tour-info">

          <h4>{name}</h4>

          <h4 className="tour-price">{price} $</h4>

        </div>

        <p>

          {readMore ? info : `${info.substring(0, 200)}...`}

          <button onClick={() => setReadMore(!readMore)}>

            {readMore ? "Show less" : "Show more"}

          </button>

        </p>

      </footer>

      <button className="delete-btn">not interested</button>

    </article>

  );

};

export default Tour;

10. Осталось написать функционал кнопки «Not interested». Поскольку переменная состояния tours находится в App.js, функцию по удалению туров из страницы нужно написать именно в главном компоненте App.js.

  const removeTour = (id) => {

    const restTour = tours.filter((el) => el.id !== id);

    setTours(restTour);

  };

Также не забываем передать эту функцию в виде свойства компоненту Tours

    <Tours removeTour={removeTour} tours={tours} />

В компоненте Tours нужно принять это свойство и передать его по цепочке компоненту Tour:

import React from "react";

import Tour from "./Tour";

const Tours = ({ tours, removeTour }) => {

  return (

    <section>

      <div className="title">

        <h2>Our tours</h2>

        <div className="underline"></div>

      </div>

      <div>

        {tours.map((tour) => {

          return <Tour key={tour.id} {...tour} removeTour={removeTour} />;

        })}

      </div>

    </section>

  );

};

export default Tours;

11. В компоненте Tour реализуем событие нажатия на кнопку «Not interested» и вызова функции removeTour.

import React, { useState } from "react";

const Tour = ({ id, image, info, price, name, removeTour }) => {

  const [readMore, setReadMore] = useState(false);

  return (

    <article className="single-tour">

      <img src={image} alt={name} />

      <footer>

        <div className="tour-info">

          <h4>{name}</h4>

          <h4 className="tour-price">{price} $</h4>

        </div>

        <p>

          {readMore ? info : `${info.substring(0, 200)}...`}

          <button onClick={() => setReadMore(!readMore)}>

            {readMore ? "Show less" : "Show more"}

          </button>

        </p>

      </footer>

      <button onClick={() => removeTour(id)} className="delete-btn">

        not interested

      </button>

    </article>

  );

};

export default Tour;

12. При удалении всех туров останется пустая страница с заголовком. Создадим условие, при котором, в случае пустой страницы, будем выводить кнопку Refresh для возврата всех туров. Реализуем это условие в компоненте App.js.

if (tours.length === 0) {

    return (

      <main>

        <div className="title">

          <h2>no tours left</h2>

          <button className="btn" onClick={fetchTours}>

            Refresh

          </button>

        </div>

      </main>

    );

  }

