React Session 1

Mohamed Emary

July 20, 2024

1 React

React is a JavaScript library for building user interfaces. It was developed by Facebook in 2013. React allows developers to create reusable UI components that can be rendered on the client side.

React is used to build single-page applications (SPAs) and dynamic web applications. It uses a component-based architecture, where each component represents a part of the user interface. Components can be composed together to create complex UIs.

One of the benifits of SPAs is that they provide a better user experience by reducing page reloads and improving performance. React uses a virtual DOM to efficiently update the UI without reloading the entire page.

React in the past was used with class components (sugar syntax) but now it is used with functional components.

1.1 Idea Behind SPAs

The main idea behind single page applications is that they enhance the user experience by reducing page reloads, reduce the number of network requests made to the server, show only the relevant content of the page, and improve the user experience.

One of the problems with SPAs is that they fetch all the data from the server at once. This can lead to slow loading times. To solve this, we use something called lazy loading. **Lazy loading** is a technique that allows developers to load only the relevant content of the page when it is needed.

1.2 React Native

React Native is a framework that allows developers to build mobile apps using React. It uses the same component-based architecture as React.

1.3 Framework vs Library

A library is a collection of functions and classes that can be used to perform specific tasks. A framework is a set of libraries and tools that provide a structure for building applications.

Framework is a complete solution for developing applications. It provides a set of tools and libraries that can be used to build applications. With frameworks, you don't need to find any extra libraries or tools to build your application.

2 How to Create a React App

To create a React app, we need to install Node.js and npm. We can install Node.js from the official website. After installing Node.js, we can create a new React app using the following command:

```
npm create vite@latest
? Project name: my-react-app
? Select a framework: React
? Select a variant: JavaScript
```

When choosing variant there will be some other options, like `TypeScript SWC`, `JavaScript SWC`. SWC stands for *speedy web compiler* which is used to compile the code faster but it has some compatibility issues, so it is better to use JavaScript only.

This will create your React app in the `my-react-app` directory. You can navigate to the directory and install the dependencies using the following commands:

```
cd my-react-app
npm install
```

To start the development server, run the following command:

```
npm run dev
```

It will show you the URL where you can access your React app in the browser.

2.1 Tailwind

To install Tailwind CSS in your React app, you can use the following command:

```
npm i -D tailwindcss postcss autoprefixer
```

Then place the following lines in your `index.css` file:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

To create a Tailwind CSS configuration file, run the following command:

```
npx tailwindcss init -p
```

This will create a `tailwind.config.js` file in your project directory. In the `content` key of the configuration file, you can specify the paths to your React components for example:

```
content: ['./src/**/*.{html,js,jsx,ts,tsx}',
  'index.html'
]
```

Now restart the development server and you should see your React app in the browser.

If you look at your `package.json` file you will see dependencies and devDependencies. Dependencies are the libraries that are required for your app to run, while devDependencies are the libraries that are required for development purposes, for example those used for testing, linting, etc.

`vite` itself is a devDependency because it is only used for development purposes. When you deploy your app, you don't need `vite` to be installed on the server.

The `package-lock.json` file is used to lock the versions of the dependencies in your project. This ensures that the same versions of the dependencies are used when the project is built on different machines. It also contain the versions of the dependencies of the dependencies.

2.2 Why did we use Vite?

`create-react-app` was used to create React apps in the past, but it used webpack as a bundler and it was slow because it uses JS. Vite uses another bundler called `esbuild` and it is much faster because it's built using go programming language.

`esbuild` & `rollup` module

3 Some Notes

- `index.html` file in root of the project is not used unless you want to change very specific things like the title of the page, favicon, add meta tags, etc. For everything else, you will be using React components inside the `src` directory.
- This `index.html` has a `div` with an `id` of `root` where the React app is mounted.
- `public` directory is used to store images of the project, like the logo of the app, etc. (we will discuss this later).
- In `sources` folder we have `index.css`, `App.css` which are used to write global styles for the app
- We also have a `App.jsx`, `main.jsx`
  - `main.jsx` is the main JS file and the entry point of the app. It's the only file that is included in the `index.html` file. It links you main component (`App.jsx`) to the `index.html` file.
  - `App.jsx` is the main/parent component of the app.
- Function components have two main characteristics:
  - The name of the component should start with a capital letter.
  - The component should return JSX.
- Sometimes you see errors under the code in VSCode, if your code is correct, this may be because of the linter.
- When creating new components place your components in a `components` directory inside the `src` directory, and each component should have its own directory because that component may also have CSS file or JS file.

- In JS, you should place the returned value next to the `return` keyword in the same line. If you didn't, the value will be ignored. So just use `()` to return the value.

Important Notes

When using modules it's better to include only one JS file in the `index.html` file, and inside that file you can import other code when needed. This is better as it reduces the number of network requests made to the server.

React has its own DOM. The actual root element is the `index.html` file, but React creates its own DOM via `ReactDOM`, which has a function `ReactDOM.createRoot()`, that creates a root to display the React app components.

The value returned from any function component **is not HTML**, it's **JSX** (JavaScript XML). JSX is a syntax that allows you to write HTML in JavaScript.

For example you can't use `class` inside JSX because it's a keyword in JavaScript, so instead you have to use `className`.

In `index.html` file we have:

```
<div id="root"></div>
```

In `main.jsx` file that is linked to `index.html` we have:

```
import ReactDOM from 'react-dom/client'; // import default so no curly
↪  brackets and you can use any name
// Import you components here ex:
import { App } from './App';


const root = document.getElementById('root');
const reactRoot = ReactDOM.createRoot(root);

reactRoot.render( // Takes only one param
  // Call your components here
  <App />
);
```

The `render` function takes the component name in empty tag after importing it at the beginning of the file. example: `<About />`

The `render` function takes only one param, so you can use `render(<App />, <About />)`, this will not work.

Example component:

```
export function About() {
  return <h1>Hello From About</h1>;
}
```

The return in the component should only return one element, so if you have multiple elements put them inside a `<> ELEMENTS </>` or use `<React.Fragment> ELEMENTS </React.Fragment>`.

The XML code can work inside `.js` files, so why to use `.jsx` files? The reason is that `.jsx` files

are expected to have XML inside it so you will get completions with the XML tags which will help you while the development process of your application.

When using a `for` property in `<label />` element for an input tag we don't use `for` instead we use `htmlFor`, similarly we use `className` instead of `class`. `for` and `class` are reserved keywords in JavaScript.

The bundler `esbuild` will convert your `.jsx` files to `.js` files.

Can i see the converted code of the bundler?