React-1

Nesli Erdoğmuş

What is React?

- React is a JavaScript library for building user interfaces.
- It runs on the client as a Single Page Application (SPA).
  - A user visits a URL and requests one minimal HTML file and one associated JavaScript file. After some network delay, the user sees the “by JavaScript rendered HTML” in the browser and starts to interact with it. Every additional page transition wouldn’t request more files from the web server, but would use the initially requested JavaScript to render the new page. Also every additional interaction by the user is handled on the client too.
- It is also referred as a framework because it is directly comparable to front-end frameworks such as Angular or Vue.
- It can be used to build full stack apps by interacting with a server/API
  - E.g. MERN stack (like MEAN but with React)

Popular web development stacks in 2022

https://survey.stackoverflow.co/2022/#technology

- React, Vue or Angular on the frontend.
- Node/JavaScript as well as PHP’s Symfony and Laravel at the backend.
- PostgreSQL, MySQL or Mongo as databases.

For mobile development:

- React Native, Android and iOS native technologies as well as Flutter.

Nothing is enough!

Redux

TS

Prettier

JSS

CSS MODULES

]v[Mobx

ESLint

REACT ROUTER

styled components

webpack

AXIOS vs FETCH

BABEL

React Query

Storybook

Why React?

- Structure the “view” layer of your application
- Reusable components with their own states
  - One way data binding
- JSX - Dynamic markup
- Interactive UIs with virtual DOM
- Very popular in industry

Setting Up a React Project

- create-react-app
  - command-line utility
  - zero-configuration starter kit for React introduced by Facebook
  - under the hood, it uses Babel and webpack
  - creates a frontend build pipeline
- After installing Node and npm (and npx that comes bundled with it):
  - Node >= 14

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

https://create-react-app.dev/docs/getting-started

Node.js and npm

- **Node.js** is a runtime environment that allows you to run JavaScript on the backend.
    - JavaScript code runs directly in a computer process itself instead of in a browser.
    - It acts as a JavaScript engine that translates your code, allowing it to be run on a physical machine.
- **npm** stands for Node Package Manager. It’s a library and registry for JavaScript software packages.
    - an online repository for the publishing of open-source Node.js projects
    - a command-line utility for interacting with said repository helping with installing packages and managing package versions and dependencies

Node.js and npm

- You define all your project’s dependencies inside your package.json file.
- Anytime you or a team member needs to get started with your project, all they have to do is run **`npm install`**.
    - This will immediately install all the necessary dependencies for your project.
    - In the package.json file, you can also specify which versions your project depends upon.
- A **package.json** file is created by your package manager (in this case npm) and exists at the root of a project.
    - To generate a package.json file you can run npm init.
    - The package.json file is in JSON format and is used for managing the project’s dependencies, scripts, versions, etc.

create-react-app dependencies

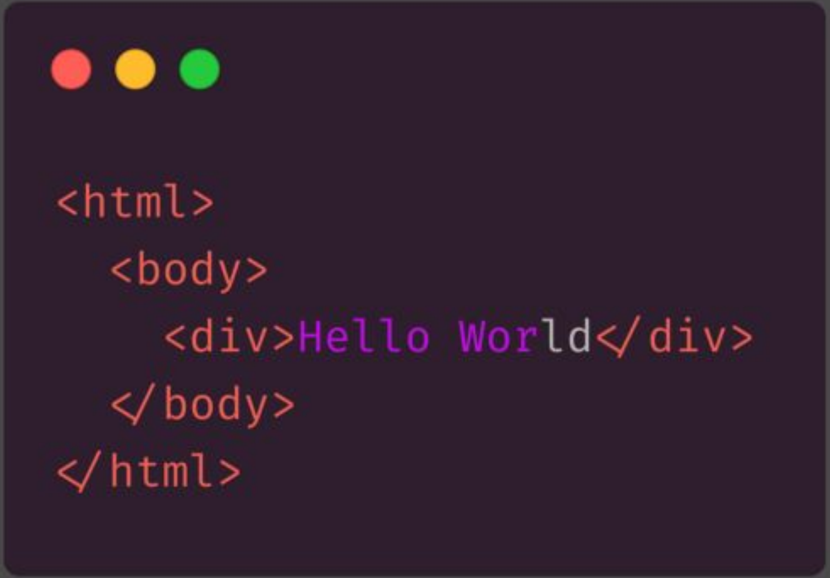
- To run your React application, you need to turn your JSX into plain JavaScript, that browser can understand.
- We need transpilers and bundlers to create, run and understand React builds on browser.
    - **Babel**: The main use is to make your code readable by older browsers.
    - **ESLint**: As a JavaScript linter, it will scan your code and flag any code errors.
    - **Jest**: This library by Facebook lets you write test scripts for your application without having to install another testing library.
    - **PostCSS** / **CSS Loader** & **Style Loader**: It handles our CSS.
    - **Webpack**: As a module bundler for JavaScript, it puts everything needed by your application together.

react-scripts

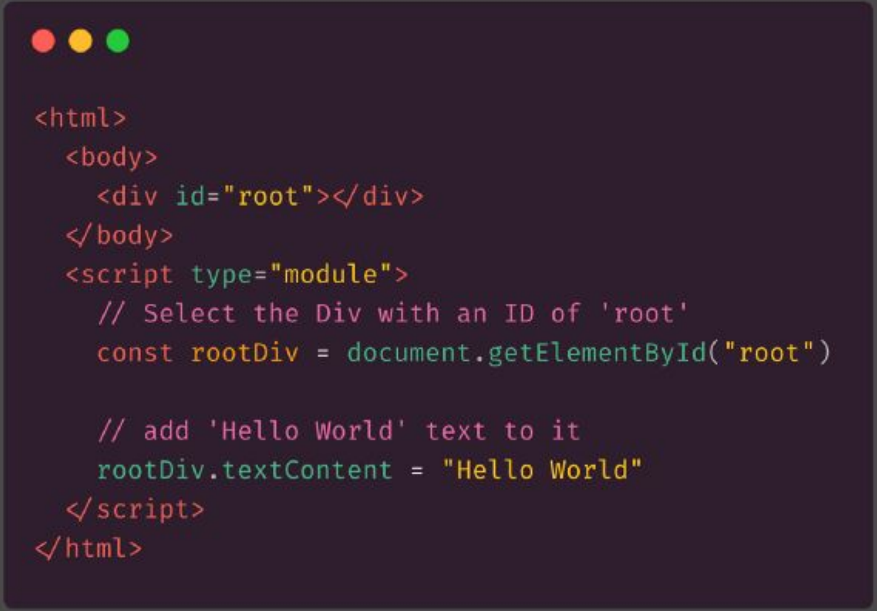
- react-scripts are simply scripts to run the build tools required to transform React JSX syntax into plain JavaScript programmatically.
    - With the **start** argument, npm will begin the process to make a development server available for your React application.
    - The **build** command will start the process of creating a production-ready React app for you.
    - The **test** command will run any test scripts that you've written using Jest.
    - The **eject** command is used to remove the dependency on react-scripts and expose the build tools and configurations for you to modify.

```
"scripts: {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
   },
```

From HTML to HTML + JS



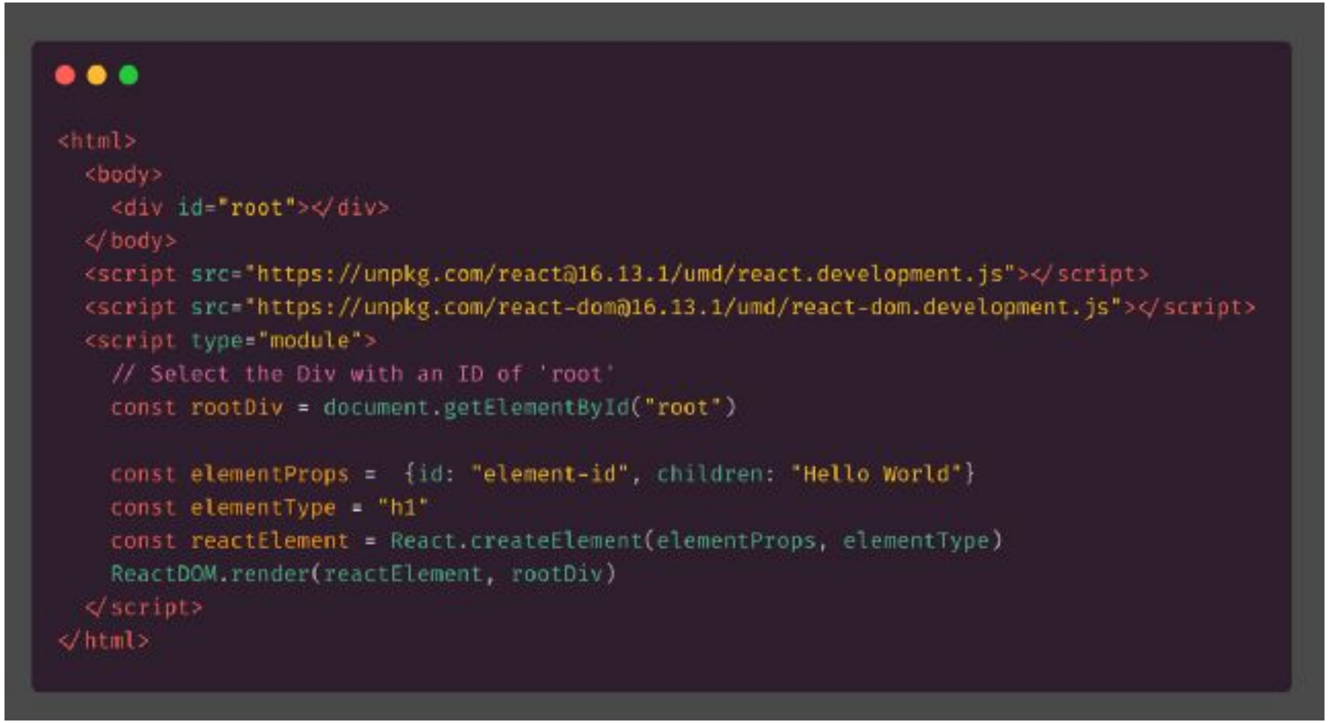
```
<html>
  <body>
    <div>Hello World</div>
  </body>
</html>
```



```
<html>
  <body>
    <div id="root"></div>
  </body>
  <script type="module">
    // Select the Div with an ID of 'root'
    const rootDiv = document.getElementById("root")

    // add 'Hello World' text to it
    rootDiv.textContent = "Hello World"
  </script>
</html>
```

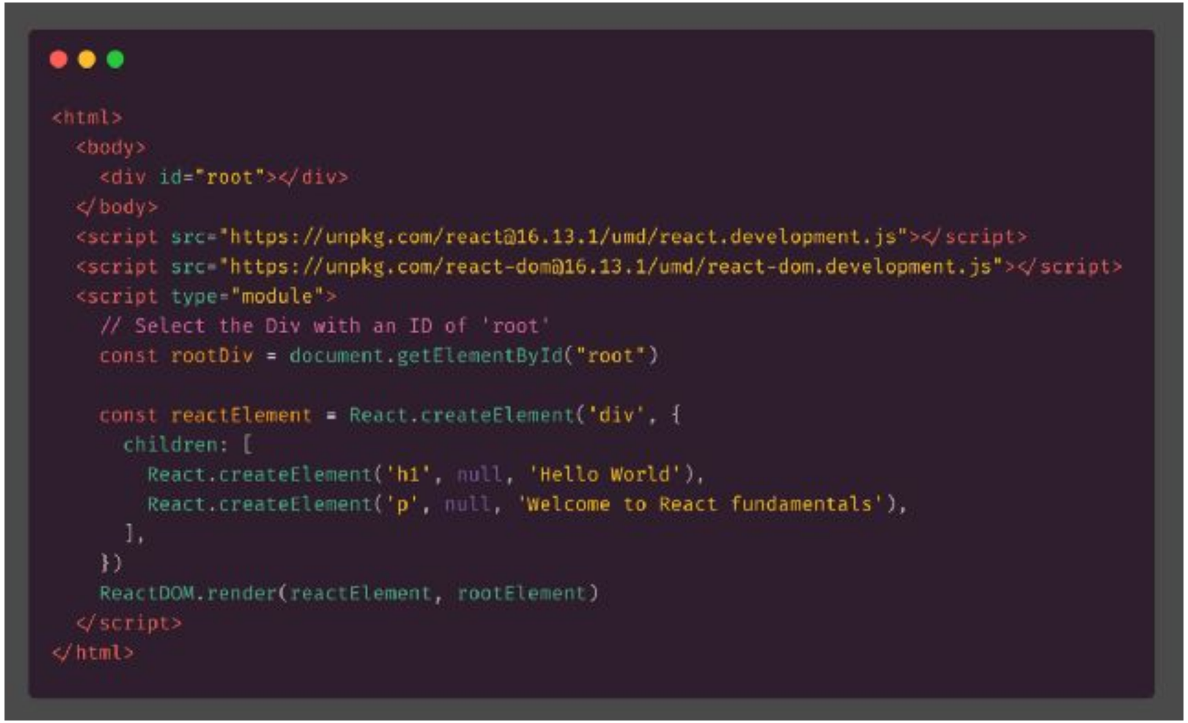
To React JS



```
<html>
  <body>
    <div id="root"></div>
  </body>
  <script src="https://unpkg.com/react@16.13.1/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.13.1/umd/react-dom.development.js"></script>
  <script type="module">
    // Select the Div with an ID of 'root'
    const rootDiv = document.getElementById("root")

    const elementProps =  {id: "element-id", children: "Hello World"}
    const elementType = "h1"
    const reactElement = React.createElement(elementProps, elementType)
    ReactDOM.render(reactElement, rootDiv)
  </script>
</html>
```

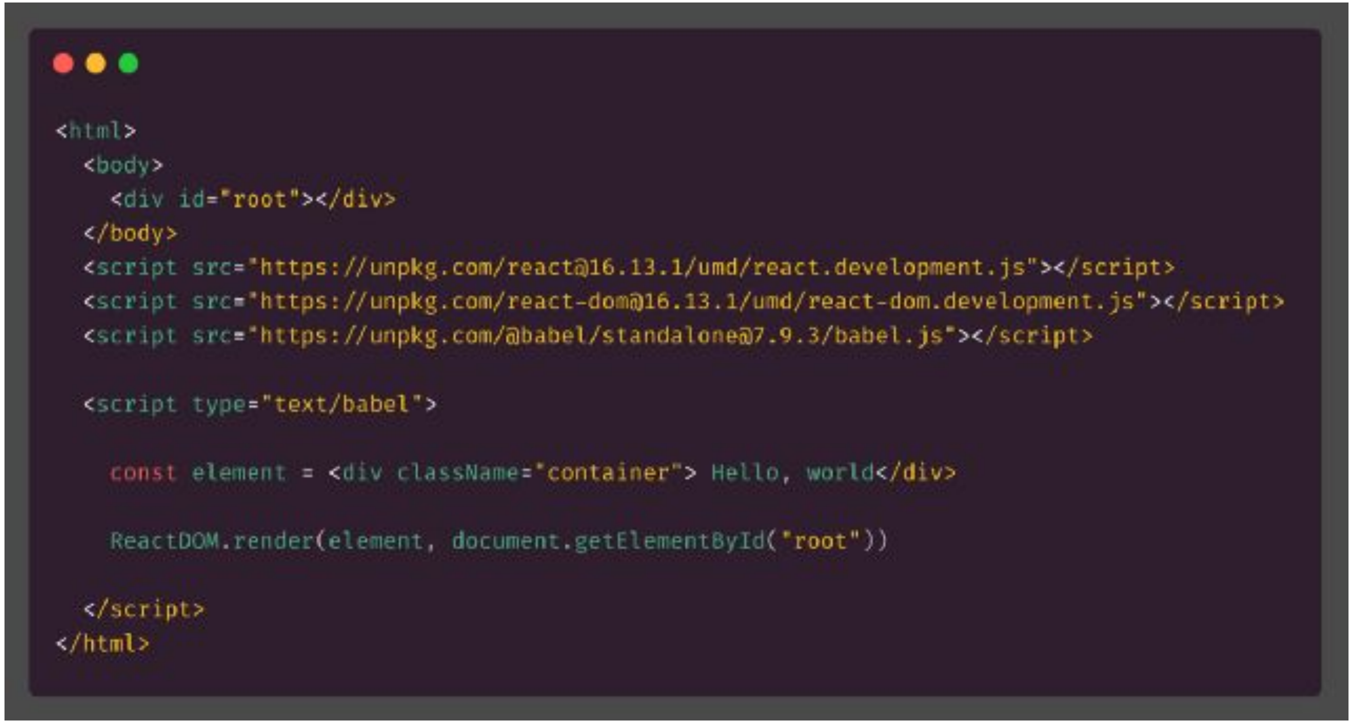
To React JS - Nested Elements



```html
<html>
  <body>
    <div id="root"></div>
  </body>
  <script src="https://unpkg.com/react@16.13.1/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.13.1/umd/react-dom.development.js"></script>
  <script type="module">
    // Select the Div with an ID of 'root'
    const rootDiv = document.getElementById("root")

    const reactElement = React.createElement('div', {
      children: [
        React.createElement('h1', null, 'Hello World'),
        React.createElement('p', null, 'Welcome to React fundamentals'),
      ],
    })
    ReactDOM.render(reactElement, rootElement)
  </script>
</html>
```

To React + JSX



```
<html>
  <body>
    <div id="root"></div>
  </body>
  <script src="https://unpkg.com/react@16.13.1/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.13.1/umd/react-dom.development.js"></script>
  <script src="https://unpkg.com/@babel/standalone@7.9.3/babel.js"></script>

  <script type="text/babel">

    const element = <div className="container"> Hello, world</div>

    ReactDOM.render(element, document.getElementById("root"))

  </script>
</html>
```

To React + JSX

JSX simply provides syntactic sugar for us to write components in a way that’s similar to HTML.

```
<MyButton color="blue" shadowSize={2}>
  Click Me
</MyButton>
```

compiles to

```
React.createElement(
  MyButton,
  {color: 'blue', shadowSize: 2},
  'Click Me'
)
```

That’s why we “`import React from 'react'`” even though we don't explicitly refer it in our code.

JSX

```
const element = <h1>Hello, world!</h1>;

const element = <img src={user.avatarUrl} />;
```

- This is neither a string not HTML. It is called JSX, and it is a syntax extension to JavaScript.
- JSX produces React “elements”.
- Instead of artificially separating *technologies* by putting markup and logic in separate files, React separates *concerns* with loosely coupled units called “components” that contain both.
- React doesn’t require using JSX.