Hello World

Using React

In the previous session, we found out React to be interesting, hence here we will understand two major script tags of React:

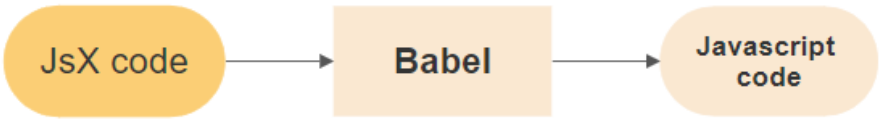
1. **React** - **React-script-tag** is an npm package that provides a **React** <**script**> **tag** which supports universal rendering. With this library, we can create react components, that is, a plain javascript object with some properties.
2. **React-DOM**- React-DOM basically converts the javascript object returned by react script tag to HTML nodes that can be rendered in the browser.

```
<html>
    <head>
        <title>
            Adding react in 1 minute!
        </title>

    </head>
    <body>
        <!-- we will put react components here -->
        <div id="app"></div>
        <!-- Load React -->
        <script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script>
        <script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
        <!-- Load our react component -->
        <script src="Likebutton.js"></script>
    </body>
</html>
```

Babel:

Babel came into the picture when we found out that the JSX file is not understandable by the browser. It is a tool that converts JSX files to simple javascript code that the browser understands. Moreover, it also converts ES6 and ES5 code to javascript code.



Instructions for using Babel:

1. Visit website https://babeljs.io/repl
2. A screen will be visible where on one side JSX code will be written and which would be converted to javascript code.

```
<div>
  Hello World!
</div>
```

```
"use strict";

/*#__PURE__*/
React.createElement("div", null, "Hello World!");
```

WebPack:

Webpack is an open-source JavaScript module bundler that is made primarily for JavaScript, but it can transform front-end assets such as HTML, CSS, and images. Webpack takes small modules like header, footer etc. and creates a single module with all the assets.

Visit Website: https://webpack.js.org/

Create-React-App

Create-React-App is a tool given by Facebook that provides us with boilerplate code and helps us to create our own react app.

Instructions to create react app:

1. Install create-react-app using **npm install -g create-react-app**
2. Go to the desktop using **cd Desktop**
3. Use command **create-react-app app_name(hello world)**
4. Use command **cd hello_world** and go to the app.
5. Use **ls** to display the list of files in the current directory.
6. And now open the file in VS Code.
7. Use **npm start** to start your first react project.

Installing React Developer Tools

It is a chrome extension that helps us in the debugging and development process.
Instructions for installation:

1. Open chrome tab and search for react dev tools.
2. Open the download link and click add to chrome.

Under inspect tools, this extension will create a new tab named components that will show us all the react components used in our app.

More on JSX

JSX, or **JavaScript XML**, is an extension to the JavaScript language syntax. Similar in appearance to HTML, JSX provides a way to structure component rendering using syntax familiar to many developers. React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript). JSX is similar to another extension syntax created by Facebook for PHP called XHP.

- **Why is a class not used as an attribute in JsX?**
  We cannot use **class** attributes in script tags. Instead of this, we use **className** because the class is a reserved keyword in javascript.

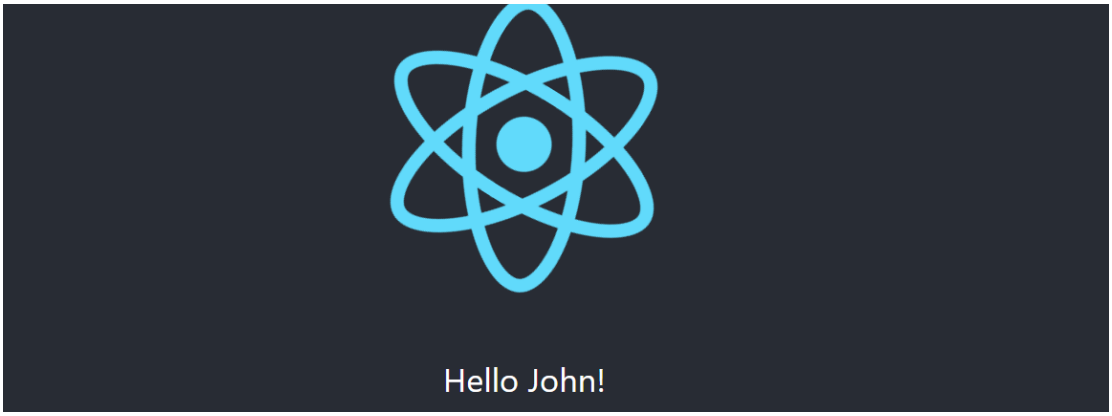
- **Using javascript variables in JsX:**
  We can use variable names instead of static text by creating variables. We can add them in a JsX file using **{variable_name}**. Here is the code snippet for your reference:

```
import logo from './logo.svg';
import './App.css';

function App() {
  const name ="John";
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Hello {name}!
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

  Note: Here, a constant variable is created named “name” and in the p tag, it is used in curly braces to display the name saved in the “name” variable.



Output

- **Fragments:**

  A common pattern in React is for a component to return multiple elements. Fragments let you group a list of children without adding extra nodes to the DOM.

  Here is a code snippet for your reference:

```
<React.Fragment>
  <h1> Hey how are you?</h1>
  <div className="App">
  <header className="App-header">
    <img src={logo} className="App-logo" alt="logo" />
    <p>
      Hello {name}!
    </p>
    <a
      className="App-link"
      href="https://reactjs.org"
      target="_blank"
      rel="noopener noreferrer"
    >
      Learn React
    </a>
  </header>
</div>
</React.Fragment>
```

- **Components Naming:**
  In react, components should be named with words starting with capital letters. If we will start words with small letters the console will show an error and the browser will render nothing.
  Here is the code snippet for your reference:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import app from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <React.StrictMode>
    <app />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Wrong Choice

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Right Choice

Conditional Rendering

What is conditional rendering?

Conditional rendering is a term to describe the ability to **render** different components based on different conditions. In some scenarios data comes via API calls that is the point where conditional rendering comes in, and we can show different data to different users.

- We use {} for writing conditions inside JsX code.
- We can use if-else conditions or && operators to apply conditions. Here is the code snippet for your reference;

```
function App() {
  const name ="John";
  const isLoggedIn=false;
  return (
    <React.Fragment>
      <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        {!isLoggedIn && <p>Hello World!</p>}
        {isLoggedIn && <p>Hello {name}!</p>}
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
    </React.Fragment>
```

Note: Here in the above code, we used a variable **isLoggedIn** which applies the condition that if it is true, then Hello {name} is displayed. Otherwise,

Hello world is displayed. Here we used **&&** notation to apply conditions. In the above code **isLoggedIn** is false so Hello World! will be displayed.

Summarizing It

Let's summarize what we have learnt in this module:

- We Learned about react script tags, babel and webpack.
- Installed **create-react-app** and react developer tools.
- Worked on more JsX functionality.
- Learned about conditional rendering and its implementation.

Some references:

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

Documentation on JSX:
https://reactjs.org/docs/jsx-in-depth.html#:~:text=Specifying%20The%20React%20Element%20Type,Foo%20must%20be%20in%20scope.

Learning fragments in detail:
https://reactjs.org/docs/fragments.html