Practical 7

Name: Sameer Gupta

Roll no: 252010036

Understanding the React framework.

1. React introduction

2. Getting started

3. ES6

4. React render HTML

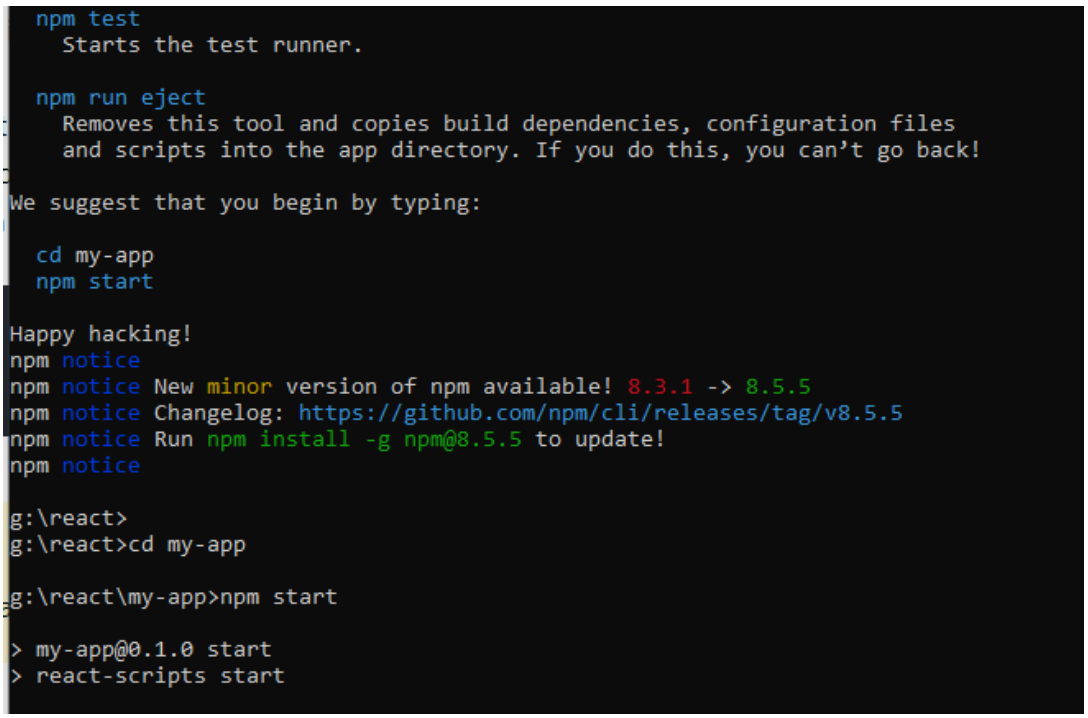
5. React JSX

React Introduction

React is a JavaScript library for building user interfaces. React is used to build single-page applications. React allows us to create reusable UI components.

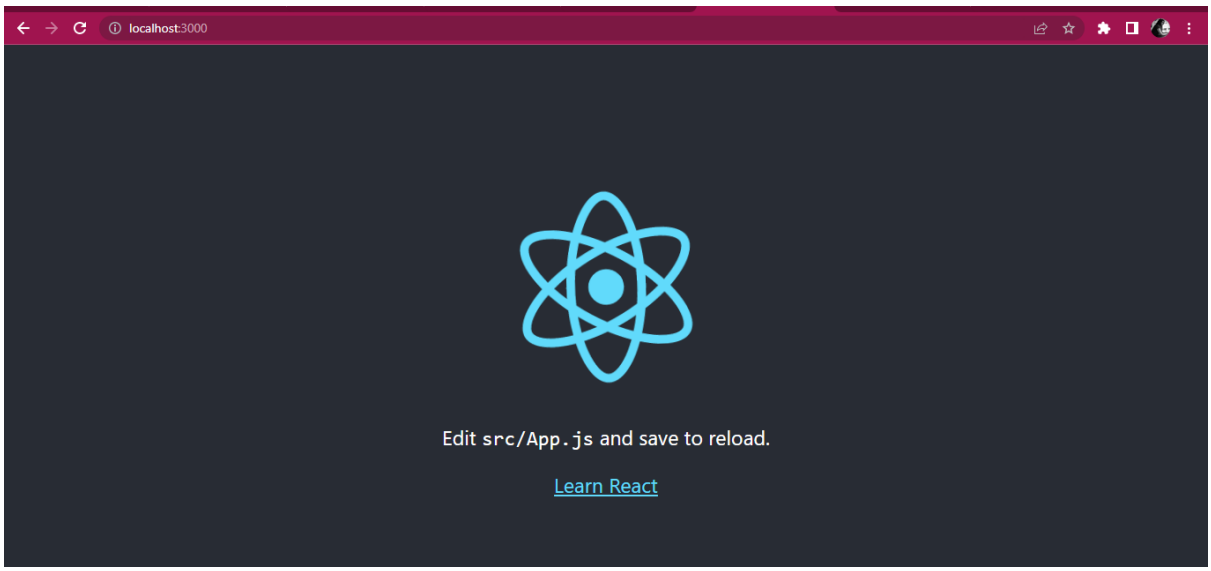
Show React" tool makes it easy to demonstrate React

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



```
Note that the development build is not optimized.
To create a production build, use npm run build.

assets by path static/ 1.49 MiB
  asset static/js/bundle.js 1.48 MiB [emitted] (name: main) 1 related asset
  asset static/js/node_modules_web-vitals_dist_web-vitals_js.chunk.js 6.92 KiB [emitted] 1 related asset
  asset static/media/logo.6ce24c58023cc2f8fd88fe9d219db6c6.svg 2.57 KiB [emitted] (auxiliary name: main)
asset index.html 1.67 KiB [emitted]
asset asset-manifest.json 546 bytes [emitted]
runtime modules 31.3 KiB 15 modules
modules by path ./node_modules/ 1.35 MiB 99 modules
modules by path ./src/ 18 KiB
  modules by path ./src/*.css 8.82 KiB
    ./src/index.css 2.72 KiB [built] [code generated]
    ./node_modules/css-loader/dist/cjs.js??ruleSet[1].rules[1].oneOf[5].use[1]!./node_modules/postcss-loader/dist/cjs.js
??ruleSet[1].rules[1].oneOf[5].use[2]!./node_modules/source-map-loader/dist/cjs.js!./src/index.css 1.37 KiB [built] [cod
e generated]
    ./src/App.css 2.72 KiB [built] [code generated]
    ./node_modules/css-loader/dist/cjs.js??ruleSet[1].rules[1].oneOf[5].use[1]!./node_modules/postcss-loader/dist/cjs.js
??ruleSet[1].rules[1].oneOf[5].use[2]!./node_modules/source-map-loader/dist/cjs.js!./src/App.css 2 KiB [built] [code gen
erated]
  modules by path ./src/*.js 5.58 KiB
    ./src/index.js 1.75 KiB [built] [code generated]
    ./src/App.js 2.46 KiB [built] [code generated]
    ./src/reportWebVitals.js 1.37 KiB [built] [code generated]
  ./src/logo.svg 3.61 KiB [built] [code generated]
webpack 5.70.0 compiled successfully in 46245 ms
```



To use React in production, you need npm which is included with Node.js.
To get an overview of what React is, you can write React code directly in HTML.
But in order to use React in production, you need npm and Node.js installed.

React Getting Started

React Directly in HTML

The quickest way start learning React is to write React directly in your HTML files.

Start by including three scripts, the first two let us write React code in our JavaScripts, and the third, Babel, allows us to write JSX syntax and ES6 in older browsers.

```
<!DOCTYPE html>
<html>
 <head>
  <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>
  <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></script>
  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
 </head>
 <body>

  <div id="mydiv"></div>

  <script type="text/babel">
   function Hello() {
    return <h1>Hello World!....Welcome to REACT tutorial</h1>;
   }

   ReactDOM.render(<Hello />, document.getElementById('mydiv'))
  </script>

 </body>
</html>
```

Setting up a React Environment

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

Modify the React Application

/myReactApp/src/App.js:

Try changing the HTML content and save the file.

Example

Replace all the content inside the with a <h1> element.

```
import React from 'react';
import ReactDOM from 'react-dom';

const myfirstelement = <h1>Hello React!</h1>

ReactDOM.render(myfirstelement, document.getElementById('root'));
```

What is ES6?

ES6 stands for ECMAScript 6.

ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015.

It brought about a significant change in the way complex apllications coulde be written, with the introduction of class declarations and ES6 modules.
Arrow functions (a concise way of writing functions), and the let and const keywords were also introduced.
ES6 also changed the way React components and props could be declared.
Without ES6, a React component would have to be created using the create-react-class module, like so:

```
var createReactClass = require("create-react-class");
var Greeting = createReactClass({
render: function () {
return <h1>Hello, {this.props.name}</h1>;
},
});
```

• The same component can be declared by using class declarations as:

```
class Greeting extends React.Component {
render() {
return <h1>Hello, {this.props.name}</h1>;
}
}
```

• Similarly props need to be initialised by declaring a getDefaultProps as a function to the createReactClass
object:

```
var Greeting = createReactClass({
getDefaultProps: function () {
return {
name: "Mary",
};
},
// ...
});
```

• When using class-based components, defaultProps is defined as a property on the component itself.

```
class Greeting extends React.Component {
// ...
}
Greeting.defaultProps = {
name: "Mary",
};
```

React Render HTML

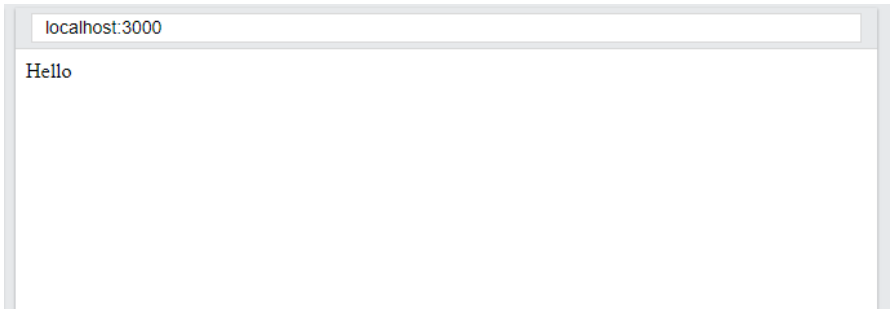
React renders HTML to the web page by using a function called ReactDOM.render().

The Render Function

The ReactDOM.render() function takes two arguments, HTML code and an HTML element. The purpose of the function is to display the specified HTML code inside the specified HTML element.
Display a paragraph inside an element with the id of "root":

```
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(pHellop, document.getElementById('root'));
```



The Root Node

The root node is the HTML element where you want to display the result.

It is like a *container* for content managed by React.

The root node can be called whatever you like:

```
<body>
  <header id="sandy"></header>
</body>
```

Display the result in the <header id="sandy"> element:

ReactDOM.render(Hallo , document.getElementById('sandy'));

React JSX

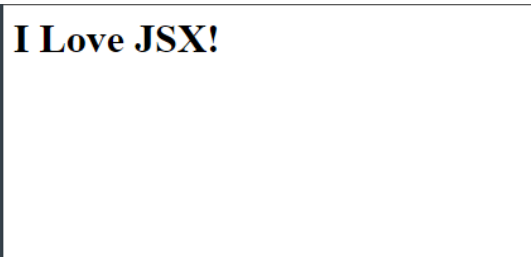
JSX stands for JavaScript XML.
JSX allows us to write HTML in React.
JSX makes it easier to write and add HTML in React.
Coding JSX
JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
JSX converts HTML tags into react elements.

Example for JSX file

```
import React from 'react';
import ReactDOM from 'react-dom';

const myelement = <h1>I Love JSX!</h1>;

ReactDOM.render(myelement, document.getElementById('root'));
```



JSX is an extension of the JavaScript language based on ES6, and is translated into regular JavaScript at runtime

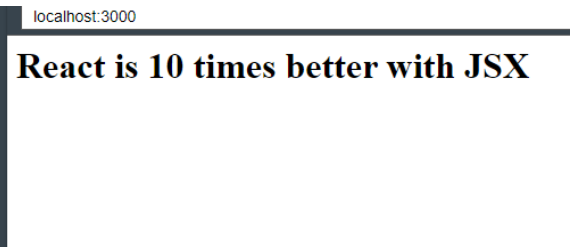
Expressions in JSX

With JSX you can write expressions inside curly braces { }.

```
import React from 'react';
import ReactDOM from 'react-dom';

const myelement = <h1>React is {5 + 5} times better with JSX</h1>;

ReactDOM.render(myelement, document.getElementById('root'));
```




```
import React from 'react';
import ReactDOM from 'react-dom';

const myelement = (
  <ul>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Cherries</li>
  </ul>
);

ReactDOM.render(myelement, document.getElementById('root'));
```

- Apples
- Bananas
- Cherries

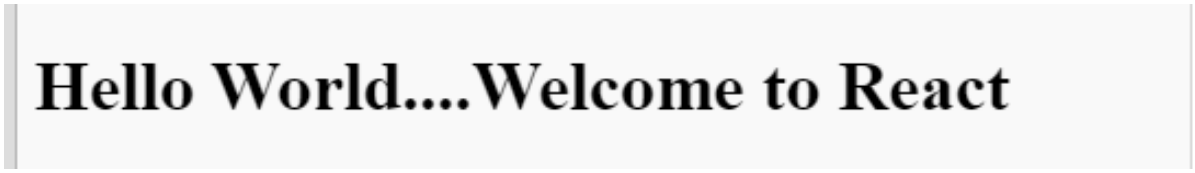
Attribute class = className

The class attribute is a much used attribute in HTML, but since JSX is rendered as JavaScript, and the class keyword is a reserved word in JavaScript, you are not allowed to use it in JSX.

Use attribute className instead.

JSX solved this by using className instead. When JSX is rendered, it translates className attributes into class attributes.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './mystyle.css';
const myelement = <h1 className="myclass">Hello World....Welcome to React</h1>;
ReactDOM.render(myelement, document.getElementById('root'));
```



Conditions - if statements

React supports if statements, but not *inside* JSX.

To be able to use conditional statements in JSX, you should put the if statements outside of the JSX, or you could use a ternary expression instead:

```
import React from 'react';
import ReactDOM from 'react-dom';

const x = 8;
let text = "Goodbye";
if (x < 10) {
  text = "Hello";
}

const myelement = <h1>{text}</h1>;

ReactDOM.render(myelement, document.getElementById('root'));
```