HAZZA INSTITUTE OF TECHNOLOGY

iqracity

HAZZA INSTITUTE OF TECHNOLOGY

MERN Stack (ES6 + REACT) Course

iqracity

Lab 13

Total Time:
3 hours

Pre-Lab Activities:

- No Pre-Lab Activity

Learning Outcomes:

- Perform the concepts of what React is and how it works.

Lab Tasks:

- React Introduction
- React Directly with HTML
- Setting up a React Environment
- Run a React App
- Modify a React App
- React Render with HTML

Student Activities:

- Explore React
- Explore React Directly with HTML
- Explore Setting up a React Environment
- Explore how to run a React App
- Explore how to modify a React App
- Explore how React Render works with HTML

Lab Solution

React Introduction:

What is React?

React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.

React is a tool for building UI components.

How does React work?

React creates a VIRTUAL DOM in memory.

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

React only changes what needs to be changed!

React finds out what changes have been made, and changes only what needs to be changed.

You will learn the various aspects of how React does this in the rest of this tutorial.

React Getting Started:

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 Directly in HTML:

The quickest way to 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.

You will learn more about JSX in the React JSX chapter.

Example:

Include three CDN's in your HTML file:

```
<!DOCTYPE html>

<html>

<head>

<script
src="https://unpkg.com/react@18/umd/react.development.js"crossorig
in></script>

<script src="https://unpkg.com/react-dom@18/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!</h1>;

     }

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

</script>

</body>

</html>
```

This way of using React can be OK for testing purposes, but for production you will need to set up a **React environment**.

Setting up a React environment:

If you have npx and Node.js installed, you can create a React application by using `create-react-app`.

If you've previously installed `create-react-app` globally, it is recommended that you uninstall the package to ensure npx always uses the latest version of `create-react-app`.

To uninstall, run this command: `npm uninstall -g create-react-app`.

Run this command to create a React application named `my-react-app`:

```
npx create-react-app my-react-app
```

The `create-react-app` will set up everything you need to run a React application.

Run the React Application:

Now you are ready to run your first *real* React application!

Run this command to move to the`my-react-app` directory:

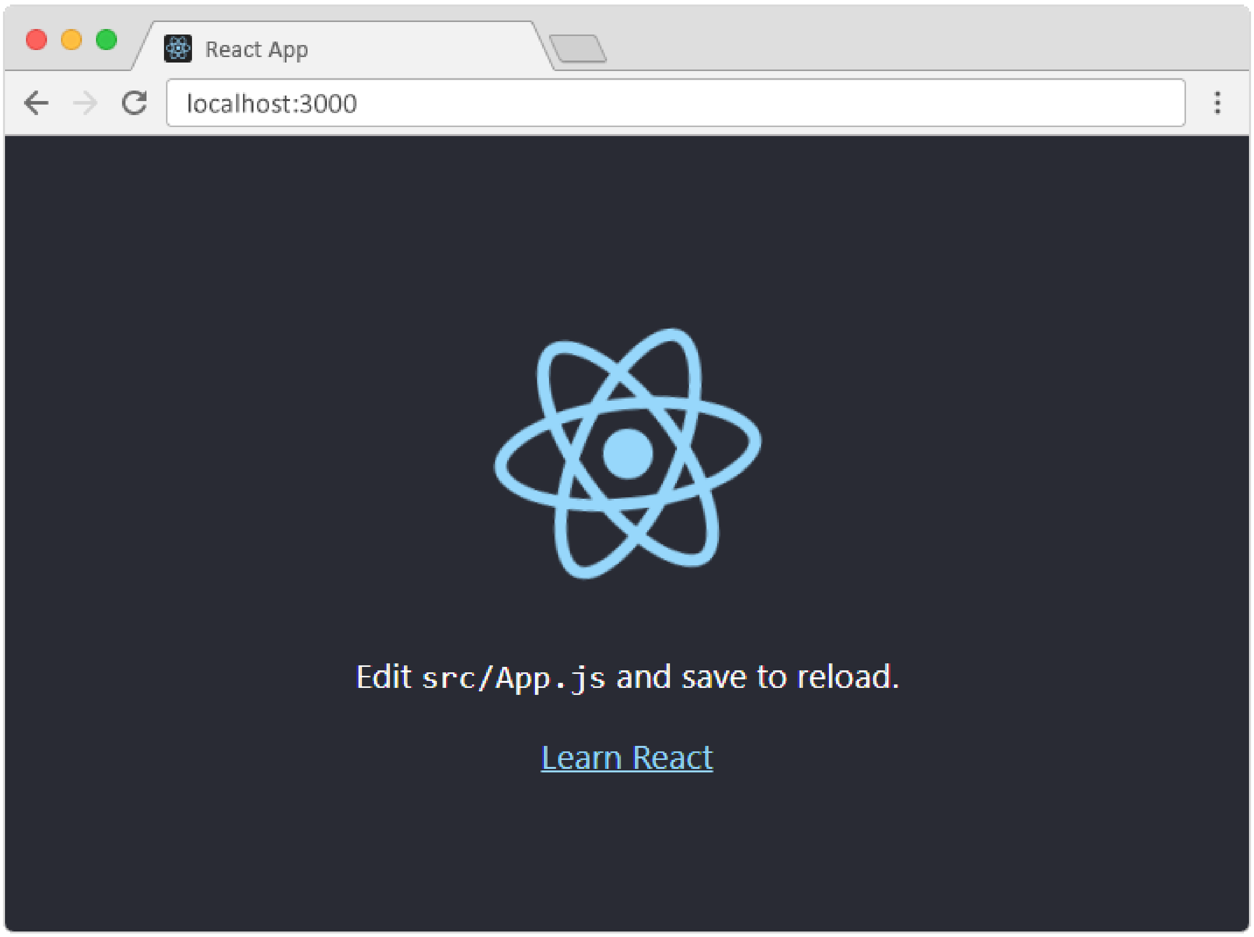
```
cd my-react-app
```

Run this command to run the React application `my-react-app`:

```
npm start
```

A new browser window will pop up with your newly created React App! If not, open your browser and type `localhost:3000` in the address bar.

The result:



Edit `src/App.js` and save to reload.

Learn React

Modify the React Application:

So far so good, but how do I change the content?

Look in the `my-react-app` directory, and you will find a `src`folder. Inside the `src`folder there is a file called `App.js`, open it and it will look like this:

/myReactApp/src/App.js:

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

functionApp(){
return(
<div className="App">
<header className="App-header">
<img src={logo}className="App-logo"alt="logo"/>
<p>
          Edit <code>src/App.js</code> and save to reload.
</p>
<a
className="App-link"
href="https://reactjs.org"
target="_blank"
rel="noopener noreferrer"
>
          Learn React
</a>
</header>
</div>
);
}


exportdefault App;
```

Try changing the HTML content and save the file.

Notice that the changes are visible immediately after you save the file, you do not have to reload the browser!

Example:

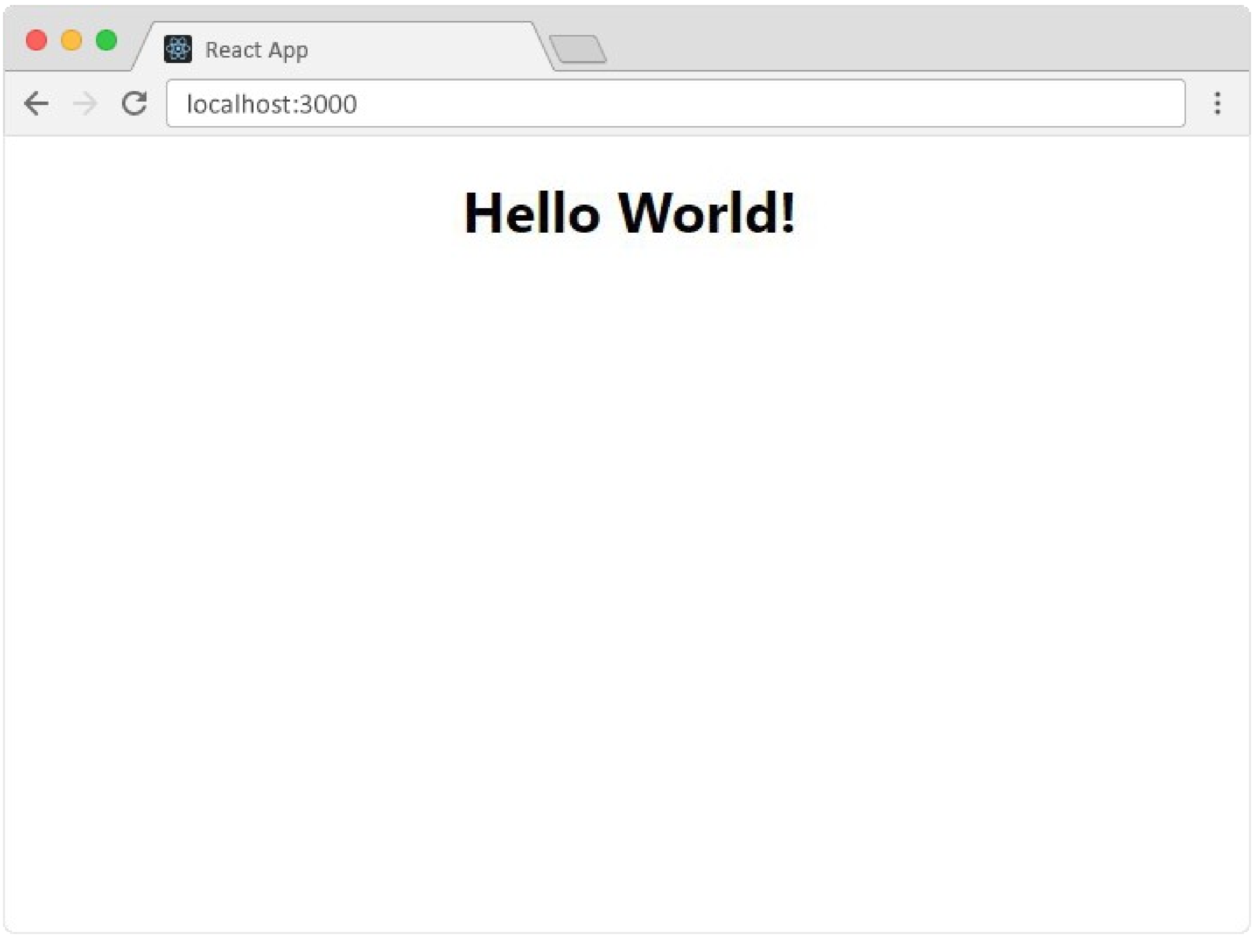
Replace all the content inside the `<div className="App">` with a `<h1>` element.

See the changes in the browser when you click Save.

```
functionApp(){

return(

<div className="App">

<h1>Hello World!</h1>

</div>

);

}

exportdefault App;
```

Notice that we have removed the imports we do not need (logo.svg and App.css).

The result:



Hello World!

React Render HTML:

React's goal is in many ways to render HTML in a web page.

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.

But render where?

There is another folder in the root directory of your React project, named "public". In this folder, there is an `index.html` file.

You'll notice a single `<div>` in the body of this file. This is where our React application will be rendered.

Example:

Display a paragraph inside an element with the id of "root":

```
ReactDOM.render(<p>Hello</p>, document.getElementById('root'));
```

The result is displayed in the `<div id="root">` element:

```
<body>

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

</body>
```

Note that the element id does not have to be called "root", but this is the standard convention.

The HTML Code:

The HTML code in this tutorial uses JSX which allows you to write HTML tags inside the JavaScript code:

Do not worry if the syntax is unfamiliar, you will learn more about JSX in the next chapter.

Example:

Create a variable that contains HTML code and display it in the "root" node:

```
const myelement = (

<table>

<tr>

<th>Name</th>

</tr>

<tr>

<td>John</td>

</tr>

<tr>

<td>Elsa</td>

</tr>

</table>

);

ReactDOM.render(myelement, 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.

It does NOT have to be a `<div>` element and it does NOT have to have the `id='root'`:

Example:

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(<p>Hallo</p>, document.getElementById('sandy'));
```