React JS

Introduction

Laiba Imran

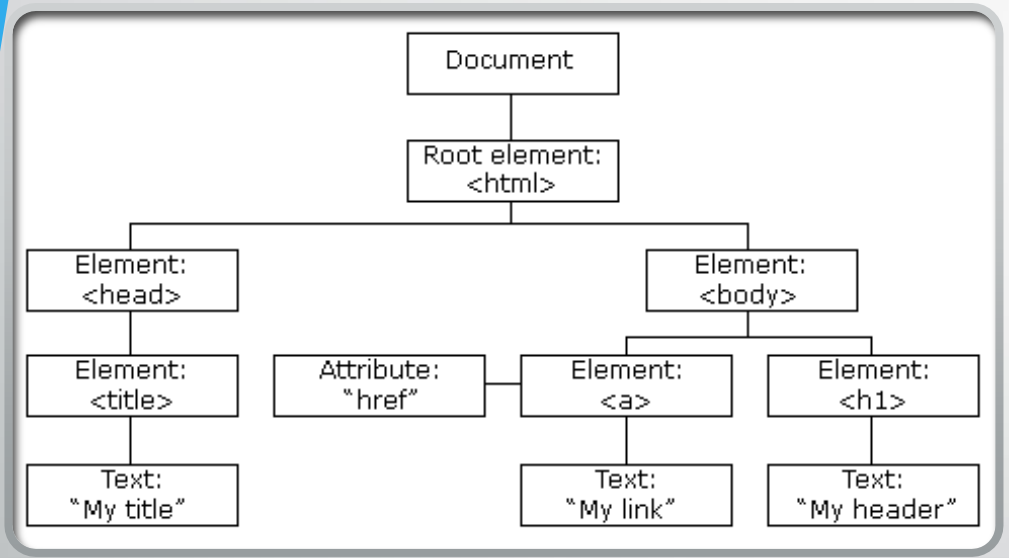
Before React

- JavaScript
- jQuery

What is React?

- React is a JavaScript Library created by Facebook.
- React is a JavaScript library for building user interfaces.
- React is a tool for Building UI components.
- React is used to build single-page applications.
- React allows us to create reusable UI components.

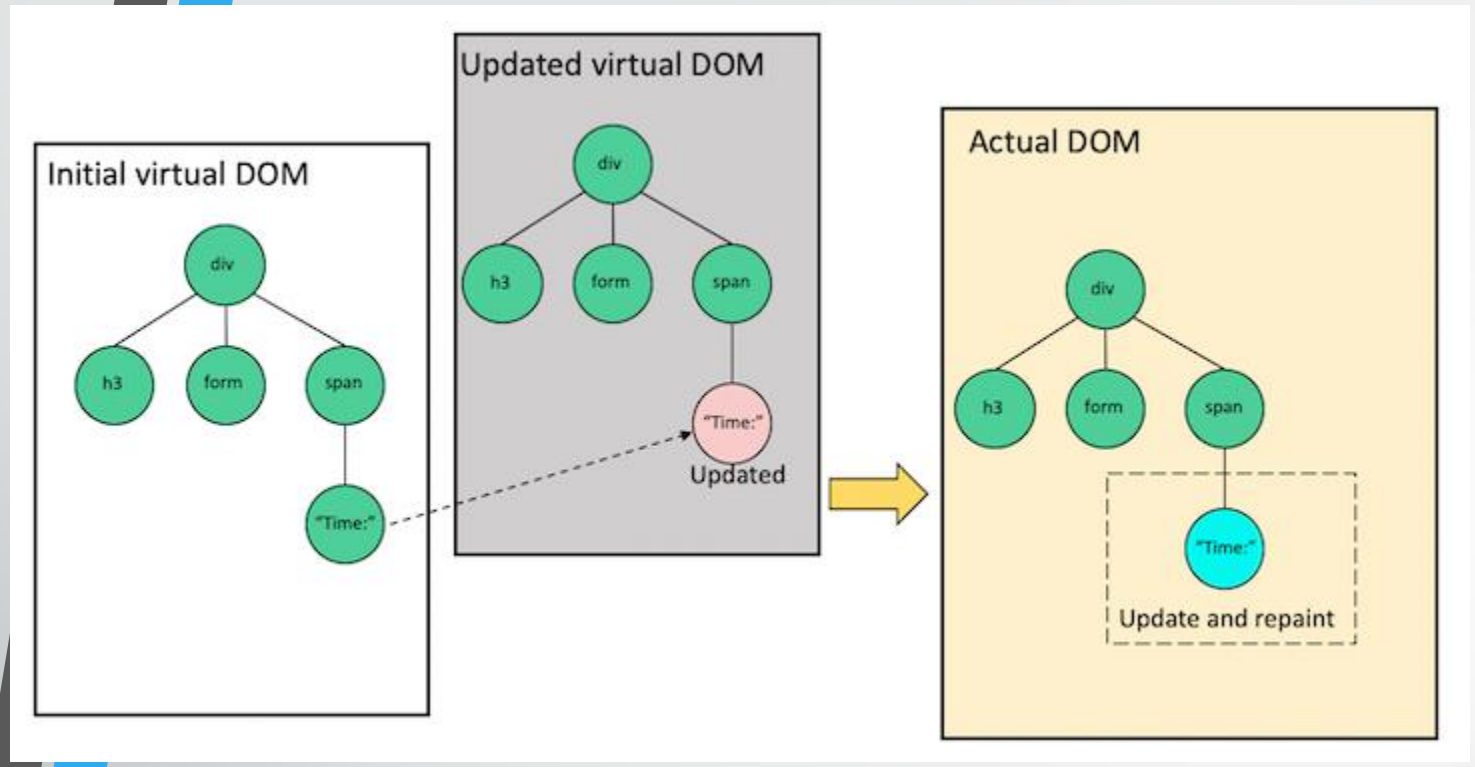
DOM (Document Object Model)

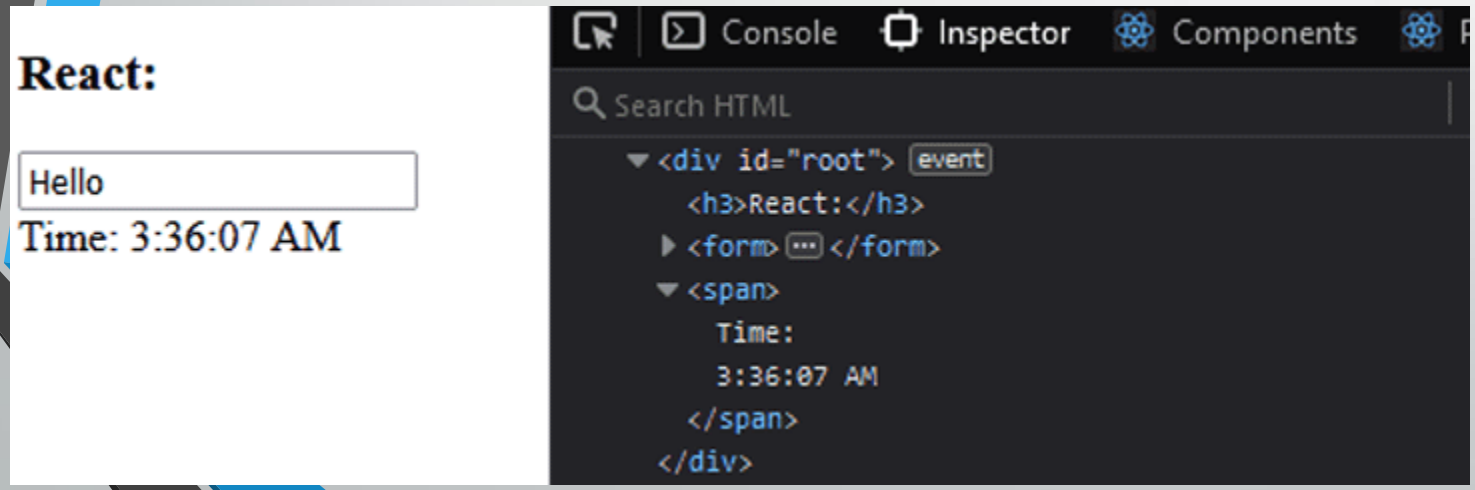


- While HTML is a text, the DOM is an in-memory representation of this text.
- The HTML DOM provides an interface (API) to traverse and modify the nodes.
- So, whenever we want to dynamically change the content of the web page, we modify the DOM.

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 in actual DOM.
  - A reconciliation process reflects the changes in virtual DOM in the real DOM





Yes, the real DOM is updated every second, but thanks to the Virtual DOM, **only the specific part that needs to be updated is changed**—making the process efficient.

Why React?

- The main objective of React is to develop User Interfaces (UI) that improves the speed of the apps.
- It uses virtual DOM (JavaScript object), which improves the performance of the app.
- The JavaScript virtual DOM is faster than the regular DOM.
- We can use React JS on the client and server-side as well as with other frameworks (Nextjs).
- It uses component that improve readability and helps to maintain larger apps.

React Features?

- **JSX** - JavaScript Extension (Its HTML like syntax)
- **Components**
- **Virtual DOM**
- **Simplicity**
- **Performance**
- **One-way Data Binding**

React Getting Started

1. Write React Directly in HTML by Include these 3 scripts:

```
<script src="https://unpkg.com/react@18/umd/react.production.min.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.production.min.js"></script>
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
```

2. Develop React Applications on Your Machine for Development and later Production

   - To use React in production, you need **NPM** and **Node.js**

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.

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

```
<!DOCTYPE html>
<html>
  <script src="https://unpkg.com/react@18/umd/react.production.min.js"></script>
  <script src="https://unpkg.com/react-dom@18/umd/react-dom.production.min.js"></script>
  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  <body>
    <div id="mydiv"></div>

    <script type="text/babel">

      class Hello extends React.Component {
        render() {
          return <h1>Hello World!</h1>
        }
      }

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

   </body>
</html>
```

Setting up a React Environment

- If you have **NPM** and **Node.js** installed, you can create a React application by first installing the **create-react-app**
- You'll need to have **Node >= 14.0.0 and npm >= 5.6** on your machine. You can verify if **Node**, **NPM** are already installed typing following commands in CLI on your machine:

```
> node -v
> npm -v
```

- To create a project, run:

```
> npm install -g create-react-app
> create-react-app my-app
```

You can install Node.js from: https://nodejs.org/

create-react-app

- You are able to create a React application by running this command:

  ```
  C:\Users\Your Name>create-react-app YourReactAppName
  ```

  - You will see following messages on console:

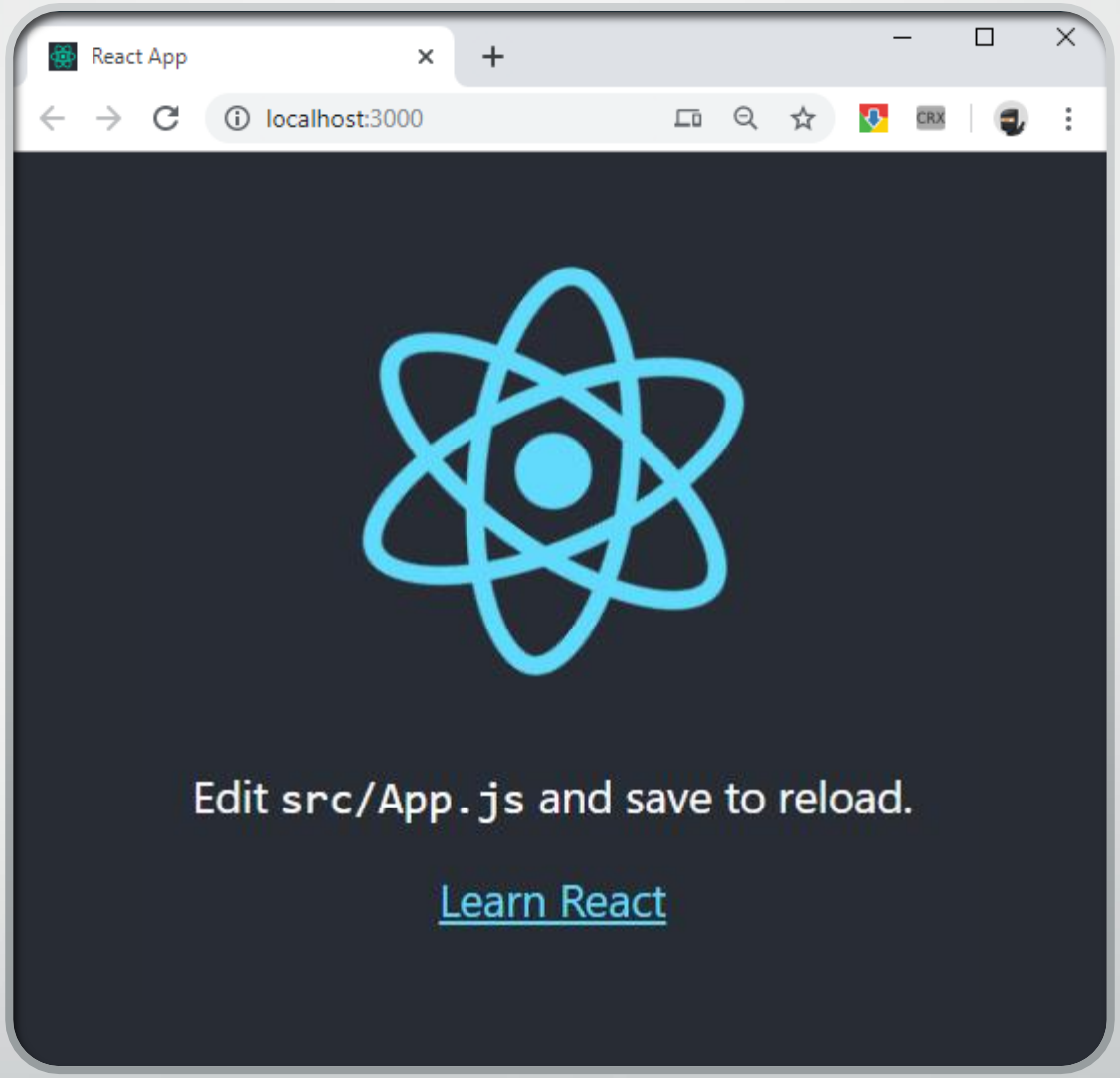
    ```
    Installing packages. This might take a couple of minutes.

    Installing react, react-dom, and react-scripts...

    [  ................] / fetchMetadata: sill resolveWithNewModule
    scheduler@0.18.0 checking installable status
    ```

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

Run the React Application

- Navigate into the React Project Directory and run following command:
  - `C:\Users\`*`UserName`*`>`**`cd YourReactAppName`**
  - `C:\Users\`*`UserName\YourReactAppName`*`>`**`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.

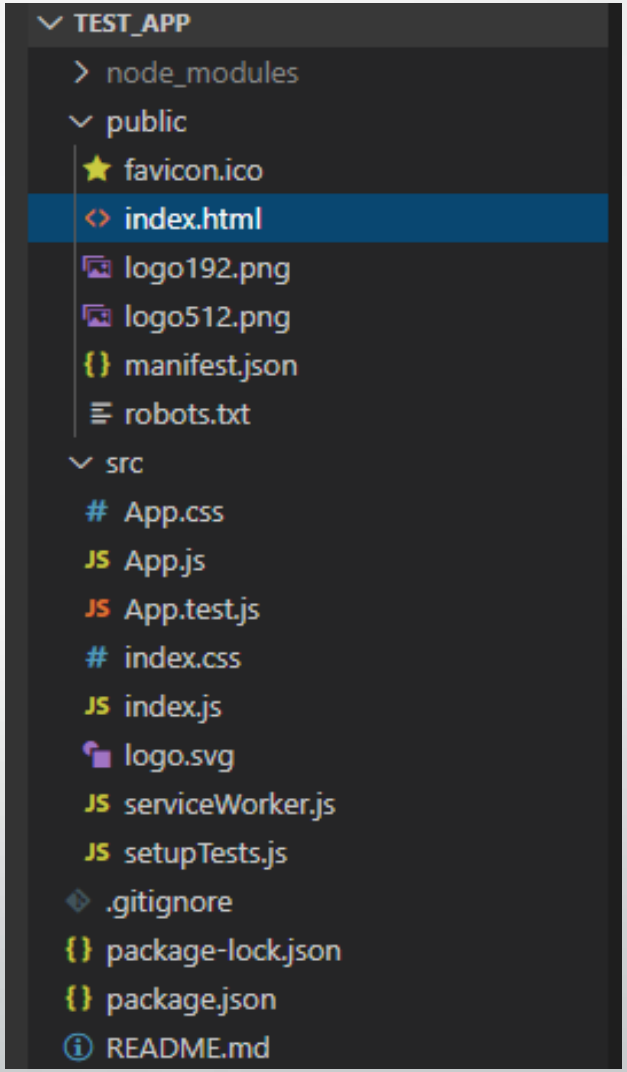
Output Will Look Like This



Folder Structure

For the project to build, these files must exist with exact filenames:

public/index.html is the page template;
src/index.js is the JavaScript entry point.



Basic App Structure

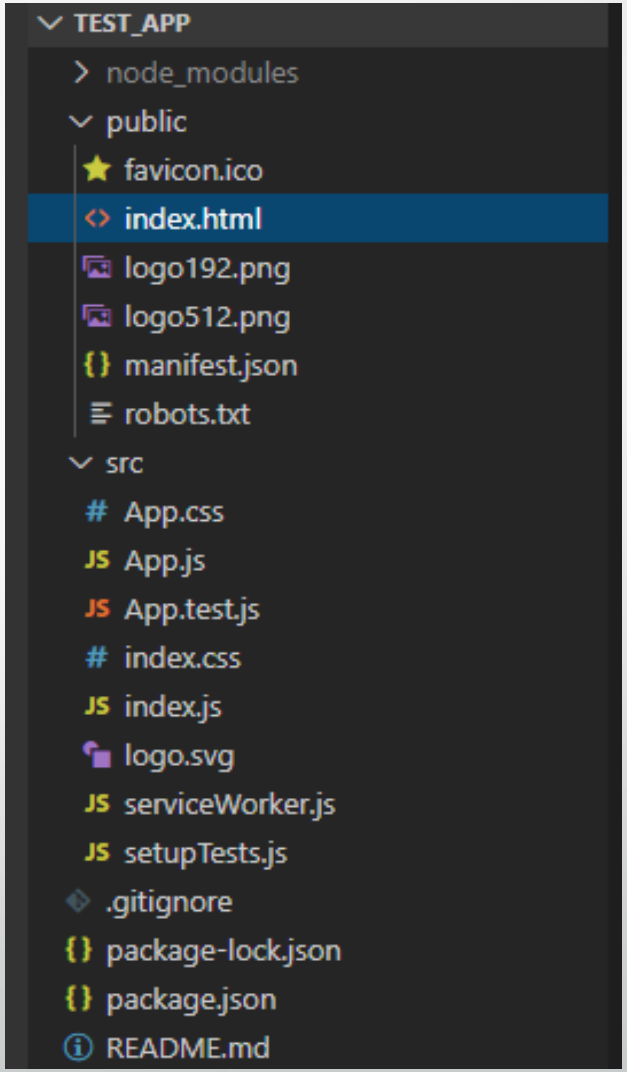
index.html →

⇩

index.js → Loads the React libraries and renders the App component in index.html

⇩

App.js → The App component



React 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.
- React uses ES6, and you should be familiar with some of the new features like:
  - Classes
  - Arrow Functions
  - Variables (let, const, var)

Classes

- ES6 introduced classes.
- Properties are assigned inside a constructor
- You can create objects using the class

```
class Car {
  constructor(name) {
    this.brand = name;
  }
}

mycar = new Car("Ford");
```

Method in Classes

```
class Car {
  constructor(name) {
    this.brand = name;
  }

  present() {
    return 'I have a ' + this.brand;
  }
}

mycar = new Car("Ford");
mycar.present();
```

Arrow Functions

- Arrow functions were introduced in ES6.
- Arrow functions allow us to write shorter function syntax

```
hello = function() {
  return "Hello World!";
}
```

```
hello = () => {
  return "Hello World!";
}
```

- It gets shorter! If the function has only one statement, and the statement returns a value, you can remove the brackets and *return* keyword

```
hello = () => "Hello World!";
```

var, let, const

- var
  - If you use var outside of a function, it belongs to the global scope.
  - If you use var inside of a function, it belongs to that function.
  - If you use var inside of a block, i.e. a for loop, the variable is still available outside of that block.
- let
  - let has a block scope
  - If you use let inside of a block, i.e. a for loop, the variable is only available inside of that loop
- const
  - const is a variable that once it has been created, its value can never change
  - const has a block scope

Render HTML - React v17

- 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().
- `ReactDOM.render()` function takes two arguments, HTML code and an HTML element
- Example
  - Display a paragraph inside the "root" element:

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

Index.Js

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

Index.html

How to upgrade to React 18

```
let container = document.getElementById('app');
ReactDOM.render(<App />, container);
```

```
let container = document.getElementById('app');
let root = ReactDOM.createRoot(container);
root.render(<App />);
```

Render HTML - React v18

- In *Index.js*, update **ReactDOM.render** to **ReactDOM.createRoot** to create a root, and render your app using root asynchronously.
- Example
  - Display a paragraph inside the "root" element:

```
const container = document.getElementById('root');

// create a root
const root = ReactDOM.createRoot(container);

//render app to root
root.render(<p>Hello</p>);
```

Index.Js

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

Index.html

Automatic Batching the State Updates

- **Before React 18**:
  - Batching only occurred within event handlers.
  - Updates outside event handlers (like in async code) caused multiple renders.
- **After React 18**:
  - Automatic batching applies to **all updates**, whether they are inside event handlers or in async callbacks like setTimeout, Promise, etc.
  - Fewer renders, better performance, and more efficient state updates.

It does NOT have to be a 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'));
```

React JSX

- JSX stands for **JavaScript XML**
- JSX allows us to write HTML in React
- JSX allows you to write code that looks like HTML but is actually JavaScript.
- JSX converts HTML tags into react elements
- Without JSX:

```
const myelement = React.createElement('h1', {}, 'I do not use JSX!’);
ReactDOM.render(myelement, document.getElementById('root'));
```

- With JSX you can write expressions inside curly braces

```
const myelement = <h1>Hello, World!</h1>;
```

References

- https://www.w3schools.com/react/default.asp
- https://reactjs.org/docs/hello-world.html
- https://github.com/facebook/create-react-app
- https://create-react-app.dev/