

Inception! (Namaste-React)

☞ Please make sure to follow along with the whole "Namaste **React**" series, starting from Episode-1 and continuing through each subsequent episode. The notes are designed to provide detailed explanations of each concept along with examples to ensure thorough understanding. Each episode builds upon the knowledge gained from the previous ones, so starting from the beginning will give you a comprehensive understanding of React development.

☞ I've got a quick tip for you. To get the most out of these notes, it's a good idea to watch **Episode-1** first. Understanding what "Akshay" shares in the video will make these notes way easier to understand.

So Let's begin our Namaste React Journey

- In this course we will study how the React concepts are actually applied into the industry
i.e. into the real-world projects.
- So, are you ready to fall in love with React????

Introducing React.

Q) What is React? Why React is known as 'React'?

React is a JavaScript Library. The name 'React' was chosen because the library was designed to allow developers to **react** to **changes in state and data** within an application, and to update the user interface in a declarative and efficient manner.

Q) What is Library?

Library is a collections of prewritten code snippets that can be used and reused to perform certain tasks. A particular JavaScript library code can be plugged into application code which leads to faster development and fewer vulnerabilities to have errors. Examples: React, jQuery

Q) What is Framework?

Framework provides a basic foundation or structure for a website or an application.

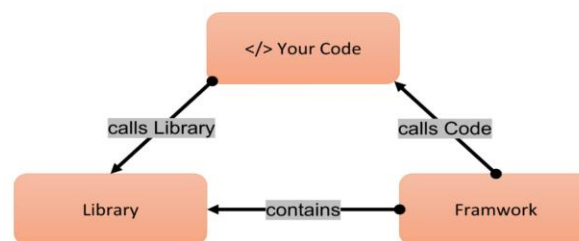
Examples: Angular

Q) Similarities between Library and Framework?

Frameworks and libraries are code written by third parties to solve regular/common problems or to optimise performance.

Q) Difference between Library and Framework?

A key difference between the two is Inversion of control. When using a library, the control remains with the developer who tells the application when to call library functions. When using a framework, the control is reversed, which means that the framework tells the developer where code needs to be provided and calls it as it requires.



* **Emmet**: Emmet is the essential toolkit for web-developers. It allows you to **type shortcuts** that are then expanded into full-fledged boilerplate code for writing **HTML** and **CSS**.

It basically generates some code for you inside VS code. (!, html:5)

Q) Create Hello World Program using only HTML?

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Document</title>
7    </head>
8    <body>
9      <div id="root">
10       <h1>Hello World! using only HTML</h1>
11     </div>
12   </body>
13 </html>
```

Q) Create Hello World Program using only JavaScript?

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Document</title>
7    </head>
8
9    <body>
10     <div id="root"></div>
11   </body>
12   <script>
13     const heading = document.createElement("h1");
14     heading.innerHTML = "Hellow World! From Javascript";
15
16     const divElement = document.getElementById("root");
17
18     divElement.appendChild(heading);
19   </script>
20 </html>
21
```

Q) Create Hello World Program using only React?

```

1  <body>
2    <div id="root"></div>
3
4    <script
5      crossorigin
6      src="https://unpkg.com/react@18/umd/react.development.js"
7    ></script>
8    <!--This file i.e. react.development.js contains the React Code which is plain JS-->
9
10   <script
11     crossorigin
12     src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
13   ></script>
14   <!--This file i.e. react-dom.development.js is useful for DOM operations or use to modify the DOM-->
15
16   <script>
17     const heading = React.createElement("h1", {}, "Hello World from React!");
18     const root = ReactDOM.createRoot(document.getElementById("root"));
19     root.render(heading);
20   </script>
21 </body>

```

* Cross Origin:

The crossorigin attribute in the script tag enables CrossOrigin Resource Sharing (CORS) for loading external JavaScript files from different origin than the hosting web page. This allows the script to access resources from the server hosting the script, such as making HTTP requests or accessing data.

Q) What is {} denotes in above code?

```

1  <div id="root">
2    <h1 id="title">Hello World!</h1>
3  </div>

```

This (id='title'), classes, etc should come under {}. Whenever I'm passing inside {}, will go as tag attributes of h1.

📌 NOTE: React will overwrite everything inside "root" and replaces with whatever given inside render.

Q) Do the below HTML code in React?

```
1 <div id="parent">
2   <div id="child1">
3     <h1>Heading 1</h1>
4     <h2>Heading 2</h2>
5   </div>
6   <div id="child2">
7     <h1>Heading 1</h1>
8     <h2>Heading 2</h2>
9   </div>
10 </div>
```

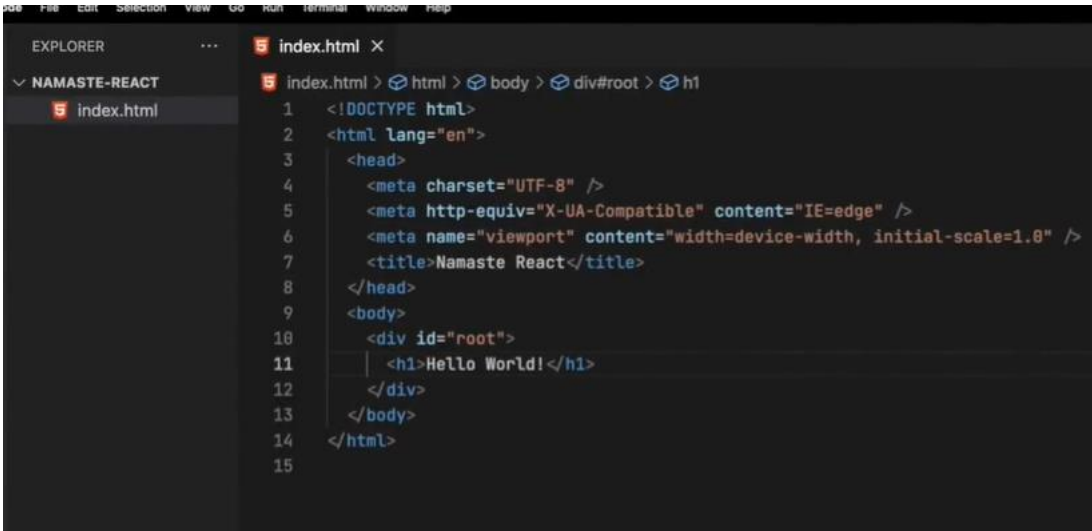
To build the structure like this in React

```
1 const parent = React.createElement("div", { id: "parent" }, [
2   React.createElement("div", { id: "child1" }, [
3     React.createElement("h1", { id: "heading_1" }, "Heading 1"),
4     React.createElement("h2", { id: "heading_2" }, "Heading 2"),
5   ]),
6   React.createElement("div", { id: "child2" }, [
7     React.createElement("h1", { id: "heading_1" }, "Heading 1"),
8     React.createElement("h2", { id: "heading_2" }, "Heading 2"),
9   ]),
10 ]);
11
12 const root = ReactDOM.createRoot(document.getElementById("root"));
13
14 root.render(parent);
15
```

Self Notes -

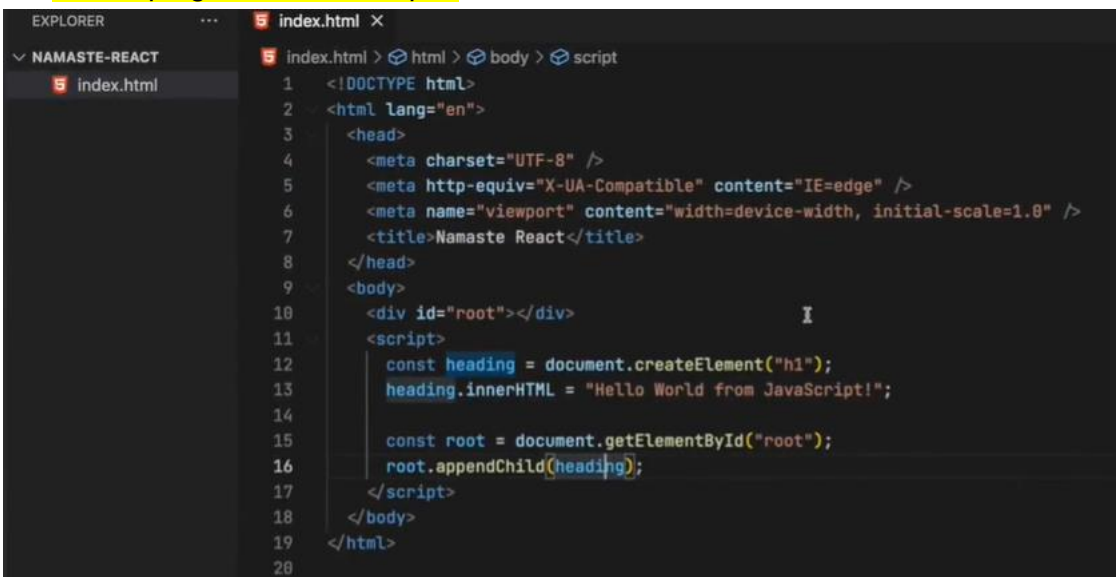
Chapter 01 – Inception

1. Basic program in HTML-



```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <title>Namaste React</title>
8   </head>
9   <body>
10    <div id="root">
11      <h1>Hello World!</h1>
12    </div>
13  </body>
14 </html>
15
```

2. Create program in JavaScript –



```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <title>Namaste React</title>
8   </head>
9   <body>
10    <div id="root"></div>
11    <script>
12      const heading = document.createElement("h1");
13      heading.innerHTML = "Hello World from JavaScript!";
14
15      const root = document.getElementById("root");
16      root.appendChild(heading);
17    </script>
18  </body>
19 </html>
20
21
22
23
24
25
26
27
28
```

How this project understands this is document, this is createElement, this is innerHTML, this is getElementById. How my code, how my browser understands what are these things?

createElement , innerHTML - These are the superpowers, browser already have it and browser

have javascript engine that execute these javascript.

Does the browser know what is react?

Browser does not understand react code.

So, what we do to use react. We will configure react in in project/ will get react in our project.

1way – CDN (content delivery network)

These are the website react is hosted and we are pulling the react in our project.

CDN is the place where react library is hosted. So, we are fetching the react and putting in the code using CDN links.

<https://legacy.reactjs.org/docs/cdn-links.html>

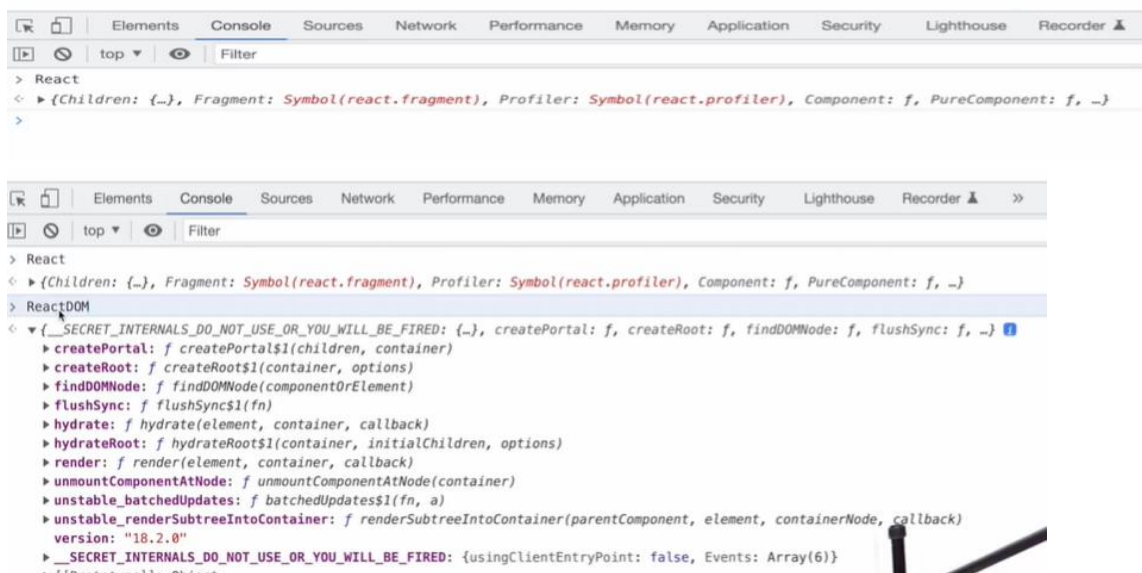
```
<script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js"></script>

<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
```

First link – this is the react development file and it is core file of react. This is the core react algorithm which is written inside it.

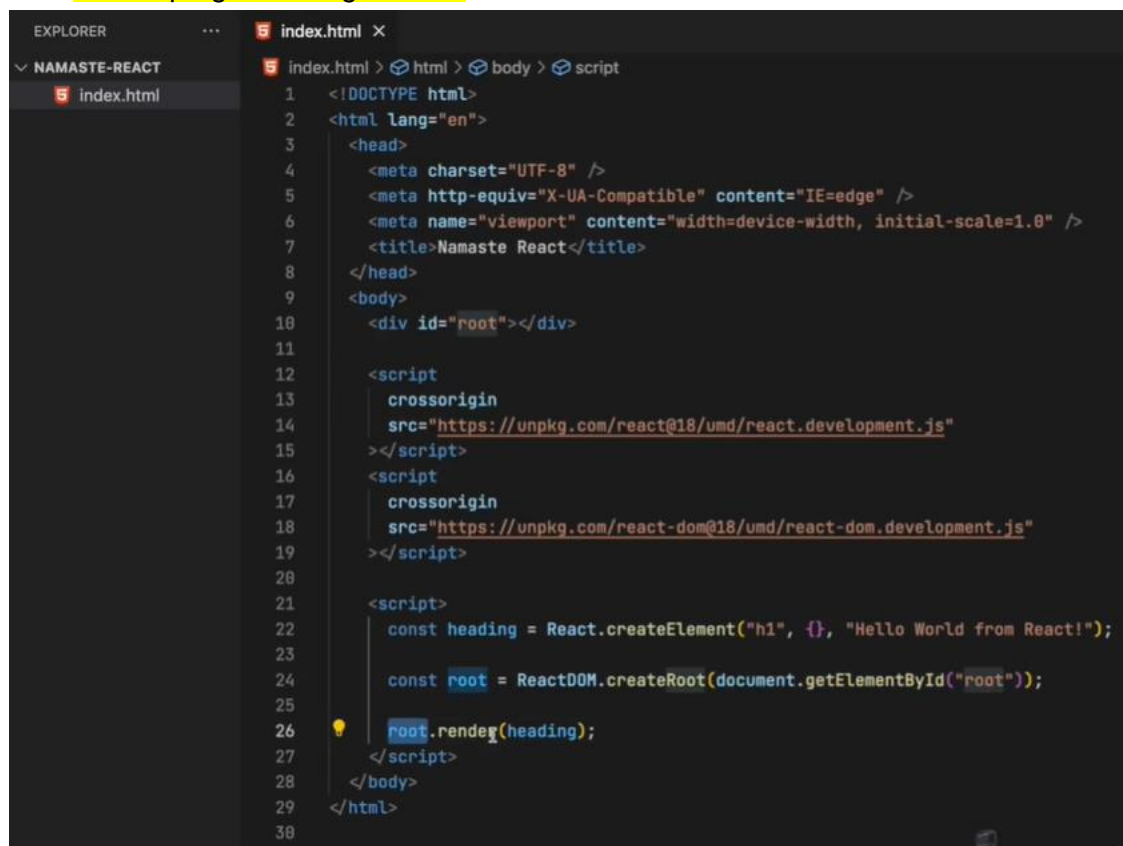
Second link – this is the react dom file which is useful for dom operations. Useful for modify the DOM.

After using CDN link in the project after that the browser understands the react.



This reacts injecting to the CDN link.

3. Create program using React –

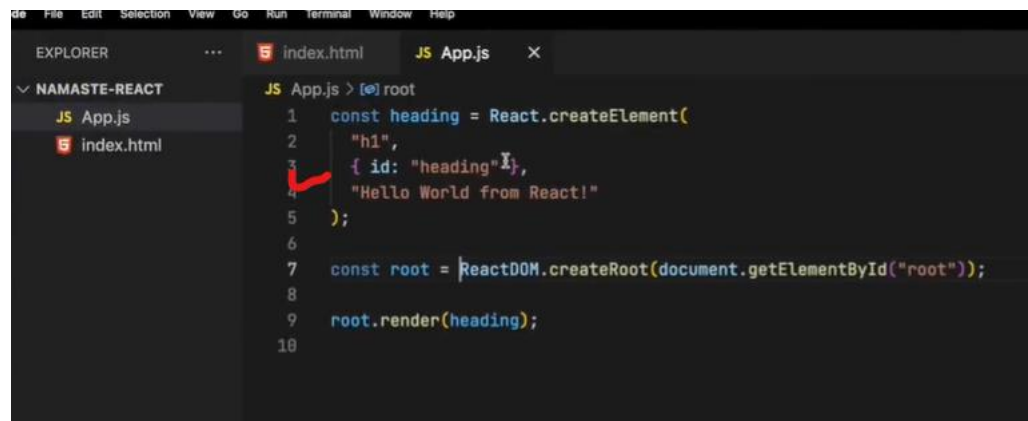


```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <title>Namaste React</title>
8   </head>
9   <body>
10    <div id="root"></div>
11
12    <script
13      crossorigin
14      src="https://unpkg.com/react@18/umd/react.development.js"
15    ></script>
16    <script
17      crossorigin
18      src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
19    ></script>
20
21    <script>
22      const heading = React.createElement("h1", {}, "Hello World from React!");
23
24      const root = ReactDOM.createRoot(document.getElementById("root"));
25
26      root.render(heading);
27    </script>
28  </body>
29 </html>
```

```
const heading = React.createElement("h1", {}, "Hello World")
```

{ } is the place where we can give attribute to a tag/class like id, class we can give here.

Creating the root is the work for create Root.// here we are creating the root and after that we render it.



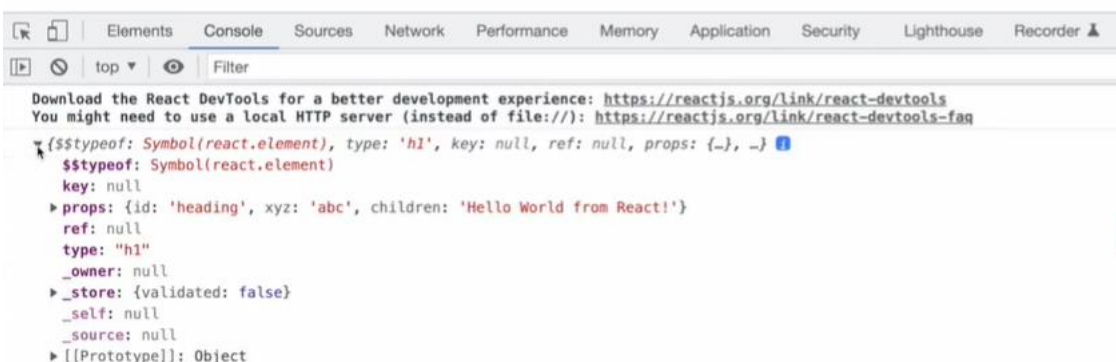
```
1 const heading = React.createElement(
2   "h1",
3   { id: "heading" },
4   "Hello World from React!"
5 );
6
7 const root = ReactDOM.createRoot(document.getElementById("root"));
8
9 root.render(heading);
10
```


We can give more attribute also.

```
< index.html JS App.js X
JS App.js > ...
1  const heading = React.createElement("h1", { id: "heading", xyz:"xyz" }, "Hello World")
2  const root = ReactDOM.createRoot(document.getElementById("root"))
3  root.render(heading)
```

What will happen when will do console.log

```
... index.html index.css JS App.js X
JS App.js > ...
1  const heading = React.createElement(
2    "h1",
3    { id: "heading", xyz: "abc" },
4    "Hello World from React!"
5  );
6
7  console.log(heading);
8
9  const root = ReactDOM.createRoot(document.getElementById("root"));
10
11  root.render(heading);
12
```



We will get the object, This is the react element and react element is the react object.

```
index.html index.css JS App.js X
JS App.js > ...
1  const heading = React.createElement(
2    "h1",
3    { id: "heading", xyz: "abc" },
4    "Hello World from React!"
5  );
6
7  console.log(heading); // object
8
9  const root = ReactDOM.createRoot(document.getElementById("root"));
10
11  root.render(heading);
12
```

How to create nested structure –

Example 1-

```

*
* <div id="parent">
*   <div id="child">
*     <h1>I'm h1 tag</h1>
*   </div>
* </div>
*
*

```

```

... index.html index.css JS App.js X
JS App.js > ...
7 * </div>
8 *
9 *
10 */
11
12 const parent = React.createElement(
13   "div",
14   { id: "parent" },
15   React.createElement(
16     "div",
17     { id: "child" },
18     React.createElement("h1", {}, "I'm an h1 tag")
19   )
20 );
21
22 console.log(parent); // object
23
24 const root = ReactDOM.createRoot(document.getElementById("root"));
25
26 root.render(parent);
27

```

ReactElement is the Object => this reacts object becomes HTML that the browser understands
 Render=> convert it into the html and puts up into the DOM. It will work behind the scenes.

Example -2

```

Go Run Terminal Window Help
index.html index.css JS App.js X
JS App.js > parent
1 /**
2  *
3  * <div id="parent">
4  *   <div id="child">
5  *     <h1>I'm h1 tag</h1>
6  *     <h2>I'm h1 tag</h2>
7  *   </div>
8  * </div>
9  *
10 * ReactElement(Object) => HTML(Browser Understands)
11 */
12
13 const parent = React.createElement(
14   "div",
15   { id: "parent" },
16   React.createElement("div", { id: "child" },
17     React.createElement("h1", {}, "I'm an h1 tag"),
18     React.createElement("h2", {}, "I'm an h2 tag"),
19   )
20 );
21
22 console.log(parent); // object
23
24 const root = ReactDOM.createRoot(document.getElementById("root"));
25
26 root.render(parent);
27

```

Example – 3



The image shows a code editor with two tabs: 'index.html' and 'index.css'. The active tab is 'JS App.js'. The code is divided into two sections. The top section (lines 1-15) shows JSX syntax for creating a parent div with two child divs, each containing an h1 and an h2 tag. The bottom section (lines 16-33) shows the equivalent JavaScript code using `React.createElement` and `ReactDOM.createRoot` to render the same structure.

```
JS App.js > ...
1  /**
2   *
3   * <div id="parent">
4   *   <div id="child">
5   *     <h1>I'm h1 tag</h1>
6   *     <h2>I'm h1 tag</h2>
7   *   </div>
8   * <div id="child2">
9   *   <h1>I'm h1 tag</h1>
10  *   <h2>I'm h1 tag</h2>
11  *   </div>
12  * </div>
13  *
14  * ReactElement(Object) => HTML(Browser Understands)
15  */
16
17  const parent = React.createElement("div", { id: "parent" }, [
18    React.createElement("div", { id: "child" }, [
19      React.createElement("h1", {}, "I'm an h1 tag"),
20      React.createElement("h2", {}, "I'm an h2 tag"),
21    ]),
22    React.createElement("div", { id: "child" }, [
23      React.createElement("h1", {}, "I'm an h1 tag"),
24      React.createElement("h2", {}, "I'm an h2 tag"),
25    ]),
26  ]);
27
28  console.log(parent); // object
29
30  const root = ReactDOM.createRoot(document.getElementById("root"));
31  |
32  root.render(parent);
33
```

It is very complicated to write it that's why JSX comes in the picture.

Lot of people think react only be written inside JSX. Core of react is this. We can write without JSX. JSX makes our life easy when we create tags.

If we change the order of the tags. It will still working? / Will the order of file matter?

Now –

```
Go Run Terminal Window Help
index.html x index.css JS App.js
index.html > html > head > link
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <link rel="stylesheet" href="./index.css" />
8     <title>Namaste React</title>
9   </head>
10  <body>
11    <div id="root"></div>
12
13    <script
14      crossorigin
15      src="https://unpkg.com/react@18/umd/react.development.js"
16    ></script>
17    <script
18      crossorigin
19      src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
20    ></script>
21
22    <script src="./App.js"></script>
23  </body>
24 </html>
25
```

After changing it –

```
index.html • index.css JS App.js
index.html > html > body
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <link rel="stylesheet" href="./index.css" />
8     <title>Namaste React</title>
9   </head>
10  <body>
11    <div id="root"></div>
12
13    <script src="./App.js"></script>
14    <script
15      crossorigin
16      src="https://unpkg.com/react@18/umd/react.development.js"
17    ></script>
18    <script
19      crossorigin
20      src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
21    ></script>
22
23  </body>
24 </html>
25
```



Now it will throw an error

That means react cdn link before the JavaScript file. The order of the file always in sequences when they are imported.

It will load react in the app, then it will load dom-react in the app, then it will load app.js in the app., then it will execute the code line by line. When it will react root.render(parent). It will render parent inside the root. If something inside the root it will replace whatever I am passing in react. So, react takes control over this div now. (it will replace not appended)

React is the small JavaScript library. It can import JavaScript file into the code.

You can use react into the existing web application project. No matter it is jQuery.

If the application is the jQuery we can import react using CDN and using it.

What happen when I use tag in the html and at the same time render it in the js

-ans Whatever inside this root (html part) replace by whatever I am passing inside the render.

First it will print Pragati but as soon as it will load react dom and then it will render root into the app and print.

Questions series-

- What is Emmet?
- Difference between a Library and Framework?
- What is CDN? Why do we use it?
- Why is React known as React?
- What is cross origin in script tag?
- What is difference between React and ReactDOM

React focuses on the component-based architecture and handling of state and props, while ReactDOM handles the interaction between React components and the browser's DOM, rendering them efficiently.

- What is difference between react.development.js and react.production.js files via CDN?

0 react.development.js: This file is meant for development purposes. It includes helpful warnings and debugging information, making it larger in size but aiding developers in identifying and fixing issues during development.

- **react.production.js**: This file is optimized for production use. It removes unnecessary warnings, debug information, and other development-specific code, resulting in a smaller file size and improved performance in production environments.

- What is async and defer? - see my Youtube video ;)

Coding -

- Set up all the tools in your laptop
 - VS Code
 - Chrome
 - Extensions of Chrome
- Create a new Git repo
- Build your first Hello World program using,
 - Using just HTML
 - Using JS to manipulate the DOM
 - Using React
 - use CDN Links
 - Create an Element
 - Create nested React Elements
 - Use root.render
- Push code to Github (Theory as well as code)
- Learn about Arrow Functions before the next class

References:

- <https://beta.reactjs.org/apis/react/createElement>
- <https://www.youtube.com/watch?v=IrHmpdORLu8>