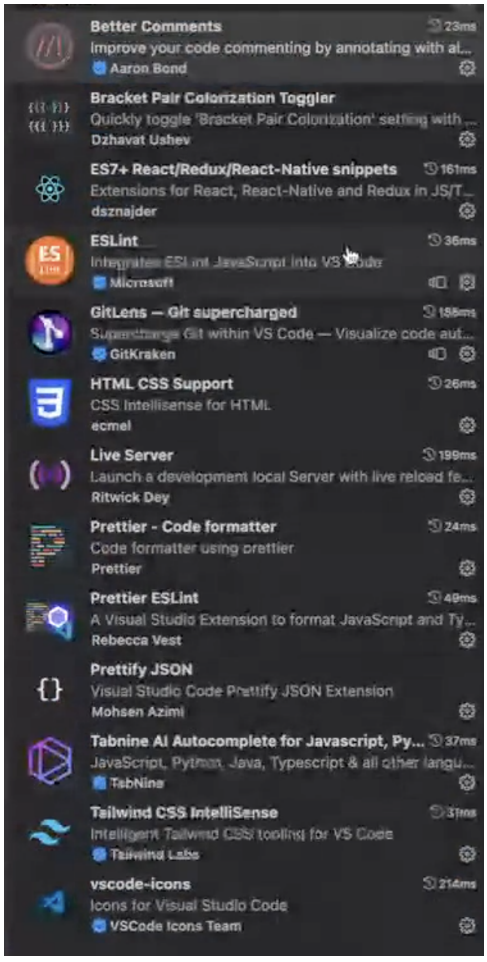
Chapter 01 - Inception

▼ Difference between library and framework

- Framework: Comes with lot of components. Eg: Routing etc
- React is a library. It takes minimum effort for library to put inside our code.

▼ Useful VS code Extensions:



▼ Homework: Emmet

- Read about Emmet.
    - Vs code emit boiler plate code https://code.visualstudio.com/docs/editor/emmet

---

Create header using:

▼ html

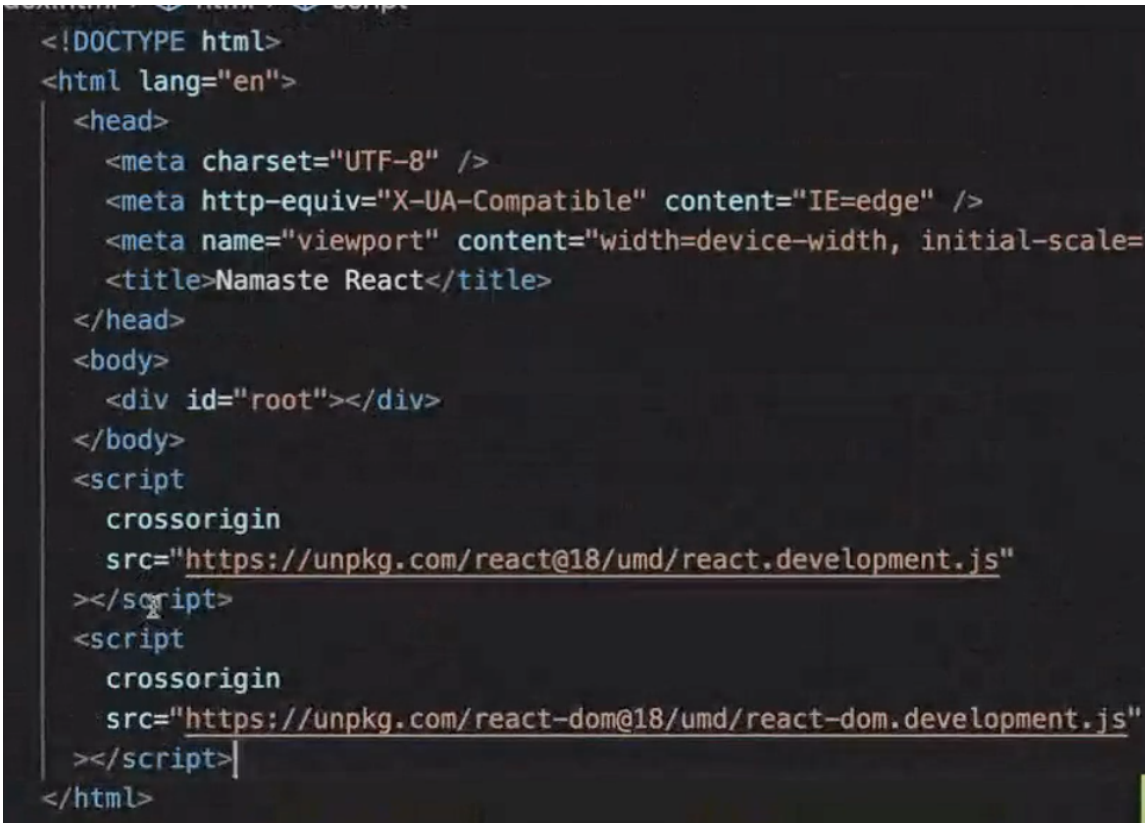
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Namaste React</title>
  </head>
  <body>
    <h1>Namaste Everyone!</h1>
  </body>
</html>
```

▼ Javascript

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Namaste React</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
  <script>
    const heading = document.createElement("h1");

    heading.innerHTML = "Namaste Everyone from JavaScript!";

    const root = document.getElementById("root");

    root.appendChild(heading);
  </script>
</html>
```

▼ React

- React CDN links: `"https://unpkg.com/react@18/umd/react.development.js", "https://unpkg.com/react-dom@18/umd/react-dom.development.js"`
- React is just a piece of javascript code
- can be injected as html script src
- This will add global object React & ReactDOM



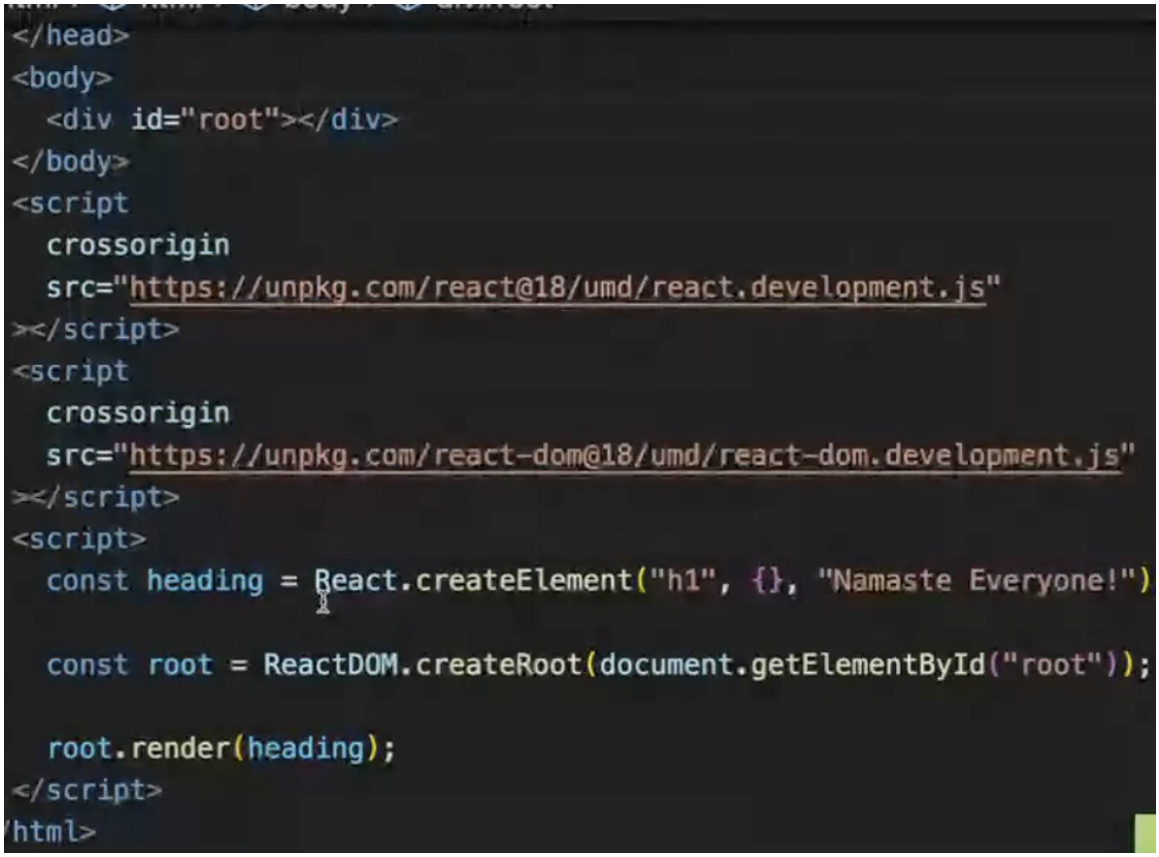
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=
    <title>Namaste React</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
  <script
    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>
</html>
```

▼ Homework: CDN

- What is CDN:
- Content delivery network
- What is cross origin

▼ Why there are two files React & ReactDOM

- React is not only for browser, it works for mobile and other platforms.
- React is core version of React.
- ReactDOM means web version of react.
- ReactDOM gives access to Browser DOM.



```
</head>
<body>
  <div id="root"></div>
</body>
<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>
<script>
  const heading = React.createElement("h1", {}, "Namaste Everyone!");

  const root = ReactDOM.createRoot(document.getElementById("root"));

  root.render(heading);
</script>
</html>
```

- React.createElement can be used to create a react element (still not in browser DOM). This is plain Javascript object. These are React tags not actual html.
- ReactDOM.createRoot() will tell which is root element in browserDOM (ReactDOM can access browser DOM).
- Though element created using React.createElement is same for all platforms, ReactDOM is specific for browser DOM. (differnt one for Mobile)

- So after creating an React element, and telling ReactDOM which DOM element in browser is root, how do we add react element to root in browser DOM?
    - root.render() does that. Bascically here ReactDOM accepting react element(Javascript object), and adding it as DOM element to browser

▼ Can we have multiple root?

- Generally for any big/small app, we will have one root and one render method. React will be injected to root element.
- root is the place where react runs, we can have anything before and after root element in html. So we can add react to even existing app (for eg: add a search element which is react). But generally we dont do that and have only one root as above.

▼ What is the empty tag {} in React.createElement above

- we can pass things like id, class etc (https://beta.reactjs.org/reference/react/createElement)

▼ What if we add elements inside root using html manually and also add an element using use react.render() ?

- element added using react will overide other elements in root.

▼ HomeWork:

- Difference between async and defer, while calling script?

https://eager.io/blog/everything-I-know-about-the-script-tag/

▼ Creating complex html with multiple children using React:

- Pass children in array param for React.createElement

```
<script>
  const heading = React.createElement(
    "h1",
    {
      id: "title",
    },
    "Heading 1"
  );

  const heading2 = React.createElement(
    "h2",
    {
      id: "title",
    },
    "Heading 2"
  );

  const container = React.createElement(
    "div",
    {
      id: "container",
    },
    [heading, heading2]
  );

  console.log(heading);

  const root = ReactDOM.createRoot(document.getElementById("root"));

  //passing a react element inside the root

  //async defer
  root.render(container);
```

▼ JSX

React wanted to write whole html & css inside Javascript.

- Because they don’t want us go to html / css file.
- React.createElement() method is used to create html & css, this is plain javascript object.

- But this is not user friendly and lot of code → JSX came into picture. So JSX is not mandatory for react.

▼ Reduce file size by spliting code into differnt file & importing