
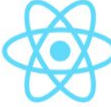




29th June 2020	REACTJS → Sundeep Sir	Woodpecker 
Revision Week DSA By Subham Joshi Sir.	Lecture flow: → Intro  React	
Topics & Explanation:		Reference
<p>Before React:</p> <ul style="list-style-type: none"> • Before React, we had Angular, which took a hit later due To usage of Typescript • It was “Enterprise-level” and now it is limited to a small circle <p>React::</p> <ul style="list-style-type: none"> • React.js is a JavaScript library. • It was developed by engineers at Facebook. <p>Why learn React?</p> <ul style="list-style-type: none"> • React is fast. Apps made in React can handle complex updates and still feel quick and responsive. • React is modular. Instead of writing large, dense files of code, you can write many smaller, reusable files. React’s modularity can be a beautiful solution to JavaScript’s maintainability problems. • React is scalable. Large programs that display a lot of changing data are where React performs best. • React is flexible. You can use React for interesting projects that have nothing to do with making a web app. People are still figuring out React’s potential. There’s room to explore. • React is popular. While this reason has admittedly little to do with React’s quality, the truth is that understanding • React will make you more employable. 		<p>Click me!</p>  <p>React JS</p> <p>React Virtual DOM vs Real DOM</p>

```
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 class="container"></div>
10    <script
11      crossorigin
12      src="https://unpkg.com/react@16/umd/react.development.js"
13    ></script>
14    <script
15      crossorigin
16      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
17    ></script>
18    <script>
19
20    </script>
21  </body>
22 </html>
23
```

we will inject our code,PARENT Element

[React JSX - W3Schools](#)

[Introducing JSX - React](#)

NOTE: Entire DOM is not changed here, instead, the changed part is compared and Changed in React, so it boosts performance.

```
      src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
    ></script>
    <script>
      const h1Element = React.createElement(
        "h1",
        null,
        "My first React H1 tag"
      );

      ReactDOM.render(h1Element, document.querySelector(".container"));
    </script>
  </body>
</html>
```

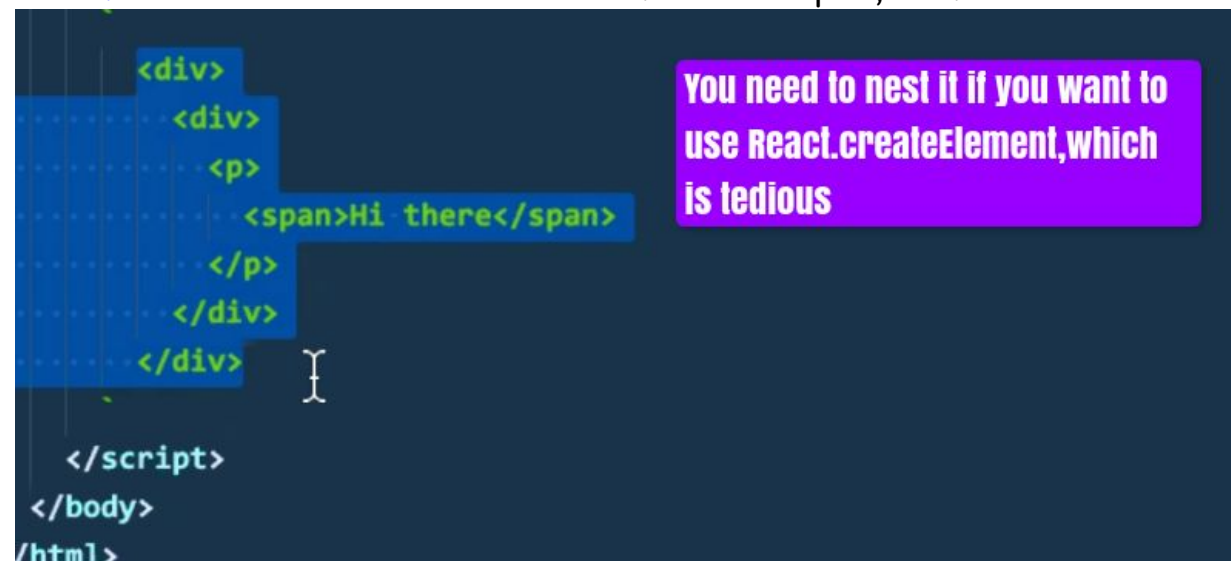
created a element and used ReactDOM to render



[Polyfill - MDN Web Docs Glossary: Definitions of Web-related terms](#)

[Babel · The compiler for next generation JavaScript](#)

`React.createElement` is tedious to use: For example ,This:



So, **JSX** comes into Picture: JSX allows us to write HTML in React. JSX makes it easier to write and add HTML to React.

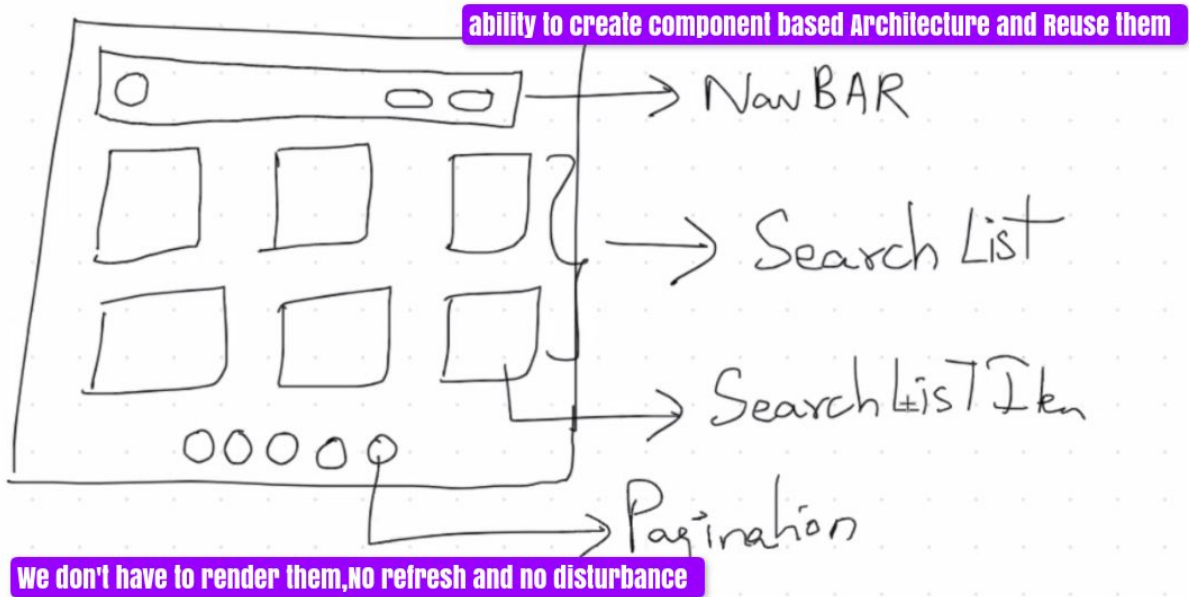


```
<h1>My first React H1 tag</h1>
```

```
1 "use strict";  
2  
3 /*#__PURE__*/  
4 React.createElement("h1", null, "My first React H1 tag");
```

JSX to Javascript for Browsers to understand --> Polyfilling with Babel

React Component-Based Model:



Functional Based Components demo

```
const boxStyle = {
  width: "100px",
  height: "100px",
  background: "red"
};
// Functional components
const App = () => {
  return (
    <div>
      <h1>Hi there</h1>
      <div style={boxStyle}>I am a box</div>
    </div>
  );
};
// const name = prompt("What is your name?");
// JSX = JavaScript XML
// const h1Element = <h1 name="something">Hi {2 > 3 ? "Good" : "Bad"}!</h1>;
ReactDOM.render(<App/>, document.querySelector("#app"));
</script>
</body>
</html>
```

Annotations in the image:

- adding style in JSX (points to `style={boxStyle}`)
- A functional component (points to `const App = () => {`)
- wrapper single element (points to `<div>`)
- component rendered (points to `ReactDOM.render`)
- customised derived component (points to `<App/>`)

Hi there

I am a box

output

<!d
<ht
><t
><t
>

Class Based Component:

```
// Class based component
class AppClass extends Component {
  constructor() {
    super();
  }

  render() {
    return (
      <>
        <h1>Hi there</h1>
        <div style={boxStyle}>I am a box</div>
      </>
    );
  }
}

// const name = prompt("What is your name?");
// JSX = JavaScript XML
// const h1Element = <h1 name="something">Hi {2 > 3 ? "Good" : "Bad"}!</h1>;
ReactDOM.render(<AppClass />, document.querySelector("#app"));
```

Break Time ⇒ 12.45 → 1.00PM

[npm vs npx — What's the Difference? - freeCodeCamp.org](#)

- Unfortunately, This Strategy is not Scalable → Create-react-app comes into picture
- So, A predefined Package, with all the dependencies and scripts with web bundlers is available for us
- [Create a New React App – React](#)

[Create a New React App – React](#)

[npm vs npx — What's the Difference? - freeCodeCamp.org](#)

`npx create-react-app <application name>`

- Please watch the Video from here (last 25 mins) to understand the file structure in Create-React App
- Install this extension [React Developer Tools - Google Chrome](#) to see the Virtual DOM

Links: (official) → React Docs

- [React.js Boilerplate](#)
- [Introducing JSX – React](#)
- [Components – React](#)

Virtual DOM Vs Real DOM

- [The DOM → Real Dom and Virtual DOM:](#)

Link to the lecture video:- [DASHBOARD](#)

Summary: Revision → ReactJS → Intro → simple React App with CDN → JSX → Fn and Class based Components