

6

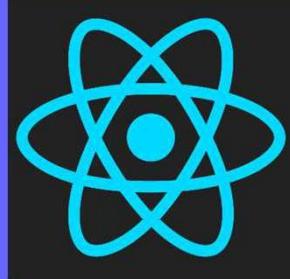
React JSX

Objectives

After completing this lesson, you should be able to do the following:

- Creating Components
- Basic Component
- Nesting Components
- Props





JSX

- JavaScript XML -- (Extension to the JavaScript Language Syntax)
- In React Lib Helps Writing XML – Like Code for Elements and Components
- Just Like XML [JSX Tags have a Tag Name, Attributes and Children]

- Is it Need to React Application
 - Ans is No [JSX is Not a Necessity to Write React Application]
- Why ?
 - JSX Makes your React Code Simpler and Elegant
- JSX Ultimately transpiles to Pure JavaScript which is Understood by the Browsers

- Lets see what does the code look like without JSX
- Let's create a react component using JSX and without using JSX
- That way we will not only understand how JSX translates to regular javascript but also appreciate how it brings out simplicity in your code

Lets Create Hello.js

- This Component will be Simple Functional Component which Render HelloMSG [Using JSX]

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure:
 - LESSON03-HELLOWORLD
 - my-app
 - node_modules
 - public
 - src
 - components
 - Greet.js
 - Hello.js
 - Welcome.js
 - App.css
 - App.js
 - App.test.js
 - index.css
 - index.js
 - logo.svg
- EDITOR:** The file `Hello.js` is open, showing the following code:

```
import React from 'react';
const Hello = () => {
  return (
    <div>
      <h1>Hello React !</h1>
    </div>
  )
}
export default Hello;
```
- TITLE BAR:** The tabs show the current file is `Hello.js`.

Include Component in App.js

The image shows a code editor interface with a dark theme. A red box highlights the code editor area. In the top bar, tabs for 'App.js', 'Hello.js', 'Welcome.js', and 'Greet.js' are visible. The code editor displays the following content:

```
my-app > src > JS App.js > ...
1 | import React, { Component } from 'react';
2 | import logo from './logo.svg';
3 | import './App.css';
4 | import { Greet } from './components/Greet';
5 | import Welcome from './components/Welcome';
6 | import Hello from './components/Hello';
7 |
8 | class App extends Component {
9 |   render() {
10 |     return (
11 |       <div className="App">
12 |         {/* <Greet />
13 |         <Welcome /> */}
14 |
15 |         <Hello />
16 |       </div>
17 |     );
18 |   }
19 | }
```

To the right of the code editor, a red box highlights a browser window. The browser's address bar shows 'neDr...'. The page content area displays the text 'Hello React !'.

Rewrite the Component Without Using JSX

- To Do So React Lib Provides createElement() Method

The screenshot shows a code editor with two tabs: 'Hello.js' and 'index.html'. The 'Hello.js' tab contains the following code:

```
JS App.js M JS Hello.js U X JS Welcome.js U JS Greet.js U

my-app > src > components > JS Hello.js > [o] Hello
1 import React from 'react';
2
3 const Hello = () => {
4     // return (
5     //   // <div>
6     //   //     <h1>Hello React !</h1>
7     //   // </div>
8     // )
9
10    return React.createElement('div', null, '<h1>Hello React !</h1>');
11
12
13 export default Hello;
```

The 'index.html' tab shows the generated HTML output:

```
<!DOCTYPE html>
<html lang="en">
  <head> ... </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">
      <div class="App"> == $0 ⌂
        <div><h1>Hello React !</h1></div>
      </div>
    </div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

A red box highlights the text '

Hello React !

' in the generated HTML, indicating that the browser has treated the JSX content as plain text.

- Output is not as Expected, Since it has Considered <h1> as Simple Text Content

Solution

```
JS App.js M JS Hello.js U X JS Welcome.js U JS Greet.js U
my-app > src > components > JS Hello.js > [?] Hello
1 import React from 'react';
2
3 const Hello = () => {
4     // return (
5     //     // <div>
6     //     //     <h1>Hello React !</h1>
7     //     // </div>
8     // )
9
10    return React.createElement('div', null, React.createElement(['h1'], null, 'Hello React !'));
11 }
12
13 export default Hello;
```

Hello React !

Importance of 2nd Parameter in createElement()

- Second parameter basically is an object of key value pairs that will be applied to the element
- For example let's say we need an **ID attribute** on this **div tag** we can specify an object key is ID and the value is hello

The image shows two side-by-side screenshots. On the left is a code editor window for a file named 'Hello.js'. The code contains a React component 'Hello' that returns a single 'div' element with an 'id' attribute set to 'helloDiv'. The code is as follows:

```
1 import React from 'react';
2
3 const Hello = () => {
4     // return (
5     //   // <div>
6     //   //   <h1>Hello React !</h1>
7     //   // </div>
8     // )
9
10    return React.createElement(
11        'div',
12        {id: 'helloDiv'},
13        React.createElement('h1', null, 'Hello React !'));
14 }
15
16 export default Hello;
```

A red arrow points from the line '12 {id: 'helloDiv'}' in the code editor to the 'id="helloDiv"' attribute in the browser's developer tools screenshot on the right.

The right screenshot shows the browser's developer tools open to the 'Elements' tab. It displays the generated HTML code. A red box highlights the 'id="helloDiv"' attribute on the first 'div' element. An arrow points from this highlighted attribute to the corresponding line in the 'Hello.js' code editor.

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">
      <div class="App">
        ... <div id="helloDiv"> == $0 <div>
          <h1>Hello React !</h1>
        </div>
      </div>
    </div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

```
JS App.js M JS Hello.js U X JS Welcome.js U JS Greet.js U
my-app > src > components > JS Hello.js > [e] default
1  import React from 'react';
2
3  const Hello = () => {
4      // return (
5      //     // <div>
6      //     //     <h1>Hello React !</h1>
7      //     // </div>
8      // )
9
10  return React.createElement(
11      'div',
12      {id: 'helloDiv' , class: 'dummy'},
13      React.createElement('h1', null, 'Hello React !'));
14}
15
16 export default Hello;
```

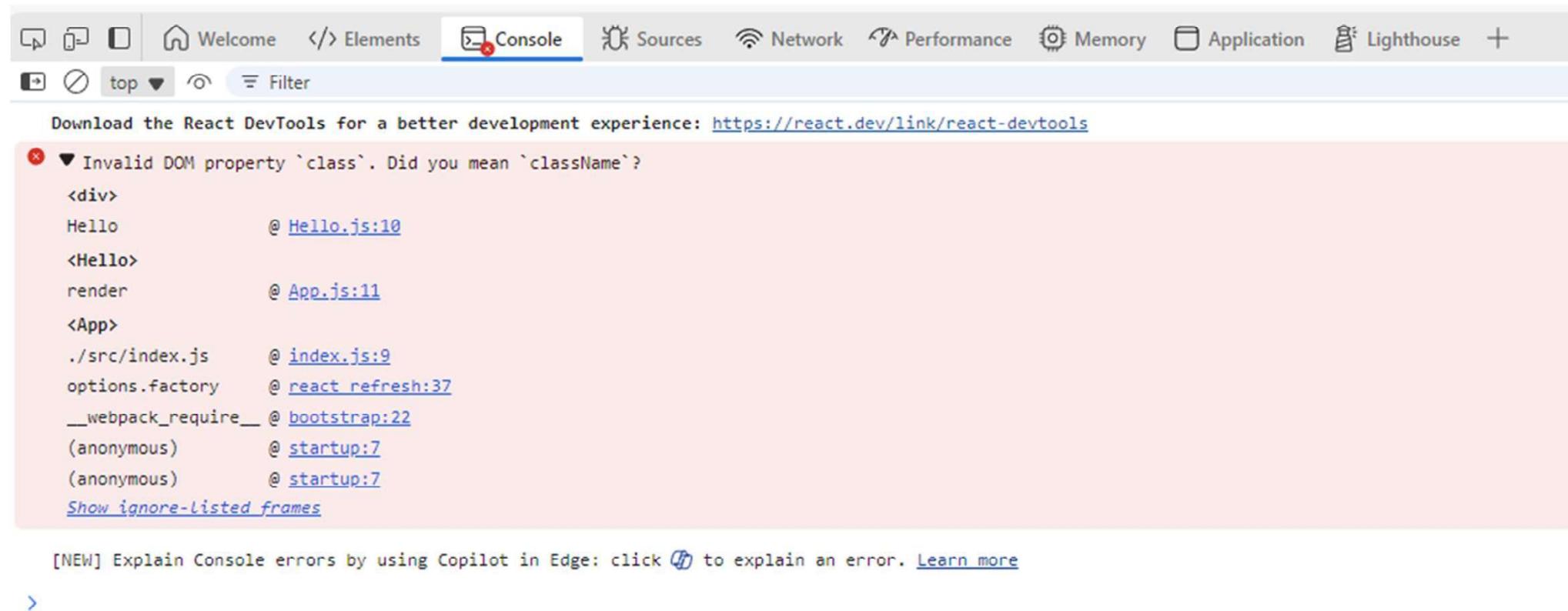
```
<!DOCTYPE html>
<html lang="en">
  <head> ... </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">
      <div class="App">
        <div id="helloDiv" class="dummy"> == $0 ⓘ
          <h1>Hello React !</h1>
        </div>
      </div>
    </div>
    <!--
        This HTML file is a template.
        If you open it directly in the browser, you will see an empty page.

        You can add webfonts, meta tags, or analytics to this file.
        The build step will place the bundled scripts into the <body> tag.

        To begin the development, run `npm start` or `yarn start`.
        To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

Error in Console

- In JS class is a Keyword, So Ended up with this.



The screenshot shows the Microsoft Edge DevTools interface with the 'Console' tab selected. A red error message is displayed:

```
Download the React DevTools for a better development experience: https://react.dev/link/react-devtools
```

x Invalid DOM property `class`. Did you mean `className`?

```
<div>
  Hello          @ Hello.js:10
<Hello>
  render        @ App.js:11
<App>
  ./src/index.js    @ index.js:9
  options.factory   @ react-refresh:37
  __webpack_require__ @ bootstrap:22
  (anonymous)       @ startup:7
  (anonymous)       @ startup:7
  Show ignore-listed frames
```

[NEW] Explain Console errors by using Copilot in Edge: click  to explain an error. [Learn more](#)

JSX is Better Than Normal createElement()

- Our example just has two elements but imagine a component with ten or a hundred elements it starts to become really clumsy
- JSX on the other hand will keep the code elegant simple and readable

JSX Differences

- Follow the Link for More
- <https://github.com/facebook/react/issues/13525>

Class -> className

for -> htmlFor

camelCase property naming convention

- onclick -> onClick
- tabindex -> tabIndex

Summary

In this lesson, you should have learned how to:

- Creating Components
- Basic Component
- Nesting Components
- Props

