

JSX

.Lect 3

1

Objectives

- Review the DOM and the virtual DOM
- Introduce JSX
- Understand basic syntax for JSX
- Introduce the transpiler

2

1

What is React

- JSX is a JavaScript extension.
- JSX makes creating React elements much easier, faster and more compact.
- It also makes the reading of React elements much easier.
- JSX code is converted(transpiled) into JavaScript.
- React components can be written without JSX, but not recommended.

3

Rendering Elements

- ReactDOM takes care of rendering of React components in web browser.

```
my-app > src > js index.js
1 import React from 'react'
2 import ReactDOM from 'react-dom'
3 import App from './App'
4
5 // version 17
6 ReactDOM.render( <App /> , document.getElementById('root'))
7
```

1 Issue:  1
[webpack-dev-server] Server started: Hot Module index.js:551
Replacement enabled, Live Reloading enabled, Progress disabled,
Overlay enabled.
[react-dom.development.js:29840](#)
Download the React DevTools for a better development experience: <https://reactjs.org/link/react-devtools>
✖ Warning: ReactDOM.render is no longer supported in React 18. Use createRoot instead. Until you switch to the new API, your app will behave as if it's running React 17. Learn more: <https://reactjs.org/link/switch-to-createRoot>

Warning with
React version 17

4

React version 18

```
my-app > src > JS index.js
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App'
4
5 ReactDOM.createRoot(document.getElementById('root')).render(<App />
6 )
```

```
my-app > {} package.json > ...
4 "private": true,
5 "dependencies": {
6   "@testing-library/jest-dom": "^5.16.5",
7   "@testing-library/react": "^13.4.0",
8   "@testing-library/user-event": "^13.5.0",
9   "react": "^18.2.0",
10  "react-dom": "^18.2.0",
11  "react-scripts": "5.0.1",
12  "web-vitals": "^2.1.4"
13 }
```

[webpack-dev-server] Server started: Hot Module Replacement enabled, Live Reloading enabled, Progress disabled, Overlay enabled.
react-dom.development.js:29840
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
React test msg
App.js:6

5

- The **ReactDOM** component makes it possible for the user interfaces to handle different screen changes required by modern web applications.
- It does this with the help of the Virtual DOM.

6

- The Document Object Model (**DOM**) is an object-oriented representation of an HTML or XML document.
- The structure of an HTML / XML document is hierarchical, so the DOM structure resembles that of a tree

7

- The DOM models a document as a set of nodes.
- The DOM is not a programming language
- Several languages have bindings to the DOM
 - JavaScript, PHP, JQuery
- Changes to the DOM cause changes in the web browser



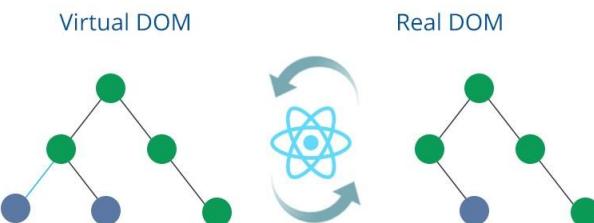
8

Virtual DOM

- DOM manipulation is slow
- DOM changes have to be checked on all levels of the DOM tree to see if the page need refreshed.
- React, takes away the details of how and when the DOM is modified from programmers.
- They created a layer between the code that the programmer writes and the DOM
- This layer is the Virtual DOM.

9

- ReactJS does not update the original DOM directly.
- In ReactJS, for every DOM object, there will be a corresponding in-memory copy created.
- This copy is called the Virtual DOM.



10

JSX

- HTML/XML-like syntax for composing React components.
- Extension of JavaScript that's used as a visual aid

```
my-💻 > src > js index.js
1   import React from 'react'
2   import ReactDOM from 'react-dom'
3   import App from './App'
4
5   ReactDOM.render( <div>
6     <h1>Hello World</h1>
7     <h2>welcome back</h2>
8   </div> , document.getElementById('root'))
9 
```



11

JSX Benefits

- Code is easier to read and maintain.
- React assumes that you use JSX and reports helpful error messages as if you are.
- Faster code—the transpiler optimises the code on the fly.
- Casual developers (e.g., designers) can modify code more easily because JSX looks like HTML.
- Less code, less errors.

12

Without JSX

```
React.createElement(  
  'h1',  
  null,  
  'Hello World',  
);
```

With JSX

```
<h1>  
Welcome  
</h1>
```

13

```
React.createElement(  
  Title,  
  {size: 8},  
  'Welcome to ',  
  React.createElement(  
    'strong',  
    null,  
    'year four',  
  ),  
);
```

```
<Title size="8">  
Welcome to  
<strong>year four</strong>  
</Title>
```

JSX looks like HTML (easier to read)

14

JSX

- XML-based syntax extension to JavaScript.
- Integral part of how React components are developed.
- React uses JSX elements to represent custom components.

```
my-app > src > JS index.js
1 import React from 'react'
2 import ReactDOM from 'react-dom'
3 import App from './App'
4
5 ReactDOM.render( <div>
6   <h1>Hello World</h1>
7   <h2>welcome back</h2>
8 </div> , document.getElementById('root'))
9
```



JSX
This is
not
HTML

15

- **import** and **export** should always be at the top level of your JavaScript file.
- The **import(s)** statements should be at the top of the module..

```
JS tester.js > ...
1 import * as myLib from './modtst1.js'
2 import { bigSubtraction, greeting } from './modtst2.js'
3
4 // Example usage: console.log(greeting());
```

16

- React has components built into it for HTML5 elements.

```
function Links() {  
    return (  
        <div>  
            <a href="//atu.ie">Donegal ATU</a>  
        </div>  
    );  
}  
  
export default Links;
```

17

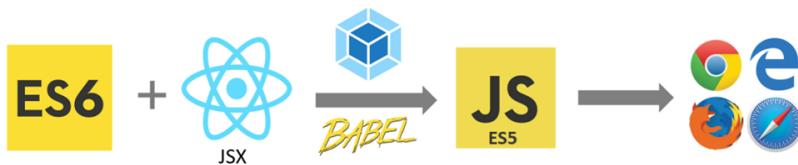
- Creating React components without using JSX (not recommended).

```
1 "use strict";  
2  
3 /*#__PURE__*/  
4 React.createElement("div", null, /*#__PURE__*/React.createElement("h1", {  
5     className: "head"  
6 }, "Head section"), /*#__PURE__*/React.createElement("h3", null, "Menu section"));
```

```
<div>  
    [<h1 className="head">Head section</h1>  
     <h3>Menu section</h3>  
    </div>
```

18

- Reverse strategy on **separation of concerns**, seeks to keep related items together.
- JSX is compiled by various transformers into standard ECMAScript.



19

Separation of concerns

- Separate logic from presentation.
 - Logic mainly outside the return statement



Traditional SoC



Component SoC

20

React transpiler

- Compilation of applications is different to other languages (like C++ or Java).
- With Java/C#/C++ the code is converted into low-level code that can be understood by the computer's software interpreter. (Java bytecode).
- **With React applications are compiled to another version of JavaScript.**
- Process called **transpilation**.
- One popular tool used for transpilation in JavaScript is called **Babel**.

21

Babeljs.io

- Built into React App
- Babel converts JSX into JavaScript that can be

The screenshot shows the Babeljs.io homepage. At the top, there is a navigation bar with links like 'Docs', 'Setup', 'Try it out', and 'Videos'. Below the navigation, a large 'BABEL' logo is displayed. The main headline reads 'Babel is a JavaScript compiler' in yellow text. Below the headline, a subtext says 'Use next generation JavaScript, today.' A yellow button at the bottom left of the main area says 'Babel 7.25 is released! Please read our blog post for highlights and changelog for more details!' In the bottom right corner of the main area, there is a small note: 'Get browser-compatible JavaScript out' followed by a snippet of code: `let yourTurn = "Type some code in here!"`. On the left side, there is a section titled 'Put in next-gen JavaScript' with a snippet of code: `let yourTurn = "Type some code in here!"`.

22

```
1 ReactDOM.createRoot(document.getElementById('root')).render(<div>
2   <App /></div>)
3 
```

```
1 ReactDOM.createRoot(document.getElementById('root')).render(<div>
2   <App /></div>)
3 
```

```
1 import { jsx as _jsx } from "react/jsx-runtime";
2 ReactDOM.createRoot(document.getElementById('root')).render(/*#__PURE__*/
3   _jsx("div", {
4     children: /*#__PURE__*/_jsx(App, {})
5   }));
6 
```

23

- When your browser executes a React application, it will only see the `React.createElement` statements required to generate the required structure.
- JSX needs to be **transpiled** into regular JavaScript before browsers can execute the code.

24

JSX syntax

- JSX similar to XML
 - ▣ All elements must be closed.
 - ▣ Elements that cannot have child nodes (so-called “empty” elements) must be closed with a slash.
 - ▣ Attributes that are strings must have quotes around them.
 - ▣ HTML elements in JSX must be written in all lowercase letters.
- Since JSX is closer to JavaScript than to HTML, React DOM uses camelCase property naming convention instead of HTML attribute names.
- React component can only return one thing

25

JSX tags may contain children:

```
my-app > src > JS App.js > App  
1  function App() {  
2    return <div>  
3      <h2 className="tst">test image</h2>  
4        
5    </div>  
6 }
```

Attribute names that contain more
than one word are camel-cased

26

Comments

- HTML comments don't work.
- However, you can put comment inside curly braces to avoid transpilation.

```
return (<>
  /* JSX comment */
```

27

- To include a variable or JavaScript in JSX.

□ Use curly braces

- Doesn't get interpreted by the transpiler.

```
const App = () => {
  console.log("React test msg");
  let today = new Date();
  let myImg = "tst.png"
  let favouriteNum = Math.floor(Math.random() * 100) + 1
  return (<>
    <h2>test2 arrow! on {today.toString()}</h2>
    <h3>my favourite number = {favouriteNum}</h3>
    <img src={myImg} />
  </>)
}
```

28

JSX variable example

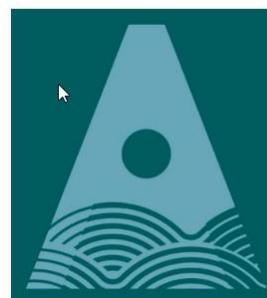
- Curly brace { } notation to output variables

```
function App() {
  const tomorrow = new Date().toDateString()
  return (
    <div className="App">
      <p>Hello <b>World!</b> {tomorrow}</p>
      <Links />
      <Links />
    </div>
  );
}
```

29

Double braces for Objects

```
return (<>
  <h2>test2 arrow! on {today.toString()}</h2>
  <h3 style={{border:"solid", width:"200px", color:"red"}}>my favour
  <img src={myImg} />
</>)
  my favourite number =
  16
```



hello

30

Expressions

- Use any JavaScript expression inside JSX
- Insert into a React element attribute values by surrounding it with curly braces.
- Expression is any valid unit of code that resolves to a value.
- Make sure you understand the difference between expressions and statements.
- We can have statement but not the return

31

conditional statement

```
let today = new Date().getDay()
let dayOfWeek = ""

if (today === 0) {
  dayOfWeek = "Sunday"
} else {
  dayOfWeek = "Other day"
}

return (<>
  <Heading />
  <h2>test2 {dayOfWeek}</h2>
  <h3 style={{border:"solid", width:"200px", color:"red"}}>my favourite number = 64</h3>
)
```

Header section

Menu section

Head section

Menu section

test2 Sunday

my favourite number = 64

32

□ Ternary condition

```
let today = new Date().getDay()
let dayOfWeek = ""

return (<>
  <Heading />
  <h2>test2 { today === 0 ? "Sunday" : "other day" }</h2>
  <h3 style={{border:"solid", width:"200px", color:"red"}}>m
    <img src={myImg} />
  </h3>
</>
)}
```

33

Summary

- Reviewed the role of the virtual DOM
- JSX is an important tool used in the development of nearly every React component.
- JSX is not HTML
- JSX code with angle brackets (<>) is easier to read than code with a lot of `React.createElement()` statements
- Add JavaScript into JSX with curly braces

34

Webography

□ Web Links

- <https://www.w3schools.com/nodejs/>
- https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm