SNS COLLEGE OF ENGINEERING

An Autonomous Institution

Coimbatore-107

19TS601-FULL STACK DEVELOPMENT

UNIT-2

REACT

1. React Introduction - React ES6

React Introduction

- React is a JavaScript library for building user interfaces.
- React is used to build single-page applications.
- React allows us to create reusable UI components.
- React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.
- React is a tool for building UI components.

React History

- Current version of React.JS is V18.2.0 (June 14, 2022).
- Initial Release to the Public (V0.3.0) was in July 2013.
- React.JS was first used in 2011 for Facebook's Newsfeed feature.
- Facebook Software Engineer, Jordan Walke, created it.
- Current version of create-react-app is v5.0.1 (April 2022).
- create-react-app includes built tools such as webpack, Babel, and ESLint.

How does React Work?

- React creates a VIRTUAL DOM in memory.
- Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.
- React only changes what needs to be changed!
- React finds out what changes have been made, and changes only what needs to be changed.

React-CDN method

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>

    <div id="mydiv"></div>

    <script type="text/babel">
      function Hello() {
        return <h1>Hello World!</h1>;
      }

      ReactDOM.render(<Hello />, document.getElementById('mydiv'))
    </script>

  </body>
</html>
```

React-CDN method

Hello World!

The first two CDN link make us write React code in our JavaScripts, and the third, Babel, allows us to write JSX syntax and ES6 in older browsers.

React ES6

- ES6 stands for ECMAScript 6.
- ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015.
- React uses ES6, and you should be familiar with some of the new features like:

- Classes
- Arrow Functions
- Variables (let, const, var)
- Array Methods like .map()
- Destructuring
- Modules
- Ternary Operator
- Spread Operator

- Classes
- ES6 introduced classes.
- A class is a type of function, but instead of using the keyword function to initiate it, we use the keyword class, and the properties are assigned inside a constructor() method.
- The first letter of class name starts with uppercase.
- The constructor function is called automatically when the object is initialized.

Example-class and object

```
<!DOCTYPE html>
<html>
<body>
<script>
class Car {
  constructor(name) {
    this.brand = name;
  }
}
const mycar = new Car("Ford");
document.write(mycar.brand);
</script>
</body>
</html>
```

Ford

Method in Classes

- Method name can be declared as user defined name inside the class.
-

```
<!DOCTYPE html>
<html>
<body>
<script>
class Car {
  constructor(name) {
    this.brand = name;
  }

  present() {
    return 'I have a ' + this.brand;
  }
}
const mycar = new Car("Ford");
document.write(mycar.present());
</script>
</body>
</html>
```

I have a Ford

- As you can see in the example above, you call the method by referring to the object's method name followed by parentheses.

Class Inheritance

- To create a class inheritance, use the extends keyword.
- A class created with a class inheritance inherits all the methods from another class.

```
<!DOCTYPE html>
<html>

<body>

<script>
class Car {
  constructor(name) {
    this.brand = name;
  }

  present() {
    return 'I have a ' + this.brand;
  }
}

class Model extends Car {
  constructor(name, mod) {
    super(name);
    this.model = mod;
  }
  show() {
    return this.present() + ', it is a ' + this.model
  }
}

const mycar = new Model("Ford", "Mustang");
document.write(mycar.show());
</script>

</body>
</html>
```

I have a Ford, it is a Mustang

The super() method refers to the parent class.

By calling the super() method in the constructor method, we call the parent's constructor method and get access to the parent's properties and methods.

Arrow Functions

- Arrow functions allow us to write shorter function syntax

Without Arrow Function

```
hello = function() {
  return "Hello World!";
}
```

With Arrow Function

```
hello = () => {
  return "Hello World!";
}
```

```
<html>
<body>
<h1>Arrow Function</h1>
<p>A demonstration of a simple arrow function.</p>
<p id="demo"></p>
<script>
hello = () => {
  return "Hello World!";
}
document.getElementById("demo").innerHTML = hello();
</script>
</body>
</html>
```

- This works only if the function has only one statement.

Arrow Functions Return Value by Default:

```
hello = () => "Hello World!";
```

Arrow Function With Parameters:

```
hello = (val) => "Hello " + val;
```

Arrow Function Without Parentheses:

```
hello = val => "Hello " + val;
```

if you have only one parameter, you can skip the parentheses as well

this keyword

- In regular functions the this keyword represented the object that called the function, which could be the window, the document, a button or whatever.
- With arrow functions, the this keyword always represents the object that defined the arrow function.

React ES6 Variables

- Before ES6 there was only one way of defining your variables: with the var keyword.
- With ES6, there are three ways of defining your variables: var, let, and const.
- If you use var outside of a function, it belongs to the global scope.
- If you use var inside of a function, it belongs to that function.
- If you use var inside of a block, i.e. a for loop, the variable is still available outside of that block.

- var has a function scope, not a block scope.
- let is the block scoped version of var, and is limited to the block (or expression) where it is defined.
- If you use let inside of a block, i.e. a for loop, the variable is only available inside of that loop.
- let has a block scope.

- const is a variable that once it has been created, its value can never change.
- const has a block scope.
- It does not define a constant value. It defines a constant reference to a value.
- Because of this you can NOT:
  - Reassign a constant value
  - Reassign a constant array
  - Reassign a constant object

- But you CAN:
  - Change the elements of constant array
  - Change the properties of constant object

```
var x = 5.6;
let x = 5.6;
const x = 5.6;
```

React ES6 Array Methods

- There are many JavaScript array methods.
- One of the most useful in React is the .map() array method.
- The .map() method allows you to run a function on each item in the array, returning a new array as the result.
- In React, map() can be used to generate lists.

```
import React from 'react';
import ReactDOM from 'react-dom/client';

const myArray = ['apple', 'banana', 'orange'];

const myList = myArray.map((item) => <p>{item}</p>)

ReactDOM.render(myList, document.getElementById('root'));
```

apple

banana

orange

React ES6 Destructuring

- Destructuring makes it easy to extract only what is needed.
- Destructuring means to break down a complex structure into simpler parts. With the syntax of destructuring, you can extract smaller fragments from objects and arrays. It can be used for assignments and declaration of a variable.

```
const vehicles = ['mustang', 'f-150', 'expedition'];

const [car, truck, suv] = vehicles;
```

When destructuring arrays, the order that variables are declared is important.

```
<html>

<body>

<script>
function calculate(a, b) {
  const add = a + b;
  const subtract = a - b;
  const multiply = a * b;
  const divide = a / b;

  return [add, subtract, multiply, divide];
}

const [add, subtract, multiply, divide] = calculate(4, 7);

document.write("<p>Sum: " + add + "</p>");
document.write("<p>Difference " + subtract + "</p>");
document.write("<p>Product: " + multiply + "</p>");
document.write("<p>Quotient " + divide + "</p>");
</script>

</body>
</html>
```

Sum: 11

Difference -3

Product: 28

Quotient 0.5714285714285714

React ES6 Spread Operator

- The JavaScript spread operator (...) allows us to quickly copy all or part of an existing array or object into another array or object.
- The spread operator is often used in combination with destructuring.

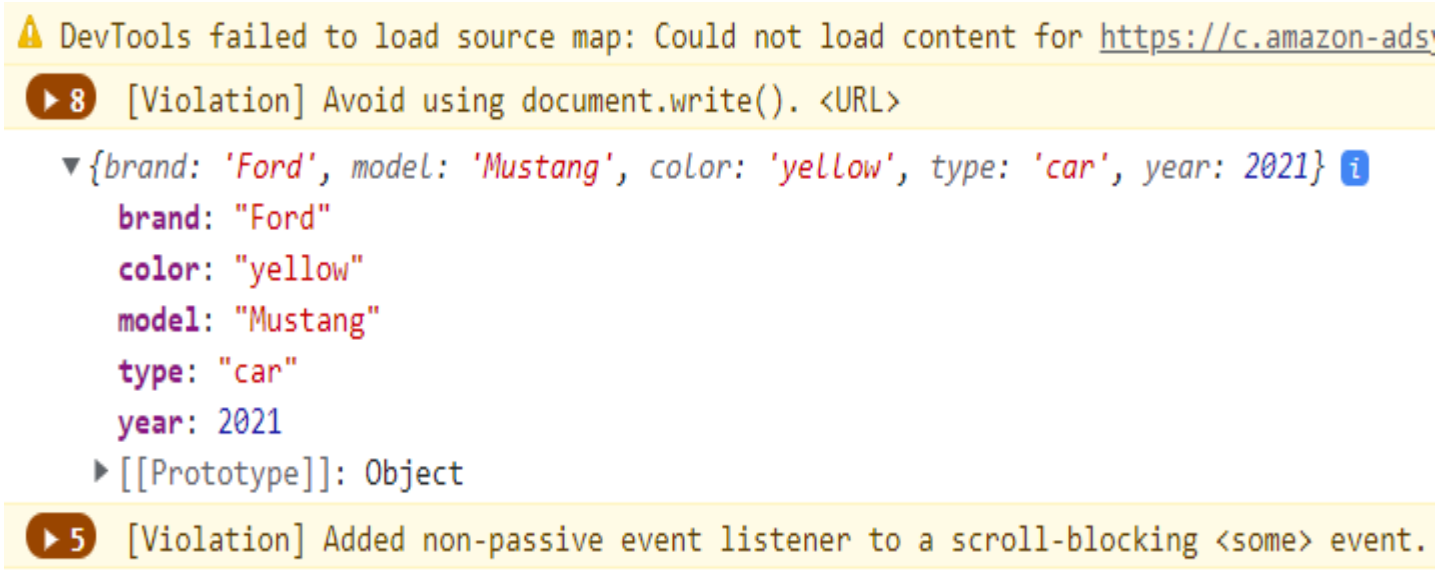
```
<html>

<body>

<script>
const numbersOne = [1, 2, 3];
const numbersTwo = [4, 5, 6];
const numbersCombined = [...numbersOne, ...numbersTwo];

document.write(numbersCombined);
</script>

</body>
</html>
```

1,2,3,4,5,6

Assign the first and second items from `numbers` to variables and put the rest in an array:

```
const numbers = [1, 2, 3, 4, 5, 6];

const [one, two, ...rest] = numbers;
```

```
<html>

<body>

<script>
const myVehicle = {
  brand: 'Ford',
  model: 'Mustang',
  color: 'red'
}

const updateMyVehicle = {
  type: 'car',
  year: 2021,
  color: 'yellow'
}

const myUpdatedVehicle = {...myVehicle, ...updateMyVehicle}

//Check the result object in the console:
console.log(myUpdatedVehicle);
</script>

<p>Press F12 and see the result object in the console view.</p>

</body>
</html>
```



Notice the properties that did not match were combined, but the property that did match, color, was overwritten by the last object that was passed, updateMyVehicle. The resulting color is now yellow.

React ES6 Modules

- JavaScript modules allow you to break up your code into separate files.
- This makes it easier to maintain the code-base.
- ES Modules rely on the import and export statements.
- Refer this concept in UNIT-1 under javascript

React ES6 Ternary Operator

- The ternary operator is a simplified conditional operator like if / else.
- Syntax: condition ? <expression if true> : <expression if false>
- Here is an example using if / else:

```
if (authenticated) {
  renderApp();
} else {
  renderLogin();
}
```

Example

With Ternary

```
authenticated ? renderApp() : renderLogin();
```

```
<html>

<body>

<h1 id="demo"></h1>

<script>
function renderApp() {
  document.getElementById("demo").innerHTML = "Welcome!";
}

function renderLogin() {
  document.getElementById("demo").innerHTML = "Please log in";
}

let authenticated = true;

authenticated ? renderApp() : renderLogin();

</script>

<p>Try changing the "authenticated" variable to false, and run the code to see what
happens.</p>
</body>
</html>
```

Welcome!

Try changing the "authenticated" variable to false, and run the code to see what happens.

Thank You