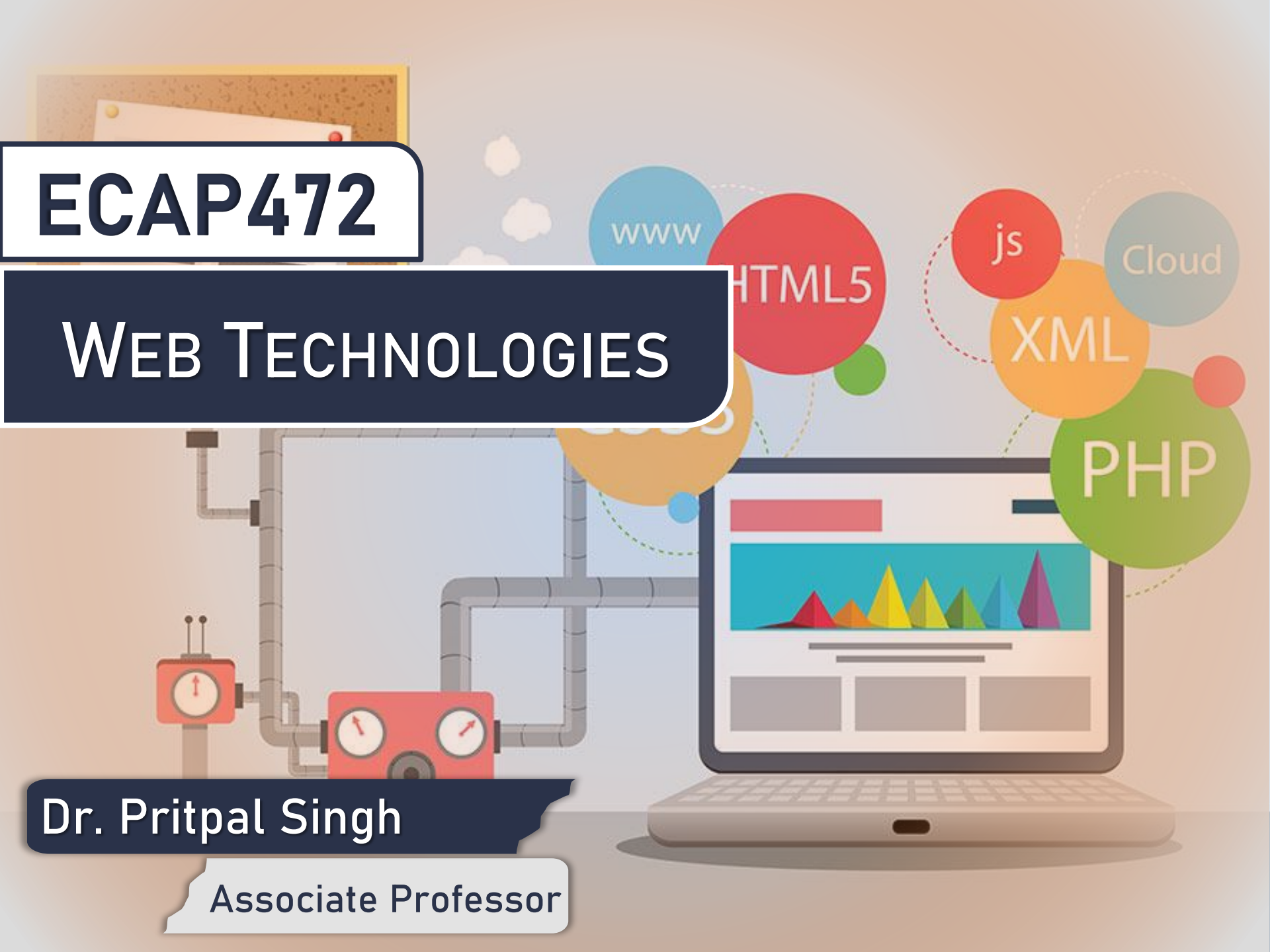
ECAP472

Web Technologies



Dr. Pritpal Singh

Associate Professor

Learning Outcomes

After this lecture, you will be able to

- Understand concepts of ES6 (ECMAScript 6).
- Go over concept of variables, classes and arrow function.

What is 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.
- ECMAScript (European Computer Manufacturers Association Script) is a scripting language based on JavaScript. Invented by Brendan Eich at Netscape, ECMAScript made its first appearance in the Navigator 2.0 browser

What is ES6?

- ES6 or ECMAScript 6 is a scripting language specification which is standardized by ECMAScript International. This specification governs some languages such as JavaScript, ActionScript, and Jscript. ECMAScript is generally used for client-side scripting, and it is also used for writing server applications and services by using Node.js.

What is ES6?

- ES6 allows you to write the code in such a way that makes your code more modern and readable. By using ES6 features, we write less and do more, so the term 'Write less, do more' suits ES6. ES6 introduces you many great features such as scope variable, arrow functions, template strings, class destructions, modules, etc

React ES6 Classes

- Classes are an essential part of object-oriented programming (OOP). Classes are used to define the blueprint for real-world object modeling and organize the code into reusable and logical parts.
- Before ES6, it was hard to create a class in JavaScript. But in ES6, we can create the class by using the class keyword. We can include classes in our code either by class expression or by using a class declaration.

React ES6 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.

Example

A simple class constructor:

```
class Car
{
  constructor(name) {
    this.brand = name;
  }
}
```

Create an object

Create an object called "mycar" based on the Car class:

```
class Car {
  constructor(name) {
    this.brand = name;
  }
}
const mycar = new Car("Ford");
```

Arrow functions

Arrow functions are introduced in ES6

which provides you a more accurate way to write the functions in JavaScript. They allow us to write smaller function syntax. Arrow functions make your code more readable and structured.

Arrow functions are anonymous functions

(the functions without a name and not bound with an identifier). They don't return any value and can declare without the function keyword. Arrow functions cannot be used as the constructors.

React ES6 Arrow Functions

Arrow functions allow us to write shorter function syntax:

Before:

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

With Arrow Function:

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

- It gets shorter! If the function has only one statement, and the statement returns a value, you can remove the brackets and the return keyword.
- Arrow Functions Return Value by Default:

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

Arrow Function With Parameters:

- If you have parameters, you pass them inside the parentheses:
- Arrow Function With Parameters:
- hello = (val) => "Hello " + val;

Advantages of Arrow Function

- It reduces code size.
- The return statement is optional for a single line function.
- Lexically bind the context.
- Functional braces are optional for a single-line statement.

What About this?

- The handling of this is also different in arrow functions compared to regular functions.
- In short, with arrow functions there are no binding of this.
- 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 were only one way of defining your variables: with the var keyword. If you did not define them, they would be assigned to the global object. Unless you were in strict mode, then you would get an error if your variables were undefined.
- Now, with ES6, there are three ways of defining your variables: var, let, and const.

var has a function scope, not a block scope.

- var
- var x = 5.6;
- 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.

let

- let x = 5.6;
- 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

- const x = 5.6;
- const is a variable that once it has been created, its value can never change.
- const has a block scope.

The keyword const is a bit misleading.

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

The keyword const is a bit misleading.

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

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.

Example

- Generate a list of items from an array:
- const myArray = ['apple', 'banana', 'orange'];
- const myList = myArray.map((item) => {item})

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.

Export

- You can export a function or variable from any file.
- Let us create a file named person.js, and fill it with the things we want to export.
- There are two types of exports:

Export

- You can export a function or variable from any file.
- Suppose to create a file named person.js, and fill it with the things we want to export.
- There are two types of exports:

Export

- You can export a function or variable from any file.
- Let us create a file named person.js, and fill it with the things we want to export.
- There are two types of exports:

Named | Default

Named Exports

You can create named exports two ways.

1. In-line individually,
2. or all at once at the bottom.

In-line individually:

person.js

```
export const name = "Jesse"

export const age = 40
```

All at once at the bottom:

person.js

```
const name = "Jesse"

const age = 40

export { name, age }
```

Import

- You can import modules into a file in two ways, based on if they are named exports or default exports.
- Named exports must be destructured using curly braces. Default exports do not.

Import from named exports

Import named exports from the file person.js:

```
import { name, age } from "./person.js";
```

Import from default exports

Import a default export from the file message.js:

```
import message from "./message.js";
```

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

Before:

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

Here is the same example using a ternary operator:

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

Vs Code Extension

- Vs Code Extension to make our React life Easy
- VS Code JavaScript (ES6) snippets
- ES7 React/Redux/GraphQL/React-Native snippets

ADVANTAGES OF REACTJS

Intuitive

ReactJS is extremely intuitive to work with and provides interactivity to the layout of any UI. Plus, it enables fast and quality assured application development that in turn saves time for both - clients and developers.

ADVANTAGES OF REACTJS

Declarative

ReactJS enables significant data changes that result in automatic alteration in the selected parts of user interfaces. Owing to this progressive functionality, there is no additional function that you need to perform to update your user interface.

Provides Reusable Components

- ReactJS provides reusable components that developers have the authority to reuse and create a new application . Reusability is exactly like a remedy for developers. This platform gives the developers the authority to reuse the components build for some other application having the same functionality. Thereby, reducing the development effort and ensuring a flawless performance.

Components Support

- ReactJS is a perfect combination of JavaScript and HTML tags. The usage of the HTML tags and JS codes, make it easy to deal with a vast set of data containing the document object model. During this time, ReactJS works as a mediator which represents the DOM and assists to decide which component needs changes to get the exact results

SEO-friendly

- React JS was introduced after immense research and improvements by Facebook. Naturally, it stands out from the crowd and allows developers to build amazing, SEO-friendly user interfaces across browsers and engines

SOME ADDITIONAL ADVANTAGES OF REACTJS:

Makes JavaScript coding easier

Extremely competent

Excellent cross-platform support

Handles dependencies

Template designing made easy

Provides amazing developer tools

UI focused designs

Easy to adopt

Practical

That’s all for now…