Skill Training
Advanced JavaScript

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com

Schedule for Advanced CSS & JS

Day1 : Advanced JS : ES6, Arrow Functions,…

Day2 : Advanced JS : OOPs, Modules, Closures

Day3 : Advanced JS : Asynchronous, Promises,…

Day4 : Advanced CSS : CSS3 Layouts, Media Queries

Day5 : Advanced CSS: UI Frameworks

Advanced JavaScript
OOPS, Modules, Closures

Narasimha
Sr. Corporate Trainer, Mentor
tnrao.trainer@gmail.com

Index – Day2

Object Oriented Programming

Narasimha

Sr. IT Trainer/Consultant

Creating Class

- Class can be created using class keyword
- Class contains collection of props and methods
- Properties holds the data of the object
- Methods are used to perform the operations on the data
- Members of the class can be access
  - ✓ through object outside the class
  - ✓ Using **this** keyword inside the class

```
class  Employee
{
            // props + methods
}
```

Working with Constructor

```
class Student
{
            constructor(id, name)
           {
                      this .sid  = id;
                      this .sname  = name;
           }
}

var obj = new Student(2566, "Smith");
```

Note :   class may only have one constructor

Inheritance

Narasimha

Sr. IT Trainer/Consultant

Inheritance

```
class Person
{
        // members of super class
}


class Student extends Person
{
        // members  of sub class

}
```

Prototype Chain

Narasimha

Sr. IT Trainer/Consultant

What is proptype chain?

- Every object in JavaScript has a built-in property, which is called its **prototype**.
- The prototype is itself an **object**, so the prototype will have its own prototype, making what's called a prototype chain.
- The chain ends when we reach a prototype that has **null** for its own prototype.

Working with Prototype details

- To get Prototype Details

```
let  object = Object.getPrototypeOf(obj);
```

- The JavaScript prototype property also allows you to add new methods to the class:

```
Employee.prototype.prop = value;
```

- Access protype details using object:

```
employeObj.__proto__
```

Closures

Narasimha

Sr. IT Trainer/Consultant

Closures

- A closure is the combination of a function bundled together with references to its surrounding state.
- In other words, a closure gives you access to an outer function's scope from an inner function.
- In JavaScript, closures are created every time a function is created, at function creation time.
- Global variables can be made local (private) with closures.

Closures

```
const add = (function () {
  let counter = 0;
  return function () {  counter += 1;  return counter; }
})();



add();
add();
add();
```

JavaScript Modules

Narasimha

Sr. IT Trainer/Consultant

JavaScript Modules

- JavaScript modules allow you to break up your code into separate files.
- This makes it easier to maintain a code-base.
- Modules also rely on **type="module"** in the <script> tag.
- Modules are imported from external files with the import statement.

JavaScript Modules

```
const empData = () => {
        ……
};

export default empData;
```

```
<script type="module">
        import message from "./empData.js";
</script>
```

Export

- The **export** declaration is used to export values from a JavaScript module.

- Exported values can then be imported into other programs with the import declaration.

- There are two types of exports: **Named** Exports and **Default** Exports.

Export

- Default
  - export default empData;

- Named
  - export const empData = () => { …. }

Import

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

-
  ```
  import empInfo from "./emp_data.js";
  import {empData} from "./emp_data.js";
  ```

Practice Hands-Ons

Narasimha
Sr. Corporate Trainer, Mentor
tnrao.trainer@gmail.com

Q & A

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com