JAVASCRIPT OOPS CONCEPTS

1. JavaScript static Method

- The JavaScript provides static methods that belong to the class instead of an instance of that class. So, an instance is not required to call the static method. These methods are called directly on the class itself.

Important points in static methods:

- The static keyword is used to declare a static method.
- The static method can be of any name.
- A class can contain more than one static method.
- If we declare more than one static method with a similar name, the JavaScript always invokes the last one.

Example1

```
<!DOCTYPE html>

<html>

<body>

<script>

class Test

{

  static display()

  {

    return "static method "

  }
}
document.writeln(Test.display());
</script>
</body>
</html>
```

Example 2

Invoking more than one static method.

```
<!DOCTYPE html>
<html>
<body>
<script>
class Test
{
  static display1()
  {
    return "static method 1"
  }
  static display2()
  {
    return "static method 2"
  }
}
document.writeln(Test.display1()+"<br>");
document.writeln(Test.display2());
</script>
</body>
</html>
```

Example3

Invoke more than one static method with similar names.

```
<!DOCTYPE html>
<html>
<body>
<script>
class Test
{
  static display()
  {
    return "static method is invoked"
  }
  static display()
  {
    return "static method is invoked again"
  }
}
document.writeln(Test.display());
</script>
</body>
</html>
```

2.JavaScript Encapsulation

- The JavaScript Encapsulation is a process of binding the data (i.e. variables) with the functions acting on that data. It allows us to control the data and validate it. To achieve an encapsulation in JavaScript: -
- Use var keyword to make data members private.
- Use setter methods to set the data and getter methods to get that data.

  For example:

```
<!DOCTYPE html>
<html>
<body>
<script>
class Student
```

```
 {
   constructor()
   {
      var name;
      var marks;  //The JavaScript Encapsulation is a process   of
binding the data (i.e. variables) with the functions acting on that
data.
                    // It allows us to control the data and validate it.
                              //Use setter methods to set the data and getter
methods to get that data.
   }
       getName()
       {
         return this.name;
       }
     setName(name)
     {
       this.name=name;
     }

     getMarks()
```

```
      {
        return this.marks;
      }
    setMarks(marks)
    {
      this.marks=marks;
    }


    }
    var stud=new Student();
     stud.setName("sandeep");
     stud.setMarks(80);
     document.writeln(stud.getName()+" "+stud.getMarks());
</script>
</body>
</html>
```

3.JavaScript Inheritance

- The JavaScript inheritance is a mechanism that allows us to create new classes on the basis of already existing classes. It provides flexibility to the child class to reuse the methods and variables of a parent class.

- The JavaScript extends keyword is used to create a child class on the basis of a parent class. It facilitates child class to acquire all the properties and behavior of its parent class.

  Example

```
<!DOCTYPE html>

    <html>

    <body>

    <script>

    class CompanyName

    {

      constructor()

      {

          this.language = "Javascript";

      }

    }                                  //The super keyword is used to call the
    parent class constructor.

                                       //this keyword refers to the object it
    belongs to.

    class Employee extends CompanyName {

      constructor(id,name) {

       super();

        this.id=id;
        this.name=name;

      }

    }

    var emp = new Employee(1,"sandeep");

    document.write(emp.id+" "+emp.name+" "+emp.language);

    </script>

    </body>

    </html>
```

4.JavaScript Polymorphism

- The polymorphism is a core concept of an object-oriented paradigm that provides a way to perform a single action in different forms. It provides an ability to call the same method on different JavaScript objects.

  For example:

```
<!DOCTYPE html>

<html>

<body>

<script>

class A

  {

     display()//The polymorphism is a core concept of an object-
oriented
//that provides a way to perform a single action in different forms.

 //example where a child class object invokes the parent class
method.

    {

      document.writeln("A is invoked");

    }

  }

class B extends A

  {

  }

var b=new B();

b.display();

</script>

</body>

</html>
```