OOPS Concepts :

class: class is a collection of variables and methods.

Java :

```
class Arth{
          //variables
               int a=10;
               int b=20;
         // methods
               function sum(){
                   return a+b;
               }
               function sub(){
                  return a-b;
               }

       }
```

JavaScript:

```
function arth(){
    //variables
            this.a=10;
            this.b=20;
    //methods
            this.add=function(){
                    return this.a+this.b
            }

            this.add=function(){
                    return this.a-this.b
            }

    }
```

There is no use, by creating a class , if you wants to call the variables and methods , we should have to create an object.

Java : Arth a=new Arth();

```
a.sum();
a.sub();
```

JavaScript: var obj=new Arth();

```
obj.sum();
obj.sub();
```

At the time of creation of object, it perform initialization(allocate memory for variables and methods), by executing the constructor.

Constructor: The method name, whose name is class name, that method behaves like a constructor.

We have two types of constructors, default, and parametrized constructor.

Object : Object is a blue print of class. At the time of object creation, it perform initialization by executing the constructor, by allocating the memory for variables and methods.

* By using new operator we can create an object.

```
Java :      Arth a=new Arth();
```

* By using dot(.) operator we can call the variables and methods .

```
                    a.sum();
                    a.sub();
JavaScript: var obj=new Arth();
                      obj.sum();
                      obj.sub();
```

Inheritance: What ever the parent has, by default available to child.

Java:

```
class Parent {
          int a=10;
  }

 class Child{
          int b=20;
          function add(){
                  return a+b;
          }
  }

Child c1=new Child();
    c1.add();
```

// It gives an error because a is not defined in Child class, i.e. defined in Parent class. Now i wants to take the properties(variable and methods) of parent class into my child class.
i.e. possible by inheritance.

JavaScript:

```
function Arth(){
        this.b=20;
        this.add=function(){
                return this.a+this.b;
                }
            }

            var obj=new Arth();
                obj.add();
```

// It gives an error because a is not defined in Arth function, Now i wants to take the properties(variable) from out side into my Arth function.ie possible by inheritance.

In **Java** we used extends keyword to inherit the properties.

```
        class Parent {
                int a=10;
            }

          class Child extends Parent{
                int b=20;
                function add(){
                    return a+b;
                }
            }

        Child c1=new Child();
            c1.add(); // 30
```

In **JavaScript** we use prototype keyword for inheritance.

```
            function Arth(){
                this.b=20;
                this.add=function(){
                    return this.a+this.b;
                }
            }

            Arth.prototype.a=10;

            var obj=new Arth();
                obj.add(); //30
```

<u>**Overloading:**</u>

Method name is same, but signature is different.
Signature is nothing but , type of arguments, no of arguments, order of arguments.

Java :

```
                class A{
                    add(int a,int b){
                        return a+b;
                    }
```

```
                              add(int a,int b,int c){
                                      return a+b+c;
                              }

                      }
```

Create Object and call the methods:

```
                       A a1=new A();
                      a1.add(2,3);   // 5
                      a1.add(2,3,4); // 9
```

In java overloading is working fine.

JavaScript :

```
        function Arth(){
                this.add=function(a,b,c){
                        return a+b+c;
                }
                this.add=function(a,b){
                        return a+b
                }
        }

        var obj=new Arth();
                obj.add(2,3,5)  // 5
                obj.add(2,3);   // 5
```

If you observe the above two method calls ,we get the output is 5. Because, JavaScript didn't considered the arguments list, the last method is overrides all the above methods, which is having the same name.

So in JavaScript , overloading is not working.

Overriding : Method heading is same, but definition is different. Overriding is possible , only at the time of inheritance only.

What ever the parent has, by default i.e. available to the child , but the child is not satisfied with the parent definition, then the child can override parent definition,with his own.

Java :

```
                      class P{
                               function arth(int a, int b){

                                      return a+b;
                                      }
                              }

                      class C extends P{
```

```
                                        function arth(int a, int b){
                                                return a*b;
                                        }
                                }
```

In the above, the parent has the method “arth” with two arguments with the functionality addition. Now my child class C extends from P. So now P is parent class. Whatever the parent has by default available to Child. So the method “arth” with 2 arguments and add functionality is available to Child. Now Child C is don’t want that addition operation. So it overrides the addition functionality with multiplication is nothing but a overriding.

JavaScript:

In JavaScript window is the parent object for all. In window there has one method alert(), which is having the functionality to open the popup , written by JavaScript . Now i wants to override that functionality with my own console message.

```
                window.alert=function(msg){
                        // for alert message functionality by default
                }

                alert(‘naresh’)//open popup with naresh
                window.alert=function(msg){
                        console.log(msg);
                }
                alert(‘naresh’) // now it not open alert message, it shows the console message,
```
means it overrides the existed functionality alert with console.

So in JavaScript overriding is possible.

Polymorphism: It is the combination of both overloading and overriding.

Abstraction : Hides the implementation , but providing services. Ex: ATM GUI , possible by using methods.

Encapsulation : Binding variables and methods in a single component is called as encapsulation. Ex : class