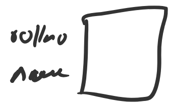
Objects

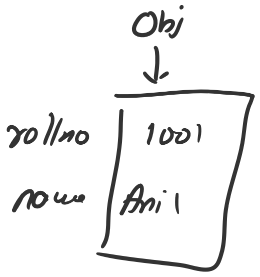
Instance of a class

```
class Student
{
    private int rollno;
    private String name;
    1) -
    2) -
}
```

```
Student s= new Student();
```



```
s.read();
s.display();
```



{ } → Object

```
let obj= { }
obj.rollno= 1001
obj.name= "Anil";
```

Key:value

```
let s= { rollno: 1001,
         name: "Anil"
       };
```

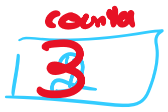
ts

```
class Student
{
    private rollno : number;
    private name : string

    constructor(rollno=0, name:" ")
        this.rollno= rollno;
        this.name= name;
    }
    public displayStudent(): void {
        console.log ("Rollno:" + this.rollno + " Name :" + this.name);
    }
```

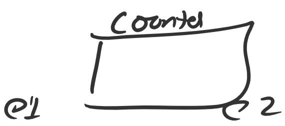
```
let s= new Student(1001, "Anil");
```

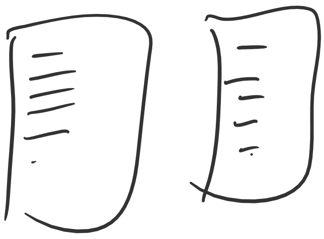
```
}

function Main(){
    let s = new Student(1001, "Anil");
    console.log(s);
}
Main()
```

```
class Employee {
    private empno : number;
    private name  : string;
    private basic : number;
    private da    : number;   // 73% of basic
    private hra : number;     // 10% of basic
    private gross: number;    // basic + da + hra
    private it : number;      // 30% of gross
    private netsal : number;  // gross - it
    private static counter : number;
    constructor( name:"", basic:0) {
        this.empno= ++counter;
        this . name = name;
        this . basic = basic;
        this.calcSalary();
    }
    private calcSalary() : void {
        this.da = this.basic * 0.73;
        this. hra= this . basic * 0.10;
        this . gross= this.basic + this.da + this.hra;
        this . it = this.gross * 0.30;
        this. netsal= this.gross - this.it;
    }

    public displayEmployee():void {
        console.log("Empno :"+ this.empno + "\n Name :"+this.name);
        console.log ("\nBasic :"+this.basic . . . . )
    }
    public static numberOfEmployees():number{
        return counter;
```



let e1 = new Emp ( )
let e2 = new Emplm();





```
          }

    }

    function Main():void {
        let e1 = new Employee(1001,"Anil",10000);
        e1.displayEmployee();
        let e2 = new Employee(1002,"Kiran",20000);
        e2.displayEmployee();
        console.log("No.of employees :" + Employee.numberOfEmployees());
    }
```

```
Class Shape {
    protected dim1 : number;
    protected dim2 : number;
    constructor ( _, _)
    {

    }
}
```

```
Class Rectange extends Shape {
    private area : number;
    constructor(d1, d2){
        super(d1,d2);
        area= d1 * d2;
    }
    }
}
```

Interface

```
interface Shape {
    public setDimension1(dim1 : number): void;
    public getDimension1 ( ): number;
    public setDimension2(dim2: number) : void;
    public getDimension2 () ;number;
    public getArea(): number;
}
abstract class Shape implements Shape
{
    protected dim1 : number;
    protected dim2: number;
    constructor(dim1 : number = 0, dim2 : number = 0)
        this.dim1 = dim1;
        this.dim2 = dim2;
    }
    public setDimension1 (dim1 : number) :void{
        this.dim1= dim1;
    }
    public getDimension1():number{
        return this dim1;
    }

}

class Rectangle extends Shape {
    private area : number;
    constructor(dim1 : number=0, dim2 : number=0)
        super(dim1, dim2);
        area = dim1 * dim2;
    }
    public getArea(): number{
        area= this.dim1 * this.dim2;
        return area;
    }
}


let s = new Rectange(10, 20);
console.log ("Area :" + s.getArea());
```

Setter & Getter Properties

```
class Student
{
    private rollno: number;
    private name : String;

    constructor(rollno : number = 0, name: String = "" ){
        this.rollno = rollno;
        this. name = name;
    }
    public setRollno(rollno: number) : void{
        this. rollno = rollno;
    }
    public getRollno ( ) : number{
        return this.rollno;
    }
    public setName(name : String) : void{
        this. name = name;
    }
    public getName() : String{
        return this.name;
    }
    public set Rollno(rollno: number)
        this.rollno = rollno;
    }
    public get Rollno(): number{
        return this.rollno;
    }
```

```
function Main()
{
    let s = new Student();
    s. setRollno (1001);          →  s. Rollno = 1001;
    s. setName("Ravi");
    console.log("Rollno:"+ s.getRollno());  →  console.log("Rollno:"+ s.Rollno);
    console.log("Name:" + S.getName());
}
```

property

~~s. setRollno() = 1001;~~

~~s. Rollno(1001);~~

```
s. Rollno = 1001;
let i = s. Rollno;
```

Chai

1) assert(expression, message)

   assert(result==290, 'Result is 290')

2) equal (actual, expected, [message])

   assert.equal(result, 290)

3) notEqual(actual , expected) ✓

4) strictEqual (actual, expected)

5) notStrictEqual (actual, expected)

6) deepEqual (actual, expected)

   assert.deepEqual({rollno:1001, name:"Amit"}, {rollno:1001, name:"Amit"})

7) notDeepEqual

8) isAbove

   assert.isAbove(result , 290) ✓

9) .isAtLeast(valueToCheck, valueToBeAtleast
   >=

10) .isBelow   <

11) .isAtMost   <=

12) .isTrue

   assert.isTrue(checkEven(8))

13) .isNotTrue

14) .isFalse

15) .isNotFalse

16) .isNull

17) .isNotNull

18) .isNaN