



JS Classes

#JavaScript Notes



Classes

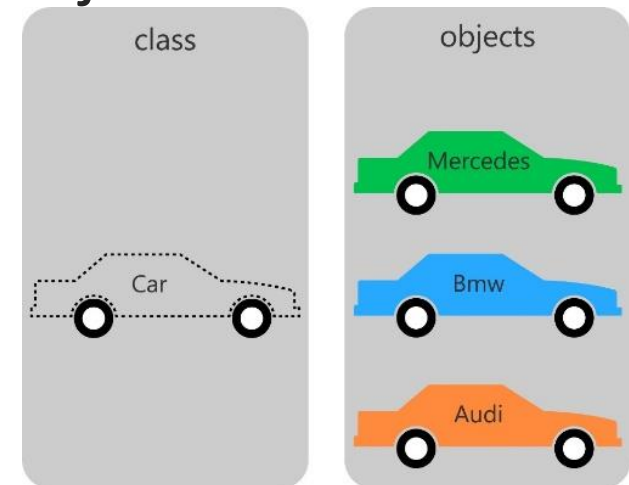
Class

- Classes are one of the features introduced in the **ES6** version of JavaScript.
- A class is a blueprint for the object. You can create an object from the class.



Class :

- Class is a programmer-defined data type, which includes local methods and local variables.
- Class is a collection of objects. Object has properties and behavior.
- First we have to define a class, where **class name should be same as filename.**
- When class is created, we can create any number of objects in that class. The object is created with the help of the new keyword.

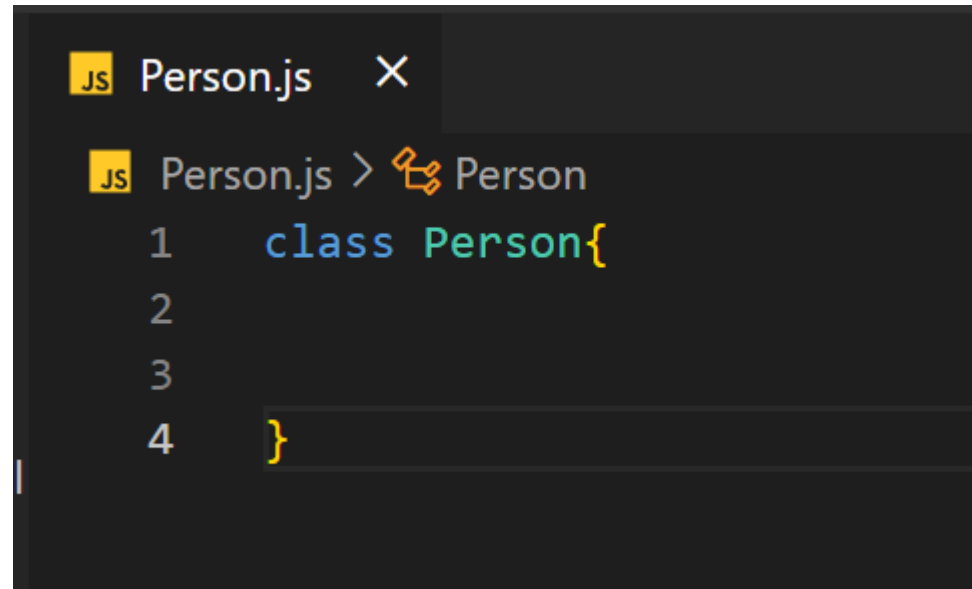


How to create classes?

- In order to create a **class**, we group the code that handles a certain topic into one place.
 1. We declare the class with the **class** keyword.
 2. We write the name of the class and capitalize the first letter.
 - Example : Car
 3. If the class name contains more than one word, we capitalize each word. This is known as **upper camel case**
 - Example : Car_Class
 4. We circle the class body within curly braces. Inside the curly braces, we put our code.
 - Class Car {}



Class Example



The screenshot shows a code editor window titled 'Person.js' with a close button. Below the title bar, there is a breadcrumb 'Person.js > Person' with a file icon. The code content is as follows:

```
1  class Person{  
2  
3  
4  }
```



Constructor

Constructor

- A constructor is a **special function that creates and initializes an object instance of a class.**
- In JavaScript, a constructor gets called when an object is created using the new keyword.
- The purpose of a constructor is to create a new object and set values for any existing object properties



| Object

How to create objects from a class?

- We can create several objects from the same class, with each object having its own set of properties.
- In order to work with a class, we need to create an object from it.
- In order to create an object, we use the **new** keyword.
- Example :
 - `bmw = new Car ();`
 - `mercedes = new Car ();`



```
JS Car.js ×
JS Car.js > ...
1 class Car{
2     constructor(){
3         console.log("Car Class Constructor Called");
4     }
5 }
6 //Object
7 myobj = new Car();
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\code\javascript_code>node Car.js
Car Class Constructor Called
```

```
JS Car.js ×
JS Car.js > ...
1 class Car{
2     constructor(){
3         console.log("Car Class Constructor Called");
4     }
5 }
6 //Object
7 myobj = new Car();
8 myobj_one = new Car();
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\code\javascript_code>node Car.js
Car Class Constructor Called
Car Class Constructor Called
```



The this keyword

- The this keyword indicates that we use the class's own methods and properties, and allows us to have access to them within the class's scope.
- The this keyword allows us to approach the class properties and methods from within the class using the following syntax:



```
JS Person.js X
JS Person.js > ...
1 class Person{
2     constructor(){
3         console.log("Constructor called");
4     }
5 }
6
7 //Create Object
8 const myobject = new Person();
9
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
D:\code\morningjs>node Person.js
Constructor called
```

```
JS Person.js X
JS Person.js > Person
1 class Person{
2
3     constructor(name){
4         this.name = name
5     }
6
7 }
```





Calling Member Variable using Object

```
JS Person.js ×
JS Person.js > ...
1  class Person{
2      constructor(name){
3          console.log("Constructor Called");
4          this.name = name
5      }
6  }
7
8  //Create Object
9  const myobject = new Person("Akash");
10 //Get Value
11 console.log(myobject.name);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
D:\code\morningjs>node Person.js
Constructor Called
Akash
```



Creating JavaScript Class with Multiple Objecy

- The constructor() method inside a class gets called automatically each time an object is created.

```
JS demo.js > ...
1  // creating a class
2  class Person {
3      constructor(name) {
4          this.name = name;
5      }
6  }
7
8  // creating an object
9  const person1 = new Person('Akash');
10 const person2 = new Person('Aarav');
11
12 console.log(person1.name); // Akash
13 console.log(person2.name); // Aarav
```

```
// creating a class
class Person {
    constructor(name) {
        this.name = name;
    }
}

// creating an object
const person1 = new Person('Akash');
const person2 = new Person('Aarav');

console.log(person1.name); // Akash
console.log(person2.name); // Aarav
```


Object with Multiple Parameter

- We can pass multiple parameter in Object
- We can access member variable using object
 - Object.variablename

```
JS Person.js X
JS Person.js > ...
1  class Person{
2      constructor(name,age){
3          console.log("Constructor Called");
4          this.myname = name
5          this.myage = age
6      }
7  }
8
9  //Create Object
10 const myobject = new Person("Akash","30");
11 //Get Value
12 console.log(myobject);
13 console.log(myobject.myname);
14 console.log(myobject.myage);
15

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  COMMENTS

D:\code\morningjs>node Person.js
Constructor Called
Person { myname: 'Akash', myage: '30' }
Akash
30
```



Access Property Outside Class Using Object

- We can access variable
 - Objectname.variablename

```
JS Person.js X
JS Person.js > Person > constructor
1 class Person{
2   constructor(name){
3     this.name = name
4   }
5 }
6
7 //Create Object
8 const myobject = new Person("Akash");
9 //Get Value
10 console.log(myobject.name);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
D:\code\morningjs>node Person.js
Akash
```





Class & Method

Javascript Class Methods

- To access the method of an object, you need to call the method using its name followed by ().

```
class Person {  
  constructor(name) {  
    this.name = name;  
  }  
  
  // defining method  
  greet() {  
    console.log(`Hello ${this.name}`);  
  }  
}
```

```
let person1 = new Person('Akash');
```

```
// accessing property  
console.log(person1.name); // Akash
```

```
// accessing method  
person1.greet(); // Hello Akash
```

```
JS demo.js > Person > constructor  
1  class Person {  
2    constructor(name) {  
3      this.name = name;  
4    }  
5  
6    // defining method  
7    greet() {  
8      console.log(`Hello ${this.name}`);  
9    }  
10 }  
11  
12 let person1 = new Person('Akash');  
13  
14 // accessing property  
15 console.log(person1.name); // Akash  
16  
17 // accessing method  
18 person1.greet(); // Hello Akash
```

Class & Method

- We can define various method and we can access using objectname.method

```
class Car{
  constructor(name,color){
    this.myname = name;
    this.mycolor = color;
  }
  getName(){
    console.log("Car Name is : "+this.myname);
  }
  getColor(){
    console.log("Car Color is : "+this.mycolor);
  }
}

const myobject = new Car("BMW","WHITE"); //Object 1
```

```
myobject.getName(); // Print Car Name
myobject.getColor(); // Print Car Color
```

```
JS Car.js > ...
1  class Car{
2    constructor(name,color){
3      this.myname = name;
4      this.mycolor = color;
5    }
6
7    getName(){
8      console.log("Car Name is : "+this.myname);
9    }
10
11   getColor(){
12     console.log("Car Color is : "+this.mycolor);
13   }
14
15 }
16
17 const myobject = new Car("BMW","WHITE"); //Object 1
18
19 myobject.getName(); // Print Car Name
20 myobject.getColor(); // Print Car Color
21
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\code\jsdemo>node Car.js
Car Name is : BMW
Car Color is : WHITE
```



Revision

```
JS Car.js > ...
1 class Car{
2   constructor(model,color){
3     this.model_year = model;
4     this.color = color;
5   }
6   //Method
7   getName(){
8     console.log("Car Name is " + this.model_year);
9   }
10  getColor(){
11    console.log("Car Color is " + this.color);
12  }
13 }
14 //Object
15 i10 = new Car("2022","White");
16 i20 = new Car("2020","Black");
17 //Access Member Variable
18 console.log(i10.color);
19 console.log(i10.model_year);
20 //Method Print
21 i20.getName();
22 i20.getColor();
```

```
White
2022
Car Name is 2020
Car Color is Black
```

```
class Car{
  constructor(model,color){
    this.model_year = model;
    this.color = color;
  }
  //Method
  getName(){
    console.log("Car Name is " + this.model_year);
  }
  getColor(){
    console.log("Car Color is " + this.color);
  }
}
//Object
i10 = new Car("2022","White");
i20 = new Car("2020","Black");
//Access Member Variable
console.log(i10.color);
console.log(i10.model_year);
//Method Print
i20.getName();
i20.getColor();
```



JS Calculator.js > ...

```
1  class Calculator {
2      constructor(no1, no2) {
3          this.no1 = no1;
4          this.no2 = no2;
5      }
6      getSum() {
7          return this.no1 + this.no2;
8      }
9      getSub() {
10         return this.no1 - this.no2;
11     }
12 }
13
14 var obj = new Calculator(10, 10);
15
16 var sum = obj.getSum();
17 console.log("Sum is " + sum);
18
19 var sum = obj.getSub();
20 console.log("Sub is " + sum);
```





Getter and Setter

Getters and Setters

- In JavaScript, getter methods get the value of an object and setter methods set the value of an object.
- use the get keyword for getter methods and set for setter methods.



Getters and Setters

```
JS demo.js > Person
1  class Person {
2      constructor(name) {
3          this.name = name;
4      }
5      // getter
6      get personName() {
7          return this.name;
8      }
9      // setter
10     set personName(x) {
11         this.name = x;
12     }
13 }
14
15 let person1 = new Person('Akash');
16 console.log(person1.name); // Akash
17
18 // changing the value of name property
19 person1.personName = 'Aarav';
20 console.log(person1.name); // Aarav
```

```
class Person {
  constructor(name) {
    this.name = name;
  }
  // getter
  get personName() {
    return this.name;
  }
  // setter
  set personName(x) {
    this.name = x;
  }
}

let person1 = new Person('Akash');
console.log(person1.name); // Akash

// changing the value of name property
person1.personName = 'Aarav';
console.log(person1.name); // Aarav
```



Get & Set

```
JS Car.js > Car
1 class Car{
2   constructor(carname){
3     this.name = carname;
4   }
5   get CarName(){
6     return this.name;
7   }
8   set CarName(x){
9     this.name = x;
10  }
11 }
12 //Object
13 myobj = new Car("BMW");
14 console.log(myobj.CarName);
15 //Assign
16 myobj.CarName = "Audi";
17 //Get
18 console.log(myobj.CarName);
19 myobj.CarName = "Honda";
20 //Get
21 console.log(myobj.CarName);
22
```

```
D:\code\javascript_code>node Car.js
BMW
Audi
Honda
```

```
class Car{
  constructor(carname){
    this.name = carname;
  }
  get CarName(){
    return this.name;
  }
  set CarName(x){
    this.name = x;
  }
}
//Object
myobj = new Car("BMW");
console.log(myobj.CarName);
//Assign
myobj.CarName = "Audi";
//Get
console.log(myobj.CarName);
myobj.CarName = "Honda";
//Get
console.log(myobj.CarName);
```



Example

```
JS Carjs > ...
1 class Car{
2     constructor(model){
3         this.model = model;
4     }
5     //Method
6     get ModelDetails()
7     {
8         return this.model;
9     }
10    set ModelDetails(value){
11        this.model = value;
12    }
13 }
14 //Object
15 tata = new Car("Nexon");
16 //Print Default Value
17 console.log(`Intial Value is ${tata.model}`)
18 //Print Method
19 var a = tata.ModelDetails;
20 console.log(`Get Name Value is ${a}`);
21 //Assign New Value
22 tata.ModelDetails = "Tiago";
23 //Print Method
24 var a = tata.ModelDetails;
25 console.log(`New Value is ${a}`);
```

```
Intial Value is Nexon
Get Name Value is Nexon
New Value is Tiago
```

```
class Car{
    constructor(model){
        this.model = model;
    }
    //Method
    get ModelDetails()
    {
        return this.model;
    }
    set ModelDetails(value){
        this.model = value;
    }
}
//Object
tata = new Car("Nexon");
//Print Default Value
console.log(`Intial Value is ${tata.model}`)
//Print Method
var a = tata.ModelDetails;
console.log(`Get Name Value is ${a}`);
//Assign New Value
tata.ModelDetails = "Tiago";
//Print Method
var a = tata.ModelDetails;
console.log(`New Value is ${a}`);
```



```
JS Calculator.js > ...
1  class Calculator {
2      constructor(no1) {
3          this.no1 = no1;
4      }
5      get MyValue(){
6          return this.no1;
7      }
8      set MyValue(x){
9          this.no1 = x;
10     }
11
12 }
13
14 var obj = new Calculator(10);
15 console.log(obj.no1); //10
16
17 obj.MyValue = 99; //Setter 99
18 console.log(obj.MyValue); //Getter 99
```

JavaScript Class Inheritance

Class Inheritance

- When the properties and the methods of the parent class are accessed by the child class, we call the concept has inheritance.
- To use class inheritance, you use the **extends** keyword.
- The child class can inherit the parent method and give own method implementation, this property is called overridden method.
- Inheritance enables you to define a class that takes all the functionality from a parent class and allows you to add more.
- Using class inheritance, a class can inherit all the methods and properties of another class.
- Inheritance is a useful feature that allows code reusability.



Class Inheritance

- Student class inherits all the methods and properties of the Person class. Hence, the Student class will now have the name property and the greet() method.

```
JS demo.js > ...
1  // parent class
2  class Person {
3      constructor(name) {
4          this.name = name;
5      }
6
7      greet() {
8          console.log(`Hello ${this.name}`);
9      }
10 }
11
12 // inheriting parent class
13 class Student extends Person {
14
15 }
16
17 //Object of Student Class
18 let student1 = new Student('Akash');
19 student1.greet();
```

```
// parent class
class Person {
    constructor(name) {
        this.name = name;
    }

    greet() {
        console.log(`Hello ${this.name}`);
    }
}

// inheriting parent class
class Student extends Person {

}

//Object of Student Class
let student1 = new Student('Akash');
student1.greet();
```


Extends to Multiple Class

```
Person.js > ...
1 // parent class
2 class Person {
3     constructor(name) {
4         this.name = name;
5     }
6     greet() {
7         console.log(`Hello ${this.name}`);
8     }
9 }
10
11 // inheriting parent class
12 class Student extends Person {
13 }
14
15 // inheriting parent class
16 class Professor extends Person {
17     subject(){
18         console.log(`I Teach Es6`);
19     }
20 }
21
22 //Object of Student Class
23 let student1 = new Student('Akash');
24 student1.greet();
25
26 let professor1 = new Professor('AkashSir');
27 professor1.greet();
28 professor1.subject();
```

Hello Akash
Hello AkashSir
I Teach Es6

```
// parent class
class Person {
    constructor(name) {
        this.name = name;
    }
    greet() {
        console.log(`Hello ${this.name}`);
    }
}
```

```
// inheriting parent class
class Student extends Person {
}
```

```
// inheriting parent class
class Professor extends Person {
    subject(){
        console.log(`I Teach Es6`);
    }
}
```

```
//Object of Student Class
let student1 = new Student('Akash');
student1.greet();
```

```
let professor1 = new Professor('AkashSir');
professor1.greet();
professor1.subject();
```



JavaScript super() keyword

- The super keyword used inside a child class denotes its parent class.

```
demo.js > ...
1  // parent class
2  class Person {
3      constructor(name) {
4          this.name = name;
5      }
6      greet() {
7          console.log(`Hello ${this.name}`);
8      }
9  }
10
11 // inheriting parent class
12 class Student extends Person {
13
14     constructor(name) {
15         console.log("Creating student class");
16         // call the super class constructor and pass in the name parameter
17         super(name);
18     }
19
20 }
21
22 let student1 = new Student('Akash');
23 student1.greet();
24 //Output
25 //Creating student class
26 //Hello Akash
```

```
// parent class
class Person {
    constructor(name) {
        this.name = name;
    }
    greet() {
        console.log(`Hello ${this.name}`);
    }
}

// inheriting parent class
class Student extends Person {

    constructor(name) {
        console.log("Creating student class");
        // call the super class constructor and pass in the name parameter
        super(name);
    }

}

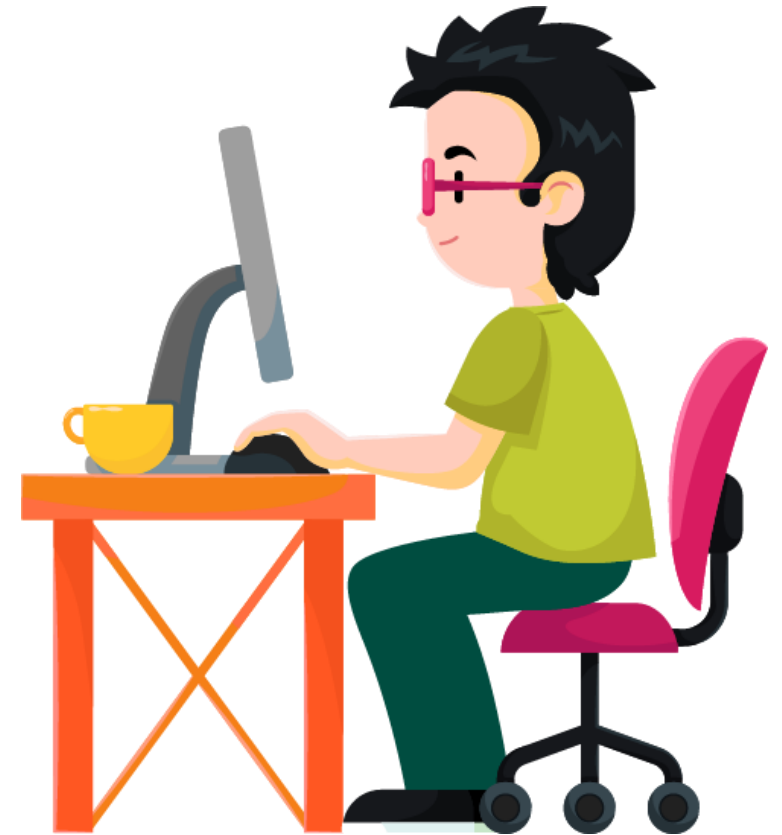
let student1 = new Student('Akash');
student1.greet();
//Output
//Creating student class
//Hello Akash
```

Get Exclusive Video Tutorials



www.apptutorials.com

<https://www.youtube.com/user/Akashtips>





Get More Details

www.akashsir.com



If You Liked It !

Rating Us Now



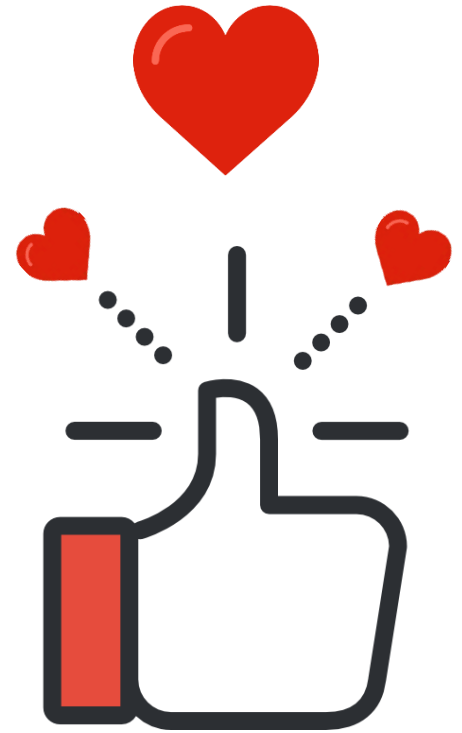
Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET



Sulekha

<https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad>



Connect With Me



Akash Padhiyar
#AkashSir

www.akashsir.com

www.akashtechlabs.com

www.akashpadhiyar.com

www.apptutorials.com

Social Info



Akash.padhiyar



Akashpadhiyar



Akash_padhiyar



+91 99786-21654



#Akashpadhiyar

#apptutorials