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.

You can think of the class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions, you build the house. House is the object.

Since many houses can be made from the same description, we can create many objects from a class.

**Creating JavaScript Class**

JavaScript class is similar to the [Javascript constructor function](https://www.programiz.com/javascript/constructor-function), and it is merely a syntactic sugar.

The constructor function is defined as:

// constructor function

function Person () {

this.name = 'John',

this.age = 23

}

// create an object

const person1 = new Person();

Instead of using the function keyword, you use the class keyword for creating JS classes. For example,

// creating a class

class Person {

constructor(name) {

this.name = name;

}

}

The class keyword is used to create a class. The properties are assigned in a constructor function.

Now you can create an object. For example,

// creating a class

class Person {

constructor(name) {

this.name = name;

}

}

// creating an object

const person1 = new Person('John');

const person2 = new Person('Jack');

console.log(person1.name); // John

console.log(person2.name); // Jack

Here, person1 and person2 are objects of Person class.

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

**Javascript Class Methods**

While using constructor function, you define methods as:

// constructor function

function Person (name) {

// assigning parameter values to the calling object

this.name = name;

// defining method

this.greet = function () {

return ('Hello' + ' ' + this.name);

}

}

It is easy to define methods in the JavaScript class. You simply give the name of the method followed by (). For example,

class Person {

constructor(name) {

this.name = name;

}

// defining method

greet() {

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

}

}

let person1 = new Person('John');

// accessing property

console.log(person1.name); // John

// accessing method

person1.greet(); // Hello John

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

**Getters and Setters**

In JavaScript, getter methods get the value of an object and setter methods set the value of an object.

JavaScript classes may include [getters and setters](https://www.programiz.com/javascript/getter-setter). You use the get keyword for getter methods and set for setter methods. For example,

class Person {

constructor(name) {

this.name = name;

}

// getter

get personName() {

return this.name;

}

// setter

set personName(x) {

this.name = x;

}

}

let person1 = new Person('Jack');

console.log(person1.name); // Jack

// changing the value of name property

person1.personName = 'Sarah';

console.log(person1.name); // Sarah

**Hoisting**

A class should be defined before using it. Unlike functions and other JavaScript declarations, the class is not hoisted. For example,

// accessing class

const p = new Person(); // ReferenceError

// defining class

class Person {

constructor(name) {

this.name = name;

}

}

As you can see, accessing a class before defining it throws an error.

**'use strict'**

Classes always follow ['use-strict'](https://www.programiz.com/javascript/use-strict). All the code inside the class is automatically in strict mode. For example,

class Person {

constructor() {

a = 0;

this.name = a;

}

}

let p = new Person(); // ReferenceError: Can't find variable: a

**Note**: JavaScript class is a special type of function. And the typeof operator returns function for a class.

For example,

class Person {}

console.log(typeof Person); // function