

# JavaScript Object Constructors

[< Previous](#)[Next >](#)

## Example

```
function Person(first, last, age, eye) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eye;  
}
```

[Try it yourself »](#)

It is considered good practice to name constructor functions with an upper-case first letter.

## Object Types (Blueprints) (Classes)

The examples from the previous chapters are limited. They only create single objects.

Sometimes we need a "**blueprint**" for creating many objects of the same "type".

The way to create an "object type", is to use an **object constructor function**.

In the example above, `function Person()` is an object constructor function.

Objects of the same type are created by calling the constructor function with the `new` keyword:

```
var myFather = new Person("John", "Doe", 50, "blue");  
var myMother = new Person("Sally", "Rally", 48, "green");
```

[Try it yourself »](#)



## The **this** Keyword

In JavaScript, the thing called **this** is the object that "owns" the code.

The value of **this**, when used in an object, is the object itself.

In a constructor function **this** does not have a value. It is a substitute for the new object. The value of **this** will become the new object when a new object is created.

Note that **this** is not a variable. It is a keyword. You cannot change the value of **this**.

## Adding a Property to an Object

Adding a new property to an existing object is easy:

### Example

```
myFather.nationality = "English";
```

Try it Yourself »

The property will be added to myFather. Not to myMother. (Not to any other person objects).

## Adding a Method to an Object

Adding a new method to an existing object is easy:





### Example

```
myFather.name = function () {  
  return this.firstName + " " + this.lastName;  
};
```

Try it Yourself »

The method will be added to myFather. Not to myMother. (Not to any other person objects).

## Adding a Property to a Constructor

HTMLCSSJAVASCRIPTMORE ▾REFERENCES ▾

Example

```
Person.nationality = "English";
```

Try it Yourself »

To add a new property to a constructor, you must add it to the constructor function:

Example

```
function Person(first, last, age, eyecolor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyecolor;  
  this.nationality = "English";  
}
```

Try it Yourself »

This way object properties can have default values.

## Adding a Method to a Constructor

Your constructor function can also define methods:

Example

```
function Person(first, last, age, eyecolor) {  
  this.firstName = first;  
  this.lastName = last;  
  this.age = age;  
  this.eyeColor = eyecolor;  
  this.name = function() {return this.firstName + " " + this.lastName;};  
}
```

Try it Yourself »

You cannot add a new method to an object constructor the same way you add a new method to an existing object.

Adding methods to an object must be done inside the constructor function:

HTMLCSSJAVASCRIPTMORE ▾REFERENCES ▾

```
function Person(firstName, lastName, age, eyeColor) {  
  this.firstName = firstName;  
  this.lastName = lastName;  
  this.age = age;  
  this.eyeColor = eyeColor;  
  this.changeName = function (name) {  
    this.lastName = name;  
  };  
}
```

The changeName() function assigns the value of name to the person's lastName property.

### Now You Can Try:

```
myMother.changeName("Doe");
```

Try it Yourself »

JavaScript knows which person you are talking about by "substituting" **this** with **myMother**.

## Built-in JavaScript Constructors

JavaScript has built-in constructors for native objects:

### Example

```
var x1 = new Object();    // A new Object object  
var x2 = new String();    // A new String object  
var x3 = new Number();    // A new Number object  
var x4 = new Boolean();   // A new Boolean object  
var x5 = new Array();     // A new Array object  
var x6 = new RegExp();    // A new RegExp object  
var x7 = new Function();  // A new Function object  
var x8 = new Date();      // A new Date object
```

Try it Yourself »

The `Math()` object is not in the list. `Math` is a global object. The `new` keyword cannot be used on `Math`.

### Did You Know?

As you can see above, JavaScript has object versions of the primitive data types `String`, `Number`, and `Boolean`. But there

ALSO:

Use object literals `{}` instead of `new Object()`.

Use string literals `""` instead of `new String()`.

Use number literals `12345` instead of `new Number()`.

Use boolean literals `true / false` instead of `new Boolean()`.

Use array literals `[]` instead of `new Array()`.

Use pattern literals `/()/` instead of `new RegExp()`.

Use function expressions `() {}` instead of `new Function()`.

### Example

```
var x1 = {};           // new object
var x2 = "";           // new primitive string
var x3 = 0;            // new primitive number
var x4 = false;        // new primitive boolean
var x5 = [];           // new array object
var x6 = /()/          // new regexp object
var x7 = function(){}; // new function object
```

Try it Yourself »

## String Objects

Normally, strings are created as primitives: `var firstName = "John"`

But strings can also be created as objects using the `new` keyword: `var firstName = new String("John")`

Learn why strings should not be created as object in the chapter [JS Strings](#).

## Number Objects

Normally, numbers are created as primitives: `var x = 123`

But numbers can also be created as objects using the `new` keyword: `var x = new Number(123)`

Learn why numbers should not be created as object in the chapter [JS Numbers](#).

## Boolean Objects

Normally, booleans are created as primitives: `var x = false`

But booleans can also be created as objects using the `new` keyword: `var x = new Boolean(false)`

## COLOR PICKER



### HOW TO

- Tabs
- Dropdowns
- Accordions
- Side Navigation
- Top Navigation
- Modal Boxes
- Progress Bars
- Parallax
- Login Form
- HTML Includes
- Google Maps