[Constructor Examples](javascript:void(0)" \o "Collapse)

var myObject = new Object();

// Creates a Date object.

var myBirthday = new Date(1961, 5, 10);

// Creates a user defined object.

var myCar = new Car();

[**Writing Constructors**](javascript:void(0))

You can create objects using the **new** operator in conjunction with predefined constructor functions such as **Object()**, **Date()**, and **Function()**. You can also create custom constructor functions that define a set of properties and methods. Here is an example of a custom constructor.

JavaScript

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_f3b2b8be-8e54-4b3d-bd98-44610991f3fd');" \o "Copy to clipboard.)

function Circle (xPoint, yPoint, radius) {

this.x = xPoint; // The x component of the center of the circle.

this.y = yPoint; // The y component of the center of the circle.

this.r = radius; // The radius of the circle.

}

When you invoke the Circle constructor, you supply values for the circle's center point and the radius. You end up with a Circle object that contains three properties. Here is how you would instantiate a Circle object.

JavaScript

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_79d3ab26-72b9-4db6-ba05-5d68aaf8a782');)

var aCircle = new Circle(5, 11, 99);

# Prototypes and Prototype Inheritance

In JavaScript, a **prototype** is a property of functions and of objects that are created by constructor functions. The prototype of a function is an object. Its main use is when a function is used as a constructor.

JavaScript

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_833a54fb-f348-4a17-996f-2f5c6959c628');)

function Vehicle(wheels, engine) {

this.wheels = wheels;

this.engine = engine;

}

In the example above, the prototype of the Vehicle function is the prototype of any object that is instantiated with the Vehicle constructor.

[**Using Prototypes to Add Properties and Methods**](javascript:void(0))

You can use the **prototype** property to add properties and methods to objects, even the ones that have already been created:

JavaScript

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_874fa640-9844-40e7-8471-b7044679b1cb');)

var testVehicle = new Vehicle(2, false);

Vehicle.prototype.color = "red";

var testColor = testVehicle.color;

The value of testColor is "red".

### Using Prototypes to Derive One Object from Another with Object.create

The **prototype** object can be used to derive one object from another. For example, you can use the [Object.create](https://msdn.microsoft.com/en-us/library/ff925952(v=vs.94).aspx) function to derive a new object Bicycle using the prototype of the Vehicle object we defined earlier (plus any new properties you need).

JavaScript

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_5f56938e-efa5-4ecb-a99d-c24ac51396fc');)

var Bicycle = Object.create(Object.getPrototypeOf(Vehicle), {

"pedals" :{value: true}

});

The Bicycle object has the properties wheels, engine, color, and pedals, and its prototype is **Vehicle.prototype**. The JavaScript engine finds the pedals property on Bicycle, and it looks up the prototype chain to find the wheels, engine, and color properties on Vehicle.

### Changing an Object's Prototype

In Internet Explorer 11, you can replace the internal prototype of an object or function with a new prototype by using the [\_\_proto\_\_](https://msdn.microsoft.com/en-us/library/dn342818(v=vs.94).aspx) property. When you use this property, you inherit the properties and methods of the new prototype along with other properties and methods in its prototype chain.

The following example shows how you can change the prototype of an object. This example shows how the object's inherited properties change when you change its prototype.

JavaScript

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_2a41cfd3-57cd-48e7-8d12-82882442307f');)

function Friend() {

this.demeanor = "happy";

}

function Foe() {

this.demeanor = "suspicious";

}

var friend = new Friend();

var foe = new Foe();

var player = new Object();

player.\_\_proto\_\_ = foe;

friend.ally = "Tom";

if (console && console.log) {

console.log(player.demeanor === "happy" ); // Returns false

console.log(player.demeanor === "suspicious"); // Returns true

console.log(player.ally === "Tom"); // Returns false

// Turn the foe to a friend.

player.\_\_proto\_\_ = friend;

console.log(player.demeanor === "happy"); // Returns true

console.log(player.demeanor === "suspicious"); // Returns false

console.log(player.ally === "Tom"); // Returns true

}

# Recursion (JavaScript)

function factorial(num)

{

// If the number is less than 0, reject it.

if (num < 0) {

return -1;

}

// If the number is 0, its factorial is 1.

else if (num == 0) {

return 1;

}

// Otherwise, call this recursive procedure again.

else {

return (num \* factorial(num - 1));

}

}

var result = factorial(8);

document.write(result);

# Variable Scope (JavaScript)

JavaScript

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_820d84eb-7333-4d48-bc2d-746d87cdc2fe');)

var aNumber = 100;

tweak();

function tweak(){

// This prints "undefined", because aNumber is also defined locally below.

document.write(aNumber);

if (false)

{

var aNumber = 123;

}

}

When JavaScript executes a function, it first looks for all variable declarations, for example, var someVariable;. It creates the variables with an initial value of **undefined**. If a variable is declared with a value, for example, var someVariable = "something";, then it still initially has the value **undefined** and takes on the declared value only when the line that contains the declaration is executed.

## Bl[ock-scoped variables](javascript:void(0))

Internet Explorer 11 introduces support for [let](https://msdn.microsoft.com/en-us/library/dn263046(v=vs.94).aspx) and [const](https://msdn.microsoft.com/en-us/library/dn251545(v=vs.94).aspx), which are block-scoped variables. For these variables, the braces {. . .} define a new scope. When you set one of these variables to a particular value, the value applies only to the scope in which it is set.