Every JavaScript object has a prototype. The prototype is also an object.

All JavaScript objects inherit their properties and methods from their prototype.

JavaScript Prototypes

All JavaScript objects inherit the properties and methods from their prototype.

Objects created using an object literal, or with new Object(), inherit from a prototype called Object.prototype.

Objects created with new Date() inherit the Date.prototype.

The Object.prototype is on the top of the prototype chain.

All JavaScript objects (Date, Array, RegExp, Function, ....) inherit from the Object.prototype.

Creating a Prototype

The standard way to create an object prototype is to use an object constructor function:

Example

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

With a constructor function, you can use the **new** keyword to create new objects from the same prototype:

Example

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

|  |  |
| --- | --- |
|  | The constructor function is the prototype for Person objects. It is considered good practice to name constructor function with an upper-case first letter. |

Adding Properties and Methods to Objects

Sometimes you want to add new properties (or methods) to an existing object.

Sometimes you want to add new properties (or methods) to all existing objects of a given type.

Sometimes you want to add new properties (or methods) to an object prototype.

Adding a Property to an Object

Adding a new property to an existing object is easy:

Example

myFather.nationality = "English";

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 also easy:

Example

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

The method will be added to myFather. Not to myMother.

Adding Properties to a Prototype

You cannot add a new property to a prototype the same way as you add a new property to an existing object, because the prototype is not an existing object.

Example

Person.nationality = "English";

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"  
}

|  |  |
| --- | --- |
|  | Prototype properties can have prototype values (default values). |

Adding Methods to a Prototype

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;};  
}

Using the **prototype** Property

The JavaScript prototype property allows you to add new properties to an existing prototype:

Example

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

The JavaScript prototype property also allows you to add new methods to an existing prototype:

Example

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