JavaScript Prototype in Plain Language

jan. 25 2013 [151](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comments)

***Prototype*** is a fundamental concept that every JavaScript developer must understand, and this article aims to explain JavaScript’s prototype in plain, detailed language. If you don’t understand JavaScript’s prototype after reading this blog post, please ask questions in the comments below. I will personally answer all questions.

To understand prototype in JavaScript you must understand objects in JavaScript. If you aren’t already familiar with objects, you should read my post [*JavaScript Objects in Detail*](http://javascriptissexy.com/javascript-objects-in-detail/). Also, know that a *property* is simply a variable defined on a function.

* Receive Updates

Top of Form



Bottom of Form

There are two interrelated concepts with **prototype** in JavaScript:

1. First, every JavaScript function has a **prototype property** (this property is empty by default), and you attach properties and methods on this prototype property when you want to implement inheritance. This prototype property is not enumerable; that is, it isn’t accessible in a for/in loop. But Firefox and most versions of Safari and Chrome have a \_\_proto\_\_ “pseudo” property (an alternative syntax) that allows you to access an object’s prototype property. You will likely never use this \_\_proto\_\_ pseudo property, but you should know that it exists and it is simply a way to access an object’s prototype property in some browsers.

The prototype property is used primarily for inheritance; you add methods and properties on a function’s prototype property to make those methods and properties available to instances of that function.

Consider this simple example of inheritance with the prototype property (more on inheritance later):

|  |  |
| --- | --- |
|  | function PrintStuff (myDocuments) { |
|  | ​this.documents = myDocuments; |
|  | } |
|  | ​ |
|  | ​// We add the print () method to PrintStuff prototype property so that other instances (objects) can inherit it:​ |
|  | PrintStuff.prototype.print = function () { |
|  | console.log(this.documents); |
|  | } |
|  | ​ |
|  | ​// Create a new object with the PrintStuff () constructor, thus allowing this new object to inherit PrintStuff's properties and methods.​ |
|  | ​var newObj = new PrintStuff ("I am a new Object and I can print."); |
|  | ​ |
|  | ​// newObj inherited all the properties and methods, including the print method, from the PrintStuff function. Now newObj can call print directly, even though we never created a print () method on it.​ |
|  | newObj.print (); //I am a new Object and I can print. |

1. The second concept with prototype in JavaScript is the **prototype attribute**. Think of the prototype attribute as a characteristic of the object; this characteristic tells us the object’s “parent”. In simple terms: An object’s prototype attribute points to the object’s “parent”—the object it inherited its properties from. The prototype attribute is normally referred to as the *prototype object*, and it is set automatically when you create a new object.To expound on this: Every object inherits properties from some other object, and it is this other object that is the object’s prototype attribute or “parent.” (You can think of the *prototype attribute* as the lineage or the parent). In the example code above, *newObj*‘s prototype is PrintStuff.prototype.

Note: All objects have attributes just like object properties have attributes. And the object attributes are *prototype, class,* and *extensible* attributes. It is this prototype attribute that we are discussing in this second example.

Also note that the \_\_proto\_\_ “pseudo” property contains an object’s prototype object (the parent object it inherited its methods and properties from).

**Constructor**  
Before we continue, let’s briefly examine the constructor. A **constructor** is a function used for initializing new objects, and you use the new keyword to call the constructor.  
For example:

|  |  |
| --- | --- |
|  | function Account () { |
|  | } |
|  | ​// This is the use of the Account constructor to create the userAccount object.​ |
|  | ​var userAccount = new Account (); |

Moreover, all objects that inherit from another object also inherit a *constructor*property. And this constructor property is simply a property (like any variable) that holds or points to the constructor of the object.

|  |  |
| --- | --- |
|  | //The constructor in this example is Object () ​ |
|  | ​var myObj = new Object (); |
|  | ​// And if you later want to find the myObj constructor:​ |
|  | console.log(myObj.constructor); // Object()​ |
|  | ​ |
|  | ​// Another example: Account () is the constructor​ |
|  | ​var userAccount = new Account (); |
|  | ​// Find the userAccount object's constructor​ |
|  | console.log(userAccount.constructor); // Account() |

**Prototype Attribute of Objects Created with new Object () or Object Literal**  
All objects created with object literals and with the Object constructor inherits from **Object.prototype**. Therefore, Object.prototype is the prototype attribute (or the prototype object) of all objects created with new Object () or with {}. Object.prototype itself does not inherit any methods or properties from any other object.

|  |  |
| --- | --- |
|  | // The userAccount object inherits from Object and as such its prototype attribute is Object.prototype.​ |
|  | ​var userAccount = new Object (); |
|  | ​ |
|  | ​// This demonstrates the use of an object literal to create the userAccount object; the userAccount object inherits from Object; therefore, its prototype attribute is Object.prototype just as the userAccount object does above.​ |
|  | ​var userAccount = {name: “Mike”} |

**Prototype Attribute of Objects Created With a Constructor Function**  
Objects created with the new keyword and any constructor other than the Object () constructor, get their prototype from the constructor function.

For Example:

|  |  |
| --- | --- |
|  | function Account () { |
|  | ​ |
|  | } |
|  | ​var userAccount = new Account () // userAccount initialized with the Account () constructor and as such its prototype attribute (or prototype object) is Account.prototype. |

Similarly, any array such as var myArray = new Array (), gets its prototype from Array.prototype and it inherits Array.prototype’s properties.

So, there are two general ways an object’s prototype attribute is set when an object is created:

1. If an object is created with an object literal (var newObj = {}), it inherits properties from Object.prototype and we say its prototype object (or prototype attribute) is Object.prototype.
2. If an object is created from a constructor function such as new Object (), new Fruit () or new Array () or new Anything (), it inherits from that constructor (Object (), Fruit (), Array (), or Anything ()). For example, with a function such as Fruit (), each time we create a new instance of Fruit (var aFruit = new Fruit ()), the new instance’s prototype is assigned the prototype from the Fruit constructor, which is Fruit.prototype.Any object that was created with new Array () will have Array.prototype as its prototype. An object created with new Fruit () will have Fruit.prototype as its prototype. And any object created with the Object constructor (Obj (), such as var anObj = new Object() ) inherits from Object.prototype.

It is important to know that in ECMAScript 5, you can create objects with an Object.create() method that allows you to set the new object’s prototype object. We will cover ECMAScript 5 in a later post.

**Why is Prototype Important and When is it Used?**  
These are two important ways the prototype is used in JavaScript, as we noted above:

1. **Prototype Property: Prototype-based Inheritance**  
   Prototype is important in JavaScript because JavaScript does not have classical inheritance based on Classes (as most object oriented languages do), and therefore all inheritance in JavaScript is made possible through the prototype property. JavaScript has a prototype-based inheritance mechanism.Inheritance is a programming paradigm where objects (or Classes in some languages) can inherit properties and methods from other objects (or Classes). In JavaScript, you implement inheritance with the prototype property. For example, you can create a Fruit function (an object, since all functions in JavaScript are objects) and add properties and methods on the Fruit prototype property, and all instances of the Fruit function will inherit all the Fruit’s properties and methods.

Demonstration of Inheritance in JavaScript:

|  |  |
| --- | --- |
|  | function Plant () { |
|  | ​this.country = "Mexico"; |
|  | ​this.isOrganic = true; |
|  | } |
|  | ​ |
|  | ​// Add the showNameAndColor method to the Plant prototype property​ |
|  | Plant.prototype.showNameAndColor = function () { |
|  | console.log("I am a " + this.name + " and my color is " + this.color); |
|  | } |
|  | ​ |
|  | ​// Add the amIOrganic method to the Plant prototype property​ |
|  | Plant.prototype.amIOrganic = function () { |
|  | ​if (this.isOrganic) |
|  | console.log("I am organic, Baby!"); |
|  | } |
|  | ​ |
|  | ​function Fruit (fruitName, fruitColor) { |
|  | ​this.name = fruitName; |
|  | ​this.color = fruitColor; |
|  | } |
|  | ​ |
|  | ​// Set the Fruit's prototype to Plant's constructor, thus inheriting all of Plant.prototype methods and properties.​ |
|  | Fruit.prototype = new Plant (); |
|  | ​ |
|  | ​// Creates a new object, aBanana, with the Fruit constructor​ |
|  | ​var aBanana = new Fruit ("Banana", "Yellow"); |
|  | ​ |
|  | ​// Here, aBanana uses the name property from the aBanana object prototype, which is Fruit.prototype:​ |
|  | console.log(aBanana.name); // Banana​ |
|  | ​ |
|  | ​// Uses the showNameAndColor method from the Fruit object prototype, which is Plant.prototype. The aBanana object inherits all the properties and methods from both the Plant and Fruit functions.​ |
|  | console.log(aBanana.showNameAndColor()); // I am a Banana and my color is yellow. |

Note that the *showNameAndColor* method was inherited by the *aBanana* object even though it was defined all the way up the prototype chain on the Plant.prototype object.

Indeed, any object that uses the Fruit () constructor will inherit all the Fruit.prototype properties and methods and all the properties and methods from the Fruit’s prototype, which is Plant.prototype. This is **the principal manner in which inheritance is implemented in JavaScript** and the integral role the prototype chain has in the process.

For more in-depth coverage on Objective Oriented Programming in JavaScript, get Nicholas Zakas’s [Principles of Object-Oriented Programming in JavaScript](https://leanpub.com/oopinjavascript) (it is only $14.99).

1. **Prototype Attribute: Accessing Properties on Objects**  
   Prototype is also important for accessing properties and methods of objects. The **prototype attribute** (or prototype object) of any object is the “parent” object where the inherited properties were originally defined.This is loosely analogous to the way you might inherit your surname from your father—he is your “prototype parent.” If we wanted to find out where your surname came from, we would first check to see if you created it yourself; if not, the search will move to your prototype parent to see if you inherited it from him. If it was not created by him, the search continues to his father (your father’s prototype parent).

Similarly, if you want to access a property of an object, the search for the property begins directly on the object. If the JS runtime can’t find the property there, it then looks for the property on the object’s prototype—the object it inherited its properties from.  
If the property is not found on the object’s prototype, the search for the property then moves to prototype of the object’s prototype (the father of the object’s father—the grandfather). And this continues until there is no more prototype (no more great-grand father; no more lineage to follow). **This in essence is the prototype chain:** the chain from an object’s prototype to its prototype’s prototype and onwards. And JavaScript uses this prototype chain to look for properties and methods of an object.  
If the property does not exist on any of the object’s prototype in its prototype chain, then the property does not exist and *undefined* is returned.

This prototype chain mechanism is essentially the same concept we have discussed above with the prototype-based inheritance, except we are now focusing specifically on how JavaScript accesses object properties and methods via the prototype object.

This example demonstrates the prototype chain of an object’s prototype object:

|  |  |
| --- | --- |
|  | var myFriends = {name: "Pete"}; |
|  | ​ |
|  | ​// To find the name property below, the search will begin directly on the myFriends object and will immediately find the name property because we defined the property name on the myFriend object. This could be thought of as a prototype chain with one link.​ |
|  | console.log(myFriends.name); |
|  | ​ |
|  | ​// In this example, the search for the toString () method will also begin on the myFriends’ object, but because we never created a toString method on the myFriends object, the compiler will then search for it on the myFriends prototype (the object which it inherited its properties from).​ |
|  | ​ |
|  | ​// And since all objects created with the object literal inherits from Object.prototype, the toString method will be found on Object.prototype—see important note below for all properties inherited from Object.prototype. ​ |
|  | ​ |
|  | myFriends.toString (); |

**Object.prototype Properties Inherited by all Objects**  
All objects in JavaScript inherit properties and methods from Object.prototype. These inherited properties and methods are *constructor, hasOwnProperty (), isPrototypeOf (), propertyIsEnumerable (), toLocaleString (), toString (), and valueOf ()*. ECMAScript 5 also adds 4 accessor methods to Object.prototype.

Here is another example of the prototype chain:

|  |  |
| --- | --- |
|  | function People () { |
|  | ​this.superstar = "Michael Jackson"; |
|  | } |
|  | ​// Define "athlete" property on the People prototype so that "athlete" is accessible by all objects that use the People () constructor.​ |
|  | People.prototype.athlete = "Tiger Woods"; |
|  | ​ |
|  | ​var famousPerson = new People (); |
|  | famousPerson.superstar = "Steve Jobs"; |
|  | ​ |
|  | ​// The search for superstar will first look for the superstar property on the famousPerson object, and since we defined it there, that is the property that will be used. Because we have overwritten the famousPerson’s superstar property with one directly on the famousPerson object, the search will NOT proceed up the prototype chain. ​ |
|  | console.log (famousPerson.superstar); // Steve Jobs​ |
|  | ​ |
|  | ​// Note that in ECMAScript 5 you can set a property to read only, and in that case you cannot overwrite it as we just did.​ |
|  | ​ |
|  | ​// This will show the property from the famousPerson prototype (People.prototype), since the athlete property was not defined on the famousPerson object itself.​ |
|  | console.log (famousPerson.athlete); // Tiger Woods​ |
|  | ​ |
|  | ​// In this example, the search proceeds up the prototype chain and find the toString method on Object.prototype, from which the Fruit object inherited—all objects ultimately inherits from Object.prototype as we have noted before.​ |
|  | console.log (famousPerson.toString()); // [object Object] |

All built-in constructors (Array (), Number (), String (), etc.) were created from the Object constructor, and as such their prototype is Object.prototype.

Check back on February 7th for a quiz on JavaScript Prototype.

**Additional Information**  
For more on JavaScript Objects, read Chapter 6 of *JavaScript: The Definitive Guide (6th Edition, May 2011)* by David Flanagan.

Be good. Sleep well. And enjoy coding.

[[](https://bovacademy.com/)](https://bovacademy.com/" \t "_blank)

[Become an Elite/Highly Paid Specialized Software Engineer (Frontend, Fullstack, etc.)](https://bovacademy.com/" \t "_blank)

[Within 8–10 Months, Earn MORE than the Avg. New CS Graduate Earns.](https://bovacademy.com/" \t "_blank)

**[New Prices (Monthly Subscription):](https://bovacademy.com/" \t "_blank)** [Basic: $250/month, Pro: $350/month, Premium: $600/month](https://bovacademy.com/" \t "_blank)

[By the founder of JavaScriptIsSexy](https://bovacademy.com/" \t "_blank)

Posted in: [16 Important JavaScript Concepts](http://javascriptissexy.com/category/16-important-javascript-concepts/), [JavaScript](http://javascriptissexy.com/category/javascript/) / Tagged: [Constructor](http://javascriptissexy.com/tag/constructor/), [Learn JavaScript](http://javascriptissexy.com/tag/learn-javascript/), [Prototype](http://javascriptissexy.com/tag/prototype/)

[Richard](http://javascriptissexy.com/author/richardb/)Thanks for your time; please come back soon. Email me here: javascriptissexy at gmail email, or use the [contact](http://javascriptissexy.com/contact/) form.

151 Comments

1. http://2.gravatar.com/avatar/e744f9aa7a2ef1fba968d71eb8810280?s=60&d=mm&r=g

keke

[February 10, 2013 at 7:28 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-2146) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=2146#respond)

well, very great, I was stuck by prototype half years ago, which made me stop learning programming. After learning so basic Python , checking the OOP concept that sparked me a litter ,now I restart my JS way. Not until after reading your article can I have so good understanding about the Prototype and OO concept . Thank you very much

1. http://0.gravatar.com/avatar/68eb051798b88732ed0c146919294d00?s=60&d=mm&r=g

Alex

[February 11, 2013 at 5:05 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-2172) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=2172#respond)

In the prototype example where you add the “amIOrganic” method to the Plant prototype, aren’t you in fact actually clobbering the first prototype object that had the showNameAndColor method with a new prototype object? To read the code it would seem you actually create and assign two distinct object literal objects and then assign them to the prototype property of Plant. Wouldn’t the second assignment clobber the first assignment in this case rather than actually adding a method to it?

* + http://0.gravatar.com/avatar/36a5db8b390708fe5494a9ef599857e8?s=60&d=mm&r=g

Amanda

[December 23, 2013 at 5:13 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-174043) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=174043#respond)

When he created amIOrganic like so:

Plant.prototype.amIOrganic = function() {…};

then the “amIOrganic” property of the prototype was set to the function defined after the = sign. However, this did not affect the showNameAndColor property because that is a separate property on the prototype. They are both part of the Plant prototype, but I think you are mistakenly assuming only one or the other can be the Plant prototype.

In the example Fruit.prototype = new Plant(), whatever properties previously existed on the Fruit prototype were overridden. In that case, the entire Fruit prototype was explicitly set to the Plant prototype.

To override(or initialize) a specific prototype property, you would specifically set that property like so:  
Plant.prototype.amIOrganic = function() {…};

1. http://0.gravatar.com/avatar/6e468a695df4050f7d9f76b9d7b4e10b?s=60&d=mm&r=g

Owen

[February 13, 2013 at 2:02 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-2263) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=2263#respond)

I’ve got pretty much the same question as Alex.

I put your plant example in jsFiddle, and amIOrganic seems to be overwriting showNameAndColor.

When I delete your bottom console.log, and replace it with:  
aBanana.amIOrganic();  
aBanana.showNameAndColor();  
amIOrganic() works fine, but there is no function called showNameAndColour();

How do you add both of these functions to Plant prototype?

* + http://0.gravatar.com/avatar/f5bbd4b34802ff5d87da29558db6255c?s=60&d=mm&r=g

alfin

[April 29, 2015 at 5:39 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306684) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306684#respond)

you can do it this way:

Plant.prototype = {  
showNameAndColor: function () {  
console.log(‘I am a ‘ + this.name + ‘ and my color is ‘ + this.color);  
},  
amIOrganic: function() {  
if(this.isOrganic) {  
console.log(‘I am organic, baby!’);  
}  
}  
}

1. http://0.gravatar.com/avatar/6e468a695df4050f7d9f76b9d7b4e10b?s=60&d=mm&r=g

Owen

[February 13, 2013 at 2:15 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-2268) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=2268#respond)

Well I learnt something new today, I managed to get the plant code working:

function Plant () {  
this.country = “Mexico”;  
this.isOrganic = true;  
}

Plant.prototype.showNameAndColor = function () {  
console.log(“I am a ” + this.name + ” and my color is ” + this.color);  
}

Plant.prototype.amIOrganic = function () {  
if (this.isOrganic)  
console.log(“I am organic, Baby!”);  
}

function Fruit (fruitName, fruitColor) {  
this.name = fruitName;  
this.color = fruitColor;  
}

Fruit.prototype = new Plant ();

var aBanana = new Fruit (“Banana”, “Yellow”);

console.log(aBanana.name);  
aBanana.amIOrganic();  
aBanana.showNameAndColor();

1. http://1.gravatar.com/avatar/7e11592034f51eee225aa1f7463b9acb?s=60&d=mm&r=g

Ben

[February 20, 2013 at 10:02 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-2657) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=2657#respond)

First off, your site is awesome. I’ve been using javascript for five years, but have never grasped it as much as I have now. That being said, I am still wholly confused.

I came here after reading chapter six of “Professional Javascript” (following the track suggested in the other post on JiS). That chapter confused me to no end – not about the above material – but rather the sections concerning “Combination Constructor/Prototype Pattern” and “Prototypal Inheritance”. Pretty confusing, as well as a LOT of typing to define objects. I can tell this is a stumbling block for many programmers because there are SO many inheritance libraries out.

Ah – going to re-read the chapter I suppose.

* + http://0.gravatar.com/avatar/912ddb0dd796437303d60f5b72340677?s=60&d=mm&r=g

Ramon Royo

[March 1, 2013 at 2:21 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-3277) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=3277#respond)

Indeed it’s a confusing theme, I printed a summary of that chapter (and another for inheritance) and I keep them both close for re-reading now and then.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[March 1, 2013 at 5:43 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-3282) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=3282#respond)

Ben and Ramon,  
You guys don’t have to worry too much with Combination Constructor/Prototype Pattern topics. In the new, updated “Learn JS Properly” roadmap, we are skipping some of that stuff, since they are not necessary.

Those topics are covered in the advanced JavaScript roadmap.

1. http://0.gravatar.com/avatar/931ecea561d31ae4ffea8192192db591?s=60&d=mm&r=g

Aarsh Patel

[February 21, 2013 at 1:18 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-2736) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=2736#respond)

I got the plant example working on my computer.

function Plant(){  
this.country = “Mexico”;  
this.isOrganic = true;  
}  
Plant.prototype = {  
showNameAndColor : function(){  
console.log(“I am a ” + this.name + ” and my color is ” + this.color);  
},  
AmIOrganic : function(){  
if(this.isOrganic){  
console.log(“I am organic, baby”);  
}  
}  
};  
function Fruit(fruitName, fruitColor){  
this.name = fruitName;  
this.color = fruitColor;  
}  
Fruit.prototype = new Plant();  
//fruit inherist all methods and properties of plant

var aBanana = new Fruit(“Banana”,”Yellow”);  
console.log(aBanana.name);  
console.log(aBanana.showNameAndColor());

I just put the two methods in on one Plant.prototype statement. Am i allowed to do that?

* + http://1.gravatar.com/avatar/a21c1c9037b94cebce07b47fde338449?s=60&d=mm&r=g

Shesh

[May 12, 2015 at 7:57 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306730) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306730#respond)

Great Aarsh, we can mix and match a lot of things.

we have to remember grouping with prototype uses JSON like : syntax and ends statements with , except last statement,  
and the function style use usual = and semicolons at the end of statement.  
A complete example with function overriding, which I hope is right. Tested and works in jsbin

function Plant(){

}

Plant.prototype = {

country: “Mexico”,  
isOrganic:true,

showNameAndColor: function(){  
console.log(“I am a “+ this.name + ” and my color is” + this.color);  
},

amIOrganic : function(){  
if(this.isOrganic){  
console.log(“I am organic, baby”);  
}  
}  
};

function Fruit(fruitName, fruitColor){  
this.name = fruitName;  
this.color = fruitColor;

//remove comments to this function to test function overriding.  
/\*this.amIOrganic = function (){  
console.log(“I am not organic, baby”);  
};\*/

}

//fruit inherits all methods and properties of plant  
Fruit.prototype = new Plant();

//create fruit object and test various variables and calls.  
var aBanana = new Fruit(“Banana”,”Yellow”);  
console.log(aBanana.name);  
console.log(aBanana.country);  
console.log(aBanana.amIOrganic());  
console.log(aBanana.showNameAndColor());

1. http://2.gravatar.com/avatar/5da4dc73f8e2d4824c0454954cef8b18?s=60&d=mm&r=g

Noah Jerreel Guillen

[March 18, 2013 at 12:14 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-4136) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=4136#respond)

This is great. I want to Master JS, and your site is my primary road map. Thanks. This really helps a lot.

@Aarsh Patel. I think so. I also did it. I think it is because it overwrites the previous prototype.

1. http://2.gravatar.com/avatar/2454d5da013ccb561721093f0224f024?s=60&d=mm&r=g

Phan Thanh

[April 23, 2013 at 10:49 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6272) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6272#respond)

Thanks a lot for your site!

I used your example Fruit on my computer(I use Chrome console to test) but I cannot call the function showNameAndColor. I don’t understand why. What may be problems?

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[April 25, 2013 at 8:37 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6407) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6407#respond)

Phan, That was my mistake, I had a typo in the code, which I just fixed.

I am very happy you have brought this to my attention so that I can fix it. You should be able to run the code now without any problems.

1. http://2.gravatar.com/avatar/2454d5da013ccb561721093f0224f024?s=60&d=mm&r=g

Phan Thanh

[April 23, 2013 at 10:51 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6273) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6273#respond)

The result of showNameAndColor function is Undefined.

Please help me for this. Thank you!

* + http://2.gravatar.com/avatar/5da4dc73f8e2d4824c0454954cef8b18?s=60&d=mm&r=g

Noah Jerreel Guillen

[April 25, 2013 at 12:34 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6389) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6389#respond)

Hello Phan Thanh, If you will notice the example above, Plant.prototype was declared twice, this means that the first declaration was overriden by the second declaration What you need to do is to put the 2 functions in one declaration and separate it with comma. 😀

* + - http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[April 25, 2013 at 8:47 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6410) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6410#respond)

Noah,

I really appreciate your comment to Phan. I am happy that you responded to him to help him. And you are spot on: I mistakenly overwrote the Prototype twice, which was not my intent—it was a typo.

I just fixed the code.

* + - * http://2.gravatar.com/avatar/5da4dc73f8e2d4824c0454954cef8b18?s=60&d=mm&r=g

Noah Jerreel Guillen

[August 1, 2013 at 8:54 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-23909) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=23909#respond)

Richard,

No problem. No worries, we all have sometimes commit a typo error. But overall, you’re blogs are really amazing. This really helped me a lot, the prototyping, closures and objects.

* + - * + http://1.gravatar.com/avatar/afba53f435099315a922f88a9f8b36f2?s=60&d=mm&r=g

Somm

[February 9, 2015 at 4:43 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306089) /

this is ok

1. http://2.gravatar.com/avatar/2454d5da013ccb561721093f0224f024?s=60&d=mm&r=g

Phan Thanh

[April 25, 2013 at 11:41 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6414) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6414#respond)

Thank you very much, Rechard and Noah!

This site is very useful and wonderful because it provided a clear roadmap to learn JS with much resources. I am studying with this roadmap efficiently :). Once again many thanks!

Regards,

Phan Thanh

1. http://2.gravatar.com/avatar/584166f7a75ff123e91708cf4727a44a?s=60&d=mm&r=g

[Vic](http://www.mind-mapping.org/)

[April 29, 2013 at 5:42 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6622) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6622#respond)

Hi Richard,

I’m confused! (It’s not hard to confuse me).

I was working through your excellent “JavaScript Objects in Detail” and soon reached 2. Prototype Pattern which told me “you should already understand JavaScript’s prototype. If you don’t, read my post JavaScript Prototype in Plain, Detailed Language”. So I broke off and came here, but found in the second paragraph “If you aren’t already familiar with objects, you should read my post JavaScript Objects in Detail.”

Is it best to complete “JavaScript Objects in Detail” first, or carry on with “JavaScript Prototype in Plain, Detailed Language”? I’ve already worked through the Codecademy part that included prototypes.

Thanks for all your work on these posts, and for the effort to answering dumb questions like this one.

Vic

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[April 29, 2013 at 6:41 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6627) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6627#respond)

Thanks for bringing this to my attention, Vic. You are correct, I do have those two paradoxical statements. I will update both posts today and fix the issue.

* + - http://2.gravatar.com/avatar/8355128ed3e985c3b193ab5a4f47e679?s=60&d=mm&r=g

Nam

[January 4, 2014 at 7:41 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-176593) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=176593#respond)

And You didnt fix it, now im stuck in a loop!

* + - * http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[January 6, 2014 at 11:55 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-177446) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=177446#respond)

Thanks, nam. I just updated the post.

The answer to the question, which I just realized I did not answer, is this: Study “JavaScript Objects in Detail” first.

1. http://0.gravatar.com/avatar/97e7e4a1bcfc72c90289295619ea3444?s=60&d=mm&r=g

Pete

[April 30, 2013 at 11:29 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6735) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6735#respond)

Started with reading about node, kept finding more and more outstanding articles.

3 hours later I find myself full of way more knowledge than I started and it’s thanks to this amazing website.

Thank you Richard for taking the time to not only detail things, but link back to previous articles you’ve written.

out-freaking-standing.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[April 30, 2013 at 11:32 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6736) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6736#respond)

Pete,

Thank you very much. You brought a huge smile to my face with your ama-freaking-zing comment 

1. http://0.gravatar.com/avatar/08cd194064137ec206b15b70fdfbd642?s=60&d=mm&r=g

Connor

[May 4, 2013 at 11:20 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6948) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6948#respond)

Several paragraphs in:  
“You will likely never used this \_\_proto\_\_”. \*use  
Hopefully, this is more helpful than annoying. Your content is the best I’ve seen on JS.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[May 5, 2013 at 1:33 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-6979) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=6979#respond)

Thank you very much, Connor. I am very happy and thankful that you have reported this typo, so definitely not annoying.

I look forward to your finding and reporting more 

1. http://2.gravatar.com/avatar/2ac965fabf6711dd734b8c8b720547af?s=60&d=mm&r=g

Willson

[May 14, 2013 at 3:22 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-7826) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=7826#respond)

Hi Richard,

In your example: console.log (famousPerson.athlete), you seemed to explain that famousPerson.athlete returned “Tiger Woods” because the athlete property was defined in People.prototype. However, is that accurate? Based on what you wrote, I would think that the famousPerson object itself has the athlete property defined on it.

On the other hand, the toString method makes sense since toString wasn’t defined on the famousPerson object itself. It will then look at People.prototype and since it’s not defined there either, it will finally look at Object.prototype.

Thanks,  
Willson

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[May 14, 2013 at 11:23 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-7846) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=7846#respond)

HIi Wilson,

Thanks for your comment.

*In your example: console.log (famousPerson.athlete), you seemed to explain that famousPerson.athlete returned “Tiger Woods” because the athlete property was defined in People.prototype. However, is that accurate? Based on what you wrote, I would think that the famousPerson object itself has the athlete property defined on it.*

You are absolutely correct, and I am very glad you found this mistake, so I can fix it before it confuses other readers.

* + - http://2.gravatar.com/avatar/2ac965fabf6711dd734b8c8b720547af?s=60&d=mm&r=g

Willson

[May 14, 2013 at 11:31 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-7847) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=7847#respond)

Thanks for confirming Richard! Awesome blog and learning tons from you!

* + - * http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[May 14, 2013 at 11:46 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-7848) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=7848#respond)

I just fixed the issue. I had forgotten to add the athlete property on the People Prototype:

|  |  |
| --- | --- |
|  | People.prototype.athlete = "Tiger Woods"; |
|  | ​ |

I am glad you like the blog. Thanks.

1. http://0.gravatar.com/avatar/3c711e7ea5df27618ee9be222a3c1ef6?s=60&d=mm&r=g

Tige

[May 23, 2013 at 6:41 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-8808) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=8808#respond)

Richard,

I agree with previous posts. I have been developing javascript code for a decade now. As a java developer, I never really understood how to apply the OO techniques I use there and apply them to javascript until this post. I have found success relying on js frameworks (Sencha, Dhtmlx) and on memorizing some of the prototype concepts without really understanding how/why it works. Thanks alot for the much needed clarity.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[May 27, 2013 at 10:35 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-9480) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=9480#respond)

You are welcome, Tige.

1. http://0.gravatar.com/avatar/376a6a2cb18b7a26bc437119bb77358c?s=60&d=mm&r=g

Nivla

[July 13, 2013 at 1:20 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-19662) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=19662#respond)

My dude is this.  
which is the diferrence if i do this.  
function Person(name,sex){  
this.name=name;  
this.sex=sex;  
}  
Person.prototype.getName= function() { return this.name}  
var Juan = new Person(“Juan”; “M”);  
Juan.getName(); //Juan

If i can do this too.  
function Person(name,sex){  
this.name=name;  
this.sex=sex;  
this.getName = .getName= function() { return this.name};  
}

var Juan = new Person(“Juan”; “M”);  
Juan.getName(); //Juan  
which is the adventaje of use prototype in this case?

Thanks!

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[July 17, 2013 at 3:15 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-20300) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=20300#respond)

I answered this exact question in the “OOP in JavaScript” post. I repost my answer below, but you can read more [“here](http://javascriptissexy.com/oop-in-javascript-what-you-need-to-know/%20rel=).

Here is my answer to another user with the same question from the OOP in JS post:  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Good question, Jurgen.

The two examples will function exactly the same with your test code. But they are fundamentally different.

In the former, all the properties and methods are defined on the instance, so the values will be unique to each instance. You can see the issue better if we use a reference type (Array or Objects). I changed your code and made variable2 an array.

|  |  |
| --- | --- |
|  | ​ |
|  | ​// Constructor Pattern​ |
|  | ​function Test(theVariable) { |
|  | this.variable = theVariable; |
|  | this.variable2 = ["Mike", "Anil"]; |
|  | this.someMethod = function () { |
|  | console.log(this.variable2 + this.variable); |
|  | } |
|  | } |
|  | ​ |
|  | ​var aTest = new Test ("-- Testing"); |
|  | aTest.variable2.push("Richard"); |
|  | aTest.someMethod(); // Mike,Anil,Richard -- Testing​ |
|  | ​ |
|  | ​var anotherTest = new Test ("-- Testing"); |
|  | anotherTest.variable2.push("Jurgen"); |
|  | ​// Array is unique to the anotherTest instance​ |
|  | anotherTest.someMethod(); //Mike,Anil,Jurgen-- Testing​ |
|  | ​ |
|  | ​// variable2 was not changed by the anotherTest instance:​ |
|  | aTest.someMethod(); // Mike,Anil,Richard-- Testing​ |
|  | ​ |

In your latter example, however, only this.variable is an instance variable, so it will be unique always. All the properties and methods defined on the prototype will be inherited by all instances of Test, and all instances will have the same values (for reference types) for those. For example:

|  |  |
| --- | --- |
|  | ​ |
|  | ​function Test(theVariable) { |
|  | this.variable = theVariable; |
|  | } |
|  | ​ |
|  | Test.prototype = { |
|  | constructor:Test, |
|  | variable2:["Mike", "Anil"], |
|  | someMethod:function () { |
|  | console.log(this.variable2 + " " + this.variable); |
|  | } |
|  | } |
|  | ​ |
|  | ​ |
|  | ​var aTest = new Test ("-- Testing"); |
|  | aTest.variable2.push("Richard"); |
|  | aTest.someMethod(); // Mike,Anil,Richard -- Testing​ |
|  | ​ |
|  | ​var anotherTest = new Test ("-- Testing"); |
|  | anotherTest.variable2.push("Jurgen"); |
|  | ​ |
|  | ​// Uses the same array from the prototype (Richard is included)​ |
|  | anotherTest.someMethod(); // Mike,Anil,Richard,Jurgen -- Testing​ |
|  | ​ |
|  | ​ |
|  | ​// variable2 changed on ALL instances:​ |
|  | aTest.someMethod(); // Mike,Anil,Richard,Jurgen -- Testing​ |
|  | ​ |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - http://0.gravatar.com/avatar/376a6a2cb18b7a26bc437119bb77358c?s=60&d=mm&r=g

Nivla

[July 18, 2013 at 7:41 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-20418) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=20418#respond)

Thank a lot! Very usefull, I understand.

* + - http://1.gravatar.com/avatar/1aa42ddf36b53a15f1ae82a4e582d28c?s=60&d=mm&r=g

Pravin Waychal

[January 18, 2014 at 10:14 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-180128) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=180128#respond)

Thanks a Lot! Even I had same question

1. http://1.gravatar.com/avatar/a3f9aabbf738531844b58821699d129e?s=60&d=mm&r=g

Nick

[July 13, 2013 at 4:11 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-19681) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=19681#respond)

Great article! Very helpful. Now I really understand the concept.

Thank you.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[July 17, 2013 at 3:16 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-20301) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=20301#respond)

Thanks, Nick.

1. http://0.gravatar.com/avatar/95f62e0517d986546ef51aa28ba56fd2?s=60&d=mm&r=g

JavaScriptLearner

[August 8, 2013 at 2:14 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-26077) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=26077#respond)

Thanks for the great article!!

One thing.

In the last example:

function People () {  
this.superstar = “Michael Jackson”;  
}  
// Define “athlete” property on the People prototype so that “athlete” is accessible by all objects that use the People () constructor.

People.prototype.athlete = “Tiger Woods”;

var famousPerson = new People ();  
famousPerson.superstar = “Steve Jobs”;

Superstart and athlete(prototype.athlete). Whats the difference between these two as both could be inherited.  
Both could be used by the object created by the People() constructor?

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[August 9, 2013 at 8:03 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-26851) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=26851#respond)

That is a good question.

Whenever a property or method is defined with “this,” as in the this.superstar = “Michael Jackson” example, the property is literally created every time a new instance is created.

For example:

|  |  |
| --- | --- |
|  | ​ |
|  | ​function People () { |
|  | ​this.superstar = "Michael Jackson"; |
|  | } |
|  | ​ |
|  | ​// When we create a new instance of People, as in this example below.​ |
|  | ​// The "superstar" property is created as a new property on the famousPerson object​ |
|  | ​var famousPerson = new People (); |
|  | ​ |

And for every instance of People you create, a new superstar property will be created.

Ordinarily, it is not best to define a property using *this* like we have it on the People constructor. Instead, this is how you would normally define use “this”:

|  |  |
| --- | --- |
|  | ​ |
|  | ​function People (superstarName) { |
|  | ​this.superstar = superstarName; |
|  | } |
|  | ​ |

On the other hand, any property defined on the prototype is indeed inherited and not created as a new property on the instance.

For a more detailed explanation of the difference between the use of *this* and the prototype to define properties and methods, see my answer above that to Nivla.

1. http://2.gravatar.com/avatar/5b777114c61182df62f5798094460dc9?s=60&d=mm&r=g

Bob

[August 19, 2013 at 6:39 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-30620) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=30620#respond)

Hi Richard

I have some PHP under my belt from working with several popular CMS’s. I decided to tackle learning JavaScript in more depth and went through ALL of CodeAcademy’s JS courses, but really didn’t have a handle on objects & constructors until I found this.

Thanks so much! Can’t wait to dig into the rest of your stuff here.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[August 21, 2013 at 12:55 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-31223) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=31223#respond)

You are welcome, Bob.

I am glad to hear this roadmap will be helpful for you.

1. http://0.gravatar.com/avatar/c2d9dfffff6c2d2be4c09997bc939929?s=60&d=mm&r=g

Dattatray Walunj

[September 12, 2013 at 6:27 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-37153) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=37153#respond)

Hi Richard,  
Great post. You have mentioned in the above post that ” the compiler will then search for it on the myFriends prototype”, I think it’s a typo or something as JavaScript is interpreted language.

Correct me if I am wrong.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[September 16, 2013 at 1:54 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-38345) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=38345#respond)

Well, this is a bit more complicated than it appears.  
First, JavaScript is widely known as an interpreted language for the most part, probably because conventionally this was how the browsers executed the code.

However, many modern JavaScript Host Environments (such as V8 in Chrome) in fact compile JavaScript. I have to research this some more to find out which specific implementations compile as opposed to interpret, but the crux of the matter is that we sometimes use “JavaScript Compiler” or “JavaScript Engine” or even “JavaScript Host Environment.”

The most common usage I have seen is “JavaScript Engine,” when referring to the browsers.

* + - http://0.gravatar.com/avatar/c2d9dfffff6c2d2be4c09997bc939929?s=60&d=mm&r=g

Dattatray Dnyaneshwar Walunj

[September 16, 2013 at 3:11 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-38369) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=38369#respond)

Hi Richard,  
Thanks for the explanation. This clarifies my doubt.

1. http://2.gravatar.com/avatar/5ce29d8776cf2a09f51689dcdbcd6775?s=60&d=mm&r=g

[archit](https://web.iiit.ac.in/~archit.mehta/)

[October 4, 2013 at 2:10 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-68640) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=68640#respond)

as you mentioned all function in javascripts are object  
what i want to know is what object itself mean , means when I can say it is(var or function) object.

normal definition of object is , for example in c++ or java when some instance of particular class call constructor and allocates memory that it is called an object and now it can access all things of class.

if you say object is object , var is also object an and function is also an object so what common things they have so that we call function an object?

* + http://0.gravatar.com/avatar/c2d9dfffff6c2d2be4c09997bc939929?s=60&d=mm&r=g

[Dattatray Dnyaneshwar Walunj](http://www.dattawalunj.com/)

[October 12, 2013 at 1:49 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-87605) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=87605#respond)

HI Archit,  
Like other languages with Classical inheritance the hierarchy in JavaScript is as follows:

See String, Date, Function, Number all are objects that means all are inherited from ‘Object’ class. Object is base class. Object has some properties which could be methods or variables.

For example: Object has a method toString() and a variable i.e. “length”, now String is inherited from Object so String also has these two properties i.e. toString() and length, in addition String has some extra methods which Object doesn’t like “toUpperCase()” and “toLowerCase()”.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 15, 2013 at 8:26 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-95960) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=95960#respond)

Good question, Archit.

And thanks much for helping out, Dattatray. You are spot on.

Archit, the reason functions are objects in JavaScript is because functions are actually derived from the Object Class (for purposes of a Java comparison).

So you could say that function is an instance of the Object Class. Function inherits properties and methods from the Object. But technically, JavaScript doesn’t have Classes, so the it is the Object constructor, not Class.

Read more here:  
[JavaScript Prototype in Plain Language](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/)

1. http://1.gravatar.com/avatar/1dc7fa252af037115367101039531b02?s=60&d=mm&r=g

Shubham

[October 22, 2013 at 2:39 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-109870) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=109870#respond)

Hi, I am just starting out on Object Oriented Javascript. Just had a quick question.  
Although i am able to see the results in my console. I am also getting an “undefined” along with the desired results.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 23, 2013 at 3:25 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-112203) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=112203#respond)

Oh, good question, Shubham. Undefined is normally returned by the browser’s console when you are not executing a function that returns a value or when you just type a statement in the console.

It is normal. Just ignore it.

The only time it is important is when you expect your function to return some value and it returns undefined instead.

For example, this function should return “sweet”.

function sweetFun () { return “sweet”}

After typing that function in the browser’s console, you will see “undefined” because we simply just declared a function—write a statement. We didn’t actually call the function yet, and that is normal behavior for the console, which I tend to disagree with, btw.

Now, when you actually call the function:  
sweetFun ()  
it will output “sweet” and you should not see undefined.

1. http://1.gravatar.com/avatar/1dc7fa252af037115367101039531b02?s=60&d=mm&r=g

Shubham

[October 22, 2013 at 2:44 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-109878) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=109878#respond)

Forgot to mention that the Above Question is related to the Plant, Fruit and a Banana example for Prototype.

* + http://1.gravatar.com/avatar/1dc7fa252af037115367101039531b02?s=60&d=mm&r=g

Shubham

[October 22, 2013 at 3:57 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-110022) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=110022#respond)

It was a Typo from my side.  
Apologies.

* + - http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 23, 2013 at 3:26 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-112204) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=112204#respond)

Oops, I answered your question before I read these last two comments. No problem.

* + - * http://1.gravatar.com/avatar/1dc7fa252af037115367101039531b02?s=60&d=mm&r=g

Shubham

[October 23, 2013 at 4:19 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-112296) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=112296#respond)

Well actually what you explained was exactly the cause of the Undefined coming in my program. Thanks for explaining

* + - * + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 29, 2013 at 2:36 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-124275) /

Good to know. You are welcome.

1. http://2.gravatar.com/avatar/8244074e29a58e7e2f8ed752e2785b3d?s=60&d=mm&r=g

Evan Hobbs

[November 5, 2013 at 8:36 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-140140) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=140140#respond)

This is truly excellent. I’ve set myself the challenge of working through one of the ’16 Javascript Concepts’ each morning before work…

One thing that I’m confused by is the difference between the constructor prototype obj and the prototype property. For instance (in Chrome console):

Obj = function() {};  
Obj.prototype.name = ‘test';  
var instance = new Obj();

console.log(Obj.prototype) // {name: ‘test’}  
console.log(instance.prototype) // undefined

Should the instance prototype attribute point to the Obj constructor? Maybe I’m reading that wrong but I’m not sure why instance.prototype would be undefined…

Thanks for any help!

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[November 10, 2013 at 2:30 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-149566) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=149566#respond)

Very good question, Evan.

As I noted in the article, the prototype property is not accessible, although some browsers allow you to access it via the \_\_proto\_\_ property. In fact, if you do this:  
console.log(instance.\_\_proto\_\_) you can see the results of your instance obj’s prototype property.

Read this for more:

*First, there is a prototype property that every JavaScript function has (it is empty by default), and you attach properties and methods on this prototype property when you want to implement inheritance.Note that this prototype property is not enumerable: it is not accessible in a for/in loop. But Firefox, and most versions of Safari and Chrome, have a \_\_proto\_\_ “pseudo” property (an alternative way) that allows you to access an object’s prototype property. You will likely never use this \_\_proto\_\_ pseudo property, but know that it exists and it is simply a way to access an object’s prototype property in some browsers.*

1. http://0.gravatar.com/avatar/96871daee4de3c8175ced641065b2305?s=60&d=mm&r=g

diavel

[November 12, 2013 at 6:25 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-154077) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=154077#respond)

I guess this tutorial is the only one where i came across the clear diff between prototype property and attribute.. its only den everything was clear.. awesome post.. Thank u very much for this write up

1. http://1.gravatar.com/avatar/492686b4403c2232cd520126484cec60?s=60&d=mm&r=g

Serge

[December 5, 2013 at 8:22 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-168479) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=168479#respond)

Hi! Thanks for nice explanation of prototype. I wonder why in this example call() to print is not working:  
function PrintStuff(myDocuments){  
this.documents = myDocuments;  
}  
PrintStuff.prototype.print = function(){  
console.log(this.documents);  
}  
var newObject = new PrintStuff(“I am a nw Object and I can print.”);  
newObject.print.call();

* + http://1.gravatar.com/avatar/1aa42ddf36b53a15f1ae82a4e582d28c?s=60&d=mm&r=g

Pravin Waychal

[January 18, 2014 at 10:32 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-180136) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=180136#respond)

I guess, instead of newObject.print.call(); you need to write newObject.print(); to call function

1. http://2.gravatar.com/avatar/e1036b9e1fcadf79f091ac7b9a62def5?s=60&d=mm&r=g

Tanvi P

[December 13, 2013 at 2:17 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-171214) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=171214#respond)

Kudos! Excellent tutorials

1. http://2.gravatar.com/avatar/583e77939c665800d4217279b17cc20c?s=60&d=mm&r=g

devi lal

[January 5, 2014 at 12:59 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-176828) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=176828#respond)

great example to clear prototype. i am gratitude to you share meaningful knowledge.

1. http://0.gravatar.com/avatar/9120ee9afdf1eb724aa557fc75700cb9?s=60&d=mm&r=g

Seung Lim

[February 7, 2014 at 11:04 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-182901) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=182901#respond)

Thank you for the tutorial. You make it so easy to understand.  
I think there will be more programmers in this world, if we have full of teachers like you who can really teach!!

1. http://0.gravatar.com/avatar/f71b99e974523cc50588f507ab9cecff?s=60&d=mm&r=g

UKONN

[February 19, 2014 at 12:29 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-184663) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=184663#respond)

is this??  
“All built-in constructors (Array (), Number (), String (), etc.) were created from the Object constructor, and as such their prototype is Object.prototype.”

because…

Array.constructor === Function.prototype.constructor; //true  
Array.\_\_proto\_\_ === Function.prototype; //true  
Array.prototype === Object.prototype; //false  
typeof Array.prototype; //”object”

1. http://0.gravatar.com/avatar/3cd0c27610b971fcb18352be0577a02f?s=60&d=mm&r=g

Celine

[February 24, 2014 at 8:19 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-185496) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=185496#respond)

Would you have any resources to elaborate on the quote “since all functions in JavaScript are objects” in your text please ?

Also you mention about a creating a post on ECMAScript 5 in a few of your articles but I was not able to retrieve one. Would you have a good resource on this too (web link or book).

They are two topics I am trying to get a better gage on. Much appreciated and thank you for all your great posts !

Celine

1. http://2.gravatar.com/avatar/e12fcd2a1ec351cacddda5c786fda3a5?s=60&d=mm&r=g

Danny

[March 26, 2014 at 11:52 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-192174) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=192174#respond)

var foo={  
name:’foo’,  
display:function(){  
console.log(this.name);  
}  
};  
var foo2 ={  
name:’foo2′  
};  
Here are two objects that are created with the object literal, i want foo2 inherits display method from foo, then invoke like foo2.display(), how can i implement? Thank you in advance.

* + http://1.gravatar.com/avatar/d66ca1712950078c17e0c85f0286524f?s=60&d=mm&r=g

Ruben

[April 13, 2014 at 1:00 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-198115) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=198115#respond)

var foo={  
name:’foo’,  
display:function(){  
console.log(this.name);  
}  
};

var foo2 = Object.create(foo, {  
name: {  
configurable: true,  
enumerable: true,  
value: ‘foo2′,  
writable: true  
}  
});

foo2.display();

1. http://1.gravatar.com/avatar/d16e7a09c9fd6fe40d0f997a5025088d?s=60&d=mm&r=g

Vipin

[April 23, 2014 at 8:45 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-206443) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=206443#respond)

brilliant !

1. http://1.gravatar.com/avatar/7230f1b2049d4803c2909996836c4438?s=60&d=mm&r=g

bill

[May 23, 2014 at 6:41 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-234687) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=234687#respond)

If you still don’t understand this “prototype” concept, just think of the “prototype” word as “parent”. For example, like Richard’s example about the Plant constructor and Fruit Constructor which creates the aBanana object, Richard said the aBanana’ s prototype attribute(or prototype object) is Fruit.prototype and Fruit inherits that prototype property from Plant constructor, you could have put it this way, aBanana’s parent is Fruit and Fruit inherits all the properties and methods from its parent also which is Plant() constructor. P/S :I didn’t know what a property was, but after reading this, I understand this concept and just want to explain it in plainer language to you guys, all credit going towards Richard still.

1. http://2.gravatar.com/avatar/e7df25fd9d495d4fc29c914cd2697656?s=60&d=mm&r=g

Vandana

[June 15, 2014 at 12:11 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-245758) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=245758#respond)

Hi Richard:

Thanks for writing the basic concepts of prototype for us.

I have a very basic doubt here. I don’t understand the purpose of assigning a method or property to a prototype when we can directly assign it to the function. For example:

function check (){  
this.a = 4;  
this.b = true;  
this.get = function get(){  
alert(this.a);  
}  
}

//now i assign a method to the prototype

check.prototype.see = function(){  
alert(this.b);  
}

//now if i create a new object of the type check

var obj = new check();  
check.see(); //assigned to prototype  
check.get(); //assigned directly to function

what is the difference in assigning a property to a function directly and to its prototype? Any new object created from check can access both the methods.

Can you please help me with this doubt?

Thanks

* + http://2.gravatar.com/avatar/5da4dc73f8e2d4824c0454954cef8b18?s=60&d=mm&r=g

Noah Jerreel Guillen

[June 23, 2014 at 3:04 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-249379) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=249379#respond)

Hello Vandana,

The difference between is that when you assign a method on a function, you are like just creating a static method, which means the method attached to this function will get re-declared for every new instance we create, while when you create a method using the prototype, the method will be declared once and shared to all instances.

Cheers!

* + - http://0.gravatar.com/avatar/34556e3808e3a027e3a40a7998216143?s=60&d=mm&r=g

shanmughasunder

[August 5, 2014 at 10:31 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-267505) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=267505#respond)

Hi Noah,  
And to add is’nt that the objects initialized from check cannot use the methods defined on this check object.  
For Example:

function check (){  
this.a = 4;  
this.b = true;  
this.get = function get(){  
alert(this.a);  
}  
}  
var test = new check();  
Is now “test” object can use the methods of check object.

Am I right.

* + - * http://2.gravatar.com/avatar/5da4dc73f8e2d4824c0454954cef8b18?s=60&d=mm&r=g

Noah Jerreel Guillen

[August 25, 2014 at 9:28 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-276730) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=276730#respond)

Hello Vandana,

No, the objects initialized to check is still usable when you add new methods.

1. http://2.gravatar.com/avatar/23e6d13a930ff10c0f42f1cd8deeaf96?s=60&d=mm&r=g

ritika

[August 12, 2014 at 2:30 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-269818) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=269818#respond)

hi,

awesome and well written blog

* + http://2.gravatar.com/avatar/23e6d13a930ff10c0f42f1cd8deeaf96?s=60&d=mm&r=g

ritika

[August 12, 2014 at 2:35 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-269824) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=269824#respond)

in the below piece of code,Could you please explain why i am not able to access the secondclass prototype property,even on creating new instances?  
function FirstClass()  
{  
this.first = “FirstProperty: you have chosen firstclass”;  
this.second = “SecondProperty: you have chosen firstclass”;  
}

FirstClass.prototype.extendone = function()  
{  
document.getElementById(“demo\_o”).innerHTML = “inside FirstClass prototype : “+this.first+ ” “+ this.second;  
}

function SecondClass(myfirst,mysecond)  
{  
this.first = myfirst;  
this.second = mysecond;  
this.myproperty =”this is the poperty of second class”;  
}

SecondClass.prototype.extendtwo = function ()  
{  
document.getElementById(“demo\_dR”).innerHTML =”new property of secondclass prototype”;  
}

SecondClass.prototype = new FirstClass(); //INHERITING METHODS AND PROPERTIES  
var obj\_first = new SecondClass(“passing myfirst”,”passing mysecond”);  
var new\_obj = new SecondClass();

obj\_first.extendone();  
obj\_first.extendtwo();  
new\_obj.extendtwo();

* + - http://1.gravatar.com/avatar/7267d4a7c726ba3e7aa1d98cf4cae928?s=60&d=mm&r=g

Feng Wang

[August 20, 2014 at 2:13 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-273866) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=273866#respond)

Hi Ritika,

It is the order of the definition of SecondClass.prototype.extendtwo and SecondClass.prototype = new FirstClass().

In JavaScript, prototype property is just an another object. When you define SecondClass.prototype = new FirstClass(), you basically replaced the prototype which hold the extendtwo function to a simple object that constructed by New FirstClass(), which simply does not have the extendtwo function at all.

Reverse the order will make things work as you expected.

I used the following code to test what I said above:

function FirstClass()  
{  
this.first = “FirstProperty: you have chosen firstclass”;  
this.second = “SecondProperty: you have chosen firstclass”;  
}

FirstClass.prototype.extendone = function()  
{  
document.getElementById(“demo\_o”).innerHTML = “inside FirstClass prototype : “+this.first+ ” “+ this.second;  
}

function SecondClass(myfirst,mysecond)  
{  
this.first = myfirst;  
this.second = mysecond;  
this.myproperty =”this is the poperty of second class”;  
}

SecondClass.prototype = new FirstClass(); //INHERITING METHODS AND PROPERTIES

SecondClass.prototype.extendtwo = function ()  
{  
document.getElementById(“demo\_dR”).innerHTML =”new property of secondclass prototype”;  
}

window.onload = function(){  
var obj\_first = new SecondClass(“passing myfirst”,”passing mysecond”);  
var new\_obj = new SecondClass();  
obj\_first.extendone();  
obj\_first.extendtwo();  
//new\_obj.extendone();  
//new\_obj.extendtwo();  
}

Hope this is helpful and happy coding!

* + - * http://2.gravatar.com/avatar/23e6d13a930ff10c0f42f1cd8deeaf96?s=60&d=mm&r=g

ritika

[August 21, 2014 at 12:40 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-274061) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=274061#respond)

thanks feng!!! it works 

1. http://1.gravatar.com/avatar/7652abf905392a4f97381ae18ac6c0c2?s=60&d=mm&r=g

Josh

[September 16, 2014 at 8:36 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-290288) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=290288#respond)

Just wanted to say a very big thank you. Had an awful moment tonight where I began to question everything I ever learnt about JavaScript objects and prototypes….your blog cleared everything up and put me back on track!

Your ability to convey difficult technical concepts in an easily understood way is incredible. Well done!

1. http://0.gravatar.com/avatar/3b4703a756a4b409f8da75560f7b5870?s=60&d=mm&r=g

[คลิปโป๊](http://iamclip.com/)

[October 7, 2014 at 5:29 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-305639) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=305639#respond)

I’m really enjoying the theme/design of your blog. Do you ever  
run into any internet browser compatibility problems? A few  
of my blog audience have complained about my blog not operating correctly in Explorer but looks great in Chrome.

Do you have any solutions to help fix this issue?

1. http://0.gravatar.com/avatar/fa4c72a694f895ff7f221b45d138cb42?s=60&d=mm&r=g

kan

[October 17, 2014 at 5:19 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-305767) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=305767#respond)

Hi,

Sometimes we add properties with use prototype keyword and sometimes add properties without prototype keyword. I want know, how to decide to use keyword protoype when adding the pro[erties and method.  
Have a any criteria to use prototype?

People.prototype.athlete = “Tiger Woods”;  
famousPerson.superstar = “Steve Jobs”;

You can also see the same senario at below link:  
<http://www.w3schools.com/js/js_object_prototypes.asp>

Please suggest me.  
thanks in advance  
kan

1. http://0.gravatar.com/avatar/f7cacc8576940894184064edde30ae20?s=60&d=mm&r=g

NicT

[November 1, 2014 at 8:12 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-305828) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=305828#respond)

Hi and thanks for the article. Hope my questions make sense:

1. Why Fruit.prototype = new Plant (); and not Fruit.prototype = Plant.prototype; ?

2. With Fruit.prototype = new Plant (); When you add more prototype properties to Fruit eg Fruit.prototype.pest=’fly';, is the pest property on the Plant prototype or the Fruit prototype ?

Thanks !

* + http://2.gravatar.com/avatar/5da4dc73f8e2d4824c0454954cef8b18?s=60&d=mm&r=g

Noah Jerreel

[November 2, 2014 at 11:46 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-305832) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=305832#respond)

Hello NicT,

1. Fruit.prototype = new Plant(); means that your are assigning a new plant instance on the fruit prototype and inheriting all the attributes and methods from the Plant class. While Fruit.prototype = Plant.prototype; you are just assigning the Plant prototype and not the whole Plant instance on the Fruit Prototype.

2. The pest property will be both on the Plant and Fruit prototype since you assign the Plant prototype to the Fruit prototype and changing the Fruit prototype will also change the Plant the prototype.

I don’t know If these answers your questions.

* + - http://0.gravatar.com/avatar/f7cacc8576940894184064edde30ae20?s=60&d=mm&r=g

NicT

[November 3, 2014 at 5:01 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-305833) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=305833#respond)

Thanks Noah. Yes you’ve understood and answered very well. The inheritance is of the whole class including non prototype members. Was thinking only the prototype was inheritable. Cheers.

* + - * http://2.gravatar.com/avatar/5da4dc73f8e2d4824c0454954cef8b18?s=60&d=mm&r=g

Noah Jerreel

[November 4, 2014 at 3:34 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-305843) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=305843#respond)

I’m glad I was able to help NicT. Cheers! 

1. http://0.gravatar.com/avatar/cb0135ecf2a19f48470a8817adaca83a?s=60&d=mm&r=g

Binh Nguyen

[November 3, 2014 at 7:56 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-305836) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=305836#respond)

Thanks, nice explanation.

1. http://0.gravatar.com/avatar/c6082d3a8a3316461824328f2d17006b?s=60&d=mm&r=g

[Jason](http://blog.hyphen-ated.com/)

[November 28, 2014 at 2:42 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-305919) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=305919#respond)

I believe

console.log(aBanana.showNameAndColor()); // I am a Banana and my color is yellow.

should be

aBanana.showNameAndColor(); // I am a Banana and my color is yellow.

(The logging is handled in said function.)

1. http://0.gravatar.com/avatar/fa4c72a694f895ff7f221b45d138cb42?s=60&d=mm&r=g

kan

[December 9, 2014 at 5:43 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-305969) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=305969#respond)

You cannot add property or method on existing object like below:-  
function Person(first) {  
this.firstName = first;  
}  
Person.nationality = “English”;  
console.log(Person.nationality): undefined;

If you want to add property or method then you have to create new object like below:-  
var famousPerson = new Person ();  
famousPerson.superstar = “Steve Jobs”;  
console.log(Person.nationality): Steve Jobs;

OR

THROUGH prototype

function Person(first) {  
this.firstName = first;  
}

Person.prototype.nationality = “English”;  
console.log(Person.nationality): English;

1. http://2.gravatar.com/avatar/e6fcc7abd2b83f3541435290b5361044?s=60&d=mm&r=g

[Javier](http://www.javiercruz.cl/)

[December 23, 2014 at 9:47 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306013) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306013#respond)

Thanks, it was usefull for me this post.

One detail, in the last code you show, under “Here is another example of the prototype chain:” the comment says ‘from which the Fruit object inherited’, but it should be say ‘from which the People object inherited’.

// In this example, the search proceeds up the prototype chain and find the toString method on Object.prototype, from which the Fruit object inherited—all objects ultimately inherits from Object.prototype as we have noted before.​  
console.log (famousPerson.toString()); // [object Object]

1. http://0.gravatar.com/avatar/62a836eb3819cfb043447850d383895e?s=60&d=mm&r=g

Gurjit singh

[January 8, 2015 at 1:50 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306033) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306033#respond)

Hi Richard,  
Great tutorial man thanks( actually earlier one are more great ones 😛 just kidding) !  
I have a little confusion on one thing you have mentioned \_\_proto\_\_ pseudo property twice both in prototype property and prototype attribute. How can one thing be pointing to 2 different things ? Surely I am getting it very wrong , plz make me correct here.  
Thanks .

1. http://1.gravatar.com/avatar/156ca7d7818b0dd16c938b0ab9c77548?s=60&d=mm&r=g

jddd

[January 27, 2015 at 8:18 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306070) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306070#respond)

got emm

1. http://1.gravatar.com/avatar/1395e76765668efc0044f086936f0924?s=60&d=mm&r=g

Gerard

[February 22, 2015 at 1:17 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306123) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306123#respond)

I think that in the last comment of your function People() example you wrote Fruit where you intended to write People.prototype.

1. http://2.gravatar.com/avatar/e9253a913228810513ed4761e1d24183?s=60&d=mm&r=g

cody

[April 17, 2015 at 8:49 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306613) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306613#respond)

How come when you call a new instance of another already defined prototype it doesn’t inherit the names assigned to the property or method? Or does it in certain situations.

This is the example.  
function PrintStuff (myDocuments) {  
​this.documents = myDocuments;  
}  
​  
​// We add the print () method to PrintStuff prototype property so that other instances (objects) can inherit it:​  
PrintStuff.prototype.print = function () {  
console.log(this.documents);  
}  
​  
​// Create a new object with the PrintStuff () constructor, thus allowing this new object to inherit PrintStuff’s properties and methods.​  
​var newObj = new PrintStuff (“I am a new Object and I can print.”);  
​  
​// newObj inherited all the properties and methods, including the print method, from the PrintStuff function. Now newObj can call print directly, even though we never created a print () method on it.​  
newObj.print (); //I am a new Object and I can print.

How come that doesn’t inherit the this.documents from that prototype? Is it because of the new Keyword in front of it?  
What would the PrintStuff prototype look like as a new instance being called like this if it was written in full instead?

Thanks

1. http://0.gravatar.com/avatar/c3e3ece4b880363510641ce353483df4?s=60&d=mm&r=g

massimo

[May 22, 2015 at 7:48 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306817) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306817#respond)

Thanks so much, the way you describe this concepts makes me able to understand them.

1. http://2.gravatar.com/avatar/89fa22cdaffd552bd7755d0d01e73f57?s=60&d=mm&r=g

[amit](http://no/)

[May 29, 2015 at 10:25 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306894) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306894#respond)

great ..

1. http://0.gravatar.com/avatar/6a043005903cf5195abac994c413bc4f?s=60&d=mm&r=g

[Jason](http://www.codedcontainer.com/)

[June 1, 2015 at 9:22 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306918) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306918#respond)

Is there a way that you can set up constructors with object literals? Also can you use prototype on object literals? If so, what would this/these look like?

[CodedContainer](http://www.codedcontainer.com/)

1. http://2.gravatar.com/avatar/53234ec4cebbe3ed2737b7cb13f3f42a?s=60&d=mm&r=g

Zinc

[June 4, 2015 at 6:46 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-306951) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=306951#respond)

I have a simple question, looking forward to your answer:  
please see the two code snippets

What are the difference of these two ways of declaring a “class” in javascript?  
From my perspective, they are the same. For any instances that instantiate the PrintStuff and PrintStuffs, print() are available. One is not using prototype. Anyway, I am just confused what the benefit of using the second method.

Thanks a lot.

// no inheritance  
function PrintStuff(doc) {  
this.name = doc;  
this.print = function() {  
console.log(“Printing ” + this.name);  
};  
}

var obj = new PrintStuff(‘without inheritance’);  
obj.print();

// use prototype  
function PrintStuffs(doc) {  
this.name = doc;  
}

PrintStuffs.prototype.print = function() {  
console.log(“Printing ” + this.name);  
};  
var obj = new PrintStuff(‘with inheritance’);  
obj.print();

1. http://0.gravatar.com/avatar/f913f43aecd2032fc16430f4c1266a53?s=60&d=mm&r=g

ABh

[June 21, 2015 at 3:11 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307099) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307099#respond)

same here i have also started reading NODE js and navigated to this article to learn basics of javascrip.  
really helpfull article. earlier i had little idea about Prototype, but now i am confident about it, just need to do some real time example.

1. http://1.gravatar.com/avatar/a10aa324170b48edf372a3b7413e210e?s=60&d=mm&r=g

JaredN

[June 26, 2015 at 1:55 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307127) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307127#respond)

“a property is simply a variable defined on a function”

I’m sorry, but what does this mean? It is completely unclear to me.

I understand that JavaScript functions are objects. Do you mean that a property is an instance variable of that function?

1. http://1.gravatar.com/avatar/7fd150f37cbc1a4009581da8f375823a?s=60&d=mm&r=g

[Jeff Nixon](http://jeffnixoncreative.com/)

[July 20, 2015 at 7:31 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307218) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307218#respond)

Hello, your description helped me make sense of prototypes and I just passed my assignment. Thank you. Based on your description, I noted that a prototype is “The inherent values (properties and methods) that a Constructor is comprised of.” Not sure if that helps anyone else, but it makes sense for me moving on. Thanks again. Cheers.

1. http://0.gravatar.com/avatar/cea8b13addf74aeea55a75996ffb2799?s=60&d=mm&r=g

Ravindra

[July 23, 2015 at 11:32 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307230) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307230#respond)

I am unable to execute this code by simply copy & paste. Due to some encoding issue, I am getting below error

Uncaught SyntaxError: Unexpected token ILLEGAL

One invisible special characters has been added at the first character for every line. Even dos2unix is unable to fix the issue. Please let me know how to fix this issue

1. http://2.gravatar.com/avatar/886ec9672488c7be0e2ba3fcd99df6a5?s=60&d=mm&r=g

[M. Junaid](http://github.com/mjunaidsalaat)

[July 31, 2015 at 6:45 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307260) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307260#respond)

Yes. Had the same issue & here is the solution I found.

> Copy Code u want to execute.  
> Paste it in a notepad  
> Just select any of the invisible char and cut (ctrl+x) it.  
> Open replace (ctrl+r) window and then in find input box ctrl+v init.  
> Select Replace all.

Done.   
Your code is clean now.

1. http://1.gravatar.com/avatar/429cb9c79c1631cda88db2ae48a7b123?s=60&d=mm&r=g

Steve

[August 6, 2015 at 7:08 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307281) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307281#respond)

Hi,

Not sure if you’ll see this as this is an old post. But I just wanted to find out why – after I tried out isPrototypeOf – there is a difference in results as shown down below. I was studying the Plant.prototype and Fruit.prototype code, and testing for myself that Fruit.prototype inherits from Plant.prototype because of this assignment:  
Fruit.prototype = new Plant ();

Plant.prototype.isPrototypeOf(aBanana);  
true  
Fruit.prototype.isPrototypeOf(aBanana);  
true  
>> Confusion begins here <<  
Plant.prototype.isPrototypeOf(Fruit);  
// false (Why does this syntax result in false vs. the syntax below just by appending ".prototype"?)  
Plant.prototype.isPrototypeOf(Fruit.prototype);  
// true

I've been trying to understand prototypes better from reading this and other sources. Please help me if I'm incorrect in thinking this: that appended .prototype is needed because Fruit is just a constructor function, whereas Fruit.prototype is the property that is sought out in the inheritance chain.

Thanks!

1. http://1.gravatar.com/avatar/4b1ceb2ba8826be079e378a3f95be061?s=60&d=mm&r=g

André Pinto

[August 8, 2015 at 6:01 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307288) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307288#respond)

Hi,  
Excelent blog!  
“// In this example, the search proceeds up the prototype chain and find the toString method on Object.prototype, from which the Fruit object inherited—all objects ultimately inherits from Object.prototype as we have noted before.​  
console.log (famousPerson.toString()); // [object Object]”  
It is famousPerson instead of Fuit?

1. http://0.gravatar.com/avatar/c708f7d2f7bc8302a14c1e2cc44b7e19?s=60&d=mm&r=g

Kurt

[August 13, 2015 at 10:23 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307298) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307298#respond)

Hello Richard,  
Thank you very much for that tutorial. It helped a lot to understand prototyping.  
But there’s still one thing I don’t get. Let’s use this example:

function Car(speed){  
this.speed = speed;  
}  
console.log(Car.prototype); //prototype is Car

var c = new Car(40);  
console.log(“toString” in c); //returns true

How can JavaScript find the toString() method? I didn’t define it in Car and Car has no prototype (it is it’s own). If inheritance in javascript is based on searching through the prototype chain, it shouldn’t find toString() in this example. Where is my mistake? Any help would be much appreciated.

Cheers,  
Kurt

1. http://1.gravatar.com/avatar/d3095176d67d8db26f45a02022857c37?s=60&d=mm&r=g

Abhinav

[August 31, 2015 at 12:21 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307334) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307334#respond)

Thanks a lot for this amazing tutorial.

1. http://0.gravatar.com/avatar/f34e063f92506450e6cf96bc0c4b782a?s=60&d=mm&r=g

[Duy Khanh](http://giammobungaz.com/tap-the-duc-giam-mo-bung-tai-nha-p2/)

[September 23, 2015 at 5:52 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307389) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307389#respond)

Hi Kurt,  
In your question  
function Car(speed){  
this.speed = speed;  
}  
console.log(Car.prototype); //prototype is Car  
var c = new Car(40);  
console.log(“toString” in c); //returns true

I think toString method is default for all Object in javascript.  
Correct me if I’m wrong

1. http://0.gravatar.com/avatar/9839cae48c825ef51e67b0bbc7de4133?s=60&d=mm&r=g

Abdul Hakim

[October 1, 2015 at 10:11 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307417) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307417#respond)

// Here, aBanana uses the name property from the aBanana object prototype, which is Fruit.prototype:​  
console.log(aBanana.name); // Banana​

Based on the explanations you have provided i believe the above comment is not correct. name is not from aBanana’s prototype.

1. http://1.gravatar.com/avatar/42a437f7e3b5aa8a566f0530ec816c25?s=60&d=mm&r=g

Santhosh

[October 26, 2015 at 6:20 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307471) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307471#respond)

nice article.. know got the idea of prototype thanx

1. http://0.gravatar.com/avatar/337922d7d01e9eb57f8dc4506810e7d4?s=60&d=mm&r=g

Toby

[November 5, 2015 at 3:28 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307502) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307502#respond)

Very nice explanation. Thank you.

1. http://0.gravatar.com/avatar/0d6238bcac6acd2a2a8d47c0ed3f1442?s=60&d=mm&r=g

Deepen

[November 6, 2015 at 5:34 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307508) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307508#respond)

Hi,

I am writing partial/curry function for my javascript code.

But I want to use javascript prototype

Can you guide me ? I have some sample example prepared

function func1(a,b) {  
console.log(a+’ ‘+b);  
}

var waitFor = setTimeout.partial(undefined,100000);

var display = func1.partial(undefined,”mehta”);

waitFor (function () {  
console.log(‘Awesome’)  
});

print(‘deepen’);

output:

Deepen Mehta

waiting for 1 sec

welcome

1. http://0.gravatar.com/avatar/0d6238bcac6acd2a2a8d47c0ed3f1442?s=60&d=mm&r=g

Deepen

[November 6, 2015 at 5:37 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307509) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307509#respond)

Hi,

I am writing partial/curry function for my javascript code.

But I want to use javascript prototype

Can you guide me ?

I have some sample example prepared.

function func1(a,b) {  
console.log(a+’ ‘+b);  
}

var waitFor = setTimeout.partial(undefined,100000);

var display = func1.partial(undefined,”mehta”);

waitFor (function () {  
console.log(‘Awesome’)  
});

print(‘deepen’);

output:

Deepen Mehta

waiting for 1 sec

welcome

1. http://0.gravatar.com/avatar/0d6238bcac6acd2a2a8d47c0ed3f1442?s=60&d=mm&r=g

Deepen

[November 6, 2015 at 5:48 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307510) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307510#respond)

Can I build something like this ?  
If yes then How?  
Function.prototype.partial = function () {

};

1. http://2.gravatar.com/avatar/8e6965b00e80dcf1cc129c97b7fa7b78?s=60&d=mm&r=g

Si

[December 1, 2015 at 7:04 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307587) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307587#respond)

Afternoon 

Sorry if I’ve got confused, but in your last example (with the ‘People’ function) the following line and comment is present. The comment states that it finds the ‘toString’ method on Object.prorototype ‘from which the Fruit object inherited’ – If I have understood your article correctly, the ‘Fruit object’ is a typo and it is the ‘People object’ that inherits from Object.prototype in this example?

// In this example, the search proceeds up the prototype chain and find the toString method on Object.prototype, from which the Fruit object inherited—all objects ultimately inherits from Object.prototype as we have noted before.​  
console.log (famousPerson.toString()); // [object Object]

1. http://1.gravatar.com/avatar/deab5302b2ed52d17cb5581dda285336?s=60&d=mm&r=g

Karl

[December 3, 2015 at 10:34 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307626) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307626#respond)

Object.prototype, from which the Fruit object inherited—all objects ultimately inherits from Object.prototype as we have noted before.​  
console.log (famousPerson.toString()); // [object Object]

Should be

Object.prototype, from which the ‘Person’ object inherited—all objects ultimately inherits from Object.prototype as we have noted before.​  
console.log (famousPerson.toString()); // [object Object]

1. http://1.gravatar.com/avatar/d46494895e271d89a81756ae4f6332ce?s=60&d=mm&r=g

[searene](http://searene.github.io/)

[December 10, 2015 at 10:18 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307709) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307709#respond)

Amazing tutorial, thanks, it helps me a lot!

1. http://2.gravatar.com/avatar/be0236bfa5a54b67067d87eaf1099cbe?s=60&d=mm&r=g

[Sri Varshan](http://www.articpost.com/)

[December 22, 2015 at 4:29 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307750) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307750#respond)

Brilliant way to explain things, you have made complex stuff pretty simple for everyone to understand. One of the best blogs that explain concepts in a clear and concise manner. Good job, will recommend this blog and bookmark it too. Good Day

1. http://1.gravatar.com/avatar/15573894f50128b93f7b9c0a65a5f17e?s=60&d=mm&r=g

Manju

[January 7, 2016 at 2:09 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307770) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307770#respond)

Thank you. It was a very nice article.

I have a following query,

If i create an object as an object function like below,

function company(){  
this.name = ‘google';  
}

company.prototype.getName = function(){alert(this.name)};

function employee(){  
this.name = ‘Manju';  
this.role = ‘Dev';  
}

employee.prototype = new company();

employee.prototype.getRole = function(){  
alert(this.role);  
}

here company object inherits the prototype from Object.prototype and i have created one more object function called employee which inherits the prototype of company object. Since above created objects are object functions I’m able to inherit the parents prototype.

My question is: What if i create an object literal instead of object function as below. If so, how will i inherit the prototype of company object from employee and also retain the properties and methods of employee?

var company = {  
name: ‘google’  
}

company.getName = function(){alert(this.name)}

var employee = {  
name: ‘Manju’,  
role: ‘Dev’  
}

employee = Object.create(company); // Here whole of an employee object is getting replaced by company object and also I’m not able to use prototype property on either employee/company object. Is it like prototype property can be used only with the Objects which are functions?

Answer to the query would be really appreciated. Thanks is advance.

1. http://0.gravatar.com/avatar/6d31bd8ee6fa9befa26252ada65127f5?s=60&d=mm&r=g

[Mahavir](http://offersday.in/)

[January 15, 2016 at 11:06 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307800) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307800#respond)

I have just tried this code and i am surprised with the output of code, can you please throw some light on it why it happens.

function PrintStuff (myDocuments) {  
this.insideFunc=function(){  
console.log(“Inside Constructor”);  
}  
}

PrintStuff.prototype.insideFunc=function(){  
console.log(“Inside Prototype”);  
}

var newObj = new PrintStuff();

//It outputs “Inside Constructor”  
newObj.insideFunc();

So by seeing the output it is confirm that anything declared within constructor have higher priority than that of declared with “prototype”, can you please add some value on this type of scenario.

Thanks   
Mahavir

1. http://2.gravatar.com/avatar/e033f0ccd9253ee16ae0cd0bf08fa670?s=60&d=mm&r=g

Riddik

[January 22, 2016 at 1:09 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307810) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307810#respond)

Hi!  
Can you please explain me code below?  
function Fruit(){  
var color=green;  
this.size=undefined;  
}  
var f=new Fruit();

1) It is not about inheritance right? Since we just create an instance “f” of Fruit class.  
2) From other side object f will get Fruit.prototype. And how should i consider that? is it inheritance or jsut simple instance creation where instance get its class members.  
3) can you please explain me way “color” property is not available as f.color? it is also constructor property but i can’t get it from instance  
4) Ihheritance is : “function Apple(){}; Apple.prototype=new Fruit(); var apple = new Apple()” it is clear for me.

1. http://0.gravatar.com/avatar/383c7d67c9ad0997e690077ee9b0f51f?s=60&d=mm&r=g

ChelBot

[February 10, 2016 at 1:40 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307865) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307865#respond)

I have been reading ‘JavaScript: the Good Parts’ and I must admit that some of the sections are confusing. These articles have helped me out immensely! Good stuff and thanks.

1. http://2.gravatar.com/avatar/2ab693511d96c51b1eff944c564ae7f5?s=60&d=mm&r=g

vikas thakur

[February 11, 2016 at 7:59 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307867) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307867#respond)

pls easy way to explain to protype….

1. http://0.gravatar.com/avatar/ce5c9e0701bc0a4e7f37cb7dbd1be8dc?s=60&d=mm&r=g

sivaragu

[March 14, 2016 at 1:13 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307950) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307950#respond)

this.country = “Mexico”; why this using

1. http://1.gravatar.com/avatar/a0831a3f2f96f472dd236a4c0fb173f9?s=60&d=mm&r=g

Paul

[March 31, 2016 at 11:46 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-307993) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=307993#respond)

‘Uncaught TypeError: aBanana.showNameAndColor is not a function’

Sums up JavaScript for me.

1. http://2.gravatar.com/avatar/5aba5aa7275e151fffd82de7021216b4?s=60&d=mm&r=g

Nasir Ahmed

[April 5, 2016 at 8:16 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-308004) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=308004#respond)

Hi,

I am very new to JavaScript, kindly guide me how to become expert in javascript.  
Kindly send me if you are having pdf materials means.

Thanks in advance  
Nasir Ahmed

1. http://2.gravatar.com/avatar/5d20a35b08d13da2527c3443c7c9aafd?s=60&d=mm&r=g

[Manoj](http://na/)

[April 13, 2016 at 1:43 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-308023) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=308023#respond)

how many diff ways we can create objects we know it is object literal and constructor function,

is using prototype another way of creating objects.

what is the diff between constructor and prototype

function Dog (breed) {  
this.breed = breed;  
// this.bark = function() {  
// console.log(“Woof”);  
// }

};

// here we make buddy and teach him how to bark  
var buddy = new Dog(“golden Retriever”);  
Dog.prototype.bark = function() {  
console.log(“Woof”);  
};  
buddy.bark();

in the example both this and dog prototype gives the same output then what is the diff and when to use what

1. http://2.gravatar.com/avatar/e6d002f32bb8f324f015301077cdc37a?s=60&d=mm&r=g

mohamed

[October 17, 2016 at 9:04 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-308373) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=308373#respond)

I was amazed by the talent, this guy called Richard, has when it comes to explaining these complicated concepts that many every expensive books failed to crack. As i was reading your about page, you said that this is “a give back ” of what you have learned from the internet but in my opinion it is more then that. You rely help me understand a lot. Thank you very much and keep up my dear friend

1. http://0.gravatar.com/avatar/6a7b6c21a0b5bf0fa9d2a09442def7c2?s=60&d=mm&r=g

Andrew Santoli

[October 31, 2016 at 10:55 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-308392) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=308392#respond)

Create a an Object constructor function called MedRecord with the properties firstName, lastName, medRecord, medID, insuranceInfo.

Define a MedRecord prototype method called changeInsurance() that accepts the name of a patient new insurance information and update object insuranceInfo property.

1. http://0.gravatar.com/avatar/698e613b52a0f55cd2c5e03c7e5d3ede?s=60&d=mm&r=g

Amar Nath Jena

[October 16, 2017 at 8:13 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-308864) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=308864#respond)

Hi Author,

How you can distribute prototype to object ? As i have read above things i got that object is prototype but prototype is not object.

1. http://2.gravatar.com/avatar/5e8644fd076385048723fa9fe57d244b?s=60&d=mm&r=g

Arthur

[November 17, 2017 at 6:04 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-308918) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=308918#respond)

This’s an awesome article!  
Great job, man!  
Many thanks!  
Appreciate it!

1. http://1.gravatar.com/avatar/1f9649a493bbda83a366173f0bc6f09d?s=60&d=mm&r=g

Mikex

[November 27, 2017 at 2:25 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-308934) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=308934#respond)

Hello,  
You mentioned ‘First, every JavaScript function has a prototype property’. Why does every JavaScript function have a prototype property?

1. http://2.gravatar.com/avatar/52e12a4a9ef57509324dfd5b082ce902?s=60&d=mm&r=g

million

[February 4, 2018 at 4:03 pm](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-309020) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=309020#respond)

oh my goodness, this topic was giving me a hard time to understand but after i get this one i give big thumb to JS and you . thank you so much.

1. http://0.gravatar.com/avatar/6e468a695df4050f7d9f76b9d7b4e10b?s=60&d=mm&r=g

Owen

[April 9, 2013 at 9:46 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-5495) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=5495#respond)

Isn’t that the whole idea?

Fruit is always organic because it is a plant? And anything else that inherits from plant, grass or flowers or cacti, they’ll all be organic too.

1. http://2.gravatar.com/avatar/b323fea23020a7e298b5d1e8754e2f33?s=60&d=mm&r=g

attila

[August 19, 2013 at 6:29 am](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/#comment-30490) / [Reply](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/?replytocom=30490#respond)

exatly