JavaScript Objects in Detail

jan. 27 2013 [176](http://javascriptissexy.com/javascript-objects-in-detail/#comments)

JavaScript’s core—most often used and most fundamental—data type is the Object data type. JavaScript has one complex data type, the Object data type, and it has five simple data types: Number, String, Boolean, Undefined, and Null. Note that these simple (primitive) data types are immutable (cannot be changed), while objects are mutable (can be changed).

[[](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)

**What is an Object**  
An object is an unordered list of primitive data types (and sometimes reference data types) that is stored as a series of name-value pairs. Each item in the list is called a *property* (functions are called *methods*).

Consider this simple object:

|  |  |
| --- | --- |
|  | var myFirstObject = {firstName: "Richard", favoriteAuthor: "Conrad"}; |

Think of an object as a list that contains items, and each item (a property or a method) in the list is stored by a name-value pair. The property names in the example above are firstName and favoriteAuthor. And the values are “Richard” and “Conrad.”

* Receive Updates

Top of Form



Bottom of Form

Property names can be a string or a number, but if the property name is a number, it has to be accessed with the bracket notation. More on bracket notation later. Here is another example of objects with numbers as the property name:

|  |  |
| --- | --- |
|  | var ageGroup = {30: "Children", 100:"Very Old"}; |
|  | console.log(ageGroup.30) // This will throw an error​ |
|  | ​// This is how you will access the value of the property 30, to get value "Children"​ |
|  | console.log(ageGroup["30"]); // Children​ |
|  | ​ |
|  | ​//It is best to avoid using numbers as property names. |

As a JavaScript developer you will most often use the object data type, mostly for storing data and for creating your own custom methods and functions.

**Reference Data Type and Primitive Data Types**  
One of the main differences between reference data type and primitive data types is reference data type’s value is stored as a reference, it is not stored directly on the variable, as a value, as the primitive data types are. For example:

|  |  |
| --- | --- |
|  | // The primitive data type String is stored as a value​ |
|  | ​var person = "Kobe"; |
|  | ​var anotherPerson = person; // anotherPerson = the value of person​ |
|  | person = "Bryant"; // value of person changed​ |
|  | ​ |
|  | console.log(anotherPerson); // Kobe​ |
|  | console.log(person); // Bryant |

It is worth noting that even though we changed *person* to “Bryant,” the *anotherPerson* variable still retains the value that person had.

Compare the primitive data saved-as-value demonstrated above with the save-as-reference for objects:

|  |  |
| --- | --- |
|  | var person = {name: "Kobe"}; |
|  | ​var anotherPerson = person; |
|  | person.name = "Bryant"; |
|  | ​ |
|  | console.log(anotherPerson.name); // Bryant​ |
|  | console.log(person.name); // Bryant |

In this example, we copied the *person* object to *anotherPerson*, but because the value in person was stored as a reference and not an actual value, when we changed the person.name property to “Bryant” the anotherPerson reflected the change because it never stored an actual copy of it’s own value of the person’s properties, it only had a reference to it.

**Object Data Properties Have Attributes**  
Each data property (object property that store data) has not only the name-value pair, but also 3 attributes (the three attributes are set to true by default):  
**—  Configurable Attribute:** Specifies whether the property can be deleted or changed.  
**— Enumerable:**Specifies whether the property can be returned in a for/in loop.  
**— Writable:**Specifies whether the property can be changed.

Note that ECMAScript 5 specifies accessor properties along with the data properties noted above. And the accessor properties are functions (getters and setters). We will learn more about ECMAScript 5 in an already-scheduled post slated for February 15.

**Creating Objects**  
These are the two common ways to create objects:

1. **Object Literals**  
   The most common and, indeed, the easiest way to create objects is with the object literal described here:

|  |  |
| --- | --- |
|  | // This is an empty object initialized using the object literal notation​ |
|  | ​var myBooks = {}; |
|  | ​ |
|  | ​// This is an object with 4 items, again using object literal​ |
|  | ​var mango = { |
|  | color: "yellow", |
|  | shape: "round", |
|  | sweetness: 8, |
|  | ​ |
|  | ​howSweetAmI: function () { |
|  | console.log("Hmm Hmm Good"); |
|  | } |
|  | } |

1. **Object Constructor**  
    The second most common way to create objects is with Object constructor. A constructor is a function used for initializing new objects, and you use the new keyword to call the constructor.

|  |  |
| --- | --- |
|  | var mango = new Object (); |
|  | mango.color = "yellow"; |
|  | mango.shape= "round"; |
|  | mango.sweetness = 8; |
|  | ​ |
|  | mango.howSweetAmI = function () { |
|  | console.log("Hmm Hmm Good"); |
|  | } |

While you can use some reserved word such as “for” as property names in your objects, it is wise to avoid this altogether.

Objects can contain any other data type, including Numbers, Arrays, and even other Objects.

**Practical Patterns for Creating Objects**  
For simple objects that may only ever be used once in your application to store data, the two methods used above would suffice for creating objects.

Imagine you have an application that displays fruits and detail about each fruit. All fruits in your application have these properties: color, shape, sweetness, cost, and a showName function. It would be quite tedious and counterproductive to type the following every time you want to create a new fruit object.

|  |  |
| --- | --- |
|  | var mangoFruit = { |
|  | color: "yellow", |
|  | sweetness: 8, |
|  | fruitName: "Mango", |
|  | nativeToLand: ["South America", "Central America"], |
|  | ​ |
|  | ​showName: function () { |
|  | console.log("This is " + this.fruitName); |
|  | }, |
|  | ​nativeTo: function () { |
|  | this.nativeToLand.forEach(function (eachCountry) { |
|  | console.log("Grown in:" + eachCountry); |
|  | }); |
|  | } |
|  | } |

If you have 10 fruits, you will have to add the **same** code 10 times. And what if you had to make a change to the nativeTo function? You will have to make the change in 10 different places. Now extrapolate this to adding objects for members on a website and suddenly you realized the manner in which we have created objects so far is not ideal objects that will have instances, particularly when developing large applications.

To solve these repetitive problems, software engineers have invented patterns (solutions for repetitive and common tasks) to make developing applications more efficient and streamlined.

Here are two common patterns for creating objects. If you have done the Learn JavaScript Properly course, you would have seen the lessons in the Code Academy used this first pattern frequently:

1. **Constructor Pattern for Creating Objects**

|  |  |
| --- | --- |
|  | function Fruit (theColor, theSweetness, theFruitName, theNativeToLand) { |
|  | ​ |
|  | this.color = theColor; |
|  | this.sweetness = theSweetness; |
|  | this.fruitName = theFruitName; |
|  | this.nativeToLand = theNativeToLand; |
|  | ​ |
|  | this.showName = function () { |
|  | console.log("This is a " + this.fruitName); |
|  | } |
|  | ​ |
|  | this.nativeTo = function () { |
|  | this.nativeToLand.forEach(function (eachCountry) { |
|  | console.log("Grown in:" + eachCountry); |
|  | }); |
|  | } |
|  | ​ |
|  | ​ |
|  | } |

1. With this pattern in place, it is very easy to create all sorts of fruits. Thus:

|  |  |
| --- | --- |
|  | var mangoFruit = new Fruit ("Yellow", 8, "Mango", ["South America", "Central America", "West Africa"]); |
|  | mangoFruit.showName(); // This is a Mango.​ |
|  | mangoFruit.nativeTo(); |
|  | ​//Grown in:South America​ |
|  | ​// Grown in:Central America​ |
|  | ​// Grown in:West Africa​ |
|  | ​ |
|  | ​var pineappleFruit = new Fruit ("Brown", 5, "Pineapple", ["United States"]); |
|  | pineappleFruit.showName(); // This is a Pineapple. |

1. If you had to change the fruitName function, you only had to do it in one location. The pattern encapsulates all the functionalities and characteristics of all the fruits in by making just the single Fruit function with inheritance.
2. Notes:  
   — An inherited property is defined on the object’s prototype property. For example: someObject.prototype.firstName = “rich”;
3. — An own property is defined directly on the object itself, for example:  
   // Let’s create an object first:  
   var aMango = new Fruit ();  
   // Now we define the mangoSpice property directly on the aMango object  
   // Because we define the mangoSpice property directly on the aMango object, it is an own property of aMango, not an inherited property.  
   aMango.mangoSpice = “some value”;
4. — To access a property of an object, we use object.property, for example:  
   console.log(aMango.mangoSpice); // “some value”
5. — To invoke a method of an object, we use object.method(), for example :  
   // First, lets add a method  
   aMango.printStuff = function () {return “Printing”;}
6. // Now we can invoke the printStuff method:  
   aMango.printStuff ();
7. **Prototype Pattern for Creating Objects**

|  |  |
| --- | --- |
|  | function Fruit () { |
|  | ​ |
|  | } |
|  | ​ |
|  | Fruit.prototype.color = "Yellow"; |
|  | Fruit.prototype.sweetness = 7; |
|  | Fruit.prototype.fruitName = "Generic Fruit"; |
|  | Fruit.prototype.nativeToLand = "USA"; |
|  | ​ |
|  | Fruit.prototype.showName = function () { |
|  | console.log("This is a " + this.fruitName); |
|  | } |
|  | ​ |
|  | Fruit.prototype.nativeTo = function () { |
|  | console.log("Grown in:" + this.nativeToLand); |
|  | } |

1. And this is how we call the Fruit () constructor in this prototype pattern:

|  |  |
| --- | --- |
|  | var mangoFruit = new Fruit (); |
|  | mangoFruit.showName(); //​ |
|  | mangoFruit.nativeTo(); |
|  | ​// This is a Generic Fruit​ |
|  | ​// Grown in:USA |

**Further Reading**  
For a complete discussion on these two patterns and a thorough explanation of how each work and the disadvantages of each, read Chapter 6 of *Professional JavaScript for Web Developers*. You will also learn which pattern Zakas recommends as the best one to use (Hint: it is neither of the 2 above).

**How to Access Properties on an Object**  
The two primary ways of accessing properties of an object are with dot notation and bracket notation.

1. **Dot Notation**

|  |  |
| --- | --- |
|  | // We have been using dot notation so far in the examples above, here is another example again:​ |
|  | ​var book = {title: "Ways to Go", pages: 280, bookMark1:"Page 20"}; |
|  | ​ |
|  | ​// To access the properties of the book object with dot notation, you do this:​ |
|  | console.log ( book.title); // Ways to Go​ |
|  | console.log ( book.pages); // 280 |

1. **Bracket Notation**

|  |  |
| --- | --- |
|  | // To access the properties of the book object with bracket notation, you do this:​ |
|  | console.log ( book["title"]); //Ways to Go​ |
|  | console.log ( book["pages"]); // 280​ |
|  | ​ |
|  | ​//Or, in case you have the property name in a variable:​ |
|  | ​var bookTitle = "title"; |
|  | console.log ( book[bookTitle]); // Ways to Go​ |
|  | console.log (book["bookMark" + 1]); // Page 20 |

Accessing a property on an object that does not exist will result in *undefined*.

**Own and Inherited Properties**  
Objects have inherited properties and own properties. The own properties are properties that were defined on the object, while the inherited properties were inherited from the object’s Prototype object.

To find out if a property exists on an object (either as an inherited or an own property), you use the in operator:

|  |  |
| --- | --- |
|  | // Create a new school object with a property name schoolName​ |
|  | ​var school = {schoolName:"MIT"}; |
|  | ​ |
|  | ​// Prints true because schoolName is an own property on the school object​ |
|  | console.log("schoolName" in school); // true​ |
|  | ​ |
|  | ​// Prints false because we did not define a schoolType property on the school object, and neither did the object inherit a schoolType property from its prototype object Object.prototype.​ |
|  | console.log("schoolType" in school); // false​ |
|  |  |
|  | ​// Prints true because the school object inherited the toString method from Object.prototype. ​ |
|  | console.log("toString" in school); // true |

**hasOwnProperty**  
To find out if an object has a specific property as one of its own property, you use the hasOwnProperty method. This method is very useful because from time to time you need to enumerate an object and you want only the own properties, not the inherited ones.

|  |  |
| --- | --- |
|  | // Create a new school object with a property name schoolName​ |
|  | ​var school = {schoolName:"MIT"}; |
|  | ​ |
|  | ​// Prints true because schoolName is an own property on the school object​ |
|  | console.log(school.hasOwnProperty ("schoolName")); // true​ |
|  |  |
|  | ​// Prints false because the school object inherited the toString method from Object.prototype, therefore toString is not an own property of the school object.​ |
|  | console.log(school.hasOwnProperty ("toString")); // false |

**Accessing and Enumerating Properties on Objects**  
To access the enumerable (own and inherited) properties on objects, you use the for/in loop or a general for loop.

|  |  |
| --- | --- |
|  | // Create a new school object with 3 own properties: schoolName, schoolAccredited, and schoolLocation.​ |
|  | ​var school = {schoolName:"MIT", schoolAccredited: true, schoolLocation:"Massachusetts"}; |
|  | ​ |
|  | ​//Use of the for/in loop to access the properties in the school object​ |
|  | ​for (var eachItem in school) { |
|  | console.log(eachItem); // Prints schoolName, schoolAccredited, schoolLocation​ |
|  | ​ |
|  | } |

**Accessing Inherited Properties**  
Properties inherited from Object.prototype are not enumerable, so the for/in loop does not show them. However, inherited properties that are enumerable are revealed in the for/in loop iteration.  
For example:

|  |  |
| --- | --- |
|  | //Use of the for/in loop to access the properties in the school object​ |
|  | ​for (var eachItem in school) { |
|  | console.log(eachItem); // Prints schoolName, schoolAccredited, schoolLocation​ |
|  | ​ |
|  | } |
|  | ​ |
|  | ​// Create a new HigherLearning function that the school object will inherit from.​ |
|  | ​/\* SIDE NOTE: As Wilson (an astute reader) correctly pointed out in the comments below, the educationLevel property is not actually inherited by objects that use the HigherLearning constructor; instead, the educationLevel property is created as a new property on each object that uses the HigherLearning constructor. The reason the property is not inherited is because we use of the "this" keyword to define the property. |
|  | \*/​ |
|  | ​ |
|  | ​ |
|  | ​function HigherLearning () { |
|  | ​this.educationLevel = "University"; |
|  | } |
|  | ​ |
|  | ​// Implement inheritance with the HigherLearning constructor​ |
|  | ​var school = new HigherLearning (); |
|  | school.schoolName = "MIT"; |
|  | school.schoolAccredited = true; |
|  | school.schoolLocation = "Massachusetts"; |
|  | ​ |
|  | ​ |
|  | ​//Use of the for/in loop to access the properties in the school object​ |
|  | ​for (var eachItem in school) { |
|  | console.log(eachItem); // Prints educationLevel, schoolName, schoolAccredited, and schoolLocation​ |
|  | } |

In the last example, note the educationLevel property that was defined on the HigherLearning function is listed as one of the school’s properties, even though educationLevel is not an own property—it was inherited.

**Object’s Prototype Attribute and Prototype Property**  
The prototype attribute and prototype property of an object are critically important concepts to understand in JavaScript. Read my post [JavaScript Prototype in Plain, Detailed Language](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/) for more.

**Deleting Properties of an Object**  
To delete a property from an object, you use the delete operator. You cannot delete properties that were inherited, nor can you delete properties with their attributes set to configurable. You must delete the inherited properties on the prototype object (where the properties were defined). Also, you cannot delete properties of the global object, which were declared with the var keyword.

The delete operator returns true if the delete was successful. And surprisingly, it also returns true if the property to delete was nonexistent or the property could not be deleted (such as non-configurable or not owned by the object).

These examples illustrate:

|  |  |
| --- | --- |
|  | var christmasList = {mike:"Book", jason:"sweater" } |
|  | ​delete christmasList.mike; // deletes the mike property​ |
|  | ​ |
|  | ​for (var people in christmasList) { |
|  | console.log(people); |
|  | } |
|  | ​// Prints only jason​ |
|  | ​// The mike property was deleted​ |
|  | ​ |
|  | ​delete christmasList.toString; // returns true, but toString not deleted because it is an inherited method​ |
|  | ​ |
|  | ​// Here we call the toString method and it works just fine—wasn’t deleted ​ |
|  | christmasList.toString(); //"[object Object]"​ |
|  | ​ |
|  | ​// You can delete a property of an instance if the property is an own property of that instance. For example, we can delete the educationLevel property from the school's object we created above because the educationLevel property is defined on the instance: we used the "this" keyword to define the property when we declare the HigherLearning function. We did not define the educationLevel property on the HigherLearning function's prototype.​ |
|  | ​ |
|  | console.log(school.hasOwnProperty("educationLevel")); true​ |
|  | ​// educationLevel is an own property on school, so we can delete it​ |
|  | ​delete school.educationLevel; true |
|  | ​ |
|  | ​// The educationLevel property was deleted from the school instance​ |
|  | console.log(school.educationLevel); undefined |
|  | ​ |
|  | ​// But the educationLevel property is still on the HigherLearning function​ |
|  | ​var newSchool = new HigherLearning (); |
|  | console.log(newSchool.educationLevel); // University​ |
|  | ​ |
|  | ​// If we had defined a property on the HigherLearning function's prototype, such as this educationLevel2 property:​ |
|  | HigherLearning.prototype.educationLevel2 = "University 2"; |
|  | ​ |
|  | ​// Then the educationLevel2 property on the instances of HigherLearning would not be own property. ​ |
|  | ​ |
|  | ​// The educationLevel2 property is not an own property on the school instance​ |
|  | console.log(school.hasOwnProperty("educationLevel2")); false​ |
|  | console.log(school.educationLevel2); // University 2​ |
|  | ​ |
|  | ​// Let's try to delete the inherited educationLevel2 property​ |
|  | ​delete school.educationLevel2; true (always returns true, as noted earlier) |
|  | ​ |
|  | ​// The inherited educationLevel2 property was not deleted​ |
|  | console.log(school.educationLevel2); University 2​ |
|  | ​ |

**Serialize and Deserialize Objects**  
To transfer your objects via HTTP or to otherwise convert it to a string, you will need to serialize it (convert it to a string); you can use the JSON.stringify function to serialize your objects. Note that prior to ECMAScript 5, you had to use a popular json2 library (by Douglas Crockford) to get the JSON.stringify function. It is now standardized in ECMAScript 5.

To Deserialize your object (convert it to an object from a string), you use the JSON.parse function from the same json2 library. This function too has been standardized by ECMAScript 5.

JSON.stringify Examples:

|  |  |
| --- | --- |
|  | var christmasList = {mike:"Book", jason:"sweater", chelsea:"iPad" } |
|  | JSON.stringify (christmasList); |
|  | ​// Prints this string:​ |
|  | ​// "{"mike":"Book","jason":"sweater","chels":"iPad"}" |
|  | ​ |
|  | ​// To print a stringified object with formatting, add "null" and "4" as parameters:​ |
|  | JSON.stringify (christmasList, null, 4); |
|  | // "{ |
|  | "mike": "Book", |
|  | "jason": "sweater", |
|  | "chels": "iPad"​ |
|  | }" |
|  | ​ |
|  | ​// JSON.parse Examples \\​ |
|  | // The following is a JSON string, so we cannot access the properties with dot notation (like christmasListStr.mike)​ |
|  | ​var christmasListStr = '{"mike":"Book","jason":"sweater","chels":"iPad"}'; |
|  | ​ |
|  | ​// Let’s convert it to an object​ |
|  | ​var christmasListObj = JSON.parse (christmasListStr); |
|  | ​ |
|  | ​// Now that it is an object, we use dot notation​ |
|  | console.log(christmasListObj.mike); // Book |

For more detailed coverage of JavaScript Objects, including ECMAScript 5 additions for dealing with objects, read chapter 6 of JavaScript: The Definitive Guide 6th Edition.

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

[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.

176 Comments

1. http://1.gravatar.com/avatar/1cb10ddfc02b9dda0ed582226c0ee008?s=60&d=mm&r=g

[moncler online shop](http://www.monclerjackendeonlineshop.com/)

[January 30, 2013 at 4:04 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-1862) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=1862#respond)

Good post, I always like them.

1. http://2.gravatar.com/avatar/bf4d36351bb61a46ff5ce24337816183?s=60&d=mm&r=g

R Sturim

[February 2, 2013 at 10:05 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-1930) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=1930#respond)

The “Fruit” constructor under the “Constructor Pattern for Creating Objects” has several coding errors. If you run the code through JSHint they’ll become very apparent. Properties should be ended with a semicolon (not a comma) and assignment should use the “=” sign, not the “:” — hope that helps.

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

Richard Bovell (Author)

[February 2, 2013 at 1:31 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-1933) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=1933#respond)

Thanks, R Sturim, for pointing out the errors. I just fixed the code.

I am not surprised there are errors in this particular post, because I didn’t get the chance to JSHint them and throughly review them for errors, which I am definitely doing for all the other example codes I post.

1. http://0.gravatar.com/avatar/9640a7686292536fdd7a3d7c4eef4580?s=60&d=mm&r=g

brandnew

[February 4, 2013 at 6:07 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-1984) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=1984#respond)

Thank you for Posts. i’m trying to get a grasp of JavaScript and i had a question related to this post in particular your example under, “Constructor Pattern for Creating Objects”.  
When i type in the console (FireFox) mangoFruit.fruitName, output is “Mango”.  
Question arises when i type mangoFruit.nativeTo, it returns:  
(function () {  
for (var eachCountry in this.nativeTo) {  
console.log(“Grown in:” + eachCountry);  
}  
})  
or when i type mangoFruit.nativeTo() it returns undefined.  
i’m not certain why this is? How is this method called to output “eachCountry” variable?

thank you

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

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

[February 4, 2013 at 7:23 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-1986) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=1986#respond)

@brandnew  
Update (you are correct):  
There was an error in the code, the this.nativeTo call in the for loop was calling the this.nativeTo function. It was not calling the this.nativeTo variable (the array) we really wanted. I just fixed the error. Here is the full code  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
function Fruit (theColor, theSweetness, theFruitName, theNativeToLand) {

this.color = theColor;  
this.sweetness = theSweetness;  
this.fruitName = theFruitName;  
this.nativeToLand = theNativeToLand;

this.showName = function () {  
console.log(“This is a ” + this.fruitName);  
}

this.nativeTo = function () {

this.nativeToLand.forEach(function (eachCountry) {  
console.log(“Grown in:” + eachCountry);  
});  
}

}

var mangoFruit = new Fruit (“Yellow”, 8, “Mango”, [“South America”, “Central America”, “West Africa”]);  
mangoFruit.showName(); // This is a Mango.  
mangoFruit.nativeTo();  
//Grown in:South America  
// Grown in:Central America  
// Grown in:West Africa

var pineappleFruit = new Fruit (“Brown”, 5, “Pineapple”, [“United States”]);  
pineappleFruit.showName(); // This is a Pineapple.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + - http://0.gravatar.com/avatar/9640a7686292536fdd7a3d7c4eef4580?s=60&d=mm&r=g

brandnew

[February 4, 2013 at 7:42 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-1987) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=1987#respond)

thanks for the response: So if i just wanted to see the output of [“South America”, “Central America”, “West Africa”] from mangoFruit i would need to type in the entire code in the console?  
i already have the entire code on a text file, i’m just running the firefox web console.

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

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

[February 4, 2013 at 9:15 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-1994) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=1994#respond)

Since you already have the Fruit function in a JS file, then you don’t have to put the entire code in FireFox. I didn’t know you had it in a JS file already.

1. http://2.gravatar.com/avatar/8c0b4c2bdb8a928240a61dfcad5f3b9b?s=60&d=mm&r=g

M Dunn

[February 12, 2013 at 8:15 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-2218) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=2218#respond)

Excellent post! Two typos, I think:  
1. To solve these repetitive problems, software engineers have invented patterns (clever was to solve common receptive tasks) to make developing applications more efficient.

– should read: (clever ways to solve common repetitive tasks)

2. In the last example, note the educationLevel property that was defined on the HigherLearning function is not listed as one of the school’s properties, even though educationLevel is not an own property—it was inherited.  
– should be: “… HigherLearning function is listed …”  
At least that is what I am seeing.

Finally, I want to make sure I understand this part:  
“To delete a property from an object, you use the delete operator. You cannot delete properties that were inherited, nor can you delete properties with their attributes set to configurable. ”

It seems that I can delete the educationalLevel property from the school object (delete school.educationLevel) even though it is inherited from HigherLearning. Indeed, when that property is deleted, it is also deleted from HigherLearning. Am I seeing this right?

Thank you!

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

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

[February 13, 2013 at 1:37 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-2261) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=2261#respond)

@M Dunn  
Thanks for pointing out the two typos; I just fixed them.

I updated the post with a detailed explanation of why we can delete the educationLevel property from the school object.

BTW, the educationLevel property was not deleted from the HigherLearning function. See the additional “delete” details I added to the post.

* + - http://2.gravatar.com/avatar/8c0b4c2bdb8a928240a61dfcad5f3b9b?s=60&d=mm&r=g

M Dunn

[February 13, 2013 at 8:22 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-2278) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=2278#respond)

Ah! I was faked out by the “true” returned when I deleted the inherited property and indeed, that property \_was\_ deleted (better to say “removed”?) from the child object. Silly me, I did not check to see that, of course, the prototype object still had the property. This makes sense now. Thank you.

BTW, you still have a “receptive” where there should be a “repetitive”.

1. http://0.gravatar.com/avatar/0c279179ad76822edbe3319da0e3c5f7?s=60&d=mm&r=g

Ariana A.

[February 28, 2013 at 6:31 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-3240) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=3240#respond)

I think you need to write a book. As someone totally new to programming I’ve had a hard time reading the books you’ve suggested. Some things are easy to grasp but once I got to Objects I wanted to bang my head against the wall. This post is so much easier to understand! I feel like I have a much better understanding now. You explained it so clearly and the examples were easy to follow. Thank you.

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

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

[March 1, 2013 at 9:49 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-3266) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=3266#respond)

Thank you very much for the encouraging words, Ariana. You are too kind.

I am very happy to hear that the article helped you to better understand JavaScript objects.

1. http://1.gravatar.com/avatar/df7807fc4c5b60d56b785f0901504e36?s=60&d=mm&r=g

[antonio](http://www.aortiz.info/)

[March 29, 2013 at 8:19 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-4951) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=4951#respond)

These are great Tuts, thank you Richard!  
I took your advice and work through almost everything here and elsewhere (in the console, editor etc.)…  
Question: This may be pretty basic but given the fact you can change variables at will, doesn’t that make them mutable? I mean if I declare…

var starship = “enterprise”;  
var starship = “reliant”; // won’t starship now be reliant?

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

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

[April 1, 2013 at 9:23 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-5124) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=5124#respond)

I think this is a great question.

When you change a variable to a new string, you are **redefining** the variable and giving it a new value. Technically, this means that in the JS engine (behind the scenes), the first variable is destroyed and it is redefined with a new string. So you didn’t actually change the variable; the original string was literally destroyed, then the new string was created and assigned to the same variable.

Here is a quick example to demonstrate that strings are immutable (cannot be changed):

|  |  |
| --- | --- |
|  | ​ |
|  | ​var myName = "Antonio"; |
|  | console.log (myName.toLowerCase()); // antonio​ |
|  | console.log (myName.replace("An", "An-")); // An-tonio​ |
|  | console.log(myName); // "Antonio"​ |
|  | ​ |

The string was never changed, because strings in JavaScript are immutable—they cannot be changed. It appears that behind the scenes a new string is being created and returned every time we execute one of the string methods like toLowerCase().

* + - http://1.gravatar.com/avatar/df7807fc4c5b60d56b785f0901504e36?s=60&d=mm&r=g

[antonio](http://aortiz.info/)

[April 5, 2013 at 11:26 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-5259) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=5259#respond)

Thank you Richard!!  
After reading Chapter 4(JS.FWD), this become much clearer!  
One last thing though regarding this;

function setName(obj){  
obj.name = “Nicholas”;  
obj = new Object();  
obj.name = “Greg”;  
}

var person = new Object();  
setName(person);  
alert(person.name); //”Nicholas”

I understand the point, this chapter was trying to convey with primitive vs. reference values, and I understand when “person” was passed in “setName” it picks up the value of “NIcholas” (it appears the argument “person” is going down the body of the function).  
So is the creation of “obj = new Object() in effect acting as a “break” in a way to stop the argument from becoming “Greg?”

I modified the code to comment out the new obj:  
function setName(obj){  
obj.name = “Nicholas”;  
//obj = new Object();  
obj.name = “Greg”;  
}  
var person = new Object();  
setName(person);  
alert(person.name); //now “Greg”

Thanks!!!!

* + - * http://1.gravatar.com/avatar/a911369e745d2560b2d51f0805fd516b?s=60&d=mm&r=g

Yo

[December 5, 2014 at 6:38 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-305935) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=305935#respond)

I tried some combinations and I think it is due the scope and preference of the Object.

1. http://0.gravatar.com/avatar/63130a7f2a9b8dc5ac71b4c062e80e20?s=60&d=mm&r=g

tomas

[April 3, 2013 at 1:45 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-5202) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=5202#respond)

hi,  
in this code:

var mangoFruit = {  
color: “yellow”,  
sweetness: 8,  
fruitName: “Mango”,  
nativeToLand: [South America, Central America],

showName: function () {  
console.log(“This is ” + fruitName);  
},  
nativeTo: function () {  
nativeToLand.forEach(function (eachCountry) {  
console.log(“Grown in:” + eachCountry);  
});  
}  
}

i’m getting this error:

Syntax error at line 5 while loading: syntax error  
ativeToLand: [South America, Central Ame  
——————–^

why is it?  
thanks

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

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

[April 4, 2013 at 10:41 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-5244) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=5244#respond)

You are correct, Tomas. I forgot to wrap the strings (inside the *nativeToLand* array) in quotes. I fixed it above.

This **incorrect** line:  
nativeToLand: [South America, Central America],

Should be this:

|  |  |
| --- | --- |
|  | ​ |
|  | nativeToLand: ["South America", "Central America"], |
|  | ​ |

Thanks for bringing this to my attention.

* + - http://0.gravatar.com/avatar/0f3e1020a9ced8906320284590a48f0d?s=60&d=mm&r=g

Soumen

[July 23, 2013 at 10:04 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-21461) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=21461#respond)

var mangoFruit = {  
color: “yellow”,  
sweetness: 8,  
fruitName: “Mango”,  
nativeToLand: [“South America”, “Central America”],

showName: function () {  
console.log(“This is ” + this.fruitName);  
},  
nativeTo: function () {  
nativeToLand.forEach(function (eachCountry) {  
console.log(“Grown in:” + eachCountry);  
});  
}  
};  
mangoFruit.nativeTo();

This gives ‘ReferenceError: nativeToLand is not defined’ in jsfiddle and firebug. Where am I wrong?

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

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

[July 31, 2013 at 1:52 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-23182) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=23182#respond)

It was an error in my code. I just fixed it. I forgot the “this” keyword. It should be:

|  |  |
| --- | --- |
|  | ​ |
|  | ​// Note the "this.nativeToLand"​ |
|  | ​this.nativeToLand.forEach(function (eachCountry) { |
|  | console.log("Grown in:" + eachCountry); |
|  | }); |
|  | ​ |

I just created a JSbin with the example:  
<http://jsbin.com/anuyoc/1/edit>

1. http://0.gravatar.com/avatar/f63f35eaa35ae498332c920d513a1817?s=60&d=mm&r=g

Gmoney

[April 18, 2013 at 1:19 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-6022) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=6022#respond)

Richard,  
Went and read chapter 6 of 3rd Ed of Zakas book and I have a question.

Under the Parasitic Constr Patt and the Durable Constructor Pattern are these things true:

the whole premise of those patterns is IF you want to override the value returned ( by using the return keyword )  
AND both patterns “hide” what created it and any information about its type other then its prototype is the native Object. It also means each property that is a function is in the global namespace too?

Is that correct ?

I see some ppl argue/debate that you should ALWAYS hide information about an objects implemention.  
Basically saying to always use the factory pattern

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

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

[April 25, 2013 at 8:13 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-6404) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=6404#respond)

There is no single pattern that should ALWAYS be used for every circumstance, but there are a couple of patterns that are more optimal, more well conceived than the others.

In my post “OOP in JavaScript: What You NEED to Know,” I discuss the two best patterns for OOP in JS: 1. Combination Constructor/ Prototype Pattern: Best for Encapsulation

2. Parasitic Combination Inheritance Pattern” Best for Inheritance.

1. http://0.gravatar.com/avatar/97a724421ce496031527d10670b4bae6?s=60&d=mm&r=g

[sajay](http://loveforpixels.com/)

[April 29, 2013 at 5:20 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-6619) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=6619#respond)

You Rocks…Love your posts…

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

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

[April 29, 2013 at 6:47 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-6630) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=6630#respond)

Thanks much, Sajay. With a cool name like Sajay, you rock too 

* + - http://0.gravatar.com/avatar/97a724421ce496031527d10670b4bae6?s=60&d=mm&r=g

[sajay](http://loveforpixels.com/)

[April 29, 2013 at 7:43 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-6633) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=6633#respond)

Richard, good to hear from you, I am taking my JS skills to advance level. I find your articles very well written, easy to follow. Keep up the good work. Thanks 

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

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

[April 30, 2013 at 11:24 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-6734) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=6734#respond)

Thank you for the wonderful words, Sajay. I am very happy to hear my blog is helping you, and that you are taking your JS skills to the advanced level.

1. http://0.gravatar.com/avatar/9f8c4a70a31cac39f42c3755b83486ac?s=60&d=mm&r=g

[Wang Lei](http://github.com/wldandan)

[May 2, 2013 at 6:44 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-6810) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=6810#respond)

Really good website for studying javascript

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

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

[May 2, 2013 at 11:06 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-6827) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=6827#respond)

Thanks for the feedback, Wang.

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

Willson

[May 14, 2013 at 5:11 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-7831) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=7831#respond)

Hi Richard,

I’m a little confused. Here’s the example where I’m confused:

function HigherLearning () {  
this.educationLevel = “University”;  
}

var school = new HigherLearning ();  
school.schoolName = “MIT”;  
school.schoolAccredited = true;  
school.schoolLocation = “Massachusetts”;

In this example, you described “educationLevel” as an inherited property. However, my understanding is that this isn’t accurate (although I could totally be wrong here). First, school.hasOwnProperty(“educationLevel”) returns true which implies that this is an own property on the school object. Second, my understanding is that when the new keyword is used with a constructor function, the invocation context is the newly created object. Therefore, the “this” keyword in the HigherLearning function refers to the newly created object (i.e. this.educationLevel refers to school.educationLevel).

Is my understanding accurate?

Thanks,  
Willson

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

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

[May 15, 2013 at 12:06 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-7849) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=7849#respond)

*In this example, you described “educationLevel” as an inherited property.*

You are correct, technically 

So, this is the technical explanation (as you correctly observed):  
Since the “this” keyword in the Constructor points to the newly created object, in this case the *school* object, and therefore the *educationLevel*property will be created on the *school* object, then it follows that educationLevel is not really inherited from the Constructor, but rather, it is a property that is actually created (new property) on the school object upon invocation of said school object.

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

Willson

[May 15, 2013 at 12:14 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-7850) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=7850#respond)

Thanks for being so prompt with your responses, and helping to clarify my confusion!

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

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

[May 15, 2013 at 12:19 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-7851) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=7851#respond)

I just updated the post and give you credit for your astute observation. 

Keep pointing out errors whenever you find them. I am happy to update the articles and rid them of all typos, grammar mistakes, errors, technical missinformation, and the like.

Thanks again.

1. http://1.gravatar.com/avatar/4fde6dee0dfb32e96507fa4cc540ccdd?s=60&d=mm&r=g

A N Sinha

[May 15, 2013 at 3:23 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-7903) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=7903#respond)

Hi Richard,

Thanks for posting the Excellent Topics!, and the way of clearing the concept. this article help me how to write code in advance level. It really helpful for programmer.

basically, I am upgrading the skills set in advance level.

thanks your very much 

A.N. Sinha 

1. http://0.gravatar.com/avatar/0c170f6f45dd3f47284d60aeb07f0517?s=60&d=mm&r=g

peacelion

[June 2, 2013 at 2:30 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-10661) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=10661#respond)

there is an error with mangoFruit.showName().

showName should be:

showName: function () {  
console.log(“This is ” + this.fruitName);  
}

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

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

[June 5, 2013 at 10:57 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-11305) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=11305#respond)

The showName method added on the Constructor in the code in the article is correct.

* + - http://0.gravatar.com/avatar/0c170f6f45dd3f47284d60aeb07f0517?s=60&d=mm&r=g

peacelion

[June 6, 2013 at 5:16 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-11360) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=11360#respond)

Correct. But the typo is just before, in the mangoFruit object definition…

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

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

[June 6, 2013 at 8:50 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-11547) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=11547#respond)

I am looking for the error, but I am not seeing it, so I need your help to find it  .

Below is the full code that I copied and pasted from the article above. You can run it in JSBin or your console and you will see that it runs just fine, with the expected results.

|  |  |
| --- | --- |
|  | ​ |
|  | ​function Fruit (theColor, theSweetness, theFruitName, theNativeToLand) { |
|  | ​ |
|  | this.color = theColor; |
|  | this.sweetness = theSweetness; |
|  | this.fruitName = theFruitName; |
|  | this.nativeToLand = theNativeToLand; |
|  | ​ |
|  | this.showName = function () { |
|  | console.log("This is a " + this.fruitName); |
|  | } |
|  | ​ |
|  | this.nativeTo = function () { |
|  | this.nativeToLand.forEach(function (eachCountry) { |
|  | console.log("Grown in:" + eachCountry); |
|  | }); |
|  | } |
|  | ​ |
|  | ​ |
|  | } |
|  | ​ |
|  | ​var mangoFruit = new Fruit ("Yellow", 8, "Mango", ["South America", "Central America", "West Africa"]); |
|  | mangoFruit.showName(); // This is a Mango.​ |
|  | mangoFruit.nativeTo(); |
|  | ​ |

* + - * + http://0.gravatar.com/avatar/0c170f6f45dd3f47284d60aeb07f0517?s=60&d=mm&r=g

peacelion

[June 7, 2013 at 5:26 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-11663) /

just a few lines before in the post:

var mangoFruit = {  
color: “yellow”,  
sweetness: 8,  
fruitName: “Mango”,  
nativeToLand: [“South America”, “Central America”],

showName: function () {  
console.log(“This is ” + fruitName);  
},  
nativeTo: function () {  
nativeToLand.forEach(function (eachCountry) {  
console.log(“Grown in:” + eachCountry);  
});  
}  
}  
—  
console.log(“This is ” + this.fruitName);  
—  
the “this” is missing

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

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

[June 7, 2013 at 2:49 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-11755) /

Thanks very much for your persistent, peacelion, I just found the error and I fixed it. The code was indeed missing the “this.” Good catch.

* + - * + http://0.gravatar.com/avatar/0c170f6f45dd3f47284d60aeb07f0517?s=60&d=mm&r=g

peacelion

[June 7, 2013 at 5:27 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-11664) /

And by the way thanx a lot for your tutorials !!! IMHO the best about JS on internet.

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

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

[June 7, 2013 at 2:50 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-11756) /

Thank you, I am very happy that the blog is proving to be helpful and instructive. I really appreciate your kind words.

1. http://1.gravatar.com/avatar/47efd43c813b6a16ba65e89c839bc2d7?s=60&d=mm&r=g

Anand Kumar

[July 10, 2013 at 5:01 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-19155) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=19155#respond)

Thanks a lot for this article and others on js. I have been struggling to learn the OOP concepts in javascript and has been too lazy to go through voluminous books but I do have spent time on google and youtubes reading articles and watching videos and then I suddenly stumbled upon your blog. You have a great simple way of explaining such complex topics. I am planning to read most of your posts and I hope you keep adding content on the website. I will refer this to all my friends who want to deep dive into the OOP concents of JS.

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

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

[July 11, 2013 at 8:01 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-19397) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=19397#respond)

Thank you very much, Anand.

1. http://2.gravatar.com/avatar/e9bcc0bf70853cbb06f99ad70aade0ea?s=60&d=mm&r=g

Gracie

[July 12, 2013 at 2:24 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-19424) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=19424#respond)

Hey! Found another error at the very beginning. Yours says:

var ageGroup = {30: “Children”, 100:”Very Old”};  
console.log(ageGroup.30) // This will throw an error  
// This is how you will access the value of the property 30, to get value “Children”  
console.log(ageGroup[“30″]}; // Children

console.log’s end parenthesis is replaced with an end curly brace instead. Thanks for the post! The track is helping me a lot so far!

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

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

[July 17, 2013 at 3:05 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-20295) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=20295#respond)

Thanks very much, Gracie.

Good catch. I just fixed the typo.

1. http://2.gravatar.com/avatar/e9bcc0bf70853cbb06f99ad70aade0ea?s=60&d=mm&r=g

Gracie Gutman

[July 12, 2013 at 2:25 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-19427) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=19427#respond)

Hey! Found another error at the very beginning. Yours says:

var ageGroup = {30: “Children”, 100:”Very Old”};  
console.log(ageGroup.30) // This will throw an error  
// This is how you will access the value of the property 30, to get value “Children”  
console.log(ageGroup[“30″]}; // Children

console.log’s end parenthesis is replaced with an end curly brace instead. Thanks for the post! The track is helping me a lot so far.

1. http://0.gravatar.com/avatar/91c90187545c42ee57b1f2dae942d304?s=60&d=mm&r=g

Ehsan

[July 28, 2013 at 11:48 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-22508) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=22508#respond)

Hi  
Thanks a lot for your great posts  
I have a question what is the usage of prototype pattern in your example?  
The only usage I have in my mind is this:  
We have one object and we don’t make its properties with prototype pattern. Then we make another object that inherits the properties of the first object now we use prototype pattern to say the second object inherits all the properties of the first one. Am I right?

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

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

[July 31, 2013 at 2:46 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-23205) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=23205#respond)

Yep, you are correct, Ehsan.

1. http://0.gravatar.com/avatar/69ea6d7b8f7fe6f558a84af746af8d05?s=60&d=mm&r=g

[Sergi](http://sergibosch.com/)

[July 31, 2013 at 12:41 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-23380) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=23380#respond)

Richard, i know you appreciate typo notifications, so here’s one for you: in the article above, the word RETAIN needs to be plural.

e.g., the anotherPerson variable still retain the value > INCORRECT  
vs  
the anotherPerson variable still retains the value > CORRECT

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

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

[July 31, 2013 at 2:01 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-23402) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=23402#respond)

Very good catch, Sergi. Thanks a lot. I just fixed it.

And you are correct that I do really appreciate typos, bugs, grammatical errors, and all such notifications.

1. http://2.gravatar.com/avatar/bb66cedafba2548a0fcb7c41ce873cd1?s=60&d=mm&r=g

Tim

[August 8, 2013 at 4:02 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-26338) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=26338#respond)

What an awesome read! Helped demistify the subject completely…and the subject on prototyping was awesome.

Thanks for the great article! I’m on week three of your javascript program and I’m learning much more than my previously failed attempts using other methods.

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

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

[August 9, 2013 at 8:07 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-26853) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=26853#respond)

Wonderful!.

1. http://2.gravatar.com/avatar/57573637fe1626ab349ba8593ff8b858?s=60&d=mm&r=g

Ramesh Kunhiraman

[August 9, 2013 at 7:35 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-26590) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=26590#respond)

The best article I came across on JavaScript. Excellent piece of work

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

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

[August 9, 2013 at 8:18 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-26856) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=26856#respond)

Those are wonderful words, Ramesh :). Thank you very much.

1. http://2.gravatar.com/avatar/5ee0e82283a434136970be4f6a77bd58?s=60&d=mm&r=g

antonio

[August 9, 2013 at 11:40 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-26673) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=26673#respond)

Hey Richard,  
Just reread this post and now I totally get it! I actually reread the post about ‘passing by value vs. reference’  
Thanks,  
antonio

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

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

[August 9, 2013 at 8:19 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-26857) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=26857#respond)

I am glad to hear your JS skills are improving. Keep up the good work.

1. http://0.gravatar.com/avatar/0200df25bb410a5bc96f6905de423cd4?s=60&d=mm&r=g

[thetrystero](http://gplus.to/thetrystero)

[September 15, 2013 at 2:26 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-38070) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=38070#respond)

howSweetAmI: function () {  
console.log(“Hmm Hmm Good”);  
}  
}

Was there a typo here? there are two closing braces, but only one opening brace.

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

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

[September 16, 2013 at 4:47 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-38395) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=38395#respond)

No, there was not a typo in that code (although I do make mistakes often  ).

Here is the full code:

|  |  |
| --- | --- |
|  | ​ |
|  | ​// This is an object with 4 items, again using object literal​ |
|  | ​var mango = { |
|  | color: "yellow", |
|  | shape: "round", |
|  | sweetness: 8, |
|  | ​howSweetAmI: function () { |
|  | console.log("Hmm Hmm Good"); |
|  | } |
|  | ​ |
|  | } // this last closing bracket belongs to the <em>mango</em> object​ |
|  | ​ |

1. http://1.gravatar.com/avatar/40fc0acae1d23d81d942ea4734314912?s=60&d=mm&r=g

Foysal Mamun

[September 30, 2013 at 7:43 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-57742) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=57742#respond)

Thanks for another good post.

1. http://1.gravatar.com/avatar/14282fa30326b5907eaffc1346e60817?s=60&d=mm&r=g

Deepak

[October 3, 2013 at 6:58 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-66316) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=66316#respond)

Thanks so much very useful.

1. http://1.gravatar.com/avatar/ab35517a21869b6513e64233d56f19d4?s=60&d=mm&r=g

Mohammed

[October 6, 2013 at 5:49 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-76204) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=76204#respond)

Great post!

1. http://1.gravatar.com/avatar/405038161d8b69a9a9e9b518e8edf3c0?s=60&d=mm&r=g

[Alessandro](http://www.alessandrosantese.com/)

[October 7, 2013 at 12:37 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-77793) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=77793#respond)

Great article, very well explained. I am a little bit more confident on objects now.

1. http://0.gravatar.com/avatar/c008eb957d2b7ece83e5ca102dccece9?s=60&d=mm&r=g

Doni Noeit

[November 11, 2013 at 6:02 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-152960) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=152960#respond)

“JavaScript has one complex data type, the Object data type, and it has five simple data types: Number, String, Boolean, Undefined, and Null. Note that these simple (primitive) data types are immutable, they cannot be changed, while objects are mutable.”

[1] So if the String datatype is immutable, what does this do?

String.prototype.hasStuff = function (s) {  
return Boolean(this.length > 0);  
}

>> “tommy”.hasStuff()  
true

[2] So if,

>>typeof(“tommy”)  
“string”  
>>typeof(4)  
“number”  
>>typeof(true)  
“boolean”  
>>typeof(undefined)  
“undefined”

and if

>>var bands = [“The Clash”, “Davis Bowie”, “Pink Floyd”]  
>>typeof(bands)  
“object”

>>typeof(null)  
“object”

then null is a datatype how?

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

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

[November 22, 2013 at 7:18 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-163145) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=163145#respond)

1. What you have done is add a method to the String prototype, which you can do for any of the data type.

By “immutable”, we mean you cannot change the instance of the data type: If you create a string instance, you cannot change it. You can alter a copy of it and save the copy to another variable, but the original string will not be changed.

For example (Immutable Data Type):

var originalStr = "Hello There";

originalStr.length = 0;

// The string did not change

console.log(originalStr); // "Hello There"

var newStr = originalStr.toLowerCase();

console.log(newStr); // "hello there"

// The original string variable never changed

console.log(originalStr); // "Hello There"

On the other hand, if we make any changes to an Object data type, such as an array, look what happens:  
(Mutable Data Types)

var originalArr =["Hello There"];

originalArr.length = 0;

// The array changed; it is now empty

console.log(originalArr); []

originalArr.push("How are you?");

// It changed again

console.log(originalArr); // ["How are you?"]

2. Yes, Null is a data type, in fact.

1. http://0.gravatar.com/avatar/fd9fad4e3ccb671bd43e63f5cfac1406?s=60&d=mm&r=g

Elhadi

[December 15, 2013 at 10:08 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-171992) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=171992#respond)

Thank you very much, the mentionned book is excellent also.

1. http://2.gravatar.com/avatar/23ef6f94c4e61e0164e2e81129eca3d3?s=60&d=mm&r=g

syed shaik shavali

[January 7, 2014 at 11:55 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-177769) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=177769#respond)

Thank you Richard.  
Its good learning blog.

Syed.

1. http://0.gravatar.com/avatar/985a9d8fa248be1df17489fe42fd6487?s=60&d=mm&r=g

Alex Andrew

[January 12, 2014 at 1:12 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-178943) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=178943#respond)

nice articels!

1. http://1.gravatar.com/avatar/a90e154319fe0a630bc8d820eda2b127?s=60&d=mm&r=g

sanjeev dhakal

[January 29, 2014 at 3:54 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-181685) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=181685#respond)

i had learned javascript for past two weeks and wile printing output i used document.write but now onward while i am going to learn more advance javascript i find console.log instead of document.write i cannot diffirentiate their working mechanism.

* + http://1.gravatar.com/avatar/49b560c31c901b112462ca8a12afd45d?s=60&d=mm&r=g

neelesh jain

[March 5, 2014 at 6:26 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-186882) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=186882#respond)

Hey,  
console.log(“xyz”); – prints “xyz” on the console of the browser, not on the browser HTML page. only the developer can see this, not the user of the website and hence, it is mainly used for debugging.  
while,  
document.write(“xyz”); – will print “xyz” on the webpage itself.

1. http://1.gravatar.com/avatar/49b560c31c901b112462ca8a12afd45d?s=60&d=mm&r=g

neelesh jain

[March 5, 2014 at 6:23 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-186881) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=186881#respond)

Thanks for the post. It was very useful for me as i am just beginning on the JS journey, it cleared quite a few concepts.  
Thankyou Again

1. http://2.gravatar.com/avatar/22ce4ce57dc096f855531675310a720c?s=60&d=mm&r=g

Pravin Waychal

[March 7, 2014 at 2:23 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-187315) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=187315#respond)

Hi Richard,

Will you please add example which shows how to get values of property in object.  
or maybe be its allready written and I couldn’t able to find it out.

var Fruit = function(color){  
this.color = color  
}

var mango = new Fruit(“yellow”);  
mango.taste = “sweet”;

for(var prop in mango){  
console.log(mango[prop]); // Prints Yellow, Sweet  
}

and thanks so much for making this complex subject so easy 

1. http://2.gravatar.com/avatar/229db2578c8c5faf8a1e962b3cab9ffd?s=60&d=mm&r=g

Mike K

[March 7, 2014 at 7:52 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-187626) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=187626#respond)

Great runthrough of objects in JS, thanks very much!

Small typo/problem:  
“To delete a property from an object, you use the delete operator. You cannot delete properties that were inherited, nor can you delete properties with their attributes set to configurable.”

Did you mean “… attributes set to NOT configurable”?

1. http://0.gravatar.com/avatar/913bfeed7ee6816a27ce103dd22e977f?s=60&d=mm&r=g

Chandra

[March 26, 2014 at 2:51 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-192089) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=192089#respond)

Hello Richard,  
Good information and site for javascript lovers.  
Is there any such website which is informative and easy to understand for jQuery, HTML5, CSS3.

1. http://1.gravatar.com/avatar/a4861a4ab4db5f729312673019d9e80d?s=60&d=mm&r=g

Tarak

[May 5, 2014 at 3:20 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-218954) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=218954#respond)

Hi Richard,

Thanks for posting the Excellent Topics!. Your explanation is also crystal clear.

-Thanx,  
Tarak

1. http://0.gravatar.com/avatar/98e4f58ddcf2072edcfa1bcafaab34f2?s=60&d=mm&r=g

Rai Butera

[May 14, 2014 at 10:36 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-228243) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=228243#respond)

I found the sections “Accessing and Enumerating Properties on Objects” and “Accessing Inherited Properties” highly confusing as you don’t seem to go over exclusively enumerating own properties, rather you seem to demonstrate inheritance in two ways.

1. http://1.gravatar.com/avatar/44fe75e3c8aeb480818d8871a381f20c?s=60&d=mm&r=g

Shaikh Siraj

[July 8, 2014 at 4:26 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-255878) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=255878#respond)

Hi Richard

Very Very good example indeed, i know quit a lot about javascript, but still i learnt so much, especially foreach, delete property good

Keep going on..

1. http://2.gravatar.com/avatar/b4638fef5ff6a7ca59239ce5678c6f9e?s=60&d=mm&r=g

MaC

[July 29, 2014 at 2:24 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-264604) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=264604#respond)

Hey kudos to the nice article. Thank you very much. Really refresh javascript

1. http://2.gravatar.com/avatar/8b38f956b5654a38c96840307674af39?s=60&d=mm&r=g

Anthony Wijnen

[July 30, 2014 at 7:05 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-265263) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=265263#respond)

Hey Richard!

First off, great stuff. I’m really enjoying your tutorials.

I was wondering about the following. You mentioned that using the constructor pattern for creating objects, you can easily make changes ones which permeate everywhere: “If you had to change the fruitName function, you only had to do it in one location. The pattern encapsulates all the functionalities and characteristics of all the fruits in by making just the single Fruit function with inheritance.”

How would I do that? Let’s say I created a couple of fruits with the Fruit() constructors and now I want them all to have a different showName() method. How would I accomplish that?

Thanks a lot!  
Anthony

1. http://1.gravatar.com/avatar/7a6b24380fc22a71f7f110ca586204a7?s=60&d=mm&r=g

Benicio

[August 20, 2014 at 9:39 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-273755) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=273755#respond)

A typo: In the section that reads ” it never stored an actual copy of it’s own value”, should be ” it never stored an actual copy of its own value”.

Sorry, I like your posts. I go ADD when I see “it’s” when it should be “its”, I stop reading and wonder why that happens and if the code might have a typo as well.

1. http://2.gravatar.com/avatar/b8f8b46e3f6c8ba0836810a15dfb345a?s=60&d=mm&r=g

Juan Llosas

[September 8, 2014 at 3:07 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-285267) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=285267#respond)

Man the guys got gypped that Christmas.. unless Mike got Professional JavaScript for Developers ofcourse.

1. http://2.gravatar.com/avatar/8eeb808fafa9276e0810921265be78e9?s=60&d=mm&r=g

asif

[September 16, 2014 at 10:53 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-290076) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=290076#respond)

why there is need for creating copy of built in object,  
ex:- var now = new Date();  
why we can not use Date() object directly.,

1. http://0.gravatar.com/avatar/c60f8966258bd6ae42eb225cbe9747a8?s=60&d=mm&r=g

ShafiShawon

[September 24, 2014 at 2:13 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-295571) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=295571#respond)

Really It’s a great post. You deserve (y). I have many confusions before reading this.

1. http://2.gravatar.com/avatar/8e74d1abdd03f2f14edbb62809ce2d59?s=60&d=mm&r=g

[Kris](http://efffeduplife.blogspot.in/)

[December 5, 2014 at 7:19 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-305938) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=305938#respond)

Great post! Under “Deleting Properties of an Object ,” you’ve written.. “nor can you delete properties with their attributes set to configurable.”

That should be “non-configurable” or configurable set to false right?

1. http://0.gravatar.com/avatar/9a9a212c92b86143ffd5f2b14e7429b5?s=60&d=mm&r=g

[Andrew](http://mosaic.ws/)

[January 11, 2015 at 6:31 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306036) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306036#respond)

I’ve been following through your advanced web developers track, and it’s been great so far, but I’ve hit a wall on objects. I don’t understand getters and setters (and accessor/data properties in general), which you don’t seem to mention, but chapter 6 does in detail.

What is the value of using a ‘value’ accessor property over simply assigning a value to an attribute normally?  
Also, what purpose do get and set fill? Get runs a function when that property is accessed (in the example, “year”), and set runs a function when that property is written to? Am I understanding correctly? I still feel shaky on objects in general, but I feel like I have a foundation to work on. I don’t feel remotely that way about these properties.

* + http://0.gravatar.com/avatar/9a9a212c92b86143ffd5f2b14e7429b5?s=60&d=mm&r=g

[Andrew](http://mosaic.ws/)

[January 11, 2015 at 6:36 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306037) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306037#respond)

If I understand correctly, get lays out the behavior for what the defined property should do when it’s called. In the case of the example in the book, on page 177, that is simply to return the same value that \_year holds. (But why not just define year to begin with and skip the \_year assignment? What do we gain from that?)

Set, on the other hand, lays out the behavior for what year should do when it’s written to. In the case of this example, it updates its own value (and \_year) as well as updating the edition accordingly.

Am I on the right track? I’m still fuzzy on the real-world utility of Object.defineProperty().

* + http://0.gravatar.com/avatar/9a9a212c92b86143ffd5f2b14e7429b5?s=60&d=mm&r=g

[Andrew](http://mosaic.ws/)

[January 11, 2015 at 6:40 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306038) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306038#respond)

Apologies to deluge with comments. I’m just questioning the real world utility of defineProperty(). For example, doesn’t this code snippet accomplish the same thing as the example on page 177, without using this property? It just seems like needless complication.

var bookTest = {  
year: 2004,  
edition: 1,

updateEdition: function(newYear) {  
if (newYear > 2004) {  
this.year = newYear;  
this.edition += newYear – 2004;  
}  
}  
};

bookTest.updateEdition(2010);  
alert(bookTest.edition);

1. http://0.gravatar.com/avatar/c673474949c68dcbb7059366022f850e?s=60&d=mm&r=g

[Gytis](http://sa.sa/)

[January 30, 2015 at 3:24 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306082) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306082#respond)

I found the use of functions together with “new” keyword interesting. Maybe you should a a bit explaining how/why this works in JS.

1. http://0.gravatar.com/avatar/c4489b2856f1c05b56a53725fcf92ebd?s=60&d=mm&r=g

[avoaja](http://biito.com/)

[February 10, 2015 at 8:44 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306093) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306093#respond)

Very good post. Written 2 years ago but still fresh.

1. http://2.gravatar.com/avatar/57e0c9e63aadc6cc167f76249f412316?s=60&d=mm&r=g

Ajay

[February 17, 2015 at 6:34 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306115) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306115#respond)

Hi,  
In the “Constructor pattern for objects” section, there needs to be a semi-colon at the end of the braces  
here:

this.showName = function () {  
console.log(“This is a ” + this.fruitName);  
};  
​  
this.nativeTo = function () {  
this.nativeToLand.forEach(function (eachCountry) {  
console.log(“Grown in:” + eachCountry);  
});  
};

Google Chrome’s console throws an ILLEGAL token exception without the semicolon.

Thank you for the explanation. It helped me a lot.

1. http://1.gravatar.com/avatar/aa65640a366a6c0a81bfed07ddd128b6?s=60&d=mm&r=g

gheorghe

[March 14, 2015 at 8:34 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306260) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306260#respond)

Hi, I really love your posts, I need to say that I love to follow you, but I have one question. I saw that you recommend several books, like “JavaScript.The Definitive Guide” or Professional JavaScript for Web Developers” ….so I took the decision to stard reading one of these, but I must choose only one, not both because there is no time for both, so why would you think is better?

* + http://1.gravatar.com/avatar/aa65640a366a6c0a81bfed07ddd128b6?s=60&d=mm&r=g

gheorghe

[March 14, 2015 at 8:39 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306261) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306261#respond)

sorry, I`ve made a mistake at the end:  
…. so, WHAT(not why) would you think is better?

1. http://1.gravatar.com/avatar/4216915565e5f02c3c15ef4f5bf601a5?s=60&d=mm&r=g

[some](http://some/)

[May 17, 2015 at 2:23 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306768) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306768#respond)

Hi,  
Could you tell what is the diffrence between using:  
this.property (or method) in the constructor and :  
Constructor.prototype.property (or method).

I will be very happy if you find time and explain me that 

1. http://0.gravatar.com/avatar/923ebf6c9914ac5f00d865be5f89a930?s=60&d=mm&r=g

Anand Rajput

[May 29, 2015 at 4:11 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-306899) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=306899#respond)

Great inline explanation of Object..you are shakespeare of programming.!!! Great Post .

1. http://1.gravatar.com/avatar/1da5d127e368a20c571a64e0db4d9fe4?s=60&d=mm&r=g

snehashis

[June 28, 2015 at 6:58 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307134) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307134#respond)

Really Good

1. http://1.gravatar.com/avatar/d6bda2598399b0725c7b5ca55d0ea0cb?s=60&d=mm&r=g

PiotrG

[June 30, 2015 at 8:08 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307138) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307138#respond)

I read all the article and comments and I;m still little confused about this.  
You write:  
“In the last example, note the educationLevel property that was defined on the HigherLearning function is listed as one of the school’s properties, even though educationLevel is not an own property—it was inherited.”

and near below you write:  
“console.log(school.hasOwnProperty(“educationLevel”)); true​  
​// educationLevel is an own property on school, so we can delete it​”

I saw the note where you add that the educationLevel property is not actually inherited by objects that use the HigherLearning constructor, so I think I get it.

But this text is confusing “educationLevel is not an own property—it was inherited” and below ” educationLevel is an own property on school, so we can delete it​”

And the second ..  
Please explain me this, because I think I don’t understand this the way I should.  
“Properties inherited from Object.prototype are not enumerable, so the for/in loop does not show them”

So why the for in loop below show me print function and author property ?

function PrintStuff (myDocuments) {  
this.documents = myDocuments;  
}

PrintStuff.prototype.print = function () {  
console.log(this.documents);  
}

PrintStuff.prototype.author = “Stephen King”;  
var newObj = new PrintStuff();

for (x in newObj) {  
console.log(x);  
}

//documents, print, author.

* + http://0.gravatar.com/avatar/fa86b2057559e7c574333a8166cf3510?s=60&d=mm&r=g

[Frank Topel](https://connexo.de/)

[February 2, 2018 at 8:50 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-309014) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=309014#respond)

“Properties inherited from Object.prototype are not enumerable, so the for/in loop does not show them”. Then you give example “PrintStuff.prototype.author”. This does not inherit from Object.prototype. It inherits from PrintStuff.prototype.

1. http://2.gravatar.com/avatar/bbc481ae8d6ec906943a2b6f5267a1df?s=60&d=mm&r=g

vasanth

[July 9, 2015 at 12:52 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307178) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307178#respond)

In the last example, note the educationLevel property that was defined on the HigherLearning function is listed as one of the school’s properties, even though educationLevel is not an own property—it was inherited.

console.log(school.hasOwnProperty(“educationLevel”)); true​  
​// educationLevel is an own property on school, so we can delete it​  
​delete school.educationLevel; true

In the first condition you said that educationLevel is not the own property of school it’s inherited. In second instance while deleting you said it has own property..Little bit confused.

1. http://2.gravatar.com/avatar/ef92759c6e2b9d891d200b389d9a712f?s=60&d=mm&r=g

Saras Arya

[September 10, 2015 at 6:14 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307367) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307367#respond)

I have two questions regarding delete.  
First You cannot delete if property of object is set to Configurable. In the post you said the three attributes i.e configurable, enumerable and writable are set to true by default. If yes then how can you delete properties of any objects without setting configurable to false. which I don’t see in any of your examples

Second You said properties of any object with var keyword cannot be deleted while you have the same in the following example. What is correct or what am I missing ? Kindly Explain?

1. http://2.gravatar.com/avatar/ef92759c6e2b9d891d200b389d9a712f?s=60&d=mm&r=g

Saras Arya

[September 10, 2015 at 6:18 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307368) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307368#respond)

I have two questions regarding delete.  
First You cannot delete if property of object is set to Configurable. In the post you said the three attributes i.e configurable, enumerable and writable are set to true by default. If yes then how can you delete properties of any objects without setting configurable to false. which I don’t see in any of your examples

Second You said properties of any object with var keyword cannot be deleted while you have the same in the following example. What is correct or what am I missing ? Kindly Explain

1. http://0.gravatar.com/avatar/f80832438dce421e1544d0ceefde7100?s=60&d=mm&r=g

Alessandro

[September 15, 2015 at 7:01 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307379) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307379#respond)

Thank you 

1. http://1.gravatar.com/avatar/1d55c8291d3bfb7391d3977bf9601a1a?s=60&d=mm&r=g

DFF

[October 7, 2015 at 3:15 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307430) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307430#respond)

Just came across this article. It’s a great resource for learning JS Objects

1. http://2.gravatar.com/avatar/5762f939cdd3e1937efe8a168de91258?s=60&d=mm&r=g

[王如锵](http://undefined/)

[October 13, 2015 at 9:28 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307438) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307438#respond)

You said that “ducationLevel is not an own property—it was inherited”.

And you said that “you cannot delete properties that were inherited”.

So I am confused why that:  
console.log(school.hasOwnProperty(“educationLevel”)); true​  
​// educationLevel is an own property on school, so we can delete it​  
​delete school.educationLevel; true

1. http://2.gravatar.com/avatar/5762f939cdd3e1937efe8a168de91258?s=60&d=mm&r=g

[王如锵](http://undefined/)

[October 13, 2015 at 9:28 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307439) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307439#respond)

You said that “educationLevel is not an own property—it was inherited”.

And you said that “you cannot delete properties that were inherited”.

So I am confused why that:  
console.log(school.hasOwnProperty(“educationLevel”)); true​  
​// educationLevel is an own property on school, so we can delete it​  
​delete school.educationLevel; true

1. http://0.gravatar.com/avatar/92e91d9d369e066477b98d730de0b521?s=60&d=mm&r=g

Bart Kleijngeld

[October 30, 2015 at 8:46 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307495) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307495#respond)

TL;DR The second time the name property is set, the obj variable points to another -the newly created- object than the first time, which dies after the function execution finishes.

Within the `setName’ function, let’s focus on the line:  
obj = new Object();

Here a new object is created and the obj variable is reassigned to point to it. When the name property is set on the line following, it is thus set on that object. Once the function finishes execution, because of its scope this object dies. Back in the global scope, the person variable points to the original object again, which had its name property set to’Nicholas’.

1. http://1.gravatar.com/avatar/4c1c475ce193740d8e6c956741df9c0d?s=60&d=mm&r=g

Vinay

[November 5, 2015 at 2:37 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307501) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307501#respond)

Hello Richard, great explanation even a 6 year old can understand easily.

kindly write a article regarding mongoose explanation as it is frequently used in node js and regular expressions if possible.

thanks , you are my mentor 

1. http://0.gravatar.com/avatar/3f009d72559f51e7e454b16e5d0687a1?s=60&d=mm&r=g

[James](http://na/)

[November 30, 2015 at 6:52 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307578) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307578#respond)

​// Prints false because we did not define a schoolType property on the school object, and neither did the object inherit a schoolType property from its prototype object Object.prototype.​  
console.log(“schoolType” in school); // false​

Can you elaborate when you referring to “neither did the object inherit schoolType ……from its prototype object” What is object Object.prototype.​ Where do we find this in the example

1. http://2.gravatar.com/avatar/8c6f40bf0432bafbc5dbce597beb97b4?s=60&d=mm&r=g

SAUMYA RANJAN SAHU

[December 3, 2015 at 2:35 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307627) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307627#respond)

nice post!! good to know the basics

1. http://1.gravatar.com/avatar/78a4864d5f1e153751f2eb1315c550f7?s=60&d=mm&r=g

[Jirayu Limjinda](http://www.jindatheme.com/)

[December 21, 2015 at 4:41 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307746) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307746#respond)

So you make me clear with javascript object  
thank you so much.

1. http://0.gravatar.com/avatar/99553ca0941eed18d49e0f07555b511c?s=60&d=mm&r=g

Khim

[December 25, 2015 at 12:24 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307755) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307755#respond)

Your JSON.stringfy example has an error that say it will print:  
// “{“mike”:”Book”,”jason”:”sweater”,”chels”:”iPad”}”    
The “chels” should be “chelsea”. Just thought it will help.

1. http://0.gravatar.com/avatar/621d69eaae2b9360752e80a3245214b5?s=60&d=mm&r=g

Kelvin

[January 13, 2016 at 5:35 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307791) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307791#respond)

What can I say, you just explained the topic better than Nicholas C, Zakas

1. http://2.gravatar.com/avatar/8b85b64ca1aba39118b45f6d83671bfd?s=60&d=mm&r=g

Tarun Nagpal

[February 12, 2016 at 10:01 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307872) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307872#respond)

Awesome

1. http://0.gravatar.com/avatar/67e354e212d780ea847b2846b40474e4?s=60&d=mm&r=g

Mahesh

[February 25, 2016 at 4:27 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307907) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307907#respond)

Hello,

I am beginner to javascript. may i know from which post should i start to learn full JS.

1. http://2.gravatar.com/avatar/25c93e8900c9dbd1ec44138342d0d5c8?s=60&d=mm&r=g

Mahesh

[February 25, 2016 at 4:30 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307908) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307908#respond)

Hello,

which is the best way to learn full JS and also please suggest me the study material

1. http://2.gravatar.com/avatar/82522d7a6258a53b13ed4cdebaea49e1?s=60&d=mm&r=g

Rahul

[March 1, 2016 at 8:59 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307919) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307919#respond)

Great tutorial , this really helped me clear my basics of object finally .

Thanks a ton !!!!

1. http://1.gravatar.com/avatar/418395e6324ba367a8770e9c1272e8bc?s=60&d=mm&r=g

Ajay

[March 9, 2016 at 5:12 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307937) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307937#respond)

It is good article sir, but there is correction in first example that native place of “Mango” is India…not an America, the scientific word of mango is “Mangifera indica”. Do search on mango on internet you will find:  
“The common mango species, or Mangifera indica, grows on the largest fruit tree in the world. Mango trees are native to northeast India, where Hindu writings about mangoes date back to 4000 B.C. Mangoes were domesticated in India, and spread to East Asia around 400 B.C. By the 10th century, mangoes were cultivated in East Africa, followed by the Philippines in the 15th century. Portuguese traders brought mango seeds along their route, spreading the trees to their African and Brazilian colonies in the 16th century.”

kindlly, update your example sir.

1. http://1.gravatar.com/avatar/4a4ce508469c999fd67e88ec38e57a12?s=60&d=mm&r=g

Venkatesh

[March 10, 2016 at 1:53 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307939) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307939#respond)

Hi,

I have little confusion in accessing the property in the object basically through bracket notation without quotes ” ”

//Or, in case you have the property name in a variable:​  
​var bookTitle = “title”;  
console.log ( book[bookTitle]); // Ways to Go​  
console.log (book[“bookMark” + 1]); // Page 20

In this how can we create a ” var bookTitle = “title”; ” inside a constructor functions so that we can access the property/variable without using notation like this book[bookTitle]

Sorry if my question is too basic  However im a starter right now in javascript 

1. http://2.gravatar.com/avatar/23c404c2ea07dd67eabb195e960494da?s=60&d=mm&r=g

Idris

[March 28, 2016 at 10:35 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-307986) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=307986#respond)

Good one .. ease to learn

1. http://2.gravatar.com/avatar/e30ab5f23def199862bb2e57b96cdd3a?s=60&d=mm&r=g

Jatni

[April 15, 2016 at 6:06 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308025) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308025#respond)

Very nice post in not so comlex term.  
Thank for posting it.

1. http://0.gravatar.com/avatar/01ed5e82797dd35bd5812fe0d958de68?s=60&d=mm&r=g

viet

[April 15, 2016 at 11:49 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308026) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308026#respond)

Iam sorry SIr, I’ve changed the code a little bit:  
I wrote:  
this.nativeToLand.forEach(function (eachCountry) {  
console.log(“Grown in:” + eachCountry);  
}  
instead of  
this.nativeTo = function () {  
this.nativeToLand.forEach(function (eachCountry) {  
console.log(“Grown in:” + eachCountry);  
});  
then it returned: this.nativeToLand is not defined. Why did it happens

1. http://0.gravatar.com/avatar/01ed5e82797dd35bd5812fe0d958de68?s=60&d=mm&r=g

viet

[April 15, 2016 at 11:49 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308027) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308027#respond)

Iam sorry SIr, I’ve changed the code a little bit:  
I wrote:  
this.nativeToLand.forEach(function (eachCountry) {  
console.log(“Grown in:” + eachCountry);  
}  
instead of  
this.nativeTo = function () {  
this.nativeToLand.forEach(function (eachCountry) {  
console.log(“Grown in:” + eachCountry);  
});  
then it returned: this.nativeToLand is not defined. Why did it happen

1. http://0.gravatar.com/avatar/01ed5e82797dd35bd5812fe0d958de68?s=60&d=mm&r=g

viet

[April 15, 2016 at 12:20 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308028) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308028#respond)

Oops, i’ve made a mistake. I reviewed my code again and found that. Sorry for that. Anyway, your blog is awesome and help me a lot. Thank u!

1. http://2.gravatar.com/avatar/50f448af1ec00a55beeb96346efaa78e?s=60&d=mm&r=g

NITESH

[April 16, 2016 at 2:26 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308029) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308029#respond)

Nice post about object and very helpfully .thanks for building my concept.

1. http://0.gravatar.com/avatar/0ebd5c8eff38efa1c795dc4d50ed2bda?s=60&d=mm&r=g

FlyingGambit

[April 18, 2016 at 11:42 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308033) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308033#respond)

An example I tried to see if it deletes inherited properties

var Person = function(name){this.name = name;}  
var john = new Person(“John”);  
Person.prototype.race = “Human”;  
console.log(john); // shows {name: “John”}  
john.hasOwnProperty(“race”); // shows false  
john.race // shows “Human”  
delete john.race // shows true  
john.race // shows “Human”

1. http://0.gravatar.com/avatar/0ebd5c8eff38efa1c795dc4d50ed2bda?s=60&d=mm&r=g

FlyingGambit

[April 18, 2016 at 11:44 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308034) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308034#respond)

What is the difference between object property and object attribute ?

1. http://0.gravatar.com/avatar/0ebd5c8eff38efa1c795dc4d50ed2bda?s=60&d=mm&r=g

FlyingGambit

[April 18, 2016 at 11:48 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308035) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308035#respond)

Your articles are great, if you can then plz include summary sections at the end, some bullet points for quick referencing. I often find myself having to read the entire article again.

1. http://1.gravatar.com/avatar/1dbab1b7055f10a09d5bd9a63aaf37da?s=60&d=mm&r=g

Jacek J.

[April 27, 2016 at 7:38 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308053) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308053#respond)

“nor can you delete properties with their attributes set to configurable” —–> should be changed to:  
” “nor can you delete properties with their attributes set to NON – configurable”

I know you don’t update it any longer (which is REALLY sad…), but the page is a really helpful and I hope you will change this so other people won’t get confused.

1. http://0.gravatar.com/avatar/0c99b646ac1e9d8f217ff8eae000b965?s=60&d=mm&r=g

ilhan

[April 28, 2016 at 1:54 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308055) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308055#respond)

Hi great article!

I have a comment on the following example:

function HigherLearning () {  
​this.educationLevel = “University”;  
}  
​  
​// Implement inheritance with the HigherLearning constructor​  
​var school = new HigherLearning ();  
school.schoolName = “MIT”;  
school.schoolAccredited = true;  
school.schoolLocation = “Massachusetts”;

​  
​//Use of the for/in loop to access the properties in the school object​  
​for (var eachItem in school) {  
console.log(eachItem); // Prints educationLevel, schoolName, schoolAccredited, and schoolLocation​  
}

In the last example, note the educationLevel property that was defined on the HigherLearning function is listed as one of the school’s properties, even though educationLevel is not an own property—it was inherited.

My question:  
Are you sure “educationLevel property” is not an own property?  
I think it would not be an own property if it was defined on HigherLearning.prototype, e.g HigherLearning.prototype.educationLevel.  
But in this case I believe it is an own property.

You can use to test it yourself.  
for (prop in school) {  
if (school.hasOwnProperty(prop)) {  
console.log(“prop: ” + prop)  
}  
}

I am looking forward to your reply

1. http://2.gravatar.com/avatar/b9be0d0f9287a7fb3da92af6bc915a9b?s=60&d=mm&r=g

giorgi

[October 12, 2016 at 3:51 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308362) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308362#respond)

Below is bad example even if string were reference type still anotherPerson would have Kobe, please correct  
—-  
// The primitive data type String is stored as a value​  
​var person = “Kobe”;  
​var anotherPerson = person; // anotherPerson = the value of person​  
person = “Bryant”; // value of person changed​  
​  
console.log(anotherPerson); // Kobe​  
console.log(person); // Bryant

1. http://0.gravatar.com/avatar/9ab26cee462bb3deee7835f9f7618850?s=60&d=mm&r=g

giorgi

[October 12, 2016 at 6:12 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308363) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308363#respond)

Also when showing two ways to create objects using constructor function and prototypes you don’t highlight what are differences with these approaches, when there are, e.g. prototype stuff is shared

1. http://0.gravatar.com/avatar/f1db2a4b6f7fd1a4c575dd7aa45093b3?s=60&d=mm&r=g

Amit

[October 19, 2016 at 1:23 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308377) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308377#respond)

I like your post

1. http://0.gravatar.com/avatar/35f2b1c295a70888fb190b8de0fd07e2?s=60&d=mm&r=g

Sachdev

[October 26, 2017 at 9:04 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308885) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308885#respond)

Thank you !!

1. http://0.gravatar.com/avatar/c89a454c7dc2283795c34af4b87e6a75?s=60&d=mm&r=g

Horace

[November 27, 2017 at 11:33 pm](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308935) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308935#respond)

Thank you. Been looking for a Magic Javascript Demystifier for centuries, and I finally found it.

1. http://0.gravatar.com/avatar/344abc6afa074233e53dd60a7ef5973b?s=60&d=mm&r=g

Yakhouba

[January 17, 2018 at 5:13 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-308994) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=308994#respond)

Very good post, but I have a question: Is Null really a type ?  
typeof null = object

Can I have an example where it is clearly shown that Null is a type ?

Thanks!

1. http://0.gravatar.com/avatar/fa86b2057559e7c574333a8166cf3510?s=60&d=mm&r=g

[Frank Topel](https://connexo.de/)

[February 2, 2018 at 8:41 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-309012) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=309012#respond)

Reference Error: “Accessing a property on an object that does not exist will result in undefined.” If the object does not access, this will throw an error “Cannot access property of undefined”. That wording in your sentence is totally ambigous.

1. http://0.gravatar.com/avatar/fa86b2057559e7c574333a8166cf3510?s=60&d=mm&r=g

[Frank Topel](https://connexo.de/)

[February 2, 2018 at 8:42 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-309013) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=309013#respond)

Reference Error: “Accessing a property on an object that does not exist will result in undefined.” If the object does not exist, this will throw an error “Cannot access property of undefined”. That wording in your sentence is totally ambigous.

1. http://2.gravatar.com/avatar/ed890afffcd4e07b157d8d35ac2d6ab2?s=60&d=mm&r=g

Vinodh

[February 14, 2018 at 9:10 am](http://javascriptissexy.com/javascript-objects-in-detail/#comment-309031) / [Reply](http://javascriptissexy.com/javascript-objects-in-detail/?replytocom=309031#respond)

You just mentioned “Also, you cannot delete properties of the global object, which were declared with the var keyword.”

And then in example a property inside a var keyword variable was deleted. Does the above statement hold good?

Trackbacks for this post

1. [JavaScript Prototype in Plain, Detailed Language | JavaScript is Sexy](http://javascriptissexy.com/javascript-prototype-in-plain-detailed-language/)
2. [16 JavaScript Concepts You Must Know Well | JavaScript is Sexy](http://javascriptissexy.com/16-javascript-concepts-you-must-know-well/)
3. [Learn Node.js Completely and with Confidence | JavaScript is Sexy](http://javascriptissexy.com/learn-node-js-completely-and-with-confidence/)
4. [Learn Backbone.js Completely | JavaScript is Sexy](http://javascriptissexy.com/learn-backbone-js-completely/)
5. [Javascript | Pearltrees](http://www.pearltrees.com/owen.mcgregor/javascript/id3615617)
6. [How to Learn JavaScript Properly | JavaScript is Sexy](http://javascriptissexy.com/how-to-learn-javascript-properly/)
7. [OOP In JavaScript: What You NEED to Know | JavaScript is Sexy](http://javascriptissexy.com/oop-in-javascript-what-you-need-to-know/)
8. [Learning JS Properly – Week 2 | coding(isBananas);](http://codingisbananas.wordpress.com/2013/04/16/learning-js-properly-week-2/)
9. [JS is Sexy Week 2- A bit on Method and Properties | B's Blog](http://www.briandavishall.com/a-bit-on-method-and-properties/)
10. [Javascript is Sexy Week 1&2- Structure at Last, I Found a Study Group | B's Blog](http://www.briandavishall.com/structure-at-last-found-a-study-group/)
11. [Abhinav (archmonk) | Pearltrees](http://www.pearltrees.com/archmonk)
12. [Apply, Call, and Bind Methods are Essential for JavaScript Professionals | JavaScript is Sexy](http://javascriptissexy.com/apply-call-and-bind-methods-are-essential-for-javascript-professionals-understand-them-well/)
13. [A Simple, Comprehensive Overview of Javascript | BetterExplained](http://betterexplained.com/articles/the-single-page-javascript-overview/)
14. [Beautiful JavaScript: Easily Create Chainable (Cascading) Methods for Expressiveness | JavaScript is Sexy](http://javascriptissexy.com/beautiful-javascript-easily-create-chainable-cascading-methods-for-expressiveness/)
15. [How to Learn JavaScript Properly - Ray Sinlao](http://www.raysinlao.com/how-to-learn-javascript-properly/)
16. [JavaScript对象详解 - javascript - 开发者第2317377个问答](http://www.w3c.com.cn/javascript%e5%af%b9%e8%b1%a1%e8%af%a6%e8%a7%a3)
17. [JavaScript对象详解 - javascript教程 - 开发者](http://www.kaifazhe.com/javascript_school/370545.html)
18. [如何正确学习JavaScript | 应酷爱网页设计](http://blog.enqoo.com/blog/2014/10/23/how-to-learn-javascript/)
19. [sexy.com | JavaScript is Sexy](http://javascriptissexy.com/sexy-com/)
20. [如何正确学习JavaScript | 产品汪－技术百科](http://pmwu.cn/tech/?p=52)
21. [「译」如何正确学习JavaScript – 千月殿 -](http://www.lyseky.com/15.html)
22. [如何正确学习JavaScript - javascript - 嗅探实事](http://ttour.sinaapp.com/posts/7843/%e5%a6%82%e4%bd%95%e6%ad%a3%e7%a1%ae%e5%ad%a6%e4%b9%a0javascript.html)
23. [如何正确学习Javascript | 17khba](http://www.17khba.me/?p=21)
24. [[译] 如何恰当地学习 JavaScript — 好JSER](http://hao.jser.com/archive/4016/)
25. [Askerov Javid | Web development](http://askerovblog.com/javascript/javascript-obekty-v-detalyax/)
26. [javascript学习路线 | w3croad](http://www.w3croad.com/85)
27. [JavaScript Concepts we should know well |](https://mapatech.wordpress.com/2015/05/15/javascript-concepts-we-should-know-well/)
28. [Program Outline | Tilly Codes](http://www.tillycodes.com/?p=44)
29. [「译」如何正确学习JavaScript - YelloWish](http://www.cnfar.cn/2015/06/18/study-javascript/)
30. [「译」如何正确学习JavaScript | 大前端](http://www.daqianduan.net/index.php/2015/06/25/sjt/)
31. [jQuery & other updates | Alexa Weidinger](http://blog.alexaweidinger.com/2015/07/16/jquery-other-updates/)
32. [Javascript – Objects | Majesty](https://nicole929chan.wordpress.com/2015/08/26/javascript-objects/)
33. [Learning MEAN Stack | Mandal's Musings](https://phisymmetry.wordpress.com/2014/11/24/learning-mean-stack/)
34. [如何正确学习JavaScript | 前端控](http://www.qianduank.com/2015/10/20/jsxuexi1020/)
35. [Code Craft – Embedding C++: Classes | Hackaday](http://hackaday.com/2015/11/06/code-craft-embedding-c-classes/)
36. [JavaScript | Pearltrees](http://www.pearltrees.com/vcardins/javascript/id15126214/item162401234)
37. [[퍼온글] 자바스크립트 배우기 - hodoogwaja](http://hodoogwaja.com/%ed%8d%bc%ec%98%a8%ea%b8%80-%ec%9e%90%eb%b0%94%ec%8a%a4%ed%81%ac%eb%a6%bd%ed%8a%b8-%ec%a0%9c%eb%8c%80%eb%a1%9c-%eb%b0%b0%ec%9a%b0%ea%b8%b0/)
38. [JavaScript Objects Overview – Tutorialspoint – sous forme d'optimisation avec la forme en ligne builder](https://wformtitanfr.wordpress.com/2016/10/05/javascript-objects-overview-tutorialspoint/)
39. [JavaScript Objects Overview – Tutorialspoint – singlepointofoptimization](https://singlepointofoptimization.wordpress.com/2016/10/06/javascript-objects-overview-tutorialspoint/)
40. [Object Expressions and Declarations – Kotlin Programming Language | launchformsinhighstyle](https://launchformsinhighstyle.wordpress.com/2016/10/06/object-expressions-and-declarations-kotlin-programming-language/)