45 Useful JavaScript Tips, Tricks a Practices

saad.mousliki

1 year ago

113 Comments

Featured, JavaScript, Web

571 1

1 126

By Saad Mousliki

As you know, JavaScript is the number one programming language in the world of mobile hybrid apps (like PhoneGap or Appcelerator), of the server side (like N has many other implementations. It's also the starting point for many new develo programming, as it can be used to display a simple alert in the web browser but (using nodebot, or nodruino). The developers who master JavaScript and write

code have become the most sought after in the job market.

In this article, I'll share a set of JavaScript tips, tricks and best practices that sho JavaScript developers regardless of their browser/engine or the SSJS (Server S interpreter.

Note that the code snippets in this article have been tested in the latest Google uses the V8 JavaScript Engine (V8 3.20.17.15).

1 – Don't forget `var` keyword when assigning a variable's value for the fi

Assignment to an undeclared variable automatically results in a global variable b global variables.

2 – use `===` instead of `==`

The `==` (or `!=`) operator performs an automatic type conversion if needed. Th will not perform any conversion. It compares the value and the type, which coul than `==`.

```
[10] === 10    // is false
[10]  == 10    // is true
'10' == 10     // is true
'10' === 10    // is false
 []   == 0     // is true
 [] ===  0     // is false
 '' == false   // is true but true == "a" is false
 '' ===   false // is false
```

3 – `undefined`, `null`, `0`, `false`, `NaN`, `''` (empty string) are all falsy.

4 – Use Semicolons for line termination

The use of semi-colons for line termination is a good practice. You won't be war because in most cases it will be inserted by the JavaScript parser. For more det use semi-colons, take a look to this artice: http://davidwalsh.name/javascript-ser

5 – Create an object constructor

```
function Person(firstName, lastName){
    this.firstName =  firstName;
    this.lastName = lastName;
}

var Saad = new Person("Saad", "Mousliki");
```

6 – Be careful when using `typeof`, `instanceof` and `constructor`.

- *typeof* : a JavaScript unary operator used to return a string that represents t

variable, don't forget that `typeof null` will return "object", and for the maj (Array, Date, and others) will return also "object".

- *constructor* : is a property of the internal prototype property, which could be
- *instanceof* : is another JavaScript operator that check in all the prototypes ch returns true if it's found and false if not.

```
var arr = ["a", "b", "c"];
typeof arr;   // return "object"
arr  instanceof Array // true
arr.constructor();  //[]
```

7 – Create a Self-calling Function

This is often called a Self-Invoked Anonymous Function or Immediately Invoked (IIFE). It is a function that executes automatically when you create it, and has th

```
(function(){
    // some private code that will be executed automatically
})();
(function(a,b){
    var result = a+b;
    return result;
})(10,20)
```

8 – Get a random item from an array

```
var items = [12, 548 , 'a' , 2 , 5478 , 'foo' , 8852, , 'Doe' , 2145 , 1

var  randomItem = items[Math.floor(Math.random() * items.length)];
```

9 – Get a random number in a specific range

This code snippet can be useful when trying to generate fake data for testing pu between min and max.

```
var x = Math.floor(Math.random() * (max - min + 1)) + min;
```

10 – Generate an array of numbers with numbers from 0 to max

```
var numbersArray = [] , max = 100;

for( var i=1; numbersArray.push(i++) < max;);  // numbers = [1,2,3 ... 1
```

11 – Generate a random set of alphanumeric characters

```
function generateRandomAlphaNum(len) {
    var rdmString = "";
    for( ; rdmString.length < len; rdmString  += Math.random().toString
    return  rdmString.substr(0, len);

}
```

12 – Shuffle an array of numbers

```
var numbers = [5, 458 , 120 , -215 , 228 , 400 , 122205, -85411];
numbers = numbers.sort(function(){ return Math.random() - 0.5});
/* the array numbers will be equal for example to [120, 5, 228, -215, 40
]  */
```

A better option could be to implement a random sort order by code (e.g. : Fisher using the native sort JavaScript function. For more details take a look to this disc

13 – A string trim function

The classic trim function of Java, C#, PHP and many other language that remov string doesn't exist in JavaScript, so we could add it to the `String` object.

```
String.prototype.trim = function(){return this.replace(/^\s+|\s+$/g, "")
```

A native implementation of the trim() function is available in the recent JavaScrip

14 – Append an array to another array

```
var array1 = [12 , "foo" , {name "Joe"} , -2458];

var array2 = ["Doe" , 555 , 100];
Array.prototype.push.apply(array1, array2);
/* array1 will be equal to  [12 , "foo" , {name "Joe"} , -2458 , "Doe" ,
```

15 – Transform the `arguments` object into an array

```
var argArray = Array.prototype.slice.call(arguments);
```

16 – Verify that a given argument is a number

```
function isNumber(n){
    return !isNaN(parseFloat(n)) && isFinite(n);
}
```

17 – Verify that a given argument is an array

```
function isArray(obj){
    return Object.prototype.toString.call(obj) === '[object Array]' ;
}
```

Note that if the toString() method is overridden, you will not get the expected res

Or use…

```
Array.isArray(obj); // its a new Array method
```

You could also use `instanceof` if you are not working with multiple frames. Ho contexts, you will get a wrong result.

```
var myFrame = document.createElement('iframe');
document.body.appendChild(myFrame);

var myArray = window.frames[window.frames.length-1].Array;
var arr = new myArray(a,b,10); // [a,b,10]

// instanceof will not work correctly, myArray loses his constructor
// constructor is not shared between frames
arr instanceof Array; // false
```

18 – Get the max or the min in an array of numbers

```
var  numbers = [5, 458 , 120 , -215 , 228 , 400 , 122205, -85411];
var maxInNumbers = Math.max.apply(Math, numbers);
var minInNumbers = Math.min.apply(Math, numbers);
```

19 – Empty an array

```
var myArray = [12 , 222 , 1000 ];
myArray.length = 0; // myArray will be equal to [].
```

20 – Don't use delete to remove an item from array

Use `splice` instead of using `delete` to delete an item from an array. Using `d` with `undefined` instead of the removing it from the array.

Instead of...

```
var items = [12, 548 ,'a' , 2 , 5478 , 'foo' , 8852, , 'Doe' ,2154 , 11
items.length; // return 11
delete items[3]; // return true
items.length; // return 11
/* items will be equal to [12, 548, "a", undefined × 1, 5478, "foo", 88
", 2154,       119]   */
```

Use...

```
var items = [12, 548 ,'a' , 2 , 5478 , 'foo' , 8852, , 'Doe' ,2154 , 11
items.length; // return 11
items.splice(3,1) ;
items.length; // return 10
/* items will be equal to [12, 548, "a", 5478, "foo", 8852, undefined ×
119]   */
```

The delete method should be used to delete an object property.

21 – Truncate an array using length

Like the previous example of emptying an array, we truncate it using the `lengt`

```
var myArray = [12 , 222 , 1000 , 124 , 98 , 10 ];
```

```
myArray.length = 4; // myArray will be equal to [12 , 222 , 1000 , 124].
```

As a bonus, if you set the array length to a higher value, the length will be chang added with `undefined` as a value. The array length is not a read only property

```
myArray.length = 10; // the new array length is 10
myArray[myArray.length - 1] ; // undefined
```

22 – Use logical AND/ OR for conditions

```
var foo = 10;
foo == 10 && doSomething(); // is the same thing as if (foo == 10) doSor
foo == 5 || doSomething(); // is the same thing as if (foo != 5) doSomet
```

The logical OR could also be used to set a default value for function argument.

```
function doSomething(arg1){
    arg1 = arg1 || 10; // arg1 will have 10 as a default value if it's r
}
```

23 – Use the map() function method to loop through an array's items

```
var squares = [1,2,3,4].map(function (val) {
    return val * val;
});
// squares will be equal to [1, 4, 9, 16]
```

24 – Rounding number to N decimal place

```
var num =2.443242342;
num = num.toFixed(4);  // num will be equal to 2.4432
```

NOTE : the `toFixed()` function returns a string and not a number.

25 – Floating point problems

```
0.1 + 0.2 === 0.3 // is false
9007199254740992 + 1 // is equal to 9007199254740992
9007199254740992 + 2 // is equal to 9007199254740994
```

Why does this happen? 0.1 +0.2 is equal to 0.30000000000000004. What you r JavaScript numbers are floating points represented internally in 64 bit binary ac standard. For more explanation, take a look to this blog post.

You can use `toFixed()` and `toPrecision()` to resolve this problem.

26 – Check the properties of an object when using a for-in loop

This code snippet could be useful in order to avoid iterating through the properti prototype.

```
for (var name in object) {
    if (object.hasOwnProperty(name)) {
        // do something with name
    }
}
```

27 – Comma operator

```
var a = 0;
var b = ( a++, 99 );
console.log(a);  // a will be equal to 1
console.log(b);  // b is equal to 99
```

28 – Cache variables that need calculation or querying

In the case of a jQuery selector, we could cache the DOM element.

```
var navright = document.querySelector('#right');
var navleft = document.querySelector('#left');
var navup = document.querySelector('#up');
var navdown = document.querySelector('#down');
```

29 – Verify the argument before passing it to `isFinite()`

```
isFinite(0/0) ; // false
isFinite("foo"); // false
isFinite("10"); // true
isFinite(10);   // true
isFinite(undefined);  // false
isFinite();   // false
isFinite(null);  // true  !!!
```

30 – Avoid negative indexes in arrays

```
var numbersArray = [1,2,3,4,5];
var from = numbersArray.indexOf("foo") ;  // from is equal to -1
numbersArray.splice(from,2);    // will return [5]
```

Make sure that the arguments passed to `splice` are not negative.

31 – Serialization and deserialization (working with JSON)

```
var person = {name :'Saad', age : 26, department : {ID : 15, name : "R&I
var stringFromPerson = JSON.stringify(person);
/* stringFromPerson is equal to "{"name":"Saad","age":26,"department":{'
   */
var personFromString = JSON.parse(stringFromPerson);
/* personFromString is equal to person object  */
```

32 – Avoid the use of `eval()` or the `Function` constructor

Use of `eval` or the `Function` constructor are expensive operations as each ti
engine must convert source code to executable code.

```
var func1 = new Function(functionCode);
var func2 = eval(functionCode);
```

33 – Avoid using `with()` (The good part)

Using `with()` inserts a variable at the global scope. Thus, if another variable h
could cause confusion and overwrite the value.

34 – Avoid using for-in loop for arrays

Instead of using…

```
var sum = 0;
for (var i in arrayNumbers) {
    sum += arrayNumbers[i];
}
```

…it's better to use…

```
var sum = 0;
for (var i = 0, len = arrayNumbers.length; i < len; i++) {
    sum += arrayNumbers[i];
}
```

As a bonus, the instantiation of `i` and `len` is executed once because it's in the
loop. Thsi is faster than using…

```
for (var i = 0; i < arrayNumbers.length; i++)
```

Why? The length of the array `arrayNumbers` is recalculated every time the loo

NOTE : the issue of recalculating the length in each iteration was fixed in the lat

35 – Pass functions, not strings, to `setTimeout()` and `setInterval()`

If you pass a string into `setTimeout()` or `setInterval()`, the string will be e
as with `eval`, which is slow. Instead of using…

```
setInterval('doSomethingPeriodically()', 1000);
setTimeout('doSomethingAfterFiveSeconds()', 5000);
```

…use…

```
setInterval(doSomethingPeriodically, 1000);
setTimeout(doSomethingAfterFiveSeconds, 5000);
```

36 – Use a switch/case statement instead of a series of if/else

Using switch/case is faster when there are more than 2 cases, and it is more ele
code). Avoid using it when you have more than 10 cases.

37 – Use switch/case statement with numeric ranges

Using a switch/case statement with numeric ranges is possible with this trick.

```
function getCategory(age) {
    var category = "";
    switch (true) {
        case isNaN(age):
            category = "not an age";
            break;
        case (age >= 50):
            category = "Old";
            break;
        case (age <= 20):
            category = "Baby";
            break;
        default:
            category = "Young";
            break;
    };
    return category;
}
getCategory(5);  // will return "Baby"
```

38 – Create an object whose prototype is a given object

It's possible to write a function that creates an object whose prototype is the give

```
function clone(object) {
    function OneShotConstructor(){};
    OneShotConstructor.prototype= object;
    return new OneShotConstructor();
}
clone(Array).prototype ;  // []
```

39 – An HTML escaper function

```
function escapeHTML(text) {
    var replacements= {"<": "&lt;", ">": "&gt;","&": "&amp;", "\"": "&q

    return text.replace(/[<>&"]/g, function(character) {
        return replacements[character];
    });
}
```

40 – Avoid using try-catch-finally inside a loop

The try-catch-finally construct creates a new variable in the current scope at run clause is executed where the caught exception object is assigned to a variable.

Instead of using…

```
var object = ['foo', 'bar'], i;
for (i = 0, len = object.length; i <len; i++) {
    try {
        // do something that throws an exception
    }
    catch (e) {
        // handle exception
    }
}
```

…use…

```
var object = ['foo', 'bar'], i;
try {
    for (i = 0, len = object.length; i <len; i++) {
        // do something that throws an exception
    }
}
catch (e) {
    // handle exception
}
```

41 – Set timeouts to `XMLHttpRequests`

You could abort the connection if an XHR takes a long time (for example, due to using `setTimeout()` with the XHR call.

```
var xhr = new XMLHttpRequest ();
xhr.onreadystatechange = function () {
    if (this.readyState == 4) {
        clearTimeout(timeout);
        // do something with response data
    }
}
var timeout = setTimeout( function () {
    xhr.abort(); // call error callback
}, 60*1000 /* timeout after a minute */ );
xhr.open('GET', url, true);

xhr.send();
```

As a bonus, you should generally avoid synchronous XHR calls completely.

42 – Deal with WebSocket timeout

Generally when a WebSocket connection is established, a server could time out seconds of inactivity. The firewall could also time out the connection after a peri

To deal with the timeout issue you could send an empty message to the server add these two functions to your code: one to keep alive the connection and the keep alive. Using this trick, you'll control the timeout.

Add a `timerID`...

```
var timerID = 0;
function keepAlive() {
    var timeout = 15000;
    if (webSocket.readyState == webSocket.OPEN) {
        webSocket.send('');
    }
    timerId = setTimeout(keepAlive, timeout);
}
function cancelKeepAlive() {
    if (timerId) {
        cancelTimeout(timerId);
    }
}
```

The `keepAlive()` function should be added at the end of the `onOpen()` meth connection and the `cancelKeepAlive()` at the end of the `onClose()` metho

43 – Keep in mind that primitive operations can be faster than function ca

For example, instead of using…

```
var min = Math.min(a,b);
A.push(v);
```

…use…

```
var min = a < b ? a : b;
A[A.length] = v;
```

44 – Don't forget to use a code beautifier when coding. Use JSLint and mi example) before going live.

45 – JavaScript is awesome: Best Resources To Learn JavaScript

- Code Academy JavaScript tracks: http://www.codecademy.com/tracks/javasc
- Eloquent JavaScript by Marjin Haverbeke: http://eloquentjavascript.net/
- Advanced JavaScript by John Resig: http://ejohn.org/apps/learn/

Conclusion

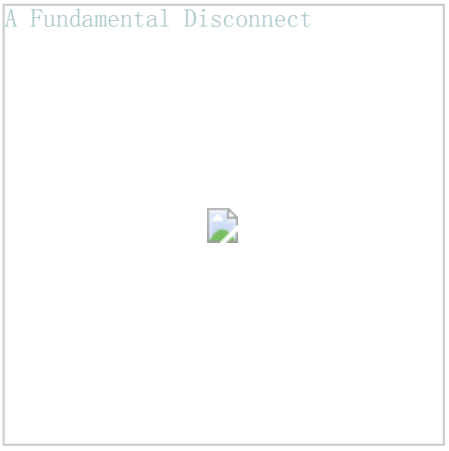
I know that there are many other tips, tricks and best practices, so if you have a
have any feedback or corrections to the ones that I have shared, please adda c

References

In this article I have used my own code snippets. Some of the snippets are inspi
and forums:

- JavaScript Performance Best Practices (CC)
- Google Code JavaScript tips
- StackOverFlow tips and tricks
- TimeOut for XHR

Related



A Fundamental Disconnect

In "Featured"

Getting Started with Modern Web Development

Henry Thompson shares some beginner resources that had an impact on his journey to learning to become a better Modern Web developer. By Henry Thompson

In "Featured"

Best of
- Week

As far a
seemed
everythi
cutting e
CSS to

In "Best

Best of JavaScript, HTML & CSS - Week of December 9, 2013

Building a P
Game witl

Comments: 98

Dimitri Nicolas

1 year ago

23 – Use the map() functio* method […]

Reply

Saad

1 year ago

Thanks Dimitri, We'll update it ASAP

Reply

William

1 year ago

#8 the array has a syntax error

// ;

Reply

Saad

1 year ago

Thanks William, We'll update it ASAP

Reply

Naveen Bhat

1 year ago

Nice article, good set of tips.

Reply

Thomas

1 year ago

Your tipps are very suggestive, without explaining the cases wh suggestions will do some complete different flow, for example t explain wha's the difference when you 'forget' the variable decl useless and somehow dangerous.

Same as with tip 40: The two variants you mention have a com one which you say to avoid executes the catch every loop inter the loop when the error occours. It depends on what's your tar use.

And last but not least you should recognize your own tips in yo don't do what you insist in tip 2 (use === instead of ==) This ap of course.

Reply

Saad

1 year ago

Thanks Thomas, I'm a developer and I know that when a deve snippets that is verified/tested and approved, he will not spend explanation and test it again, he need some things which could quickly in his code.

Regards

Reply

Murthy

1 year ago

Saad, I'm a developer too and have moved to web from wi recently. Sorry to say but I agree with Thomas. I understan effort in writing this article but would definitely appreciate if would rather understand the concept before "copy/pasting" a much more elaborate article. Thanks

Reply

Saad

1 year ago

Thanks Murthy, I think that I'll update the article with some

other articles too, hope that I find time for that …

Reply

Richard Waddell

1 year ago

Saad, Thanks for the great tips. I actually appreciate the br
work out the rest for myself if I need to.

On #40 I'd say the advice should be that exception handlin
should be used only for exception handling, not error hand
time I would use exception handling in an inner loop is in s
process where you don't want some unanticipated error in
entire run. Once an exception has been logged however, y
code (obviously) or code to anticipate what is now a known
instead of throwing an exception. In my experience some c
concept confused and think that coding for exceptions is ex
only expensive if it actually has to handle an exception.

Reply

Fred

13 days ago

Saad, I second your motion that these snippets don't need
elaborating on appropriate use cases. Afterall, your title sta

-Great job! And Thanks!!

Reply

SyntaxError

1 year ago

Since there are no strict comparison operators in JavaScript, 3

Reply

duq

1 year ago

There *are* strict comparisons in JS, see #2.

Reply

Javier

1 year ago

Sounds like T Maul

Reply

gotofritz

1 year ago

What on earth is the benefit of #27 – Comma operator??

Reply

Saad

1 year ago

IMHO it could be considered as a trick that should be known by you could see it somewhere and don't understand it

Reply

gotofritz

1 year ago

…errrr… a trick to achieve what?

in that case you should include all the golfing tricks too

I'm sorry, but to me that's utterly pointless

Reply

atmin

1 year ago

Check https://javascriptweblog.wordpress.com/2011/04/04/ operator/ for use cases.

Example:

if (expression) { doSomething(); return something; } else { return somethingElse; }

vs

return expression ? (doSomething(), something) : (doSome somethingElse);

Reply

Saad

1 year ago

Thanks atmin for this illustration

Reply

VeX

1 year ago

Just as a reference, doesn't #22 implement #2 either.

Reply

Saad

1 year ago

The second best practice will be updated by a clear explanatio inconvenience

Reply

Lazurowy

1 year ago

#43 I think speed of
A.push(v);
and
A[A.length] = v;
depend on browser

http://jsperf.com/push-vs-length-test

Reply

Saad

1 year ago

I have seen the performance tests, an I have noticed that wher
push, it has about twice op/s, but when push gain it is not more

Thanks

Reply

Avron Polakow

1 year ago

(See: http://www.avronp.com/nsn/nsn_stringvb/rsrcs/js/foundat
In addition to Trim() check out all the vb functions (and more) b
****

functions – Trim(mystring);
methods – mystring.trim();
eg:
// .trim | Trim() | trims string on left and right
// .ltrim | LTrim() | left trim string
// .rtrim | Rtrim() | right trim string
// .left | Left() | Returns left part of string
// .right | Right() | Returns right part of string
// .nolf | Nolf() | removes linefeeds (ASCII 10)
// .nocr | Nocr() | removes carriage returns (ASCII 13)
// .nocrlf | Nocrlf() | removes carriagereturns + linefeeds (ASCII
// .nolfcr | Nolfcr() | removes linefeeds + carriage returns (ASCI
// .nosp | Nosp() | removes spaces
// .nosquotes | Nosquotes() | removes single quotes
// .crbr | Crbr() | replaces carriage returns
// .brcr | Brcr() | carriage returns replaces
// .lpad | Lpad() | left pads a string with a fill string to a count
// .rpad | Rpad() | right pads a string with a fill string to a count
// .flat | Flat() | on each line removes leading and trailing space
// .degremlin | Degremlin() | removes gremlins from a string

// | | Default gremlins: Some Non alphanumerics
// | | Optional: Parameter can contain list of gremlins
// .upperhtml | Upperhtml() | Convert HTML tags to uppercase
// .lowerhtml | Lowerhtml() | Convert HTML tags to lowercase
// .htmlattribs | Htmlattribs()| FORMAT HTML tags (quotes/case
// .format | Format() | formats a string to a pattern | format can
any formatting pattern required
// .count | Count() | counts occurrences of a pattern in a string
// .strreverse | Strreverse() | reverses a string
// .isnumber | Isnumber() | checks to see if a string is a number
// .isnotdigit | Notdigit() | checks to see if a string is not a digit

Reply

Johan

1 year ago

Great list. Typo in #20 – description refers to split it should be s
in code example.

Reply

Saad

1 year ago

Thanks Johan :D, yes it was my mistake, we will fixe it ASAP

Reply

Sjeiti

1 year ago

Where is requestAnimationFrame? (setInterval is bad practice)

Reply

Saad

1 year ago

As I said in the article introduction, I'm sharing tips that are rela

method will not work for example in server side SSJS, It's relat
not yet implemented by all browsers, http://caniuse.com/#feat=

Reply

SyntaxError

1 year ago

On a related note, the JSON object didn't exist before IE8.
would need a polyfill.

Reply

Kev

1 year ago

who says setInterval() is bad practice? I had major issues (like
for several seconds) with RAF on iOS (6 and 7) and had to aba
YMMV.

Reply

Matthew Wilcoxson

11 months ago

While SetInterval may be bad practice, requestAnimationFram
when updating something visible on screen (i.e. animating) but
purposes.

Use SetTimer repeatedly instead, as is done in #42 (although i
perfect timer isn't essential and setInterval could work!)

Reply

Christian

1 year ago

Hi Saad.

Great "little" compilation of tips & tricks.

I've stumbled over a few minor typo's etc. And I just saw some

been commented on while I wrote. Passing them on anyway

#14 – Append an array to another array.
Using push this way is actually limited by max number of argun
Chrome).
So for huge arrays this trick will throw an exception.

#20 – Don't use delete to remove an item from array
A minor typo – "Use split instead of …" (split => splice)

#22 – Use logical AND/ OR for conditions

"The logical AND could …" => "The logical OR could …"

Also the function has fallen victim to some automatic spell chec
and uppercase 'Arg1′)

#26 – Check the properties of an object when using a for-in loc
"if (object.hasOwnProperty(name))" is necessary only on objec
you want to disregard prototype properties.
Object literals can safely be run though using for..in.

#27 – Comma operator
I agree with your note that developers should know how to use
Actually, many use it in var lists and they should have explicit k
great difference between your example and the var list use.

#30 – Avoid negative indexes in arrays
"… passed to indexOf are …" => "… passed to splice are …"

An interresting use of out-of-bounds indeces is: "for (var i=0; m
element lookup outside the range will return "undefined" (of co
array to contain no other falsy items

#33 – Avoid using with() (The good part)
A good example would be

```
function f(object) {
var x = 2;
with (object)
++x;
return x;
}
```

Calling f for any object not having a 'x' property will make f alwa

Calling f({x:7}) (an object with property ‘x’ being numerical – or
8.
Calling f with any object having a non-numerical ‘x’ will return 2
Unless applied with care using “with” is best suited for code ob
😀

#34 – Avoid using for-in loop for arrays
Last line: “Why? The length of the array arrayNumbers is recal
loop iterates.”
I doubt length is recalculated. However, the object property loo
than local variable lookup.
Having a sufficiently good optimizing interpreter no discernable

Reply

Saad

1 year ago

Thanks Christian for the notes, many benchmarking tests were
demonstrate those tips, and as you may know JavaScript imple
(JavaScript Runtime Engines) : the browsers (client side), and
:Server-Side JavaScript) implementation, it’s why the results c
depending in the implementation, in the list, I have chosen the
the max of browsers and SSJS Server, I add also that many tip
#42 …etc.) will have a native implementation in the latest Java
ES7 (work in progress)
it’s not a Bible 😉

Reply

Mike McNally

1 year ago

#12 is **not** a good way to shuffle an array. In fact it really ha
behavior at all. Use a Fisher-Yates shuffle.

Reply

Michael Martin-Smucker

1 year ago

Indeed, Array.prototype.sort isn't the way to go. If anyone is int
(or if you want to see a good Javascript implementation of Fish
answers to this StackOverflow question: http://stackoverflow.co

Reply

Don Burks

1 year ago

Typo in #14, you need a colon between name and "Joe" in: var
{name: "Joe"} , -2458];

Otherwise, fantastic list.

Reply

Debjeet

1 year ago

#11 has a reference error

Reply

Saad

1 year ago

Yes, I have just seen it.

Thanks Debjeet

Reply

Tamm Kwun

1 year ago

Great article!

Reply

raj kimor

1 year ago

Very useful list, thanks !

Reply

Aamir Afridi

1 year ago

For 14, use concat
var one = [“Cecilie”, “Lone”];
var two = [“Emil”, “Tobias”, “Linus”];
var both = one.concat(two);

Reply

dotnetCarpenter

1 year ago

+1

Reply

raj kimor

1 year ago

Hi Aamir, the tip is about appending and not concatenating, the
be a good option too

thanks

Reply

P

1 year ago

Correction for #11:

var rdmstring = “”;
Should be

var rdmString = “”; // because currently when you call rdmStrin
error due to being undefined

Reply

shaun

1 year ago

#43 is missing the colon in your conditional operator. Not sure
pointed this out. Thanks for the article!!

Also a good read on flippinawesome: http://flippinawesome.org
javascript-native-array-functions/

Reply

Andrei

1 year ago

For 24 see http://stackoverflow.com/questions/10015027/javas
rounding. There is no easy answer.

Reply

Sebastian Otto

1 year ago

Example 10 will not contain 0 in the numbersArray

Reply

Joel

1 year ago

#10 Would return [1…99] instead of the indicated [0,1,2,3 … 1
————-

```
var numbersArray = [] , max = 100;
for( var i=1; numbersArray.push(i++) < max;); // numbers = [0,1
```

Reply

Saad

1 year ago

It returns [1,2,3 … 100] and not to 99,

it will be fixed ASAP

Thanks

Reply

Kir6

11 months ago

Actually, this is a trick and not a good practice.
Reading too fast, one would think it would return [1 … 99] b
because the push method is an instruction that is irremedia
the result of the conditional for-loop parameter is.
100 is a value that is pushed in-extremis. After that, the tes
already too late.
You are not tricking the interpreter, but the reader. And tha
The conditional segment should stay as clear as possible c

I like your blog article. But it lacks of reminders about “gooc
practices”.

In good practice, devs should avoid the ternary operator as
are good practice in limited use at the beginning of method
and such. Not in the core implementations. And should not
logic. Use the good old if (){} else {}. And choose to prefer
advices.

If you really need other devs to be unable to read you, or t
an obfuscator !
Ternary branching are also hard to debug with breakpoints

A[A.length] = v; // is not a good practice either. But it depen
structure or your browser or your real array end. Some wo
objects anyway. And not caring about the indices. Never as
never be trimmed down or spliced. You may one day be su
is shorter than expected.

Keep up the good work in compiling those tricks anyway. T
sheets though.

Reply

Robin

1 year ago

Technically, #24 returns “2.4432”, it is a string, not a number. S
another number to n after the toFixed, it would simply add it to
concatenation.

```
var n = 2.414233;
n = n.toFixed(4);
console.log(typeof n); // string
```

Reply

Ankit Garg

1 year ago

Nice article man !! Thanks.

Reply

Bill

1 year ago

What happened to tip #3?

Reply

remotesynth

1 year ago

Nice catch. My fault. Fixed.

Reply

Robin

1 year ago

Nice article. Flipping Awesome.

Reply

Kevin

1 year ago

Crockford's "The Good Parts" recommends against using the r
(under the "Bad Parts" – "new" and "The Constructor Invocatio
also against the ++ increment operator in tip #34 (under the "B
Stefanov's "JavaScript Patterns" book suggests using the += in
of ++ (e.g., i += 1 instead of i++) in the "Essentials – for Loops"
Parts" gives several recommended alternates to the new opera
Inheritance". Thanks for all of the tips, it's always nice to review
reinforcement and finding new gems!

Reply

JasonHuang

1 year ago

nice article ! but with some typo witch will easily mislead new fis
for remove a item from a array with slice or delete . typoed the

Reply

Swapneel

1 year ago

These are very handy tips. Thanks for sharing!

Reply

Brennan Young

1 year ago

#4 – Not very persuasive about semicolons. It's "good practice" bite you if you write a return value on the next line after the key

function greeting () {
return
"hello world"
}

This function will actually return undefined because a semicolo return. AFAIK this is the main reason for enforcing the semicol more effective rule would be "begin your return value on the sa return". If people only did that, they could be as sloppy as they

Your advice to use jslint is admirable. I wish you had used it on you didn't use the single var pattern in the longer snippets, and of errors to declare vars anywhere but at the top of the function I am sure you know this, as will many readers.

Otherwise, thanks for a good article. Some useful tricks.

Reply

Konrad Dzwinel

1 year ago

Bugfixes/Improvements:
– It's hard (and silly) to remember the tip without understanding please add missing explanations.
– syntax highlighting!
– #6 – this is a good example of a tip that misses proper explan
– #7 `})(10,20)` – missing semicolon (see point #4)
– #13 extending build-in objects is controversial
– #14 `{name "Joe"}` -> `{name "Joe"}`
– #16 linking to the author of this code would be nice (http://stackoverflow.com/questions/18082/validate-numbers-in-
– #19 and #21 are the same thing
– #22 `Function doSomething(arg1){ ` -> `function doSomethin 10;` -> `arg1 = arg1 || 10;`
– #22 It's worth mentioning that last example is not entirely true 0 or false it will be replaced with default value.

– #27 interesting, but not useful
– #30 “Make sure that the arguments passed to indexOf are nc
sure that the arguments passed to splice are not negative.”? I’
you meant here.
– #34 “The length of the array arrayNumbers is recalculated ev
iterates.” – it’s no longer true for modern JS engines
– #35 `setTimeOut` -> `setTimeout`
– #36 “Avoid using it when you have more than 10 cases.” – wi
– #39 link to the source: https://gist.github.com/leommoore/471
– #41 “synchronous Ajax” -> “synchronous XHR”. A in Ajax star
– #42 `var min = a **`var min = a < b ? a : b; `**
– #45 this resource is locked and outdated (it's from 2011)

Reply

Saad

1 year ago

Thanks Konrad for this good review, As I said the list is open to
and improvments, I’ll check this list and others, after that updat

For #45, have you could share a list of good resources if you h

Reply

John

1 year ago

Re: 13 – A String Trim Function: JavaScript does have a string
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/Trim

Reply

Christian

1 year ago

You wrote «slice» instead of «splice». Also JS does have a ‘trim
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/Trim

– otherwise this is a great write-up!

Reply

Bernard

1 year ago

#22 in the first example doSomething() always return somethin variable will take the returned value, even undefined. In the ca won't.

Nice article. You wrote you didnt want explanations. They or lin external explanations would be useful.

Regards

Reply

dotnetCarpenter

1 year ago

#10 should be:
var max = 100, numbersArray = new Array(max); // faster
for(var i=1; numbersArray.push(i++) < max;); // numbers = [0,1

If you use jQuery you could do:
var numbersArray = jQuery.map(new Array(100), function(el, i)
= [0,1,2,3 … 100]

I know for a fact, that the D3 library has changes all its Array in Array([array length]) solely for performance reasons.

Reply

raj kimor

1 year ago

This code :
var max = 100, numbersArray = new Array(max); // faster

will create an array of 100 undefined items, As I said in the top to any frameworks (JQuery, D3 …etc), only native JavaScript c

Thanks for sharing

Reply

Saad

1 year ago

Thanks dotnetCarpenter, I'll update the comment the creat
100, and not from 0.

Reply

dotnetCarpenter

1 year ago

#19 If the array contain references to DOM nodes or titanium p
memory leak in older IE and Appcelerator.
Instead you have to explicitly null the references before emptyi
var myArray = [12 , objReference, 1000];
myArray.forEach(explNull); // forEach is part of ecmaScript5
myArray.length = 0; // myArray will be equal to [].
function explNull(item) {
item = null; // reference is freed for the garbage collector
}
In ecmaScript3 (IE8 and before) you can polyfil it with:
if(!Array.prototype.forEach) {
Array.prototype. forEach = function forEach(fn, context) { // nar
to debug
for(var i = 0, len = this.length; i < len; ++i)
fn.call(context, this[i], i, this);
};
}

Reply

dotnetCarpenter

1 year ago

#43 has a typo. "var min = a < b ? a b;" should be "var min = a

Nice list. I didn't know about the trick in #37. Very nice

Reply

Cody

1 year ago

var z = 1; // ONE
function fn(x, y){ console.log('@args:', x, y, z); return x + y + z;

fn((z && (z++, 4)), 6); // returns 12

var z = 0; // ZERO
function fn(x, y){ console.log('@args:', x, y, z); return x + y + z;

fn((z && (z++, 4)), 6); // returns 6

Kinda neat-o!

Reply

Saad

1 year ago

Cody, Thanks for this snippet

Reply

Yaw

1 year ago

Tip #14 looks good but array1.concat(array2) is the natural wa one in JavaScript

Reply

Yaw

1 year ago

Tip #16

A little tricky, silly implementation I learnt from Lea Verou

function isNumber(n) {
return n === +n;
}

Reply

Grégoire

1 year ago

29 – isFinite(und*i*fined); // false

Reply

Kev

1 year ago

23 – map() should never be an automatic choice for looping ov
overhead on every iteration. Only use map() if you don't care a

Reply

SanSYS

1 year ago

36 – Use a switch/case statement instead of a series of if/else

please see test http://jsperf.com/if-vs-switch-statement
switch more slow than if statement

Reply

SanSYS

1 year ago

look and second test http://jsperf.com/if-vs-switch-statement2

Reply

SanSYS

1 year ago

43 – Keep in mind that primitive operations can be faster than
VanillaJS.

Keep in mind that examples should be tested http://jsperf.com/

Reply

Saad

1 year ago

Thanks I'll check the given jsperf tests

Reply

justin

1 year ago

Great reference for when you need something quick. Thanks f

One typo… You're missing a : on…
var min = a < b ? a : b;
A[A.length] = v;

Reply

Bas Slagter

1 year ago

In tip 34 ' Avoid using for-in loop for arrays' you state that one s
instead of a for-in loop.
Of course this is true and your exampled alternative is indeed a
this one:

for(var i=myArray.length; i–;){
// Do something under the awareness that we are looping back
}

This actually is an even faster way to loop through the array sir
check the value of i against the length of the array. This is beca
(falsy) at one point and the loop will end. Maybe it's a minor pe
it.

Reply

SanSYS

1 year ago

What do you think about this method?

for (var i = myArray.length, item; item = myArray[–i];){
// do something with item
}

Notice: array should not contain values interpreted like false (e
zero…)

Reply

mm

1 year ago

can you say the reason about 37 ?

Reply

MJ

11 months ago

In 42, you have used `cancelTimeout,` which is not a JS functi
probably want is `clearTimeout`

Reply

Saad

11 months ago

Thanks, I just see it, It was my mastake, it'll be updated ASAP

Reply

Stan88

11 months ago

It's awesome man!! Thanks for article!

Reply

Kir6

11 months ago

#3 I would add an exemple.
This is good practice to not be lazy and to write for example: "i
of "if (!val) {}".
As you stated: undefined, 0, "", null, NaN and false, are false.
But you don't want to know if this is false, in that case, you wou

specifically what is.
Is this because this is binary false, because it's null or undefine
So write it.

Of course, "if (!val){}" stays valid and can be used if you know

Reply

Nadeem Jamali

11 months ago

Very nice….

Reply

Giacomo Paita

10 months ago

Hi! I found this article really useful! Can i translate it, mentionio
actual author, and put it in my blog? http://www.paitadesign.co

Reply

Saad

10 months ago

Hi Giacomo,
Ok, I hope that it could be helpful for people who doesn't under

Reply

Giacomo Paita

10 months ago

Thanks! As soon as i can i will start, once i end the post i w

Reply

kumar

9 months ago

very useful list.

Thanks.

Reply

Yogeesh R

7 months ago

Good Tips & Tricks. Waiting for some more . . .

Reply

Dimitar Ivanov

7 months ago

Great list! See few more: http://zinoui.com/blog/javascript-best-

Reply

Nick

4 months ago

KeepAlive should probably have an else statement that calls C
websocket is unavailable

Reply

john

3 months ago

Great list.

P.s Ignore the haters. If they weren't ignorant and/or pedantic t
the list and learn a thing or two.

Reply

Eric Elliott

15 days ago

ES5 shim

5 No. No, really. Don't do it.

http://ericleads.com/2012/09/stop-using-constructor-functions-i
http://ericleads.com/2013/01/javascript-constructor-functions-v
https://medium.com/javascript-scene/the-two-pillars-of-javascri

7 Don't do this with modules anymore. Instead, use node-style
(the latter with ES6 transpiler).

13 Use ES5 shim instead.

14 Use Array.prototype.concat() instead.

15 Can be safely replaced with [].slice.call(arguments). This is
worry about perf. I've tested it heavily. It won't slow you down c
memory.

17 Use ES5 `Array.isArray()` instead. Never use `instance`
that it breaks across execution contexts.

18 Cool trick! =)

19 Cleaner approach: myArray = [];

20 Be careful about mutating arrays. If other parts of the code
could cause bugs. Copy the array if it's feasible.

21 As with 20, you may want to use Array.prototype.slice() and
instead of mutating an existing array.

22 Code readability is much more important than code writeabi
defaults / overrides pattern, it's safer to use named paramaters
`defaults, options)` or `$.extend({}, defaults, optio`

36 Try using an action object, instead. See http://ericleads.com
considered-harmful/

38 This is standard in ES5 as `Object.create()`. Use that, in

39 This is unsafe. Use a security library, instead. See WASP re
https://github.com/mapbox/sanitize-caja, https://github.com/pur
remember to penetration test: IronWASP http://blog.ironwasp.c

43 You're optimizing the wrong things. http://ericleads.com/201
the-wrong-things/

Additional suggestions: The biggest and most common mistake
making is that they often miss two of JavaScript's most importa

OO and functional programming. See “JavaScript Training Suc
https://medium.com/javascript-scene/javascript-training-sucks-

I notice you got a lot of other suggestions. Some of them good
version 2 of this article after you’ve had time to absorb all these

Reply

Sid

12 days ago

I like this article and translate it into Chinese: http://chensd.com
javascript-tips-tricks-and-best-practices.html

Reply

Pingbacks: 15

1. Web Development | Annotary
   1 year ago
2. 45 Useful JavaScript Tips, Tricks and Best Practices
   1 year ago
3. Tweet Parade (no.01 Jan 2014) - Best Articles of Last Week | gonzoblog
   1 year ago
4. 45 Useful JavaScript Tips, Tricks and Best Practices | CompkSoft
   1 year ago
5. Monday Trends – Links Digest 6/1/14 | Monday Trends
   1 year ago
6. Miscellaneous | Annotary
   1 year ago
7. Interesting Web Dev Things 13 Jan 2014 | Joseph Khaw
   1 year ago
8. JavaScript best practices, SVGs, typography | The Treehouse Show Episode
   1 year ago

9. JavaScript best practices, SVGs, typography | The Treehouse Show Episode
   1 year ago

10. Ape Boy Studio | JavaScript best practices, SVGs, typography | The Treehou
    11 months ago

11. make your javascript clean | Keep Learning
    2 months ago

12. This week's JavaScript news, issue 213 | blog1
    14 days ago

13. Useful JavaScript Tips and Tricks 2015 - Codemics: Tech News, Hardware F
    12 days ago

14. JavaScript奇技淫巧45招 | 前端头条
    10 days ago

15. JavaScript奇技淫巧45招 | 一本尘封已久的土册子
    1 day ago

Leave a Reply

Name Email

Type here

POST COMMENT

Notify me of follow-up comments by email.

Notify me of new posts by email.