

JAVASCRIPT

THE BAD PARTS

@felixge

DISCLAIMER

I love JavaScript

But if we could go back in time ...

PARSEINT

```
> parseInt('42')  
42  
> parseInt('07')  
7  
> parseInt('08')  
0  
> parseInt('010')  
8  
> parseInt('08', 10)  
8
```

If the input string begins with "0", radix is eight (octal). This feature is non-standard, and some implementations deliberately do not support it (instead using the radix 10). For this reason always specify a radix when using parseInt.

https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/parseInt

FLOATS - ONLY

```
> 0.1 + 0.1 + 0.1
0.3000000000000004
> Math.pow(2, 53) === (Math.pow(2, 53) + 1)
true
> Math.random() === 1 // 1 in 2^62 cases
true
```

Note that as numbers in JavaScript are IEEE 754 floating point numbers with round-to-nearest-even behavior, these ranges, excluding the one for `Math.random()` itself, aren't exact, and depending on the bounds it's possible in extremely rare cases (on the order of 1 in 2^{62}) to calculate the usually-excluded upper bound.

https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Math/random

THIS

```
> obj = {method: function() {return this;}}
{ method: [Function] }
> obj.method() === obj
true
> method = obj.method
[Function]
> method() === obj
false
> method() === this
true
> obj2 = {method: obj.method}
{ method: [Function] }
> obj2.method() === obj
false
> obj2.method() === obj2
true
```

In general, the object bound to this in the current scope is determined by how the current function was called, it can't be set by assignment during execution, and it can be different each time the function is called. ES5 introduced the bind method to fix a function's this regardless of how it's called.

[https://developer.mozilla.org/en/JavaScript/
Reference/Operators/this](https://developer.mozilla.org/en/JavaScript/Reference/Operators/this)

ARGUMENTS.

```
> method = function(){return arguments;}  
[Function]  
> method('hi')[0]  
'hi'  
> method('hi').length  
1  
> method('a', 'b', 'c').length  
3  
> method('a', 'b', 'c').pop()  
TypeError: Object #<Object> has no method  
'pop'
```

The `arguments` object is not an array. It is similar to an array, but does not have any array properties except `length`. For example, it does not have the `pop` method. However it can be converted to a real array:

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

https://developer.mozilla.org/en/JavaScript/Reference/Functions_and_function_scope/arguments#Description

SETTIMEOUT

```
var start = Date.now();
setTimeout(function() {
    console.log(Date.now() - start);
}, 0);
```

- Firefox 9: 25 - 100 ms
- Chrome 16: 1 - 10ms
- Safari 5: 2 - 8 ms
- Node.js: 0ms

OMG! NODE

INFINITELY.

FASTER!!! 111!!

0
(M 4a58e) ~/Documents/Presentation

You need to restart your computer. Hold down the Power button until it turns off, then press the Power button again.

Redémarrez l'ordinateur. Enfoncez le bouton de démarrage jusqu'à l'extinction, puis appuyez dessus une nouvelle fois.

Debe reiniciar el ordenador. Mantenga pulsado el botón de arranque hasta que se apague y luego vuelva a pulsarlo.

Sie müssen den Computer neu starten. Halten Sie den Ein-/Ausschalter gedrückt bis das Gerät ausgeschaltet ist und drücken Sie ihn dann erneut.

コンピュータの再起動が必要です。電源が切れるまでパワー ボタンを押し続けてから、もう一度パワー ボタンを押します。

Historically browsers implement `setTimeout()` "clamping": successive `setTimeout()` calls with delay smaller than the "minimum delay" limit are forced to the use at least the minimum delay. The minimum delay, `DOM_MIN_TIMEOUT_VALUE`, is 4 ms (stored in a preference in Firefox: `dom.min_timeout_value`), with a `DOM_CLAMP_TIMEOUT_NESTING_LEVEL` of 5ms.

In fact, 4ms is specified by the HTML5 spec and is consistent across browsers released in 2010 and onward. Prior to (Firefox 5.0 / Thunderbird 5.0 / SeaMonkey 2.2) , the minimum timeout value for nested timeouts was 10 ms.

[https://developer.mozilla.org/en/DOM/
window.setTimeout#Minimum_delay_and_timeout_nesting](https://developer.mozilla.org/en/DOM/window.setTimeout#Minimum_delay_and_timeout_nesting)

- <http://ejohn.org/blog/how-javascript-timers-work/>
- <http://hacks.mozilla.org/2011/08/animating-with-javascript-from-setinterval-to-requestanimationframe/>
- <http://paulbakaus.com/2011/10/17/on-accurately-measuring-fps/>

DATE

```
> date = new Date  
Thu, 19 Jan 2012 18:23:42 GMT  
> date.getDay()  
4  
> date.getDate()  
19  
> date.getMonth()  
0
```

The value returned by `getMonth` is an integer between 0 and 11. 0 corresponds to January, 1 to February, and so on.

https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Date/getMonth

"Should array indices start at 0 or 1? My compromise of 0.5 was rejected without, I thought, proper consideration."

-- Stan Kelly-Bootle

"Dear Stan, we would love to hear more of your ideas!"

-- The JavaScript Community

THANKS

<http://wtfjs.com/>