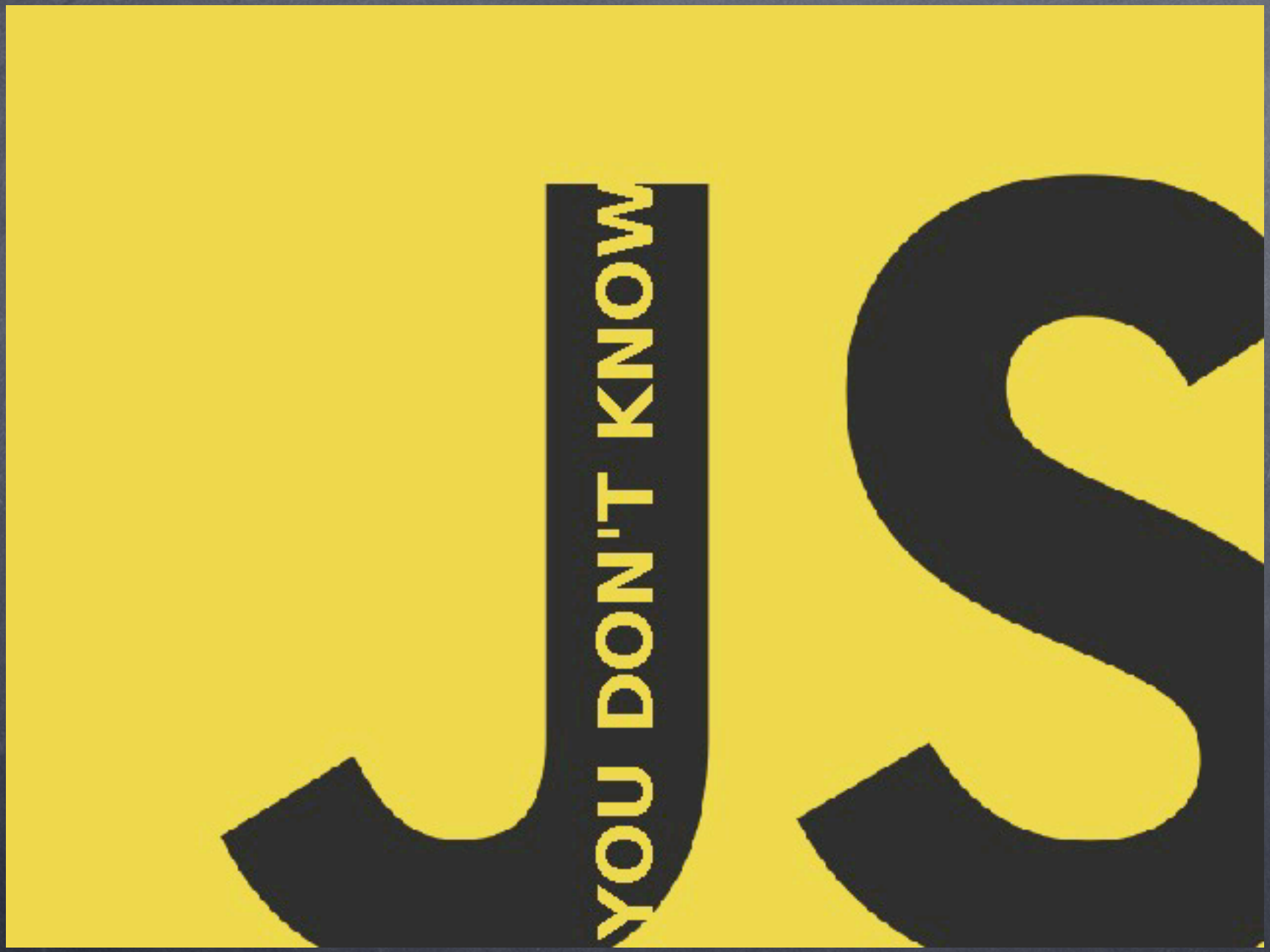
Kyle Simpson
@getify
http://getify.me

- LABjs
- grips
- asynquence

Kyle Simpson
@getify
http://getify.me

http://speakerdeck.com/getify



http://YouDontKnowJS.com

https://developer.mozilla.org/en-US/docs/JavaScript

https://github.com/rwldrn/idiomatic.js

http://www.ecma-international.org/ecma-262/5.1/

https://github.com/tc39/ecma262#current-proposals

Agenda

Values & Types

- Primitive Types
- Natives
- Coercion

- undefined
- string
- number
- boolean
- object
- function
- null

Primitive Types

```
1 typeof foo;              // "undefined"
2 typeof "foo";            // "string"
3 typeof 123;              // "number"
4 typeof true;             // "boolean"
5 typeof { a: 1 };         // "object"
6 typeof function() { alert(e); };    // "function"
```

Primitive Types (literals)

```
2
3 typeof null;     // "object" ... WAT!?
4
```

Primitive Types (literals)

Quiz

```
1 var foo;
2 typeof foo;                    // ???
3
4 var bar = typeof bar;
5 bar;                           // ???
6 typeof bar;                    // ???
7
8 typeof typeof 2;               // ???
```

Quiz

```
1 var foo;
2 typeof foo;                    // "undefined"
3
4 var bar = typeof bar;
5 bar;                           // "undefined"
6 typeof bar;                    // "string"
7
8 typeof typeof 2;               // "string"
```

- NaN ("not a number")
- Infinity, -Infinity
- null
- undefined (void)
- +0, -0

Primitive Types: special values

NaN

```
1 var a = "a" / 2;
2
3 a;                    // NaN
4 typeof a;             // "number"
5 isNaN(a);             // true
6
7 isNaN("foo");         // true! WAT!?
```

Primitive Types: special values

NaN

```
1 if (!Number.isNaN) {
2     Number.isNaN = function(num) {
3         return (
4             typeof num === "number" &&
5             window.isNaN(num)
6         );
7     }
8 }
```

Primitive Types: special values

NaN

```
1 if (!Number.isNan) {
2       Number.isNaN = function(num) {
3           return num !== num;
4       };
5 }
```

Primitive Types: special values

Negative Zero

```
1 var foo = 0 / -3;
2 foo === -0;                  // true ... yay?
3 foo === 0;                   // true ... doh!
4 0 === -0;                    // true ... grr!
5 (0/-3) === (0/3);            // true ... sigh
6
7 foo;                         // 0 ... WAT!?
```

Primitive Types: special values

Negative Zero

```
1 function isNeg0(x) {
2     return x===0 && (1/x)===-Infinity;
3 }
4
5 isNeg0(0 / -3);       // true ... yay!
6 isNeg0(0 / 3);        // false ... yay!
```

ES6: Object.is(..)

```
1 Object.is( "foo", NaN );      // false
2 Object.is( NaN, NaN );        // true
3
4 Object.is( 0, -0 );           // false
5 Object.is( -0, -0 );          // true
```

Primitive Types: special values

Quiz

```
 1 var baz = 2;
 2 typeof baz;                    // ???
 3 var baz;
 4 typeof baz;                    // ???
 5 baz = null;
 6 typeof baz;                    // ???
 7
 8 baz = "baz" * 3;
 9 baz;                           // ???
10 typeof baz;                    // ???
11
12 baz = 1 / 0;
13 baz;                           // ???
14 typeof baz;                    // ???
```

Primitive Types: special values

Quiz

```
 1 var baz = 2;
 2 typeof baz;                    // "number"
 3 var baz;
 4 typeof baz;                    // "number"
 5 baz = null;
 6 typeof baz;                    // "object"
 7
 8 baz = "baz" * 3;
 9 baz;                           // NaN
10 typeof baz;                    // "number"
11
12 baz = 1 / 0;
13 baz;                           // Infinity
14 typeof baz;                    // "number"
```

Primitive Types: special values

- String
- Number
- Boolean
- Function
- Object
- Array
- RegExp
- Date
- Error

Native Types?
Native Objects?

Natives

- String()
- Number()
- Boolean()
- Function()
- Object()
- Array()
- RegExp()
- Date()
- Error()

Native Functions?

Native Constructors

- new String()
- new Number()
- new Boolean()
- new Function()
- new Object()
- new Array()
- new RegExp()
- new Date()
- new Error()

Quiz

```
1  var foo = new String("foo");
2  foo;                                  // ???
3  typeof foo;                           // ???
4  foo instanceof String;                // ???
5  foo instanceof string;                // ???
6
7  foo = String("foo");
8  typeof foo;                           // ???
9
10 foo = new Number(37);
11 typeof foo;                           // ???
```

Natives

Quiz

```
 1 var foo = new String("foo");
 2 foo;                                  // ???
 3 typeof foo;                           // "object"
 4 foo instanceof String;                // true
 5 foo instanceof string;                // false; ref error!
 6
 7 foo = String("foo");
 8 typeof foo;                           // "string"
 9
10 foo = new Number(37);
11 typeof foo;                           // "object"
```

Natives

```
 1 var foo;
 2
 3 foo = new Array(1,2,3); // don't!
 4
 5 foo = [1,2,3];          // do!
 6
 7 foo = new Object();     // don't!
 8 foo.a = 1;
 9 foo.b = 2;
10 foo.c = 3;
11
12 foo = { a:1, b:2, c:3 };// do!
```

Natives: literals

```
1 var foo;
2
3 foo = new RegExp("a*b","g");
4
5 foo = /a*b/g;
6
7 foo = new Date();
8
9 // no date literal!
```

Natives: literals

Coercion

Abstract Operations

Coercion

ToString

Coercion: Abstract Operations

null	"null"
undefined	"undefined"
true	"true"
false	"false"
3.14159	"3.14159"
0	"0"
-0	"0"

Coercion: ToString

ToString: toString()

Coercion: ToString

[]	""
[1,2,3]	"1,2,3"
[null,undefined]	","
[[[],[],[]],[]]	",,,"
[,,,,]	",,,"

Coercion: ToString (Array)

{} "[object Object]"

{a:2} "[object Object]"

Coercion: ToString (Object)

ToNumber

""	0
"0"	0
"-0"	-0
" 009 "	9
"3.14159"	3.14159
"0."	0
".0"	0
"."	NaN
"0xaf"	175

Coercion: ToNumber

false	0
true	1
null	0
undefined	NaN

Coercion: ToNumber

ToNumber (object):
ToPrimitive (number)

Coercion: ToNumber (Array/Object)

ToPrimitive

valueOf()
toString()

Coercion: ToPrimitive

[""]	0
["0"]	0
["-0"]	-0
[null]	0
[undefined]	0
[1,2,3]	NaN
[[[[]]]]	0

Coercion: ToNumber/ToPrimitive (Array)

ToBoolean

Coercion: Abstract Operations

Falsy	Truthy
""	"foo"
0, +0, -0	23
null	{ a:1 }
NaN	[1,3]
false	true
undefined	function(){..}
	...

Coercion: ToBoolean

Explicit

Implicit

Coercion

Explicit: it's obvious from the code that you're doing it.

Explicit Coercion

string ⟷ number

Explicit Coercion

```
1  var foo = "123";
2  var baz = parseInt(foo,10);
3  baz;                                  // 123
4
5  baz = Number(foo);
6  baz;                                  // 123
7
8  baz = +foo;                           // explicit?
9  baz;                                  // 123
10
11 baz = 456;
12 foo = baz.toString();
13 foo;                                  // "456"
14
15 foo = String(baz);
16 foo;                                  // "456"
```

Explicit Coercion

$* \longrightarrow$ boolean

Explicit Coercion

```
1  var foo = "123";
2  var baz = Boolean(foo);
3  baz;                                  // true
4
5  baz = !!foo;
6  baz;                                  // true
7
8  // explicitly implicit!
9  baz = foo ? true : false;
10 baz;                                  // true
```

Explicit Coercion

```
1 var now = +new Date();
2 // now = Date.now(); // ES5 only!
3
4 var foo = "foo";
5 // ~N -> -(N+1)
6 if (~foo.indexOf("f")) {
7     alert("Found it!");
8 }
```

Explicit (or Implicit?) Coercion

Implicit: happens as a side effect of some other operation

Implicit Coercion

Implicit: evil?

```
1  var foo = "123";
2  var baz = foo - 0;
3  baz;                          // 123
4
5  baz = foo - "0";
6  baz;                          // 123
7
8  baz = foo / 1;
9  baz;                          // 123
10
11 baz = 456;
12 foo = baz + "";
13 foo;                          // "456"
14
15 foo = baz - "";
16 foo;                          // 456 ... WAT!?
```

Implicit Coercion

```
1  var foo = "123";
2  if (foo) {                          // yup
3      alert("Sure.");
4  }
5
6  foo = 0;
7  if (foo) {                          // nope
8      alert("Right.");
9  }
10 if (foo == false) {                 // yup
11     alert("Yeah.");
12 }
13
14 var baz = foo || "foo";
15 baz;                                // "foo"
```

Implicit Coercion

```
 1 var foo = "123";
 2 if (foo == true) {                    // nope!
 3     alert("WAT!?");
 4 }
 5
 6 foo = [];
 7 if (foo) {                            // yup
 8     alert("Sure.");
 9 }
10 if (foo == false) {                   // yup!
11     alert("WAT!?");
12 }
```

Implicit Coercion

```
1  "0" == null;              // false
2  "0" == undefined;         // false
3  "0" == false;             // true -- UH OH!
4  "0" == NaN;               // false
5  "0" == 0;                 // true
6  "0" == "";                // false
7
8  false == null;            // false
9  false == undefined;       // false
10 false == NaN;             // false
11 false == 0;               // true -- UH OH!
12 false == "";              // true -- UH OH!
13 false == [];              // true -- UH OH!
14 false == {};              // false
15
16 "" == null;               // false
17 "" == undefined;          // false
18 "" == NaN;                // false
19 "" == 0;                  // true -- UH OH!
20 "" == [];                 // true -- UH OH!
21 "" == {};                 // false
22
23 0 == null;                // false
24 0 == undefined;           // false
25 0 == NaN;                 // false
26 0 == [];                  // true -- UH OH!
27 0 == {};                  // false
```

7/24

17/24

Implicit Coercion

```
1 "0" == false;          // true -- UH OH!
2 false == 0;            // true -- UH OH!
3 false == "";           // true -- UH OH!
4 false == [];           // true -- UH OH!
5 "" == 0;               // true -- UH OH!
6 "" == [];              // true -- UH OH!
7 0 == [];               // true -- UH OH!
```

NEVER use == false or == true!

```
1 [] == ![];             // true
2
```

Ask Yourself:

1. Can either value be true or false?
2. Can either value ever be [], "", or 0?

```
1 42 == "43";                    // false
2 "foo" == 42;                   // false
3 "true" == true;                // false
4
5 42 == "42";                    // true
6 "foo" == [ "foo" ];            // true
```

Implicit Coercion: The Safe Parts

Primitive ⟷ Native

Implicit Coercion

```
1 var foo = "123";
2 foo.length;                    // 3
3
4 foo.charAt(2);                 // "3"
5
6 foo = new String("123");
7 var baz = foo + "";
8 typeof baz;                    // "string"
```

Implicit Coercion

"Any sufficiently advanced technology is indistinguishable from magic."

--Arthur C. Clarke

Implicit Coercion

"Any sufficiently complex technology
is indistinguishable from magic."
--many devs

Implicit Coercion

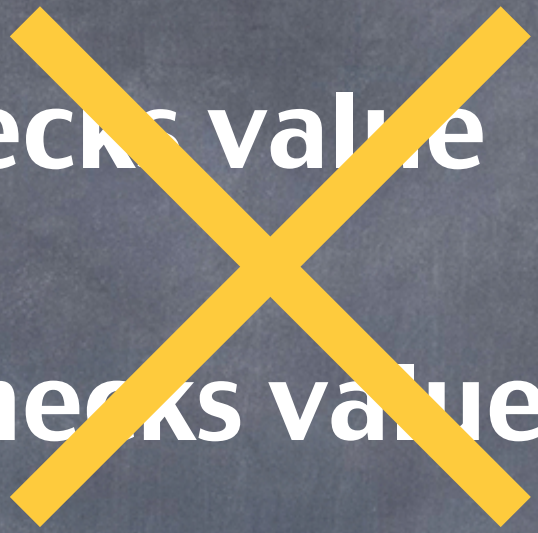
"Any sufficiently confusing technology is indistinguishable from magic."

--many devs

Implicit Coercion

== VS. ===

Implicit Coercion



== checks value

=== checks value and type

== allows coercion

=== disallows coercion

Implicit Coercion

```
1  var foo = [];
2  var baz = "";
3  if (foo == baz) {          // yup!
4      alert("Doh!");
5  }
6  if (foo === baz) {         // nope
7      alert("Phew.");
8  }
9
10 foo = 0;
11 if (foo == "") {           // yup!
12     alert("Argh!");
13 }
14 if (foo === "") {          // nope
15     alert("Phew.");
16 }
```

Implicit Coercion: == vs. ===

Implicit Coercion

http://davidwalsh.name/fixing-coercion

```
Number("") === 0
Number(false) === 0
Number(true) === 1
Number(null) === 0

String([]) === ""
String([null]) === ""
String([undefined]) === ""
```

Value Coercion: The Bad Parts

http://getify.github.io/coercions-grid/

	String(x)	x + ''	x.toString()	Number(x), +x	x * 1	x + 0
0	'0'	'0'	'0'	0	0	0
.0	'0'	'0'	'0'	0	0	0
-0	'0'	'0'	'0'	-0	-0	0
NaN	'NaN'	'NaN'	'NaN'	NaN	NaN	NaN
''	''	''	''	0	0	'0'
' '	' '	' '	' '	0	0	' 0'
'\n\n'	'\n\n'	'\n\n'	'\n\n'	0	0	'\n\n0'
null	'null'	'null'		0	0	0
undefined	'undefined'	'undefined'		NaN	NaN	NaN
true	'true'	'true'	'true'	1	1	1
false	'false'	'false'	'false'	0	0	0
/ /	'/ /'	'/ /'	'/ /'	NaN	NaN	'/ /0'
Infinity	'Infinity'	'Infinity'	'Infinity'	Infinity	Infinity	Infinity
-Infinity	'-Infinity'	'-Infinity'	'-Infinity'	-Infinity	-Infinity	-Infinity
'Infinity'	'Infinity'	'Infinity'	'Infinity'	Infinity	Infinity	'Infinity0'
'-Infinity'	'-Infinity'	'-Infinity'	'-Infinity'	-Infinity	-Infinity	'-Infinity0'
function(){}	'function (){}'	'function (){}'	'function (){}'	NaN	NaN	'function (){}0'
Symbol('')	'Symbol()'		'Symbol()'			
[]	''	''	''	0	0	'0'
[0]	'0'	'0'	'0'	0	0	'00'
[.0]	'0'	'0'	'0'	0	0	'00'
[-0]	'0'	'0'	'0'	0	0	'00'
[NaN]	'NaN'	'NaN'	'NaN'	NaN	NaN	'NaN0'
['']	''	''	''	0	0	'0'
[' ']	' '	' '	' '	0	0	' 0'
['\n\n']	'\n\n'	'\n\n'	'\n\n'	0	0	'\n\n0'
[null]	''	''	''	0	0	'0'

Infinity + ''

Value Coercion: The Bad Parts

but... but...

Implicit Coercion: == helpful?

```
1  var foo = "3";
2  if (foo === 3 || foo === "3") { // yup
3      alert("Thanks, but...");
4  }
5
6  if (foo == 3) {                 // yup!
7      alert("That's nicer!");
8  }
9
10 if (typeof foo === "string") {  // yup
11     alert("typeof always returns a string");
12 }
```

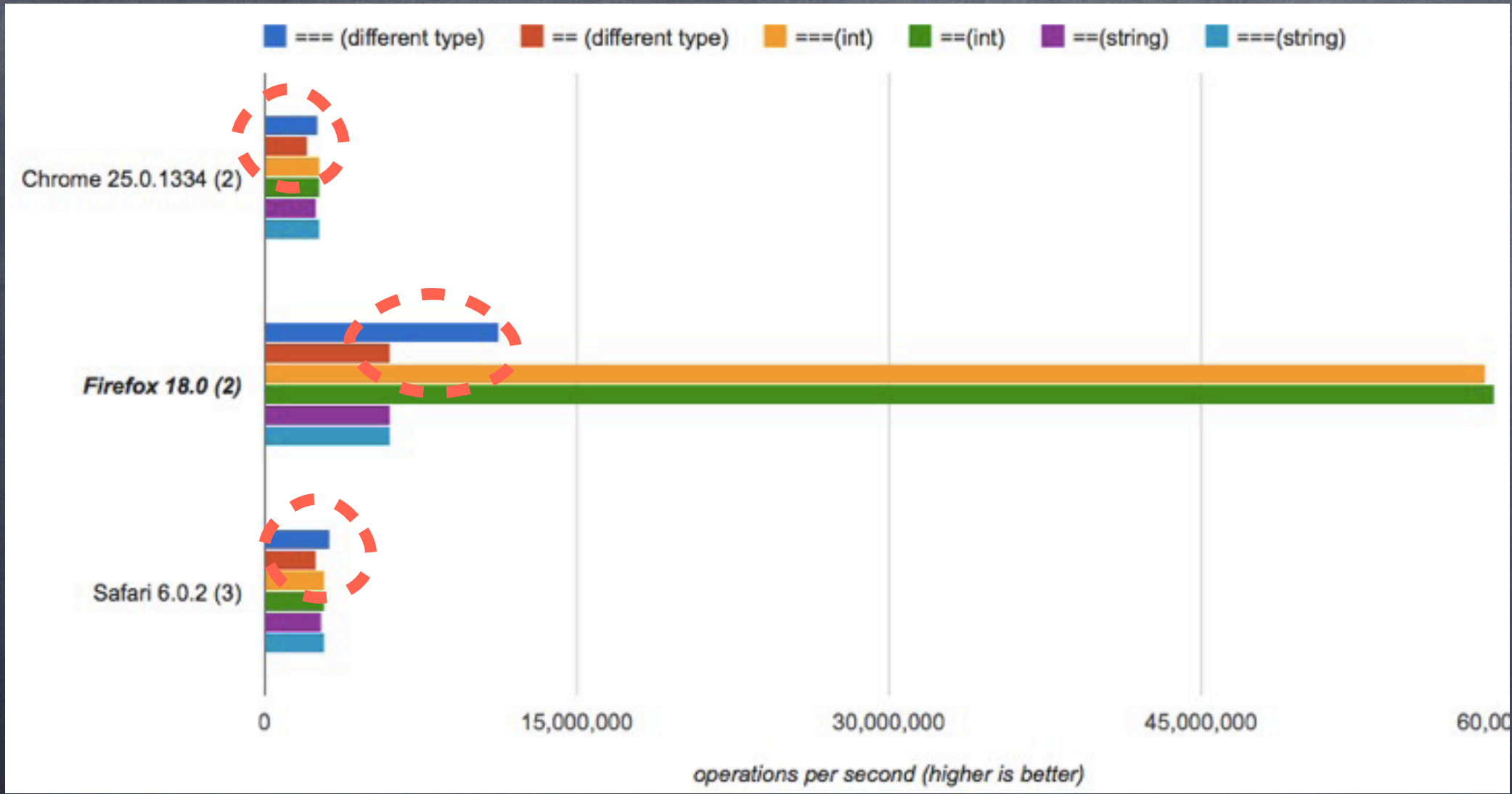
Implicit Coercion: == helpful?

```
 1 var foo;
 2 if (foo == null) {          // yup!
 3     alert("Thanks!");
 4 }
 5
 6 foo = null;
 7 if (foo == null) {          // yup
 8     alert("Thanks, again!");
 9 }
10
11 foo = false;
12 if (foo == null) {          // nope!
13     alert("Phew!");
14 }
```

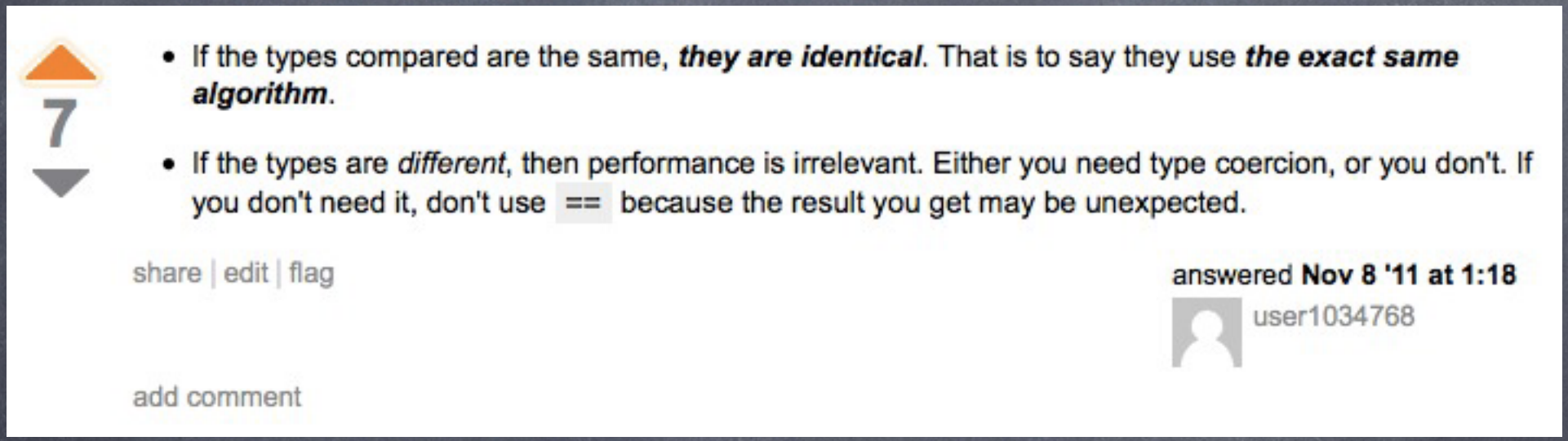
Implicit Coercion: == helpful?

What about the performance of == or ===?

http://jsperf.com/triple-equals-vs-double-equals/5



Implicit Coercion: == vs. === performance



Implicit Coercion: == vs. === performance

#FUD: using == in your code is dangerous. Avoid it!

#protip: use === where it's safer and use == where it's more helpful.

#FUD: JavaScript's implicit coercion is a flaw in the design of the language. Avoid it!

#protip: use explicit coercion where it's safer and use implicit coercion where it's more helpful.

http://jscoercion.qfox.nl/

Kyle Simpson
@getify
http://getify.me

Thanks!

Questions?