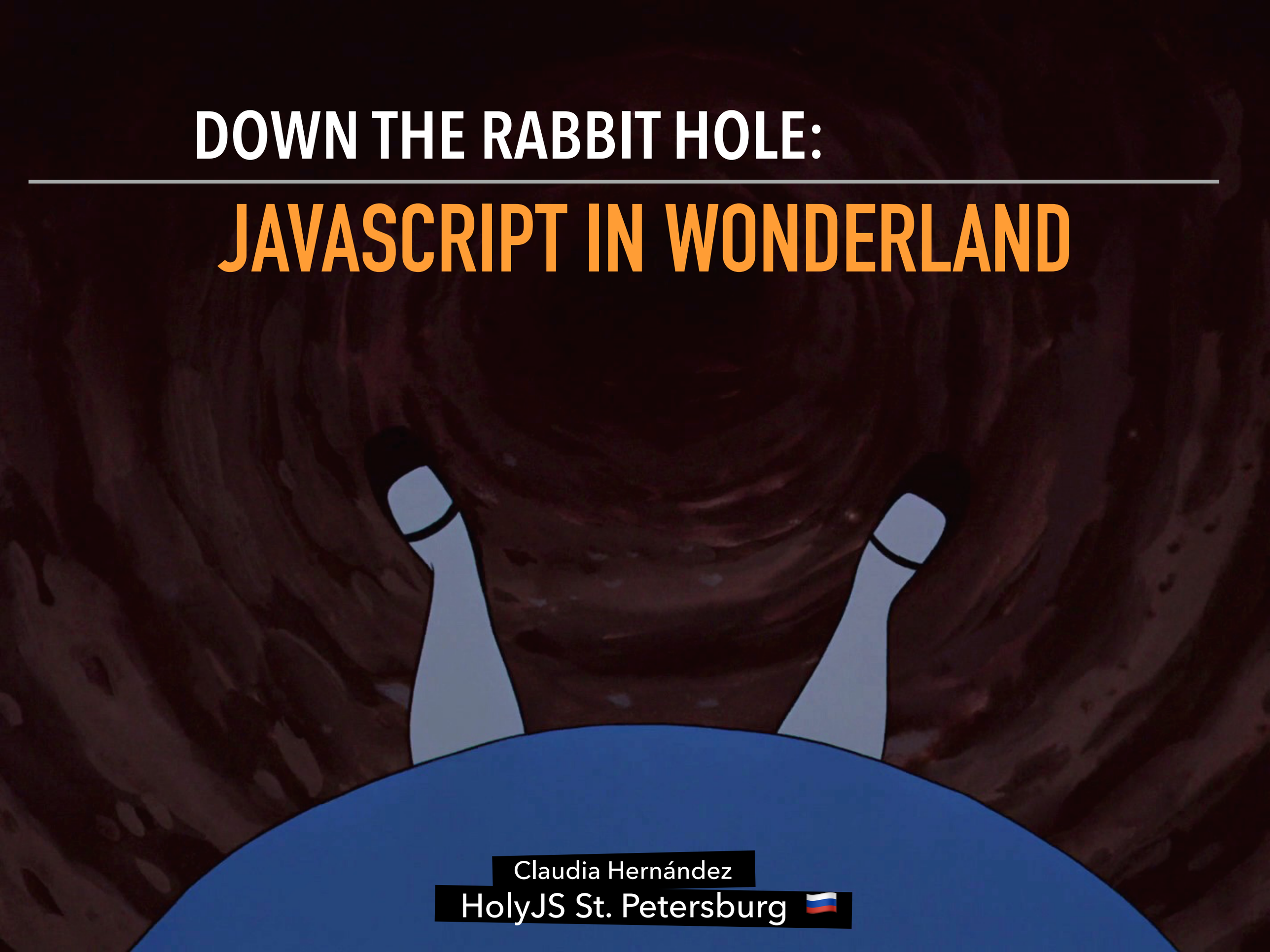
DOWN THE RABBIT HOLE:

JAVASCRIPT IN WONDERLAND



Claudia Hernández

HolyJS St. Petersburg 🇷🇺

CLAUDIA HERNÁNDEZ

@koste4

claudia.hernandez@dailymotion.com

dailymotion

dailymotion

Vidéo, chaîne...

ENVOYER UNE VIDÉO

CONNEXION

Créez un compte ou connectez-vous pour des recommandations personnalisées

Connexion

Recommandées pour vous

Catégories

- Musique
- Jeux vidéo
- Sport
- News
- Cinéma

Parcourir

Qui suivre

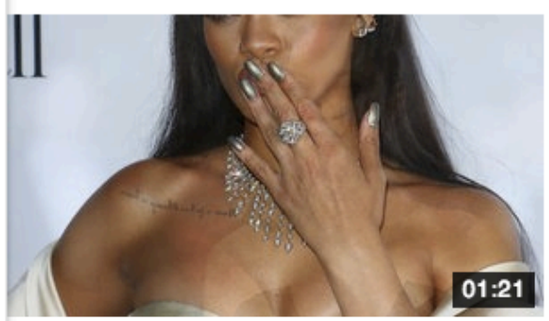
- Dailymotion Jobs
- Charles Asselin
- Chefclub

Recommandées pour vous | Abonnements

Les meilleures idées sucrées avec de la pâte feuilletée [volume 2]

par Chefclub

RECOMMANDÉE pour vous



Cannes 2017: les actrices qui vont faire briller la Croisette

par lalibre

RECOMMANDÉE pour vous

Foot - Bleus : Deschamps sur

Fakear - Boiler Room Paris

Old XP

Symphony + Twig

New XP

React + Redux + Apollo + GraphQL

dailymotion

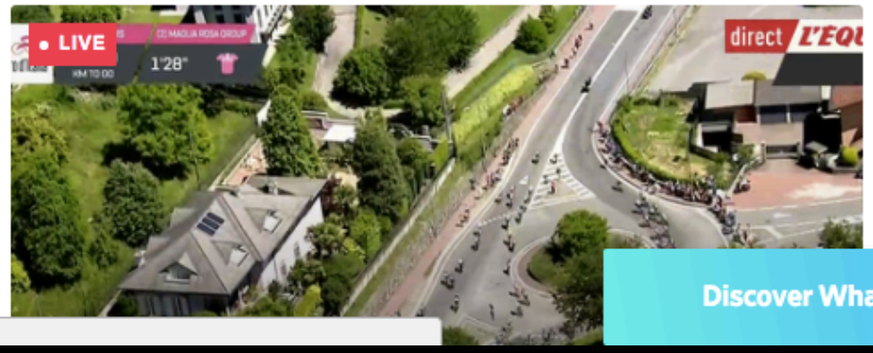
For You Explore

Find videos, Channels...

Sign in

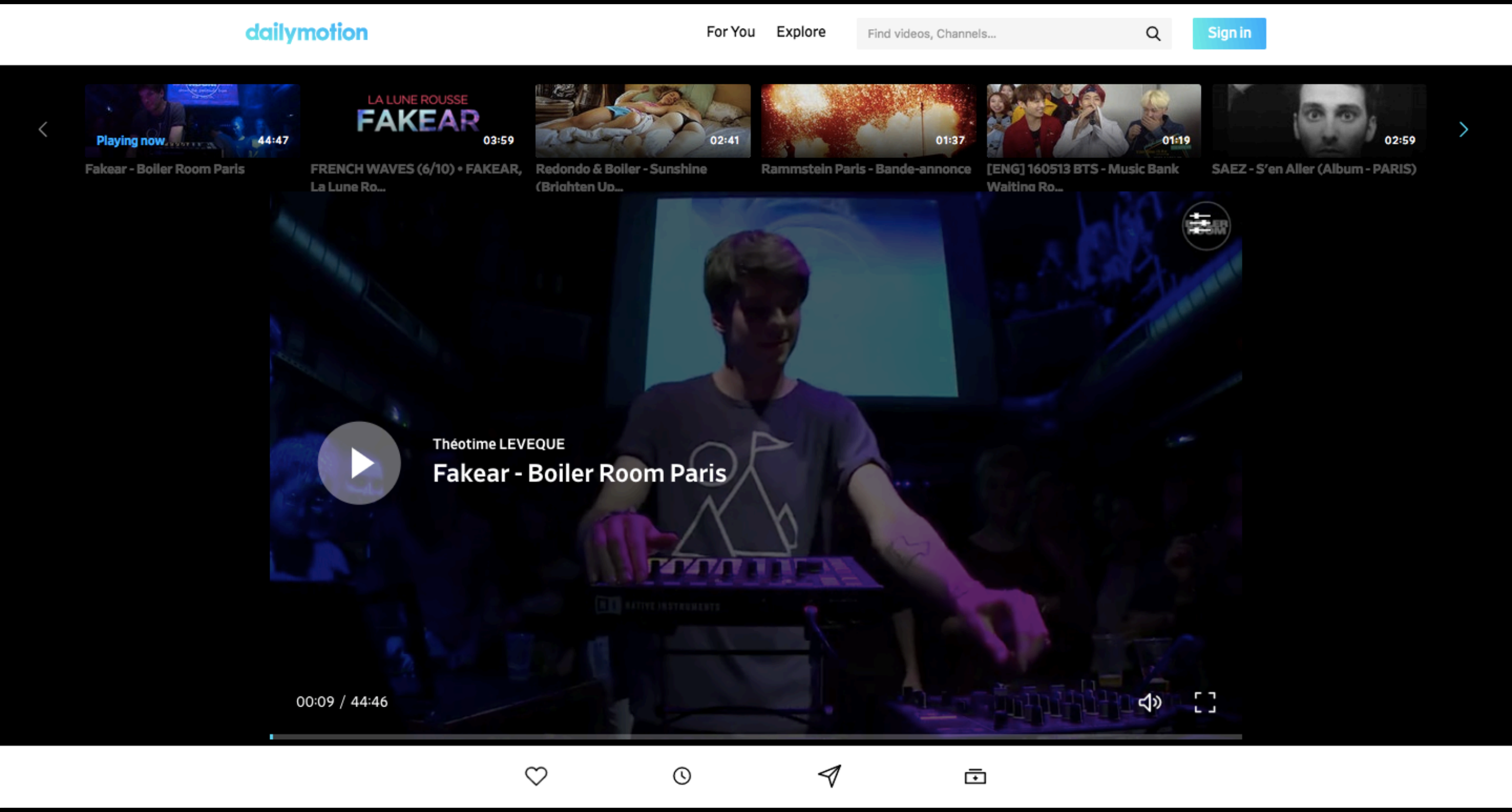
Suivez tout ce qu'il se passe

Trending Lives



Discover What's New

localhost:3000/video/x5lxyqo


dailymotion
For You
Explore
Find videos, Channels...
Sign in
Playing now
44:47
Fakear - Boiler Room Paris
LA LUNE ROUSSE
FAKEAR
03:59
FRENCH WAVES (6/10) • FAKEAR, La Lune Ro...
02:41
Redondo & Boiler - Sunshine (Brighten Up...
01:37
Rammstein Paris - Bande-annonce
01:19
[ENG] 160513 BTS - Music Bank Waiting Ro...
02:59
SAEZ - S'en Aller (Album - PARIS)
Théotime LEVEQUE
Fakear - Boiler Room Paris
00:09 / 44:46

dailymotion

For You Explore Find videos, Channels... Sign in

Les sujets qui font du bruit

Emmanuel Macron

+ Follow

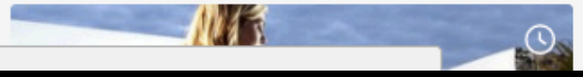
Des topics des plus épiques

Édouard Philippe

+ Follow

Les sujets à creuser

Benjamin Biolay

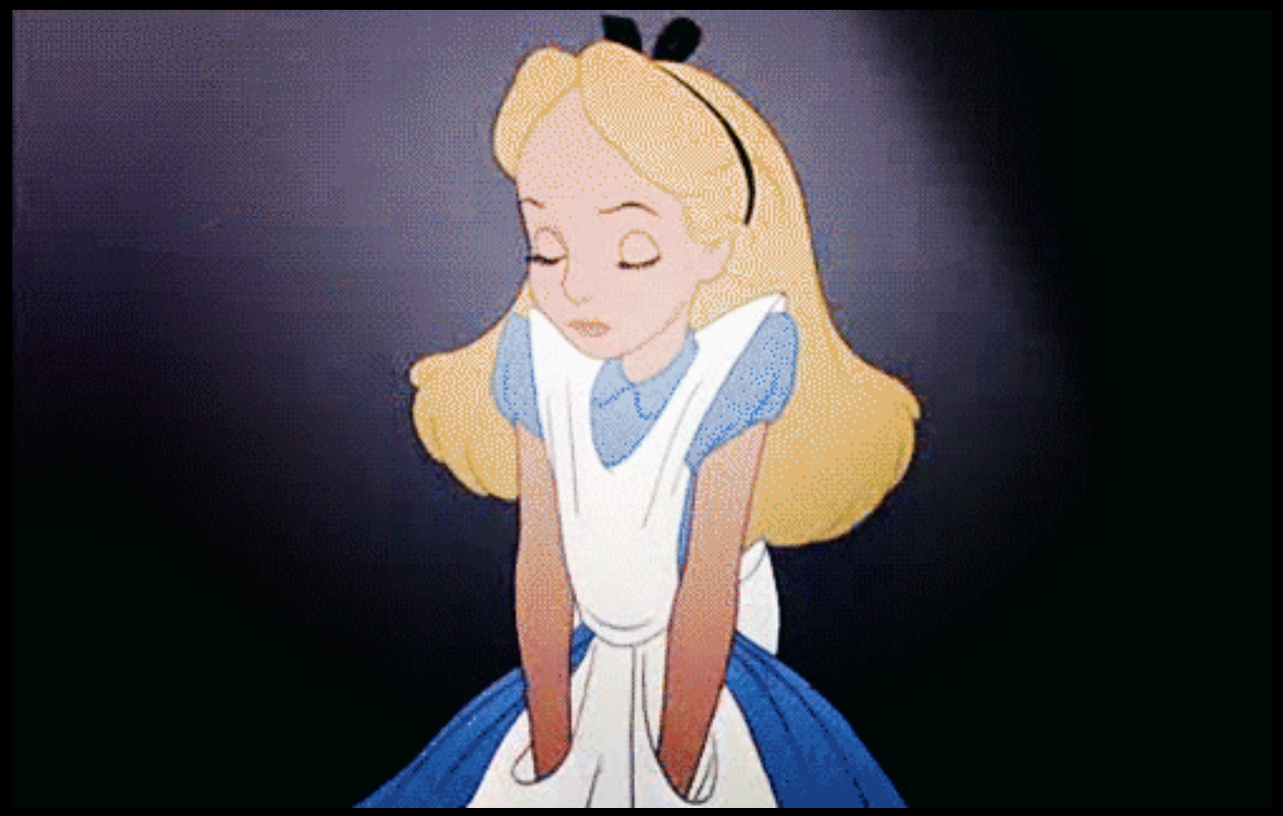


Discover What's New

localhost:3000/video/x5mxkih

Coming soon...


Oh, she's curious.



```
Math.min() < Math.max()  // false
```

```
Math.max() // -Infinity
Math.min() //  Infinity
```

```
.1+.2 === .3 // false
```

```
.1+.2    // 0.30000000000000004
```

```
const a = 012
console.log(a)
```

```
const a = 012
console.log(a) // 012
```

```
const a = 012
console.log(a) // 12
```

```
const a = 012
console.log(a) // 10
```

```
[1,2,3] === [1,2,3] //false
```

??????lolwat

claudia ᕦ(̇ᐡ)ᕤ 🇷🇺
@koste4

JS Pro's & Con's

Pro: Easy to learn.
Con: Easy to fuckup.

😂

@slobodan_ #holyjs

RETWEETS 3 LIKES 6

4:32 AM - 2 Jun 2017 from Saint Petersburg, Russia

CURIOUSER AND CURIOUSER

♣ NaN ♣

```
typeof NaN // number
```

```
const foo = 2 / 'bar' // NaN
```

0/0 // NaN

```
Math.sqrt(-9) // NaN
```

```
typeof NaN // number
```

Mathematical operations can't lead to an **error** or **crash** in JavaScript

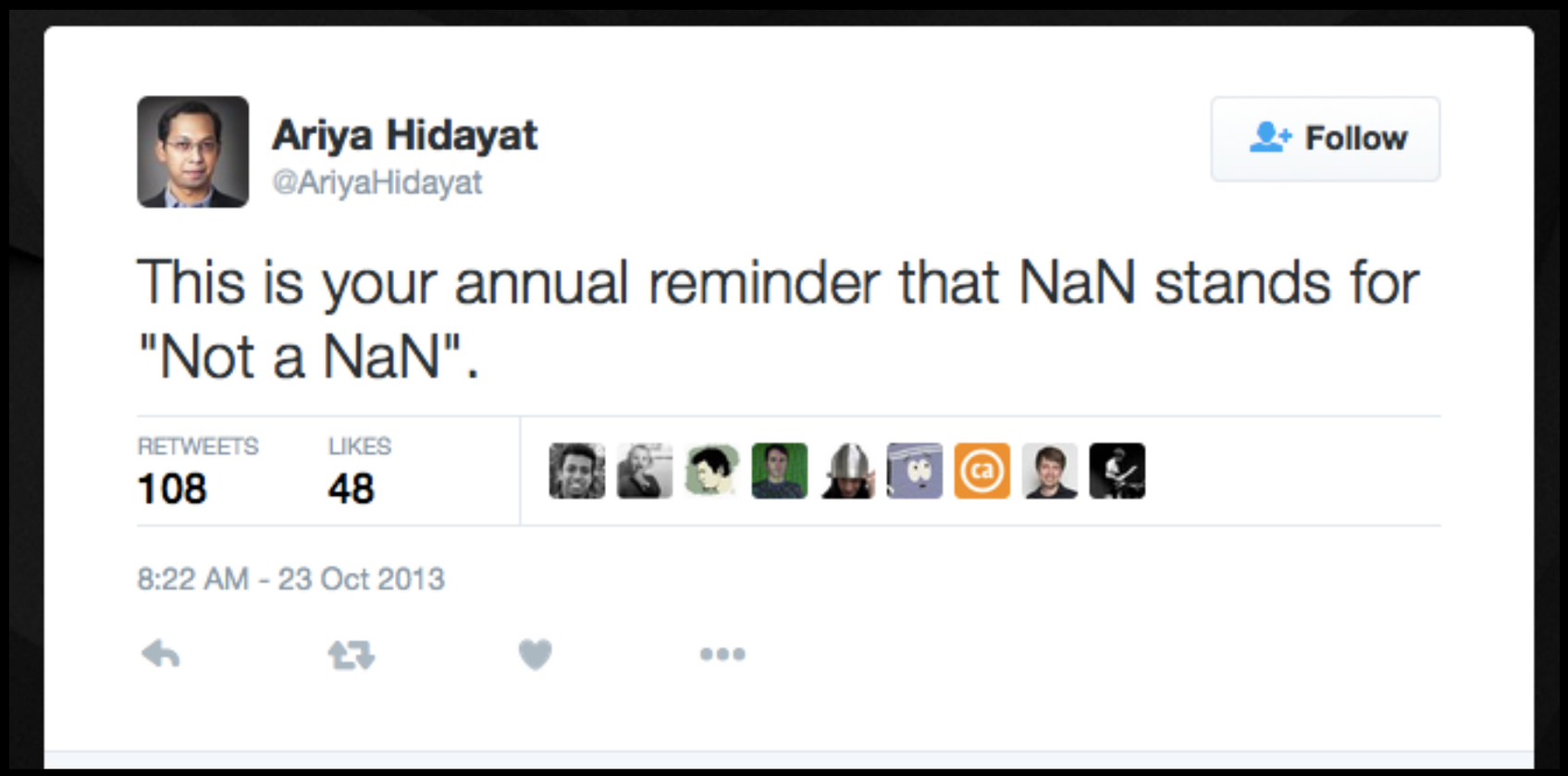
```
NaN === NaN // false
```

NaN isn’t just a Javascript thing…

NaN is actually defined by the **IEEE754** floating-point standard

16,777,214

NaN !== NaN


Ariya Hidayat
@AriyaHidayat
Follow
This is your annual reminder that NaN stands for "Not a NaN".
RETWEETS
108
LIKES
48
8:22 AM - 23 Oct 2013

```
isNaN(NaN) // true
```

~ operator

What's a bit-wise operator?

Operator that treats its operands as a sequence of 32 bits (zeroes and ones) rather than as decimal numbers. They return standard Javascript numerical values.

Bitwise AND &

Bitwise OR |

The following table summarizes JavaScript's bitwise operators:

Operator	Usage	Description
Bitwise AND	a & b	Returns a one in each bit position for which the corresponding bits of both operands are ones.
Bitwise OR	a | b	Returns a one in each bit position for which the corresponding bits of either or both operands are ones.
Bitwise XOR	a ^ b	Returns a one in each bit position for which the corresponding bits of either but not both operands are ones.
Bitwise NOT	~ a	Inverts the bits of its operand.
Left shift	a << b	Shifts a in binary representation b (< 32) bits to the left, shifting in zeroes from the right.
Sign-propagating right shift	a >> b	Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off.
Zero-fill right shift	a >>> b	Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off, and shifting in zeroes from the left.

Signed 32-bit integers

The operands of all bitwise operators are converted to signed 32-bit integers in two's complement format. Two's complement format means that a number's negative counterpart (e.g. 5 vs. -5) is all the number's bits inverted (bitwise NOT of the number, a.k.a. ones' complement of the number) plus one. For example, the following encodes the integer 314:

```
00000000000000000000000100111010
```

The following encodes ~314, i.e. the ones' complement of 314:

IN THIS ARTICLE

Bitwise NOT

~

9 = 00000000000000000000000000001001

~9 = 11111111111111111111111111110110

-(N+1)

```
console.log(~-2) // 1
console.log(~-1) // 0
console.log(~0)  // -1
console.log(~1)  // -2
console.log(~2)  // -3
```

```
~~1.2543    // 1
~~4.9       // 4
~~(-2.999)  // -2
```

```
const float = 281.0901

~~float          // 281
float | 0        // 281
float << 0       // 281
float >> 0       // 281
float >>> 0      // 281
```

Done. Ready to run again.

Run again

Testing in Chrome 57.0.2987 / Mac OS X 10.12.3		
Test		**Ops/sec**

<< 0	`var r, i;`
	`for (i = 0; i < 10000; i += 1) {`
	`  r = n[i] << 0;`
	`}`
		112,999
		±3.43%
		fastest

| 0	`var r, i;`
	`for (i = 0; i < 10000; i += 1) {`
	`  r = n[i] | 0;`
	`}`
		115,993
		±2.74%
		fastest

~~	`var r, i;`
	`for (i = 0; i < 10000; i += 1) {`
	`  r = ~~n[i];`
	`}`
		114,693
		±2.51%
		fastest

Math.floor()	`var r, i;`
	`for (i = 0; i < 10000; i += 1) {`
	`  r = Math.floor(n[i]);`
	`}`
		31,557
		±1.74%
		73% slower

Compare results of other browsers

https://jsperf.com/math-floor-vs-bitwise-double-not/2

Bitwise NOT

~

```
const teaParty = ['madHatter', marchHare', 'dormouse']
```

```
if (teaParty.indexOf('marchHare') >= 0) {
    // marchHare in the teaParty
}

if (teaParty.indexOf('marchHare') != -1) {
    // marchHare in the teaParty
}

if (teaParty.indexOf('marchHare') < 0) {
    // marchHare not in the teaParty
}

if (teaParty.indexOf('marchHare') == -1) {
    // marchHare not in the teaParty
}
```

```
if (~teaParty.indexOf('marchHare')) {
  // marchHare in the teaParty
} else {
  // marchHare not in the teaParty
}
```

```
if (teaParty.includes('marchHare')) {
  // marchHare in the teaParty
} else {
  // marchHare not in the teaParty
}
```

♣ for loops ♣

```
for(;;) {}
```

```
for (initialization; condition; iteration) {
    // code
}
```

```
let i = 0
for (; i < 9; i++) {}
```

```
for (let i = 0;; i++) {
  if (i > 3) break;
  // code
}
```

```
for(;i--;) {}
```

```
for(;;) {} = for(; true ;) = while(true) {}
```

undefined

```
const cheshireCat

console.log(cheshireCat == undefined) // true
```

```
undefined in window // true
```

```
undefined = "we are all mad here"

console.log(undefined)
```

```
// IE8 & below
undefined = "we are all mad here"
console.log(undefined)
// "we are all mad here"

// Modern browsers
undefined = "we are all mad here"
console.log(undefined) // undefined
```

Syntax

```
undefined
```

Description

`undefined` is a property of the *global object*, i.e. it is a variable in global scope. The initial value of `undefined` is the primitive value `undefined`.

In modern browsers (JavaScript 1.8.5 / Firefox 4+), `undefined` is a non-configurable, non-writable property per the ECMAScript 5 specification. Even when this is not the case, avoid overriding it.

A variable that has not been assigned a value is of type undefined. A method or statement also returns `undefined` if the variable that is being evaluated does not have an assigned value. A function returns `undefined` if a value was not `returned`.

Since `undefined` is not a `reserved word`, it can be used as an identifier (variable name) in any scope other than the global scope.

```
// logs "foo string"
(function(){ var undefined = 'foo'; console.log(undefined, typeof undefined); })

// logs "foo string"
(function(undefined){ console.log(undefined, typeof undefined); })('foo');
```

Examples

Strict equality and `undefined`

```
const undefined = 5

console.log(undefined) // 5
```

What have we learned so far ?

Not a Number is a Number

~ is the bitwise NOT operator

all parts in a for loop are optional

undefined can be defined (sometimes)

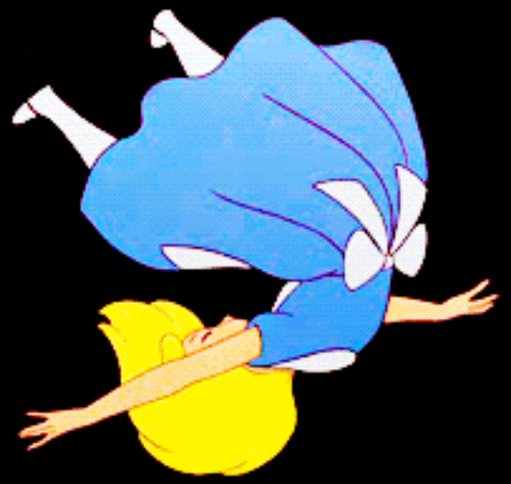
ARRAY#SORT FUN

```
myArray = [33, 2, 98, 25, 4]

myArray.sort()

// [ 2, 25, 33, 4, 98 ]
```

lexicographical sorting

'**dictionary**' or '**telephone book**'

not numerical order



```
colors = ['red', 'blue']
colors.sort() // ['blue', 'red']

numbers = [80, 9]
numbers.sort() // [80, 9]
```

```
colors = ['red', 'blue']
colors.sort() // ['blue', 'red']

numbers = ['80', '9']
numbers.sort() // ['80', '9']
```

UNICODE CODE POINT VALUE

80 = 56 48

9 = 57 00

str.codePointAt(pos)

```
const emojis = ['🍄', '🃏', '👑']
emojis.sort() // ['🃏', '🍄', '👑']

const wtfJS = [390, '😂', 1, '2315']
wtfJS.sort() // [1, '2325', 390, '😂']
```

```
// [ 2, 25, 33, 4, 98 ]
```

```
function compare (a, b) {
    if (a < b ) return -1 // a comes first than b
    else if (a > b) return 1 // b comes first than a
    else return 0  // a and b are left unchanged
}
```

```
myArray = [33, 2, 98, 25, 4]

myArray.sort( (a,b) => a - b )

// [ 2, 4, 25, 33,  98 ]
```

Have you ever wondered which algorithm is used in JS native sort method ?

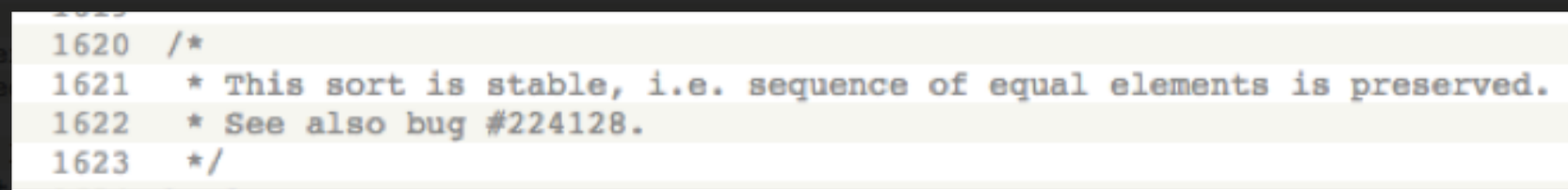
```
1620 /*
1621  * This sort is stable, i.e. sequence of equal elements is preserved.
1622  * See also bug #224128.
1623  */
1624 bool
1625 js_MergeSort(void *src, size_t nel, size_t elsize,
1626              JSComparator cmp, void *arg, void *tmp,
1627              JSMergeSortElemType elemType)
1628 {
1629     void *swap, *vec1, *vec2;
1630     MSortArgs msa;
1631     size_t i, j, lo, hi, run;
1632     int cmp_result;
1633
1634     JS_ASSERT_IF(JS_SORTING_VALUES, elsize == sizeof(Value));
1635     bool isValue = elemType == JS_SORTING_VALUES;
1636
1637     /* Avoid memcpy overhead for word-sized and word-aligned elements. */
1638 #define COPY_ONE(p,q,n) \
1639     (isValue ? (void)(*(Value*)p = *(Value*)q) : (void)memcpy(p, q, n))
1640 #define CALL_CMP(a, b) \
1641     if (!cmp(arg, (a), (b), &cmp_result)) return JS_FALSE;
1642 #define INS_SORT_INT 4
1643
1644     /*
1645      * Apply insertion sort to small chunks to reduce the number of merge
1646      * passes needed.
1647      */
1648     for (lo = 0; lo < nel; lo += INS_SORT_INT) {
1649         hi = lo + INS_SORT_INT;
1650         if (hi >= nel)
1651             hi = nel;
1652         for (i = lo + 1; i < hi; i++) {
1653             vec1 = (char *)src + i * elsize;
1654             vec2 = (char *)vec1 - elsize;
1655             for (j = i; j > lo; j--) {
1656                 CALL_CMP(vec2, vec1);
1657                 /* "<=" instead of "<" insures the sort is stable */
1658                 if (cmp_result <= 0) {
1659                     break;
1660                 }
1661
1662                 /* Swap elements, using "tmp" as tmp storage */
1663                 COPY_ONE(tmp, vec2, elsize);
1664                 COPY_ONE(vec2, vec1, elsize);
1665                 COPY_ONE(vec1, tmp, elsize);
1666                 vec1 = vec2;
1667                 vec2 = (char *)vec1 - elsize;
1668             }
1669         }
1670     }
1671 #undef CALL_CMP
1672 #undef COPY_ONE
1673
```

SPIDER MONKEY

MERGESORT SORT

https://hg.mozilla.org/mozilla-central/file/28be8df0deb7/js/src/jsarray.cpp

```
/*
 * This sort is stable, i.e. sequence of equal elements is preserved.
 * See also bug #224128.
 */
bool
js_MergeSort(void *src, size_t nel, size_t elsize,
             JSComparator cmp, void *arg, void *tmp,
             JSMergeSortElemType elemType)
{
    void *swap, *vec1, *vec2;
    MSortArgs msa;
    size_t i, j, lo, hi, run;
    int cmp_result;

    JS_ASSERT_IF(JS_SORTING_VALUES, elsize == sizeof(Value));
    bool isValue = elemType == JS_SORTING_VALUES;

    /* Avoid memcpy overhead for word-sized and word-aligned elements. */
#define COPY_ONE(p,q,n) \
    (isValue ? (void)(*(Value*)p = *(Value*)q) : (void)memcpy(p, q, n))
#define CALL_CMP(a, b) \
    if (!cmp(arg, (a), (b), &cmp_result)) return JS_FALSE;
#define INS_SORT_INT 4

    /*
     * Apply inse
     * passes nee
     */
    for (lo = 0;
        hi = lo +
        if (hi >= nel)
            hi = nel;
        for (i = lo + 1; i < hi; i++) {
            vec1 = (char *)src + i * elsize;
            vec2 = (char *)vec1 - elsize;
            for (j = i; j > lo; j--) {
                CALL_CMP(vec2, vec1);
                /* "<=" instead of "<" insures the sort is stable */
                if (cmp_result <= 0) {
                    break;
                }

                /* Swap elements, using "tmp" as tmp storage */
                COPY_ONE(tmp, vec2, elsize);
                COPY_ONE(vec2, vec1, elsize);
                COPY_ONE(vec1, tmp, elsize);
                vec1 = vec2;
                vec2 = (char *)vec1 - elsize;
            }
        }
    }
#undef CALL_CMP
#undef COPY_ONE
```



https://hg.mozilla.org/mozilla-central/file/28be8df0deb7/js/src/jsarray.cpp

```
/*
 * This sort is stable, i.e. sequence of equal elements is preserved.
 * See also bug #224128.
 */
bool
js_MergeSort(void *src, size_t nel, size_t elsize,
             JSComparator cmp, void *arg, void *tmp,
             JSMergeSortElemType elemType)
{
    void *swap, *vec1, *vec2;
    MSortArgs msa;
    size_t i, j, lo, hi, run;
    int cmp_result;

    JS_ASSERT_IF(JS_SORTING_VALUES, elsize == sizeof(Value));
    bool isValue = elemType == JS_SORTING_VALUES;

    /* Avoid memcpy overhead for word-sized and word-aligned elements. */
#define COPY_ONE(p,q,n) \
    (isValue ? (void)(*(Value*)p = *(Value*)q) : (void)memcpy(p, q, n))
#define CALL_CMP(a, b) \
    if (!cmp(arg, (a), (b), &cmp_result)) return JS_FALSE;
#define INS_SORT_INT 4

    /*
     * Apply i
     * passes
     */
    for (lo =
        hi = l
        if (hi >= nel)
            hi = nel;
        for (i = lo + 1; i < hi; i++) {
            vec1 = (char *)src + i * elsize;
            vec2 = (char *)vec1 - elsize;
            for (j = i; j > lo; j--) {
                CALL_CMP(vec2, vec1);
                /* "<=" instead of "<" insures the sort is stable */
                if (cmp_result <= 0) {
                    break;
                }

                /* Swap elements, using "tmp" as tmp storage */
                COPY_ONE(tmp, vec2, elsize);
                COPY_ONE(vec2, vec1, elsize);
                COPY_ONE(vec1, tmp, elsize);
                vec1 = vec2;
                vec2 = (char *)vec1 - elsize;
            }
        }
    }
#undef CALL_CMP
#undef COPY_ONE
```

```
/*
 * Apply insertion sort to small chunks to reduce the number of merge
 * passes needed.
 */
```

SPIDER MONKEY

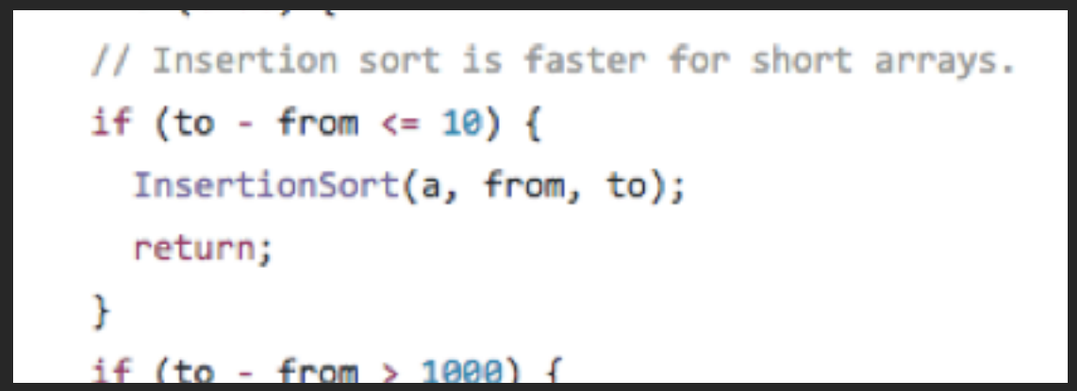
MERGESORT SORT

https://hg.mozilla.org/mozilla-central/file/28be8df0deb7/js/src/jsarray.cpp

```
759  function QuickSort(a, from, to) {
760    var third_index = 0;
761    while (true) {
762      // Insertion sort is faster for short arrays.
763      if (to - from <= 10) {
764        InsertionSort(a, from, to);
765        return;
766      }
767      if (to - from > 1000) {
768        third_index = GetThirdIndex(a, from, to);
769      } else {
770        third_index = from + ((to - from) >> 1);
771      }
772      // Find a pivot as the median of first, last and middle element.
773      var v0 = a[from];
774      var v1 = a[to - 1];
775      var v2 = a[third_index];
776      var c01 = comparefn(v0, v1);
777      if (c01 > 0) {
778        // v1 < v0, so swap them.
779        var tmp = v0;
780        v0 = v1;
781        v1 = tmp;
782      } // v0 <= v1.
783      var c02 = comparefn(v0, v2);
784      if (c02 >= 0) {
785        // v2 <= v0 <= v1.
786        var tmp = v0;
787        v0 = v2;
788        v2 = v1;
789        v1 = tmp;
790      } else {
791        // v0 <= v1 && v0 < v2
792        var c12 = comparefn(v1, v2);
793        if (c12 > 0) {
794          // v0 <= v2 < v1
```

https://github.com/v8/v8/blob/master/src/js/array.js

```
function QuickSort(a, from, to) {
  var third_index = 0;
  while (true) {
    // Insertion sort is faster for short arrays.
    if (to - from <= 10) {
      InsertionSort(a, from, to);
      return;
    }
    if (to - from > 1000) {
      third_index = GetThirdIndex(a, from, to);
    } else {
      third_index = from + ((to - from) >> 1);
    }
    // Find a pivot as the median of first, last and middle element.
    var v0 = a[from];
    var v1 = a[to - 1];
    var v2 = a[third_index];
    var c01 = comparefn(v0, v1);
    if (c01 > 0) {
      // v1 < v0, so swap them.
      var tmp = v0;
      v0 = v1;
      v1 = tmp;
    } // v0 <= v1.
    var c02 = comparefn(v0, v2);
    if (c02 >= 0) {
      // v2 <= v0 <= v1.
      var tmp = v0;
      v0 = v2;
      v2 = v1;
      v1 = tmp;
    } else {
      // v0 <= v1 && v0 < v2
      var c12 = comparefn(v1, v2);
      if (c12 > 0) {
        // v0 <= v2 < v1
```



https://github.com/v8/v8/blob/master/src/js/array.js

```
555
556     function mergeSort(array, valueCount, comparator)
557     {
558         var buffer = [ ];
559         buffer.length = valueCount;
560
561         var dst = buffer;
562         var src = array;
563         for (var width = 1; width < valueCount; width *= 2) {
564             for (var srcIndex = 0; srcIndex < valueCount; srcIndex += 2 * width)
565                 merge(dst, src, srcIndex, valueCount, width, comparator);
566
567             var tmp = src;
568             src = dst;
569             dst = tmp;
570         }
571
572         if (src != array) {
573             for(var i = 0; i < valueCount; i++)
574                 array[i] = src[i];
575         }
576     }
577
578     function bucketSort(array, dst, bucket, depth)
579     {
580         if (bucket.length < 32 || depth > 32) {
581             mergeSort(bucket, bucket.length, stringComparator);
582             for (var i = 0; i < bucket.length; ++i)
583                 array[dst++] = bucket[i].value;
584             return dst;
585         }
586
587         var buckets = [ ];
588         for (var i = 0; i < bucket.length; ++i) {
589             var entry = bucket[i];
590             var string = entry.string;
591             if (string.length == depth) {
592                 array[dst++] = entry.value;
593                 continue;
594             }
595
596             var c = string.charCodeAt(depth);
597             if (!buckets[c])
598                 buckets[c] = [ ];
599             buckets[c][buckets[c].length] =
600         }
601
```

NITRO

MERGE SORT

https://trac.webkit.org/browser/trunk/Source/JavaScriptCore/builtins/ArrayPrototype.js

Microsoft / ChakraCore

Watch 410 | Star 6,084 | Fork 700

Code | Issues 195 | Pull requests 35 | Projects 1 | Wiki | Pulse | Graphs

CHAKRA

QUICKSORT SORT

Tree: 4e26a257a3 | ChakraCore / lib / Common / DataStructures / QuickSort.h | Copy path

obastemur QuickSort: improve perf. and move our custom qsort_s

3 contributors

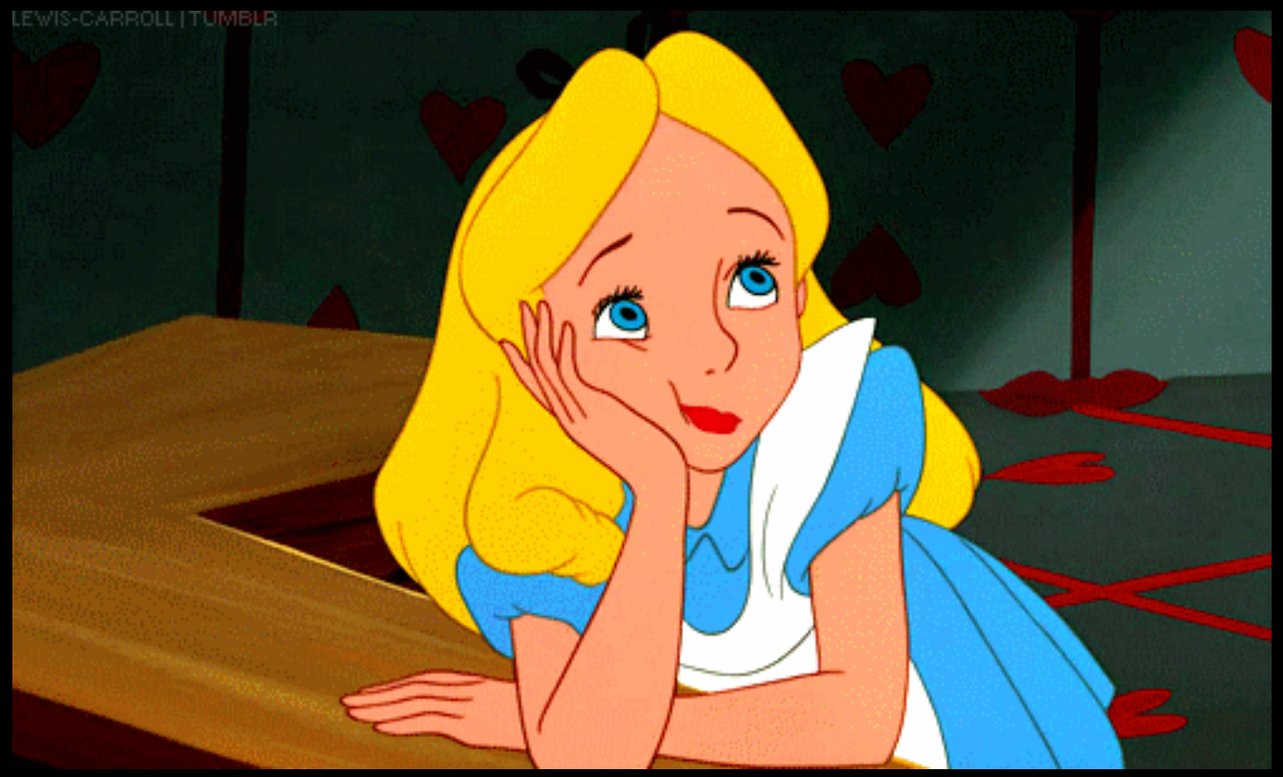
283 lines (257 sloc) | 7.32 KB | Raw | Blame | History

```
//----------------------------------------------------------------------------
// Copyright (C) Microsoft. All rights reserved.
// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.
//----------------------------------------------------------------------------

#pragma once
namespace JsUtil
{

// Sorting Networks - BEGIN
#define CCQ_SORT2(a, b)                 \
    if (comparer(context,               \
        base + (b * size),              \
        base + (a * size)) < 0)         \
    {                                   \
        CCQ_SWAP(base + (a * size),     \
          base + (b * size), size);     \
    }

#define CCQ_SORT3() \
    CCQ_SORT2(0, 1) \
    CCQ_SORT2(1, 2) \
    CCQ_SORT2(0, 1)
```

https://github.com/Microsoft/ChakraCore/blob/master/lib/Common/DataStructures/QuickSort.h

JS Engine	Sort Algorithm
SpiderMonkey FIREFOX	Insertion Sort (for short arrays) Merge Sort
V8 CHROME	Insertion Sort (for short arrays) Quick Sort
Nitro SAFARI	Merge Sort
Chakra INTERNET EXPLORER	QuickSort

1. Insertion Sort
2. Merge Sort
3. Quick Sort


LEWIS-CARROLL | TUMBLR

```
1  function InsertionSort(arr) {
2    for(let i = 1; i < arr.length; i++) {
3      let value = arr[i]
4      for (let j = i - 1; j >= 0 && arr[j] > value; j--) {
5        arr[j+1] = arr[j]
6      }
7      arr[j+1] = value
8    }
9    return arr
10 }
```

10 LOC

```
1  function MergeSort(arr) {
2    if (arr.length < 2) {
3      return arr
4    }
5    let middle = Math.floor(arr.length / 2)
6    let left = arr.slice(0, middle)
7    let right = arr.slice(middle)
8    return merge(MergeSort(left), MergeSort(right))
9  }
10
11 function merge(left, right) {
12   let result = [],
13       i = 0,
14       j = 0
15   while(i < left.arr.length && j < right.arr.length) {
16     if(left[i] < right[j]) {
17       result.push(left[i++])
18     } else {
19       result.push(right[j++])
20     }
21   }
22   return result.concat(left.slice(i)).concat(right.slice(j))
23 }
```

23 LOC

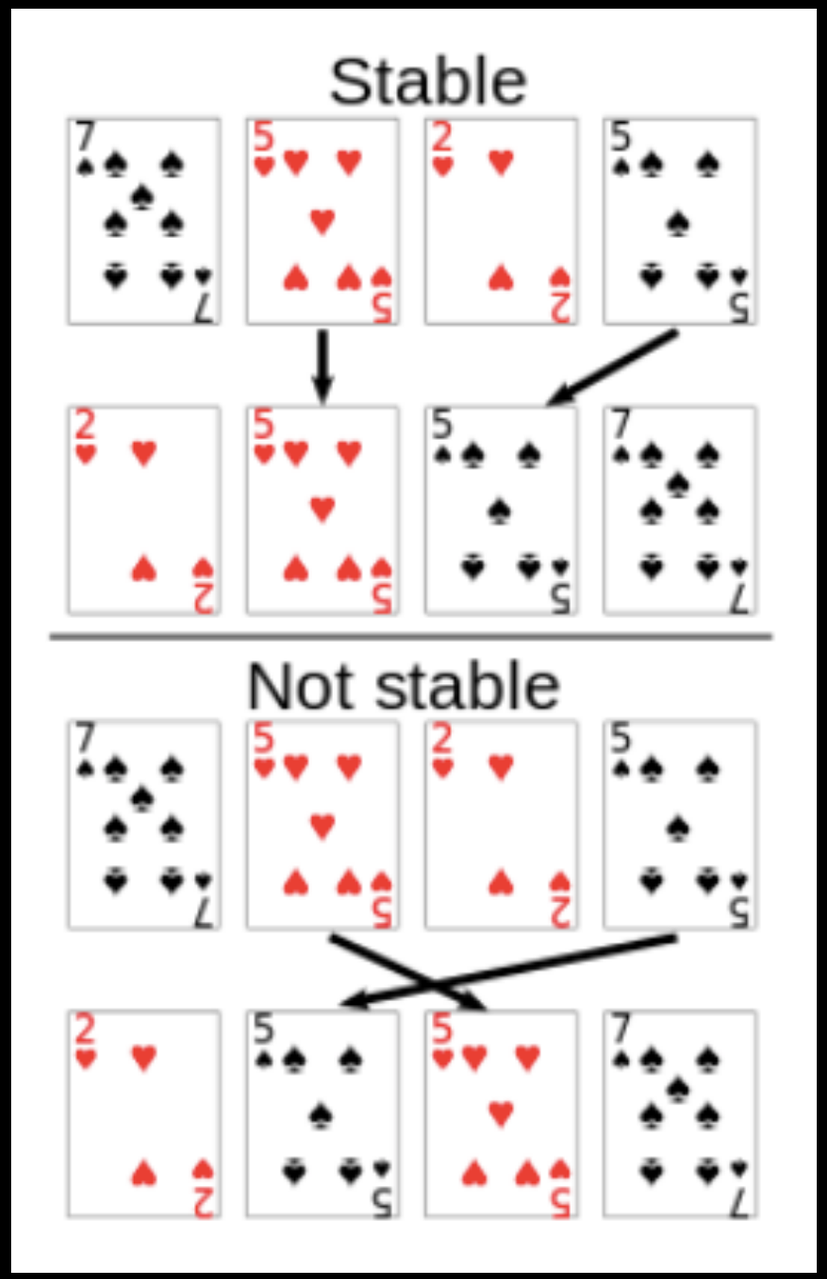
```
function QuickSort(arr, left = 0, right = arr.length - 1) {
  if(arr.length > 1) {
    let index = partition(arr, left, right)
    if(left < index - 1) {
      QuickSort(arr, left, index - 1)
    }
    if(index < right) {
      QuickSort(arr, index, right)
    }
  }
  return arr
}

function partition(arr, left, right) {
  let pivot = arr[Math.floor((right + left) / 2)
                ],
  while(left <= right) {
    while(arr[left] < pivot) {
      left++
    }
    while(arr[right] > pivot) {
      right--
    }
    if(left <= right) {
      [arr[left], arr[right]] = [arr[right], arr[left]]
      left++
      right--
    }
  }
  return left
}
```

31 LOC

```
1  Array.prototype.InsertionSort = function() {
2    /* your implementation here */
3  }
4
5  Array.prototype.MergeSort = function() {
6    /* your implementation here */
7  }
8
9  Array.prototype.QuickSort = function() {
10   /* your implementation here */
11 }
12
13 myArray.InsertionSort()
14 myArray.MergeSort()
15 myArray.QuickSort()
```


Most everyone's mad here.

Stability



A stable sort is one in which equivalent elements retain their relative positions after sorting

Array.prototype.sort()

IN THIS ARTICLE

The **`sort()`** method sorts the elements of an array *in place* and returns the array. The sort is not necessarily stable. The default sort order is according to string Unicode code points.

```
var fruit = ['cherries', 'apples', 'bananas'];
fruit.sort(); // ['apples', 'bananas', 'cherries']

var scores = [1, 10, 21, 2];
scores.sort(); // [1, 10, 2, 21]
// Note that 10 comes before 2,
// because '10' comes before '2' in Unicode code point order.

var things = ['word', 'Word', '1 Word', '2 Words'];
things.sort(); // ['1 Word', '2 Words', 'Word', 'word']
// In Unicode, numbers come before upper case letters,
// which come before lower case letters.
```

```
const people = [
  { name: 'Simon Blackburn', age: 32 },
  { name: 'Lewis Carroll', age: 42 },
  { name: 'Gertrude Chataway', age: 26 },
  { name: 'Charles Dodgson', age: 11 },
  { name: 'Martin Gardner', age: 11 },
  { name: 'Reginald Hargreaves', age: 54 },
  { name: 'Arthur Hughes', age: 6 },
  { name: 'Alice Liddell', age: 26 },
  { name: 'Skeffington Lutwidge', age: 26 },
  { name: 'George MacDonald', age: 17 },
  { name: 'Lorina Reeve', age: 26 },
  { name: 'Gabriel Rossetti', age: 26 },
  { name: 'John Tenniel', age: 32 },
  { name: 'Donald Thomas', age: 12 }
]
```

```
people.sort( (a,b) => {
    return a.age - b.age
})
```

V8

```
{ name: 'Arthur Hughes', age: 6 },
{ name: 'Charles Dodgson', age: 11 },
{ name: 'Martin Gardner', age: 11 },
{ name: 'Donald Thomas', age: 12 }
{ name: 'George MacDonald', age: 17 },
{ name: 'Gabriel Rossetti', age: 26 },
{ name: 'Alice Liddell', age: 26 },
{ name: 'Lorina Reeve', age: 26 },
{ name: 'Skeffington Lutwidge', age: 26 },
{ name: 'Gertrude Chataway', age: 26 },
{ name: 'John Tenniel', age: 32 },
{ name: 'Simon Blackburn', age: 32 },
{ name: 'Lewis Carroll', age: 42 },
{ name: 'Reginald Hargreaves', age: 54 }
```

```
{ name: 'Arthur Hughes', age: 6 },
{ name: 'Charles Dodgson', age: 11 },
{ name: 'Martin Gardner', age: 11 },
{ name: 'Donald Thomas', age: 12 }
{ name: 'George MacDonald', age: 17 },
{ name: 'Gabriel Rossetti', age: 26 },
{ name: 'Alice Liddell', age: 26 },
{ name: 'Lorina Reeve', age: 26 },
{ name: 'Skeffington Lutwidge', age: 26 },
{ name: 'Gertrude Chataway', age: 26 },
{ name: 'John Tenniel', age: 32 },
{ name: 'Simon Blackburn', age: 32 },
{ name: 'Lewis Carroll', age: 42 },
{ name: 'Reginald Hargreaves', age: 54 }
```

Performance

Let's do some benchmarking !

10 elements

Testing in Chrome 56.0.2924 / Mac OS X 10.12.3		
Test		Ops/sec
Array#sort	`arr.sort(numSort)`	1,967,829 ±4.39% 93% slower
Insertion Sort	`arr.InsertionSort()`	28,520,017 ±2.19% fastest
Merge Sort	`arr.MergeSort()`	109,299 ±5.77% 100% slower
Quick Sort	`arr.QuickSort()`	880,522 ±15.73% 97% slower

100 elements

Testing in Chrome 56.0.2924 / Mac OS X 10.12.3		
Test		**Ops/sec**
Array#sort	`arr.sort(numSort)`	127,999 ±4.21% 97% slower
Insertion Sort	`arr.InsertionSort()`	4,014,363 ±2.83% fastest
Merge Sort	`arr.MergeSort()`	10,559 ±2.76% 100% slower
Quick Sort	`arr.QuickSort()`	96,941 ±2.25% 98% slower

1,000 elements

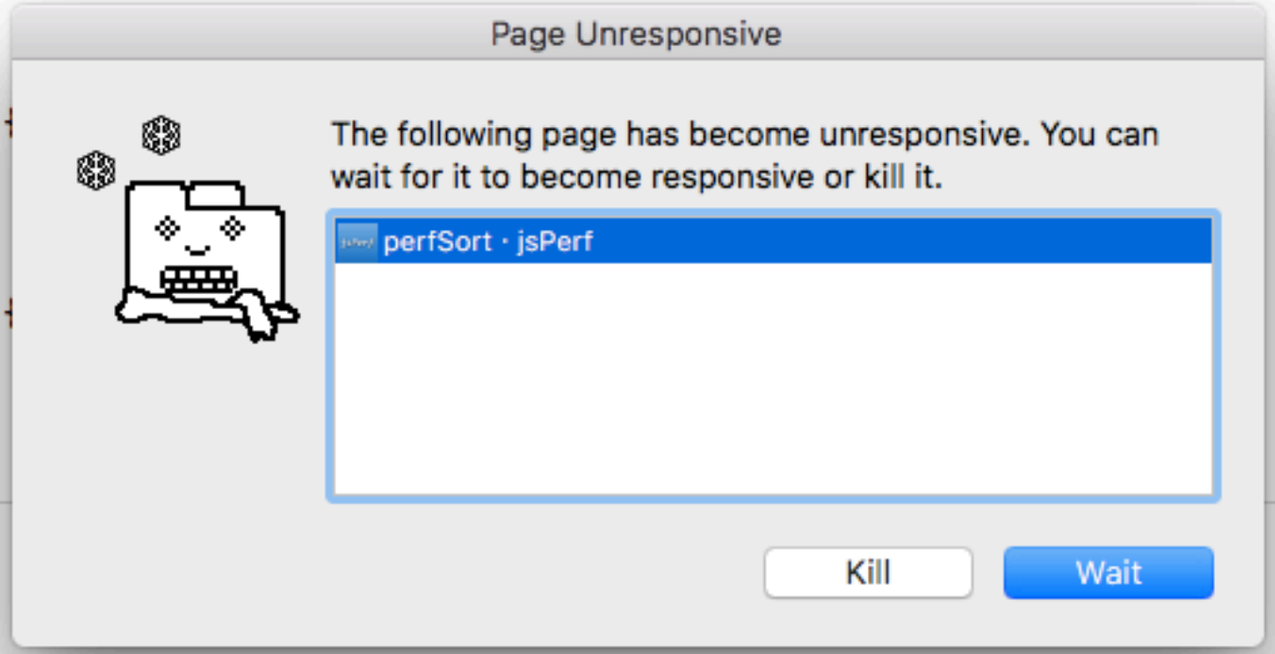
Testing in Chrome 56.0.2924 / Mac OS X 10.12.3		
Test		Ops/sec
Array#sort	`arr.sort(numSort)`	5,601 ±16.34% 98% slower
Insertion Sort	`arr.InsertionSort()`	314,407 ±8.32% fastest
Merge Sort	`arr.MergeSort()`	711 ±7.00% 100% slower
Quick Sort	`arr.QuickSort()`	6,316 ±6.74% 98% slower

100,000 elements

Testing in Chrome 56.0.2924 / Mac OS X 10.12.3		
Test		**Ops/sec**
Array#sort	`arr.sort(numSort)`	8.61 ±14.79% 78% slower
Insertion Sort	`arr.InsertionSort()`	0.19 ±10.41% 99% slower
Merge Sort	`arr.MergeSort()`	6.16 ±5.21% 83% slower
Quick Sort	`arr.QuickSort()`	35.75 ±3.74% fastest

1,000,000 elements

```
while(i <= j) {

  while(arr[i] < pivot) {
    i++
  }

  while(arr[j] > pivot) {
    j--
  }
```



Test runner

Insertion Sort × 1 (0 samples)

Stop running

Testing in Chrome 56.0.2924 / Mac OS X 10.12.3		
Test		**Ops/sec**
Array#sort	`arr.sort(numSort)`	completed
Insertion Sort	`arr.InsertionSort()`	pending…
Merge Sort	`arr.MergeSort()`	pending…
Quick Sort	`arr.QuickSort()`	pending…

You can edit these tests or add even more tests to this page by appending `/edit` to the URL.

1,000,000 elements

Testing in Chrome 56.0.2924 / Mac OS X 10.12.3		
Test		Ops/sec
Array#sort	`arr.sort(numSort)`	1.00 ±21.05% 56% slower
Merge Sort	`arr.MergeSort()`	0.45 ±11.46% 79% slower
Quick Sort	`arr.QuickSort()`	1.98 ±3.99% fastest

10,000,000 elements

Testing in Chrome 56.0.2924 / Mac OS X 10.12.3		
Test		Ops/sec
Array#sort	`arr.sort(numSort)`	0.08 ±34.87% 66% slower
Merge Sort	`arr.MergeSort()`	0.05 ±13.15% 76% slower
Quick Sort	`arr.QuickSort()`	0.18 ±5.01% fastest


FYDisneyMisfits|Mostamazing

V8

```
759  function QuickSort(a, from, to) {
760    var third_index = 0;
761    while (true) {
762      // Insertion sort is faster for
763      if (to - from <= 10) {
764        InsertionSort(a, from, to);
765        return;
766      }
767      if (to - from > 1000) {
768        third_index = GetThirdIndex(a, from, to);
769      } else {
770        third_index = from + ((to - from) >> 1);
771      }
772      // Find a pivot as the median of first, last and middl
773      var v0 = a[from];
774      var v1 = a[to - 1];
775      var v2 = a[third_index];
776      var c01 = comparefn(v0, v1);
777      if (c01 > 0) {
778        // v1 < v0, so swap them.
779        var tmp = v0;
780        v0 = v1;
781        v1 = tmp;
782      } // v0 <= v1.
783      var c02 = comparefn(v0, v2);
784      if (c02 >= 0) {
785        // v2 <= v0 <= v1.
786        var tmp = v0;
787        v0 = v2;
788        v2 = v1;
789        v1 = tmp;
790      } else {
791        // v0 <= v1 && v0 < v2
792        var c12 = comparefn(v1, v2);
793        if (c12 > 0) {
794          // v0 <= v2 < v1
```

NITRO

```
554  }
555
556  function mergeSort(array, valueCount,
557  {
558      var buffer = [ ];
559      buffer.length = valueCount;
560
561      var dst = buffer;
562      var src = array;
563      for (var width = 1; width < valueCount; width *= 2) {
564          for (var srcIndex = 0; srcIndex < valueCount; srcInde
565              merge(dst, src, srcIndex, valueCount, width, comp
566
567          var tmp = src;
568          src = dst;
569          dst = tmp;
570      }
571
572      if (src != array) {
573          for(var i = 0; i < valueCount; i++)
574              array[i] = src[i];
575      }
576  }
577
578  function bucketSort(array, dst, bucket, depth)
579  {
580      if (bucket.length < 32 || depth > 32) {
581          mergeSort(bucket, bucket.length, stringComparator);
582          for (var i = 0; i < bucket.length; ++i)
583              array[dst++] = bucket[i].value;
584          return dst;
585      }
586
587      var buckets = [ ];
588      for (var i = 0; i < bucket.length; ++i) {
589          var entry = bucket[i];
590          var string = entry.string;
591          if (string.length == depth) {
592              array[dst++] = entry.value;
593              continue;
594          }
595
596          var c = string.charCodeAt(depth);
597          if (!buckets[c])
598              buckets[c] = [ ];
599          buckets[c][buckets[c].length] = entry;
600      }
```

Javascript Inception?!

Self Hosting

Implementing parts of a language
in that very language itself

- Array methods all take a callback as an argument
- They iterate over the list and invoke the callback for every step
- Execution has to switch between compiled C++ and interpreted JS
- Context switch is expensive
- Staying in the same execution context and within the same language allows us to have additional nice things

JS Engine	Sort Algorithm	Self-Host ?
SpiderMonkey FIREFOX	Insertion Sort (for short arrays) Merge Sort	No
V8 CHROME	Insertion Sort (for short arrays) Quick Sort	Yes
Nitro SAFARI	Merge Sort	Yes
Chakra INTERNET EXPLORER	QuickSort	No

- Unneeded overhead due to the actual algorithm implementation in every engine
- Native implementations just do more in terms of error handling and features than the simple self-implementations

A MAD JS-PARTY

Disclaimer

The following content is designed to challenge and amuse programmers, not made to be suitable for practical use

non alphanumeric js

♠ vocabulary ♠

[ ] access arrays/strings and object properties

( ) call functions and avoid errors

{ } to get the string "[object Object]"

+ append strings, sum, and cast things to numbers

! cast things to booleans

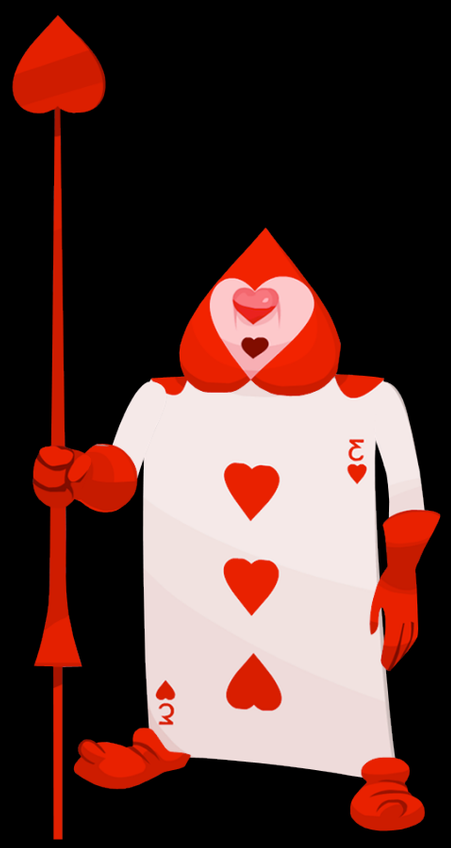

10
Careful! She's
stark raving mad!

Falsey	false, 0, "", null, undefined, NaN
Truthy	*All other values* "0", "false", [], {}

the basics

![[] = false

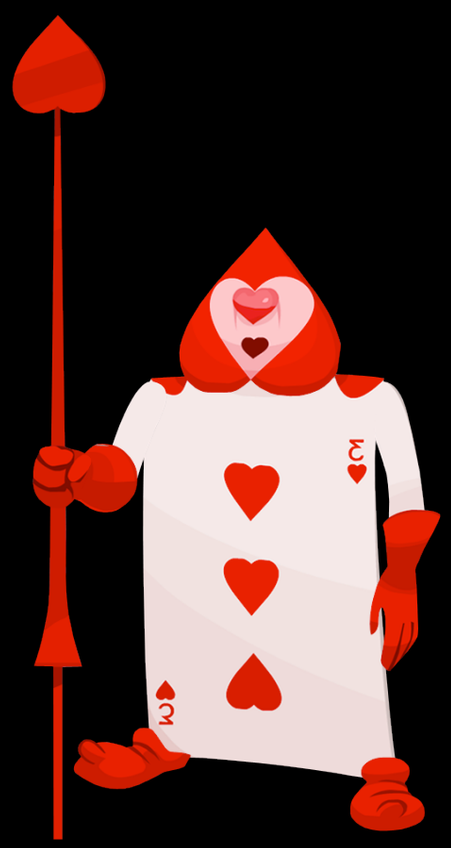
!![] = true



the basics

+![] = 0

+!![] = 1



0 = +![]

1 = +!![]

2 = !![]+!![]

3 = !![]+!![]+!![]

4 = !![]+!![]+!![]+!![]

5 = !![]+!![]+!![]+!![]+!![]

6 = !![]+!![]+!![]+!![]+!![]+!![]

7 = !![]+!![]+!![]+!![]+!![]+!![]+!![]

8 = !![]+!![]+!![]+!![]+!![]+!![]+!![]+!![]

9 = !![]+!![]+!![]+!![]+!![]+!![]+!![]+!![]+!![]

— ♠ strings ♠ —

```
![] + [] = "false"
!![] + [] = "true"
```

— ♠ strings ♠ —

```
![] + "" = "false"
!![] + "" = "true"
```

— ♠ strings ♠ —

```
+!![] + [] = "1"
!![]+!![] + [] = "2"
```

+(("1")+("2")+("3"))

```
+((+!![]+[])+(!![]+!![]+[])+(!![]+!![]+!![]+[]))
```

[][[]] = undefined

+{} = NaN

[]+{} = "[object Object]"

"false"[0] = "f"

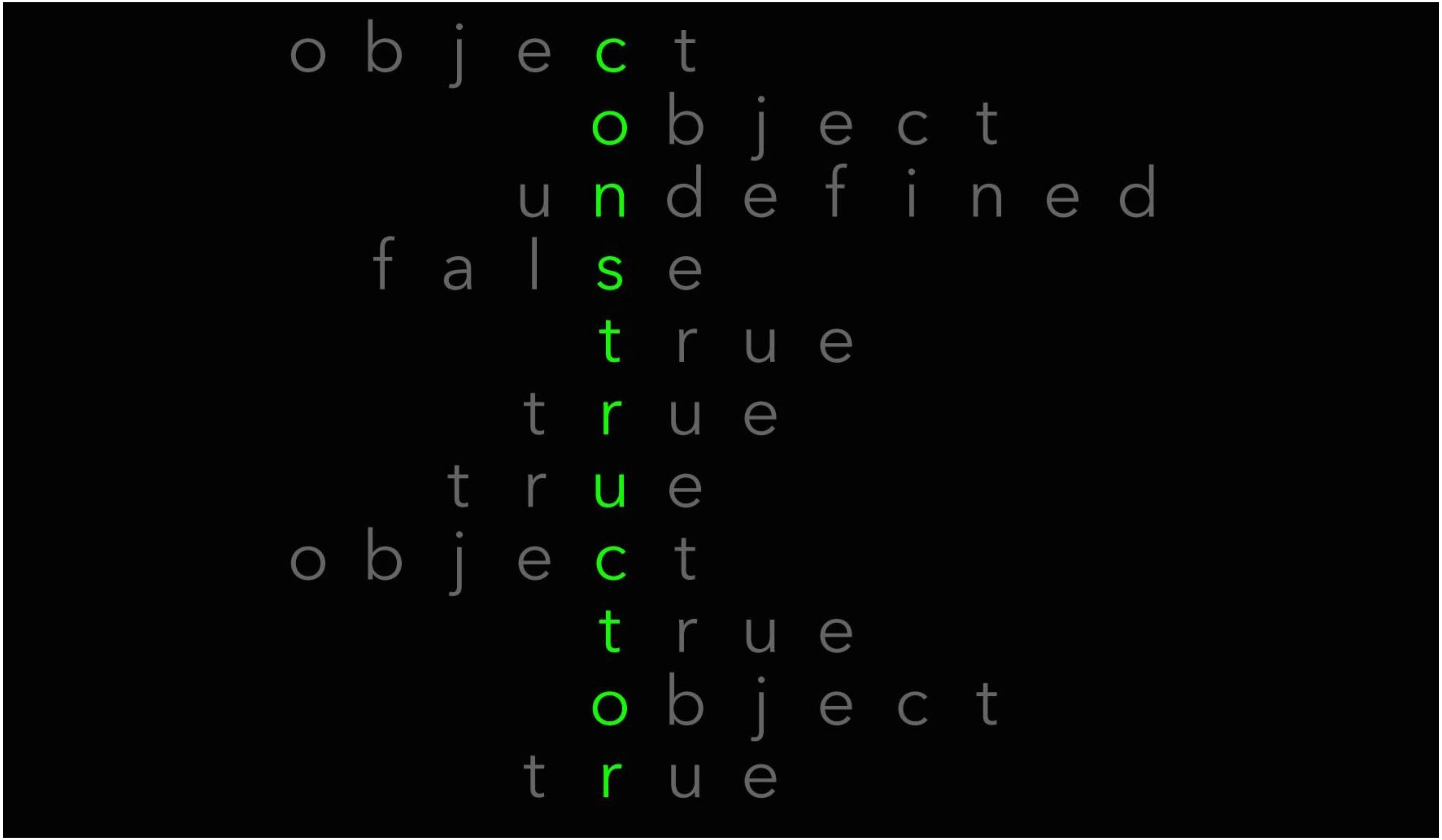
"undefined"[5] = "i"

"false"[2] = "l"

"true"[0] = "t"

"true"[3] = "e"

"true"[1] = "r"



object
object
undefined
false
true
true
true
object
true
object
true

call, concat, constructor, join, slice, sort, filter…

[]["filter"]["constructor"]("alert('1')")()

[]["filter"]["constructor"]("alert('1')")()

[]["filter"]["constructor"]("alert('1')")()

```
> []["filter"]
<· function filter() { [native code] }
```

```
function["constructor"]( "alert('1')" )()
```

```
function["constructor"]( "alert('1')" )()
```

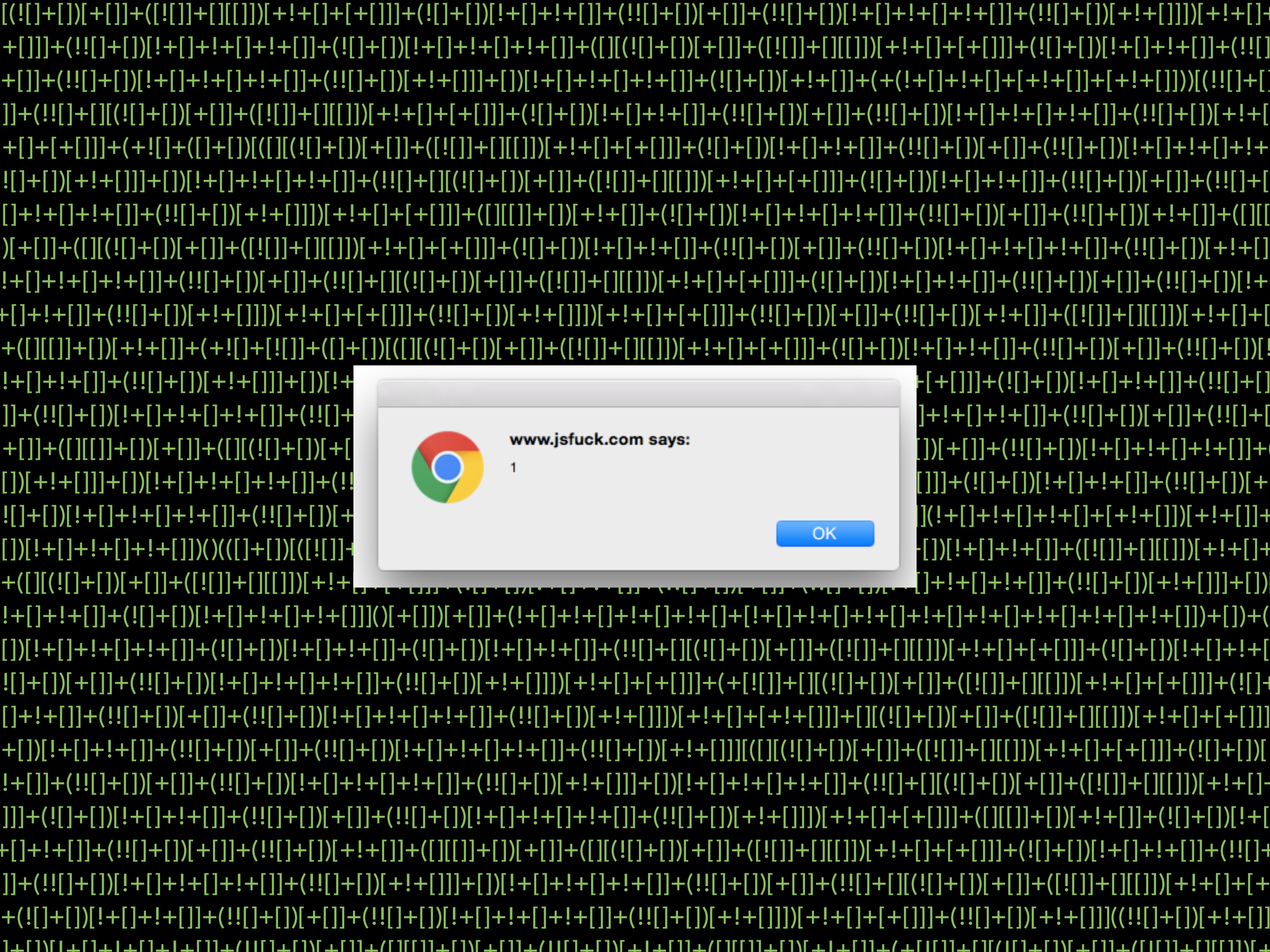
```
> []["filter"]["constructor"]
<· function Function() { [native code] }
```

```
Function( "alert('1')" )()
```

```
> [][“filter”][“constructor”]( “alert('1')” )
<· function anonymous() {
  alert('1')
  }
```

(function(){alert('1')})()

```
(function(){alert('1')})()
```


www.jsfuck.com says:
1
OK


Oh dear.
Everything is so confusing.

Hieroglyphy demo

Insert the script

```
alert(1)
```

Your output

```
[][(![]+[])[!+[]+!![]+!![]]+([]+{})[+!![]]+(!![]+[])[+!![]]+(!![]+
[])[+[]]][([]+{})[!+[]+!![]+!![]+!![]+!![]]+([]+{})[+!![]]+([][[]]+
```

Convert number | Convert script | Convert string

Y 5 submit Tweet

Home Blog Github CV email

by Patricio Palladino
http://patriciopalladino.com/files/hieroglyphy/

JSF*ck demo

encode JavaScript with only 6 letters - []()!+

Enter any JavaScript source:

```
alert("Hello, JavaScript")
```

Target: Firefox,Chrome,Safari,Opera

```
(+□)[(□[(!□+□)[+□]+([!□]+□[□])[+!+□+[+□]]+(!□+□)[!+□+!+□]+(!+□+□)[+□]+(!+□+□)[!+
□+!+□+!+□]+(!+□+□)[+!+□]]+□)[!+□+!+□+!+□]+(!+□+□[(!□+□)[+□]+([!□]+□[□])[+!+□+[+□]]+
(!□+□)[!+□+!+□]+(!+□+□)[+□]+(!+□+□)[!+□+!+□+!+□]+(!+□+□)[+!+□]])[+!+□+[+□]]+(□[□]+□)[+!+
□]+(!□+□)[!+□+!+□+!+□]+(!!□+□)[+□]+(!!□+□)[+!+□]+(□[□]+□)[+□]+(□[(!□+□)[+□]+([!□]+
□[□])[+!+□+[+□]]+(!□+□)[!+□+!+□]+(!+□+□)[+□]+(!+□+□)[!+□+!+□+!+□]+(!+□+□)[+!+□]]+□)[!+
□+!+□+!+□]+(!!□+□)[+□]+(!+□+□[(!□+□)[+□]+([!□]+□[□])[+!+□+[+□]]+(!□+□)[!+□+!+□]+(!+
□+□)[+□]+(!+□+□)[!+□+!+□+!+□]+(!+□+□)[+!+□]])[+!+□+[+□]]+(!!□+□)[+!+□]][(□[(!□+□)[+□]+
([!□]+□[□])[+!+□+[+□]]+(!□+□)[!+□+!+□]+(!+□+□)[+□]+(!+□+□)[!+□+!+□+!+□]+(!+□+□)[+!+□]]+
□)[!+□+!+□+!+□]+(!+□+□[(!□+□)[+□]+([!□]+□[□])[+!+□+[+□]]+(!□+□)[!+□+!+□]+(!+□+□)[+□]+
(!+□+□)[!+□+!+□+!+□]+(!+□+□)[+!+□]])[+!+□+[+□]]+(□[□]+□)[+!+□]+(!□+□)[!+□+!+□+!+□]+(!!□+
□)[+□]+(!!□+□)[+!+□]+(□[□]+□)[+□]+(□[(!□+□)[+□]+([!□]+□[□])[+!+□+[+□]]+(!□+□)[!+□+!+
□]+(!+□+□)[+□]+(!+□+□)[!+□+!+□+!+□]+(!+□+□)[+!+□]]+□)[!+□+!+□+!+□]+(!!□+□)[+□]+(!+□+
□[(!□+□)[+□]+([!□]+□[□])[+!+□+[+□]]+(!□+□)[!+□+!+□]+(!+□+□)[+□]+(!+□+□)[!+□+!+□+!+
□]+(!+□+□)[+!+□]])[+!+□+[+□]]+(!!□+□)[+!+□]]((!□+□)[+!+□]+(!□+□)[!+□+!+□]+(!+□+□)[!+□+!+
□+!+□]+(!!□+□)[+!+□]+(!!□+□)[+□]+(□[(□[(!□+□)[+□]+([!□]+□[□])[+!+□+[+□]]+(!□+□)[!+□+!+
□]+(!+□+□)[+□]+(!+□+□)[!+□+!+□+!+□]+(!+□+□)[+!+□]]+□)[!+□+!+□+!+□]+(!+□+□[(!□+□)[+□]+
([!□]+□[□])[+!+□+[+□]]+(!□+□)[!+□+!+□]+(!+□+□)[+□]+(!+□+□)[!+□+!+□+!+□]+(!+□+□)[+!+□]])
[+!+□+[+□]]+(□[□]+□)[+!+□]+(!□+□)[!+□+!+□+!+□]+(!!□+□)[+□]+(!!□+□)[+!+□]+(□[□]+□)[+□]+
```

40228 letters

[eval] [Permalink]

Thanks : Kenji Aiko

[utf-8.jp/]

by Kenji Aiko

http://utf-8.jp/public/jsfuck.html

6chars.js

Convert any Javascript code to ([!+]) chars (currently working only on Chrome)

Enter your code here (5000 chars max)

```
var name = prompt("What is your name ?");
alert("Hello "+name);
```

Convert

Result:

6chars.js

http://syllab.fr/projets/experiments/sixcharsjs/

JScrewIt

JScrewIt converts plain JavaScript into JSFuck code, which uses only six different characters to write and run any code: `! ( ) + [ ]`

Use the form below to convert your own script. Uncheck `Executable code` to obtain a plain string.

Input

```
alert(1)
```

Compatibility: All supported engines ☑ Executable code

Output

```
[][(![]+[])[+[]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!![]+!![]]+(!![]+[]
)[+[]]+(!![]+[])[!![]+!![]+!![]]+(!![]+[])[+!![]]][(+(+!![]+[+([][(![]+[]
)[+[]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!![]+!![]]+(!![]+[])[+[]]+(!!
[]+[])[!![]+!![]+!![]]+(!![]+[])[+!![]]]+[])[+[]]])+[!![]]+[][(![]+[])[+[
]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!![]+!![]]+(!![]+[])[+[]]+(!![]+[
])[!![]+!![]+!![]]+(!![]+[])[+!![]]])[+!![]+[+[]]]+(!!(+([][(![]+[])[+[]]
+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!![]+!![]]+(!![]+[])[+[]]+(!![]+[])
[!![]+!![]+!![]]+(!![]+[])[+!![]]]+[])[+[]]+!![])+[][(![]+[])[+[]]+([![]]
+[][[]])[+!![]+[+[]]]+(![]+[])[!![]+!![]]+(!![]+[])[+[]]+(!![]+[])[!![]+!
![]+!![]]+(!![]+[])[+!![]]])[+!![]+[+!![]]]+([][[]]+[])[+!![]]+(![]+[])[!
```

2084 chars Run this

Links

Documentation and source code on GitHub | JSFuck introduction and alternatives on Esolang | jsfuck.com | jQuery Screwed

by Francesco Trotta
http://jscrew.it/

JSFuck

JSFuck is an esoteric and educational programming style based on the atomic parts of JavaScript. It uses only six different characters to write and execute code.

It does not depend on a browser, so you can even run it on Node.js.

Use the form below to convert your own script. Uncheck "eval source" to get back a plain string.

alert(1) Encode ☑ Eval Source

```
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!!
[]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]]([][(![]+[])
[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+
(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!!
[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+
[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[+!+[]+
[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+
(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])
[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+
[]+!+[]+!+[]]+(!![]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+
```

1227 chars Run This

Links

- Share on Twitter, Google+
- View source on GitHub
- Follow @aemkei (Martin Kleppe)
- Original discussion at Sla.ckers.org

Alternatives

by Martin Kleppe
http://www.jsfuck.com/

jQuery *Screwed*

jQuery JavaScript library made using only six different characters: `!` `(` `)` `+` `[` `]`

Usage

You can use jQuery Screwed anywhere in your code like the regular jQuery.

```
<script src="jquery-2.2.1.screwed.js"></script>
```

```
<script>
$(function(){
    alert("You are using jQuery Screwed!");
});
</script>
```

See the section Browser Support for information about supported browsers.

Creation

jQuery Screwed was created with JScrewIt. Below are the steps to recreate it manually.

- Install Node.js if you haven't done so yet.
- Install JScrewIt: run `npm install -g jscrewit`
- Download jQuery - current stable version is 2.2.1: http://code.jquery.com/jquery-2.2.1.min.js
- Replace path names as appropriate and run

https://github.com/fasttime/jquery-screwed

https://raw.githubusercont

Claudia

https://raw.githubusercontent.com/fasttime/jquery-screwed/master/jquery-2.2.1.screwed.js


```
[][(![]+[])[+[]]+([![]]+[][[]])[+!!]+[+[]]]+(![]+[])[!!]+!![]+(!![]+[])[+[]]+(!![]+[])[!!]+!![]+!![]+(!![]+[])[+!![]]][(+(+!![]+[+(([![]+[])[+[]]+([![]]+[][[]])[+!![]+[+[]]]+
(![]+[])[!!]+!![]+(!![]+[])[+[]]+(!![]+[])[!!]+!![]+!![]+(!![]+[])[+!![]]+[])[+[]]])+!![]+[(![]+[])[+[]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!!]+!![]+(!![]+[])[+[]]+(!!
[]+[])[!!]+!![]+!!]+(!![]+[])[+[]]+(!!(+([][(![]+[])[+[]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!!]+!![]+(!![]+[])[+[]]+(!![]+[])[!!]+!![]+!!]+(!![]+[])[+!!
[]]]+[])[+[]]+!!])+[][(![]+[])[+[]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!!]+!![]+(!![]+[])[+[]]+(!![]+[])[!!]+!![]+!![]+(!![]+[])[+!!]])[+!![]+[+!!]]]+([][[]]+[])[+!!]]+(![
[]+[])[!!]+!![]+!!]+(!!(+([][(![]+[])[+[]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!!]+!![]+(!![]+[])[+[]]+(!![]+[])[!!]+!![]+!!]+(!![]+[])[+!!]])[+!![]+[+[]]]+(![]+[])[!!]+!!
[]+!!]+(!!(+([][(![]+[])[+[]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!!]+!![]+(!![]+[])[+[]]+(!![]+[])[!!]+!![]+!!]+(!![]+[])[+!!]])[+!![]+[+
[]]]+(!!(+([][(![]+[])[+[]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!!]+!![]+(!![]+[])[+[]]+(!![]+[])[!!]+!![]+!!]+(!![]+[])[+!!]])[+!![]+[+!!]]]+([][[]]+[])[+!!]])+[][(![]+[])[+[]]+
([![]]+[][[]])[+!!]+[+[]]+(![]+[])[!!]+!!]+(!!(+([][(![]+[])[+[]]+([![]]+[][[]])[+!![]+[+[]]]+(![]+[])[!!]+!![]+(!![]+[])[+[]]+(!![]+[])[!!]+!![]+!!]+(!![]+[])[+!!]])[+!![]+[+!!]]]+(![]+[])[!!]+!!]+(+
[])+[][[]]+![]+(+[])+![]+[][[]]+!!]+![]+!!]+![]+!!]+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+!!]+![]+[][[]]+![]+(+!!])+![]+(+[])+(+!!])+![]+(+[![]])+![]+(+[![]])+(+[])+![]+!!
[]+(+[![]])+![]+(+[])+(+[])+![]+(+[])+(+!!])+![]+(+!!])+(+[![]])+![]+![]+[][[]]+!!]+![]+!!]+![]+!!]+(+[![]])+![]+(!!])+!!])+(+[])+![]+[][[]]+(+[])+![]+(+[])+[][[]]+!!]+(+!!])+!!]+
(+!!])+(+[])+![]+![]+(+[])+(+[])+![]+(![]+[])[+[]]+![]+(!!]+[])[+[]]+![]+(+!!])+(+!!])+![]+(+[])+(+[])+![]+[][[]]+(+[])+![]+(+!!])+!!])+![]+![]+![]+(+[![]])+!!]+![
[]+(+[![]])+!!]+![]+!!]+!!]+(+[![]])+![]+(+[])+(+[])+![]+(+[])+(+!!])+![]+(+!!])+(+[![]])+![]+![]+[][[]]+!!]+![]+!!]+(+[])+![]+(!!]+!!])+![]+(!!]+!!])+![]+!!]+![
[]+!!]+!!]+(+[])+![]+(+!!])+(+[])+![]+![]+(+[])+(+[])+![]+(![]+[])[+[]]+![]+(!!]+[])[+[]]+![]+(+!!])+(+!!])+![]+(+[])+(+[])+![]+[][[]]+(+[])+![]+(+[])+[][[]]+![]+(+!!])+!!]+![
[]+![]+!!]+[][[]]+![]+![]+(!!]+[])[+!!]]+![]+(+!!])+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+![]+!!]+![]+!!]+(+!!])+![]+([][[]]+[])[+[]]+![]+(+!!])+(+!!])+![]+(+[])+(+[])+![]+[]
[[]]+(+[])+![]+(+[])+[][[]]+![]+(+!!])+!!]+![]+![]+!!]+[][[]]+![]+![]+(!!]+[])[+!!]]+![]+(+!!])+(+[])+![]+(+[])+(+[])+![]+(+[])+!!]+![]+!!]+![]+!!]+(+!!])+![]+(!!]+!!])+![]+!
[]+[][[]]+![]+!!]+![]+[][[]]+![]+[][[]]+(+[])+![]+(+[])+(+[])+![]+[][[]]+!!]+![]+(+[])+[][[]]+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+(+[])+!!]+![]+!
[]+!!]+![]+[][[]]+![]+(+!!])+![]+(+[])+!!]+!!]+![]+(!!]+!!]+!!])+![]+(+[![]])+![]+(+!!])+[][[]]+![]+(![]+[])[+[]]+![]+(+[])+[][[]]+![]+(+[])+![]+[][[]]+!!]+![]+!!]+![]+!!
[]+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+!!]+![]+[][[]]+![]+(+[![]])+![]+(+[![]])+(+[])+![]+!!]+(+[])+![]+(![]+[])[+[]]+![]+!!]+!!]+![]+(+[])+!!]+![]+(+[])+!!]+![]+!
[[]]+![]+[][[]]+(+[])+![]+(+[])+(+[])+![]+[][[]]+!!]+![]+(+[])+[][[]]+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+[][[]]+[][[]]+![]+(+[![]])+(+[])+![]+(
[])+![]+!!]+(!!]+!!])+![]+!!]+(!!]+[])[+[]]+![]+(+[])+!!]+![]+(+[])+!!]+!!]+![]+(!!]+!!]+!!])+![]+(+[![]])+![]+(+!!])+[][[]]+![]+(![]+[])[+[]]+![]+(+[])+[][[]]+![]+(+[])+![]+[][[]]+!!]+![]+!!]+!!]+![]+!!
[]+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+!!]+![]+[][[]]+![]+(+[![]])+![]+(+[![]])+(+[])+![]+!!]+(+[])+![]+(+[])+(+[])+![]+(+[])+!!]+![]+!!]+!!]+![]+(+[])+!!]+![]+(+[])+!!]+![]+!
[]+![]+[][[]]+(+[])+![]+(+[])+(+[])+![]+[][[]]+!!]+![]+(+[])+[][[]]+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+[][[]]+[][[]]+![]+(+[![]])+(+[])+![]+(+
[])+![]+!!]+(!!]+!!])+![]+!!]+(!!]+[])[+[]]+![]+(+[])+!!]+![]+(+[])+!!]+!!]+![]+(!!]+!!]+!!])+![]+(+[![]])+![]+(+!!])+[][[]]+![]+(![]+[])[+[]]+![]+(+[])+[][[]]+![]+(+[])+![]+[][[]]+!!]+![]+!!]+!!]+![]+!!
[]+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+!!]+![]+[][[]]+![]+(+[![]])+![]+(+[![]])+(+[])+![]+!!]+(+[])+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+[][[]]+[][[]]+![]+(+[![]])+(+[])+![]+(+
[])+![]+!!]+(!!]+!!])+![]+!!]+(!!]+[])[+[]]+![]+(+[])+!!]+![]+(+[])+!!]+!!]+![]+(!!]+!!]+!!])+![]+(+[![]])+![]+(+!!])+[][[]]+![]+(![]+[])[+[]]+![]+(+[])+[][[]]+![]+(+[])+![]+[][[]]+!!]+![]+!!]+!!]+![]+!!
[]+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+!!]+![]+[][[]]+![]+(+[![]])+![]+(+[![]])+(+[])+![]+!!]+(+[])+![]+(+[])+(+[])+![]+(+[])+!!]+![]+!!]+!!]+![]+(+[])+!!]+![]+(+[])+!!]+![]+!
[[]]+![]+[][[]]+(+[])+![]+(+[])+(+[])+![]+[][[]]+!!]+![]+(+[])+[][[]]+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+[][[]]+[][[]]+![]+(+[![]])+(+[])+![]+(+
[])+![]+!!]+(!!]+!!])+![]+!!]+(!!]+[])[+[]]+![]+(+[])+!!]+![]+(+[])+!!]+!!]+![]+(!!]+!!]+!!])+![]+(+[![]])+![]+(+!!])+[][[]]+![]+(![]+[])[+[]]+![]+(+[])+[][[]]+![]+(+[])+![]+[][[]]+!!]+![]+!!]+!!]+![]+!!
[]+![]+[][[]]+(+[])+![]+!!]+![]+(+[![]])+![]+(+[])+!!]+!!]+![]+(+[])+!!]+!!]+![]+(+[])+[][[]]+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+[][[]]+![]+!!]+![]+(+
[])+(+[])+![]+(!!]+!!])+![]+!!]+!!]+!!]+(+!!])+![]+!!]+(![]+[])[+[]]+![]+(![]+[])[+!!]]+![]+(!!]+!!])+!!]+![]+!!]+(+!!])+![]+(!!]+!!])+!!]+![]+(+[])+[][[]]+![]+(+[])+[]
[[]]+!!]+![]+(+[![]])+(+[![]])+![]+(+[])+[][[]]+!!]+![]+(+[])+[][[]]+!!]+![]+(!!]+!!])+!!]+![]+(!!]+[])[+!!]]+![]+(+[![]])+[][[]]+(+[])+![]+(!!]+!!]+!!])+![]+[][[]]+(+[![]])+!
[]+(+[![]]+[])[+[]]+![]+[][[]]+(+!!])+![]+(![]+[])[+!!]]+![]+(!!]+!!])+![]+(+!!])+![]+(!!]+!!])+!!]+![]+(+[])+[][[]]+![]+(+[])+[][[]]+!!]+![]+[][[]]+(+[![]])+!
[]+(+[])+[][[]]+!!]+![]+[]+(+[])+[][[]]+!!]+![]+(!!]+!!])+![]+(!!]+[])[+[]]+![]+(+[])+!!]+![]+(+[])+!!]+![]+(+[![]])+![]+(+[![]])+(+[])+![]+(+[![]])+[][[]]+![]+[][[]]+![]+[]
(+[])+!!]+![]+(!!]+[])[+[]]+![]+[][[]]+!!]+![]+(!!]+!!])+![]+(![]+[])[+!!]]+![]+[][[]]+(+[![]])+![]+(+!!])+![]+[][[]]+(+[])+![]+(!!]+!!])+![]+[][[]]+![]+!!]+(+[])+![]+(+
[[]]+(+[])+![]+(+[])+(+[])+![]+[][[]]+!!]+![]+(+[])+[][[]]+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+!!])+![]+![]+(!!]+!!])+![]+[][[]]+!!]+![]+!!]+![]+(+!!
[])+![]+(+!!])+!!]+![]+!!]+!!]+(+[])+![]+[][[]]+!!]+![]+(+[])+![]+(+!!])+!!]+!!]+![]+(+!!])+![]+(+!!])+(+[![]])+![]+(!!]+!!])+![]+!!]+(+[])+![]+!!]+![]+[][[]]+![]+!!]+![]+(+
[]+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+!!]+![]+[][[]]+![]+(+[![]])+(+[])+![]+!!]+(+[])+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+[][[]]+![]+(+[])+!!]+![]+(+[])+!!]+![]+!
[]+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+!!]+![]+[][[]]+![]+(+[![]])+![]+(+[![]])+(+[])+![]+!!]+(+[])+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+[][[]]+![]+(+[])+!!]+![]+(+[])+!!]+![]+!
[]+(+[])+![]+(+!!])+(+[])+![]+(+!!])+(+[![]])+![]+(+[])+[][[]]+!!]+![]+(+[])+[][[]]+!!]+![]+(!!]+!!])+!!]+![]+(!!]+[])[+!!]]+![]+(!!]+!!])+!!]+![]+(+[])+[][[]]+![]+(+[])+[]
[[]]+!!]+![]+(+[![]])+(+[![]])+![]+(+[])+[][[]]+!!]+![]+(+[])+[][[]]+!!]+![]+(!!]+!!])+!!]+![]+(!!]+[])[+!!]]+![]+[][[]]+(+[])+![]+(!!]+!!]+!!])+![]+[][[]]+(+[![]])+!
[]+(+[![]]+[])[+[]]+![]+[][[]]+(+!!])+![]+(![]+[])[+!!]]+![]+(!!]+!!])+![]+(+!!])+![]+(!!]+!!])+!!]+![]+(+[])+[][[]]+![]+(+[])+[][[]]+!!]+![]+(+[![]])+(+[![]])+!
[]+(+[])+[][[]]+!!]+![]+(+[])+[][[]]+!!]+![]+(!!]+!!])+![]+(+[])+[][[]]+!!]+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+(+[![]])+(+[])+![]+(+[![]])+[][[]]+![]+[][[]]+![]+
(+[])+!!]+![]+(!!]+[])[+[]]+![]+[][[]]+!!]+![]+(!!]+!!])+![]+(![]+[])[+!!]]+![]+(+[![]])+![]+(+!!])+![]+[][[]]+(+[])+![]+(!!]+!!])+![]+(+!!])+![]+[][[]]+(+[![]])+!
[]+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+!!]+![]+[][[]]+![]+(+[![]])+(+[])+![]+!!]+(+[])+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+[][[]]+![]+(+[])+!!]+![]+(+[])+!!]+![]+!
[]+(+[])+![]+(+!!])+(+[])+![]+(+!!])+(+[![]])+![]+(+[])+[][[]]+!!]+![]+(+[])+[][[]]+!!]+![]+(!!]+!!])+!!]+![]+(!!]+[])[+!!]]+![]+(!!]+!!])+!!]+![]+(+[])+[][[]]+![]+(+[])+[]
[[]]+!!]+![]+(+[![]])+(+[![]])+![]+(+[])+[][[]]+!!]+![]+(+[])+[][[]]+!!]+![]+(!!]+!!])+!!]+![]+(!!]+[])[+!!]]+![]+[][[]]+(+[])+![]+(!!]+!!]+!!])+![]+[][[]]+(+[![]])+!
[]+(+[![]]+[])[+[]]+![]+[][[]]+(+!!])+![]+(![]+[])[+!!]]+![]+(!!]+!!])+![]+(+!!])+![]+(!!]+!!])+!!]+![]+(+[])+[][[]]+![]+(+[])+[][[]]+!!]+![]+(+[![]])+(+[![]])+!
[]+(+[])+[][[]]+!!]+![]+(+[])+[][[]]+!!]+![]+(!!]+!!])+!!]+![]+(!!]+[])[+!!]]+![]+(!!]+!!])+!!]+![]+(+[])+[][[]]+![]+(+[])+[][[]]+!!]+![]+[][[]]+(+[![]])+!
[]+(+[])+![]+(+[])+(+[])+![]+(+[])+![]+!!]+!!]+![]+[][[]]+![]+(+[![]])+(+[])+![]+!!]+(+[])+![]+(+!!])+(+!!])+![]+![]+(+[])+![]+!!]+![]+(+[![]])+![]+!!]+![]+[][[]]+![]+(+[])+!!]+![]+(+[])+!!]+![]+!
[]+(+[])+![]+(+!!])+(+[])+![]+(+!!])+(+[![]])+![]+(+[])+[][[]]+!!]+![]+(+[])+[][[]]+!!]+![]+(!!]+!!])+!!]+![]+(!!]+[])[+!!]]+![]+(!!]+!!])+!!]+![]+(+[])+[][[]]+![]+(+[])+[]
[[]]+!!]+![]+(+[![]])+(+[![]])+![]+(+[])+[][[]]+!!]+![]+(+[])+[][[]]+!!]+![]+(!!]+!!])+!!]+![]+(!!]+[])[+!!]]+![]+[][[]]+(+[])+![]+(!!]+!!]+!!])+![]+[][[]]+(+[![]])+!
[]+(+[![]])+![]+!!]+![]+!!]+(+!!])+![]+(+[])+(+[![]])+![]+(+!!])+![]+(+!!])+(+[])+![]+(!!]+!!])+![]+!!]+(+[])+![]+!!]+![]+(+!!])+![]+!!]+![]+(+[])+(+[])+![]+(+!!])+(+!!
[])+![]+!!]+!!]+(+!!])+![]+!!]+(+[])+(+[])+![]+!!]+!!]+(+!!])+![]+(+!!])+![]+[][[]]+!!]+!!]+![]+(!!]+!!])+![]+!!]+!!]+(+!!])+![]+!!]+(![]+[])[+[]]+![]+(+[])+!!]+![]+!!]+![]+[]
```

Martin Kleppe
@aemkei

Following

Achievement unlocked: blog.checkpoint.com /2016/02/02/eba … - But don't blame me or my jsfuck.com for hacking @eBay!

About JSFuck

Written by Martin Kleppe, this technique, which uses non-alphanumeric characters, allows the attacker to bypass IDSs, IPSs and WAFs payload sanitation. Only 6 different characters are used: [](]!+

The following basic vocabulary helps us write anything

RETWEETS 9 LIKES 22

11:53 AM - 2 Feb 2016

eBay Platform Exposed to Severe Vulnerability

by Oded Vanunu posted 2016/02/02

Vulnerability Discovery

Check Point security researcher Roman Zaikin recently discovered a vulnerability that allows attackers to execute malicious code on eBay users' devices, using a non-standard technique called "JSF**k." This vulnerability could allow cyber criminals to use eBay as a phishing and malware distribution platform.

To exploit this vulnerability, all an attacker needs to do is create an online eBay store. In his store details, he posts a maliciously crafted item description. eBay prevents users from including scripts or iFrames by filtering out those HTML tags. However, by using **JSF**k**, the attacker is able to create a code that will load an additional JS code from his server. This allows the attacker to insert a remote controllable JavaScript that he can adjust to, for example, create multiple payloads for a different user agent.

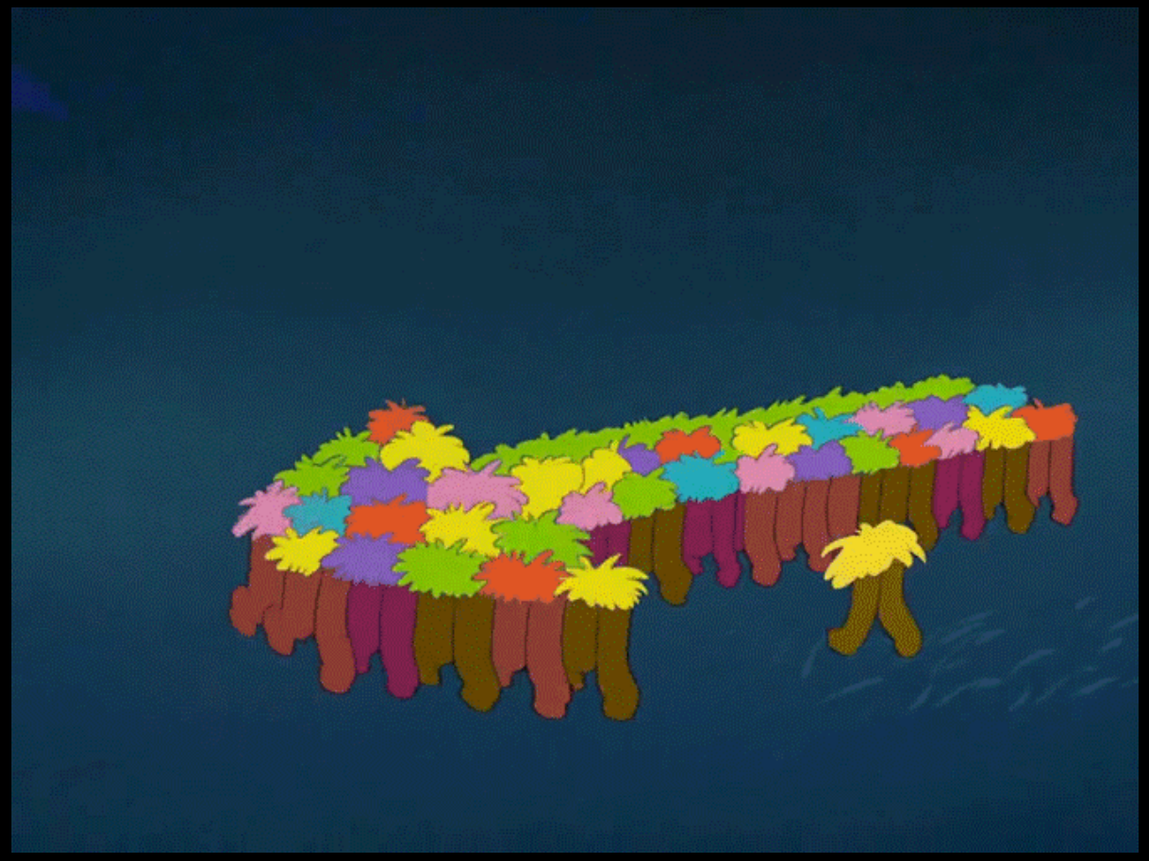
eBay performs simple verification, but only strips alpha-numeric characters from inside the script tags. The **JSF**k** technique allows the attackers to get around this protection by using a very limited and reduced number of characters.

http://blog.checkpoint.com/2016/02/02/ebay-platform-exposed-to-severe-vulnerability/

*"eBay **prevents** users from including **scripts** or **iFrames** by **filtering** out those **HTML tags**. However, by using **JSF*ck** the attacker can insert a remote controllable JS that can, for example, **create multiple payloads for a different user agent**"*


VINTAGEGAL
It would be so nice if something would make sense for a change.

FOLLOWING THE MOMERATHS



```
function foo () {
    setTimeout (function () {
      console.log('name: ', this.name)
    }, 100)
}

// name:
foo.call( { name: "alice"} )
```

```
function foo () {
    const self = this
    setTimeout (function () {
      console.log('name: ', self.name)
    }, 100)
}

// name: alice
foo.call( { name: "alice"} )
```

Arrow functions don’t have a this at all

this

arguments

super (ES6)

new.target (ES6)

```
function foo () {
    setTimeout ( () => {
      console.log('name: ', this.name)
    }, 100)
}

// name: alice
foo.call( { name: "alice"} )
```

```
function foo() {
  return () => {
    console.log('id:', this.id)
  }
}

const arrowfn = foo.call ( { id: 42 } )

setTimeout (arrowfn.bind ( { id:
100 } ), 100) // id: 42
```

WAS IT ALL A DREAM?

not really ...

but, why should I care?

knowledge is power

pushing the limits

breaking rules

playing is learning

power of the language

THE BEST JAVASCRIPT DEVELOPERS ARE THOSE WHO OBSESS ABOUT LANGUAGE, WHO EXPLORE AND PLAY WITH IT EVERYDAY AND IN DOING SO DEVELOP THEIR OWN IDIOMS AND THEIR OWN VOICE.

Angus Croll
on 'If Hemingway wrote Javascript'

СПАСИБО !

CLAUDIA HERNÁNDEZ

@koste4

claudia.hernandez@dailymotion.com

dailymotion