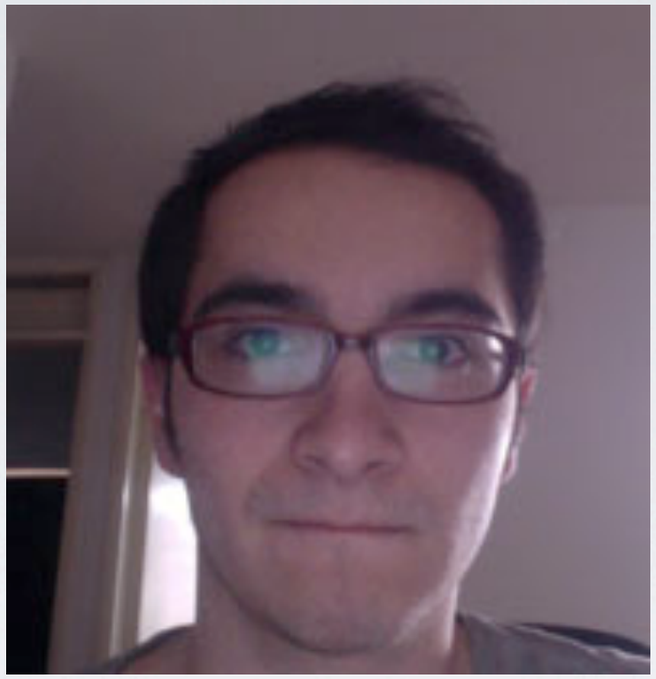
INTRODUCCIÓN A LA PROGRAMACIÓN Y JAVASCRIPT

David Avellaneda
@davsket
#mejorandoweb

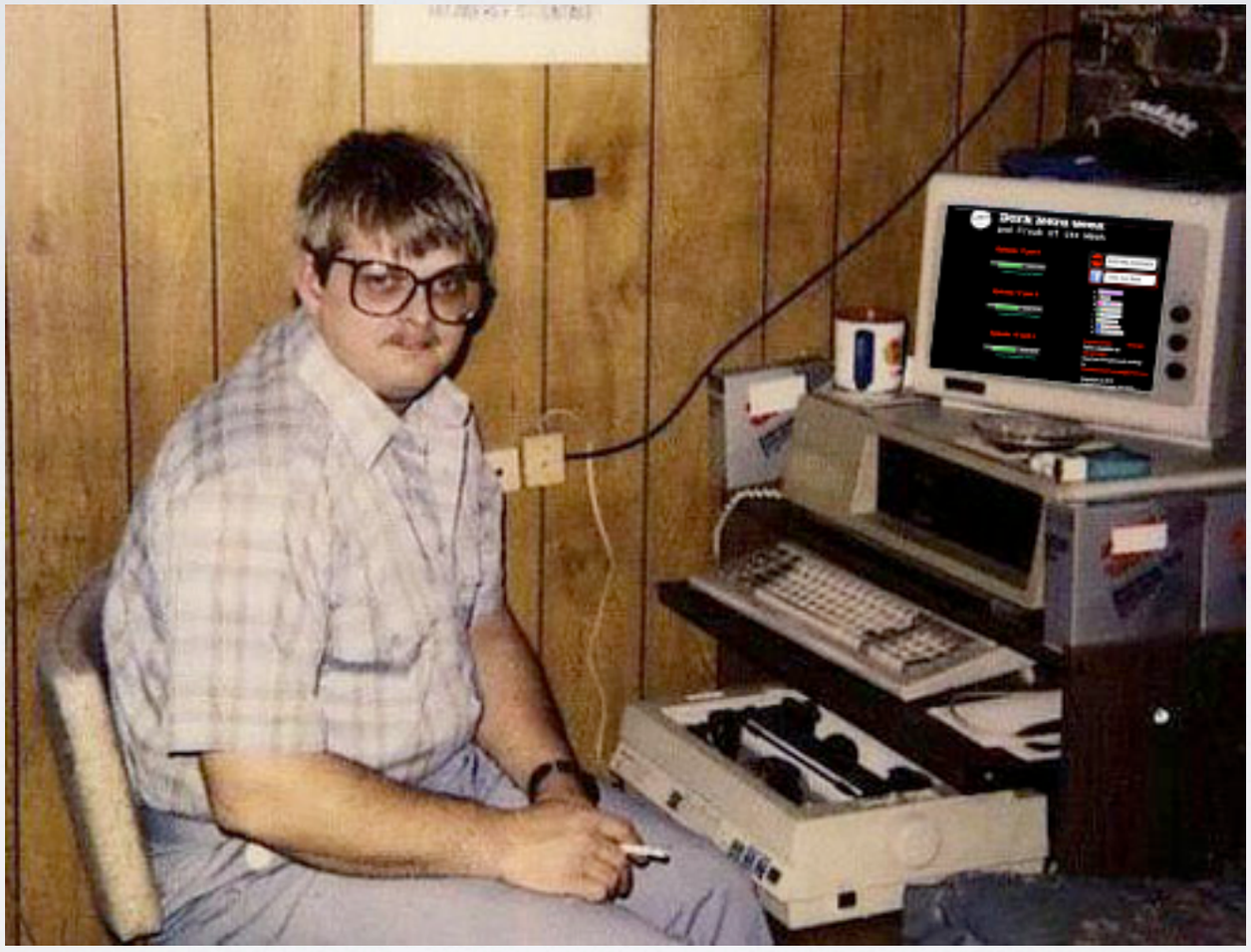


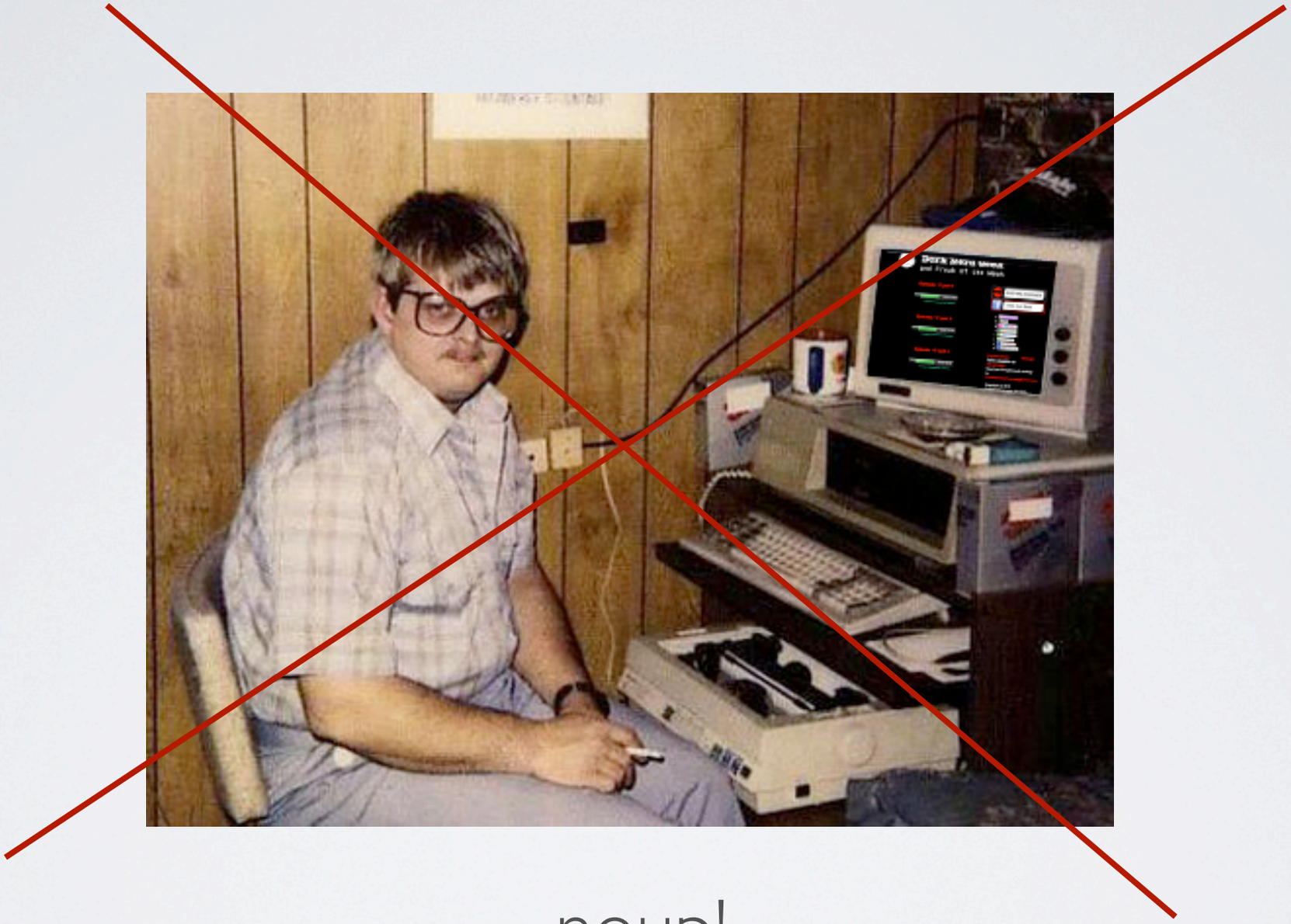
Ingeniero de Sistemas y Computación
Co-fundador de Monoku (@monoku)
Co-fundador de Monomi (@monomi_stores)
Co-organizador de BogotáJS (@BogotaJS)
Co-organizador de JSConfCo (@jsconfco)

@davsket - github.com/davsket

OBJETIVOS DEL CURSO

- Aprender a programar
- Aprender a hacerlo con JavaScript
- Conocer las herramientas básicas para entender cualquier algoritmo en JavaScript
- Poder interactuar con eventos y manipular el DOM a gusto





noup!

HERRAMIENTAS DEL CURSO

Google Chrome

SublimeText (please)

http://github.com/davsket/mejorando.la

http://davsket.me/mejorando.la

INTRODUCCIÓN A LA PROGRAMACIÓN

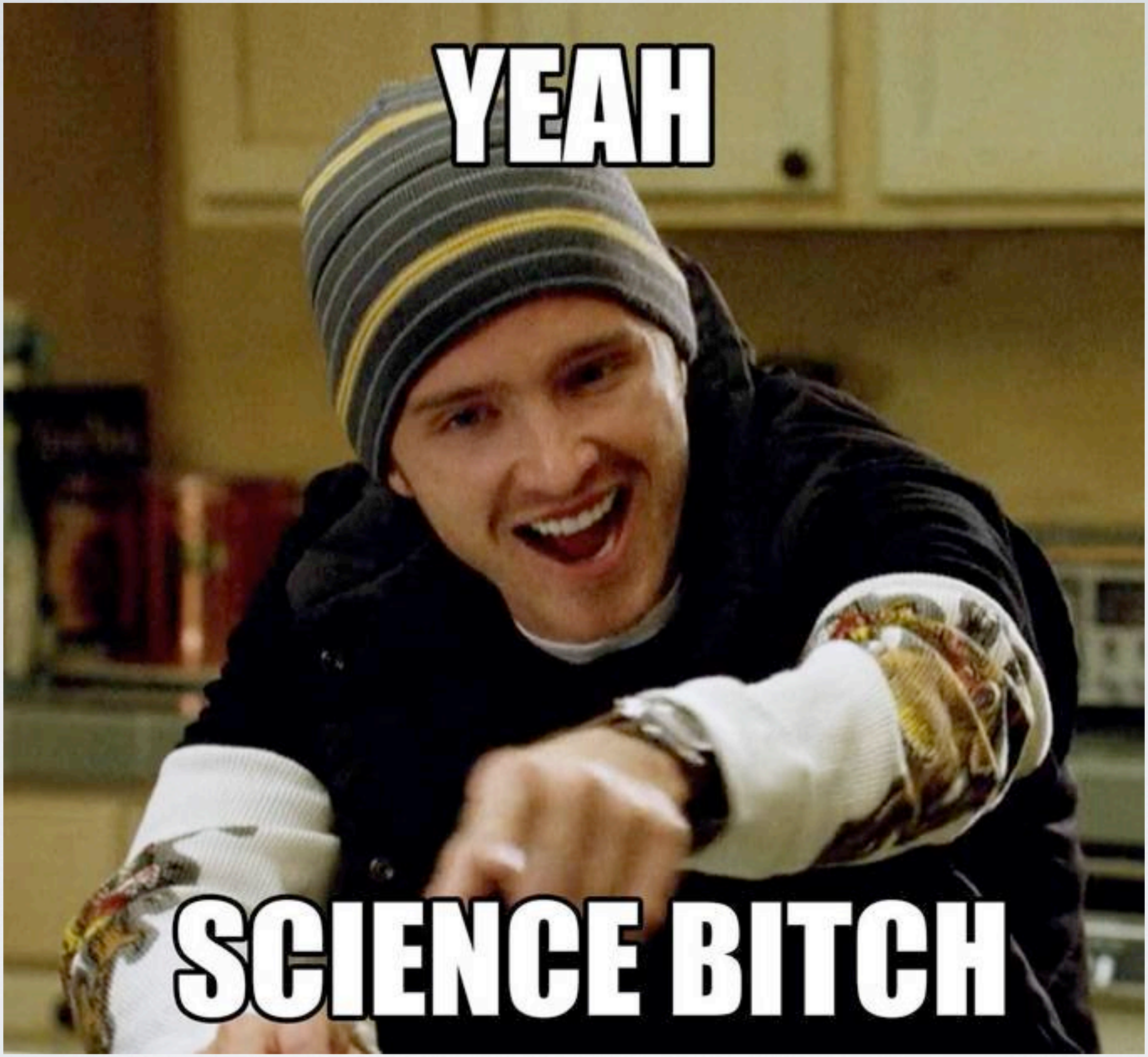
VARIABLES

Una variable es un nombre simbólico o un identificador que puede contener un tipo de información/valor.

VARIABLE

```
a = 2
b = 3
c = a + b
```

Como en álgebra


YEAH
SCIENCE BITCH

VARIABLE

```
var a = 20
```

Nombre

Valor (opcional)

VARIABLE

```
var a = 20
```

Siempre definan sus variables con **`var`**!

No hacerlo puede generar conflictos en el scope global

VARIABLE

```
var a = 20;              // número
var b = 'Hola';          // cadena de texto
var c = true;            // boleano
var d = [1,2,3];         // arreglo
var e = {};              // objeto
```

VARIABLE

Arreglos

```
var a = [20,3,8]
```

VARIABLE

Arreglos

```
var a = [20,3,8]
```

0

VARIABLE

Arreglos

```
var a = [20,3,8]
```

0 1

VARIABLE

Arreglos

```
var a = [20,3,8]
```

0 1 2

VARIABLE

Arreglos

```
var a = [20,3,8]
```

0 1 2

```
a[0] // 20
```

VARIABLE

Arreglos

```
var a = [20,3,8]
```

0 1 2

```
a[0] // 20
a[2] // 8
```

VARIABLE

Arreglos

```
var a = [20,3,8]
```

0 1 2

```
a[0] // 20
a[2] // 8
a.length // 3
```

VARIABLE

Arreglos

```
var a = [20,3,8]
```

0 1 2

```
a[0] // 20
a[2] // 8
a.length // 3
a[3] // undefined
```

VARIABLE

Arreglos

```
var a = ['a',3,true]
```

VARIABLE

Objetos

```
var a = {b:'hola'}
```

VARIABLE

Objetos

```
var a = {b:'hola'}

a.b // hola
```

VARIABLE

Objetos

```
var a = {b:'hola'}

a.b // hola
a['b'] // hola
```

VARIABLE

Objetos

```
var a = {b:'hola'}

a.b // hola
a['b'] // hola
a.z // undefined
```

VARIABLE

Objetos

```
var a = {'a b':'hola'}
```

VARIABLE

Objetos

```
var a = {'a b':'hola'}

a.a b      // error!!
```

VARIABLE

Objetos

```
var a = {'a b':'hola'}

a.a b      // error!!
a['a b']   // hola
```

VARIABLE

```
var a = 2
var b = 3
var c = a + b
alert(c)          // 5
```

Ejemplo 1

CONDICIONALES

Expresión que dependiendo de una condición ejecuta diferentes opciones de código.

CONDICIONALES

```
si a > b:
    max = a
si no:
    max = b
```

c será igual al mayor entre **a** y **b**

CONDICIONALES

```
if(a > b){ max = a }
```

Expresión boleana

Cuerpo
Ejecutable si la
expresión es válida

EXPRESIÓN BOLEANA?

EXPRESIÓN BOLEANA?

EXPRESIONES BOLEANAS

```
if(a > b)
```

a es mayor que **b**

```
if(a == b)
```

a es equivalente a **b**

```
if(a === b)
```

a es equivalente tan
a **b** (porque '2' ==

```
if(a != b)
```

a es diferente a **b**

```
if(a)
```

a existe, es decir: es diferente a **0**, **null**, **undefined** o **''**

Demostrar que por no ser tipado...

'2' == 2!!
0 == false
undefined == null

CONDICIONALES

Algunas expresiones boleanas en JavaScript

`if(a>2 && a<4)`	**a** mayor que **2 y** menor que **4**
`if(a>2 || a<4)`	**a** mayor que **2 o** menor que **4**
`if(!a)`	**no a**, es decir: si a es **0**, **null**, **undefined** o **''**

CONDICIONALES

```
if(a>b){
   //a>b
}else if(a<b){
   //a<b
}else{
   //a==b
}
```

Si a es mayor que b

De lo contrario;
si a es menor que b

Si ninguna de las
anteriores

Si entra en una no entra en las siguientes,
y siempre va en orden

CONDICIONALES

```
if(a > b){
  //a>b
}else if(a < b){
  //a<b
}else{
  //a==b
}
```

puede ir cero o varias veces

si va, va solo una vez,
no puedes tener dos else

CONDICIONALES

Agrupación en un var

```
var a = 2,
    b = 3,
    max

if( a > b ){
    max = a
}else{
    max = b
}
alert( max ) // 3
```

Ejemplo 2

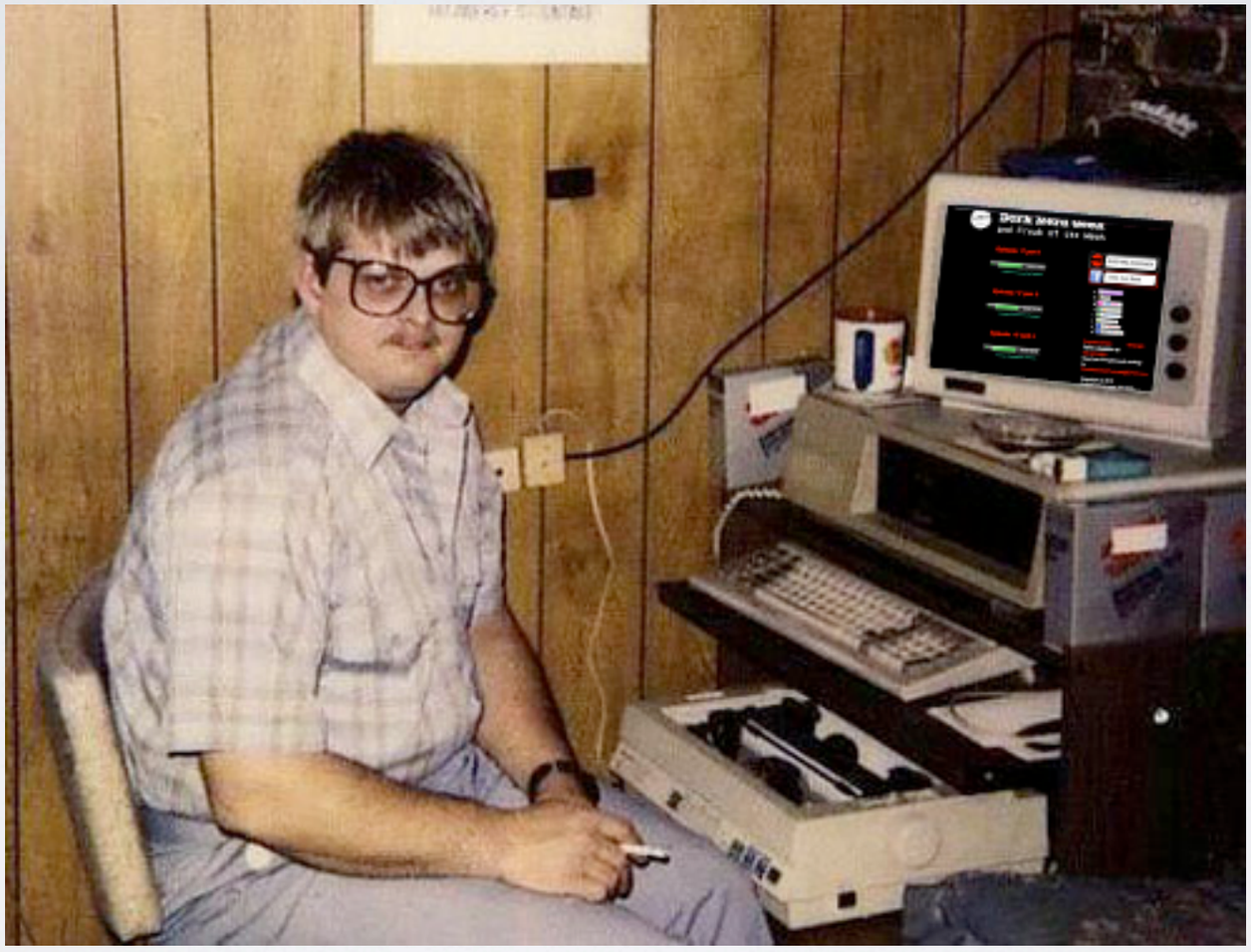
FUNCIONES

Es un conjunto de instrucciones
que son agrupadas bajo un identificador
para ser ejecutadas cuando sea necesario.

[DRY]

"divide y vencerás"

WARNING



Cada slide que pasa ustedes se acercan más a este individuo :D

FUNCIONES

```
max a,b:
    si a>b:
        :a
    si no:
        :b
```

la función **max** devuelve el máximo entre dos números

FUNCIONES

```
function max(a,b){ ... }
```

Nombre de la función (opcional)

Parámetros (opcionales)

Cuerpo: grupo de instrucciones a ejecutar

FUNCIONES

```
function max(a,b){
    if(a>b){
        return a
    }
    return b
}
alert(max(2,3)) // 3
```

Ejemplo 3

CICLOS

Repetición continua de un conjunto de instrucciones siempre y cuando tenga validez una condición.

CICLOS

condición
índice o centinela

```
i = 1
a = 0
mientras i<5:
    a = a + i
    i = i + 1
```

al final **i** será igual a 5 y **a** será igual a 10

1+2+3+4 = 10

CICLOS

```
i = 0
a = [1,2,3,2]
sum = 0
mientras i<a.largo:
  sum = sum + a[i]
  i = i + 1
```

CICLOS

```
i = 0
a = [1,2,3,2]
sum = 0
mientras i<a.largo:
  sum = sum + a[i]
  i = i + 1
```

al final **i** será igual a 5 y **sum** será igual a 8

CICLOS

for

```
for( var i ; i<l ; i++ ){ ... }
```

CICLOS

for

```
for( var i ; i<l ; i++ ){ ... }
```

Espacio para definición de variables (opcional)

CICLOS

for

```
for( var i ; i<l ; i++ ){ ... }
```

Espacio para definición de variables (opcional)

Condición para entrar al ciclo

CICLOS

for

```
for( var i ; i<l ; i++ ){ ... }
```

Espacio para definición de variables (opcional)

Condición para entrar al ciclo

Cuerpo: grupo de instrucciones a ejecutar

CICLOS

for

```
for( var i ; i<l ; i++ ){ ... }
```

Espacio para definición de variables (opcional)

Condición para entrar al ciclo

Expresión a ejecutar al final del ciclo (opcional)

Cuerpo: grupo de instrucciones a ejecutar

CICLOS

while

```
while( i<l ){ ... }
```

CICLOS

while

```
while( i<l ){ ... }
```

Condición
para entrar
al ciclo

CICLOS

while

```
while( i<l ){ ... }
```

Condición para entrar al ciclo

Cuerpo: grupo de instrucciones a ejecutar

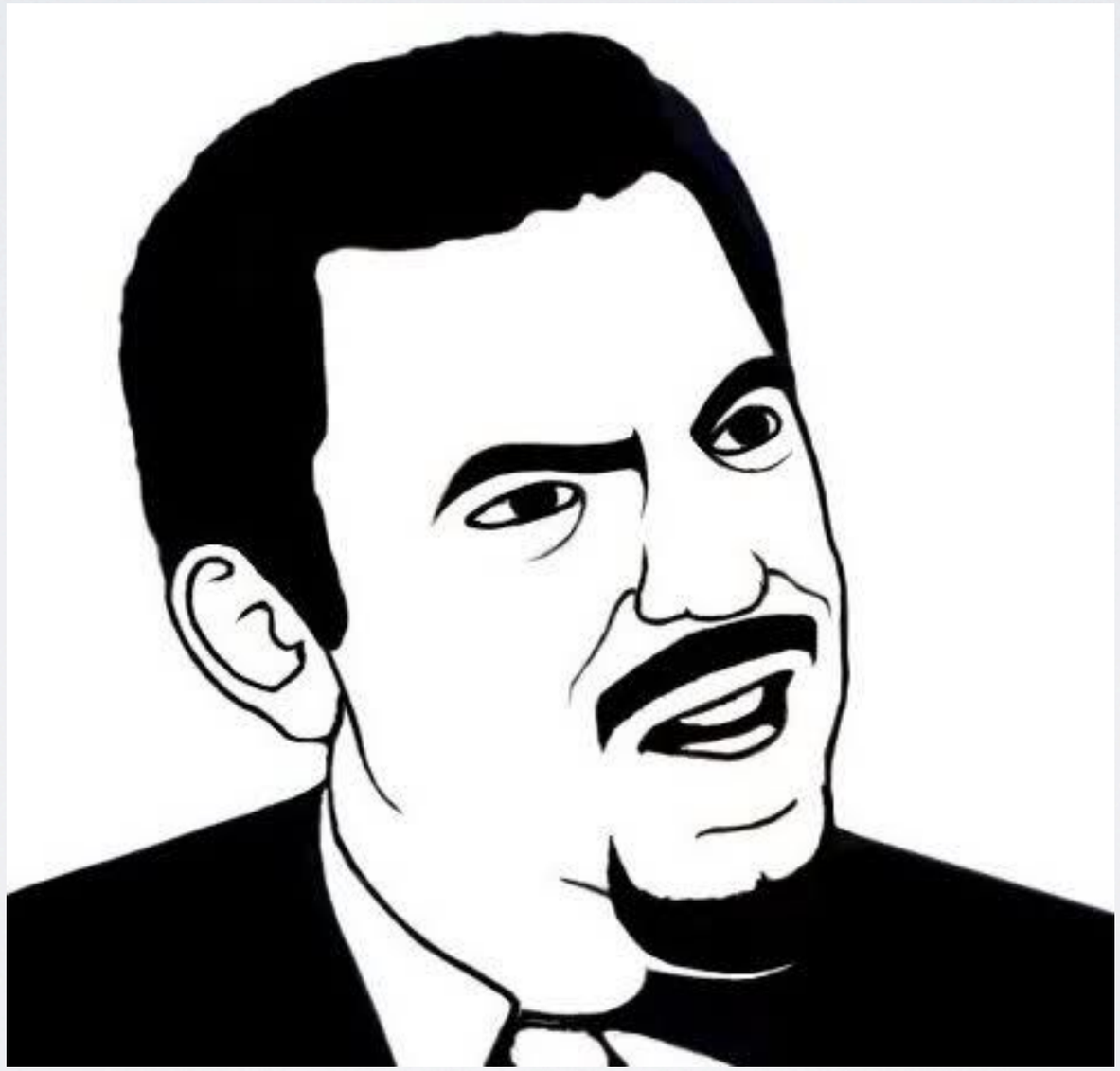
CICLOS

Importante aclarar lo del length hasta 4 y posiciones de 0 a 3, por eos el <

```
var a = [1,2,3,2],
    sum = 0
for(var i=0; i<a.length; i++){
    sum += a[i]
}
alert(sum) // 8
```

Ejemplo 4

FIN INTRO PROGRAMACIÓN



FIN INTRO PROGRAMACIÓN

JAVASCRIPT

Brendan Eich

JAVASCRIPT

- Nació en 1995 en Netscape (Mozilla)
- Implementaciones: Rhino (Java), SpiderMonkey (Firefox), V8 (Chrome y NodeJS), WebKit, Carakan (Opera), Chakra (IE)

JAVASCRIPT

Mostrar no tipado en Chrome

JAVASCRIPT

Mostrar no tipado en Chrome

No es tipado

JAVASCRIPT

Mostrar no tipado en Chrome

No es tipado

Es interpretado (no compilado)

JAVASCRIPT

Mostrar no tipado en Chrome

No es tipado

Es interpretado (no compilado)

Es orientado a prototipos

JAVASCRIPT

Mostrar no tipado en Chrome

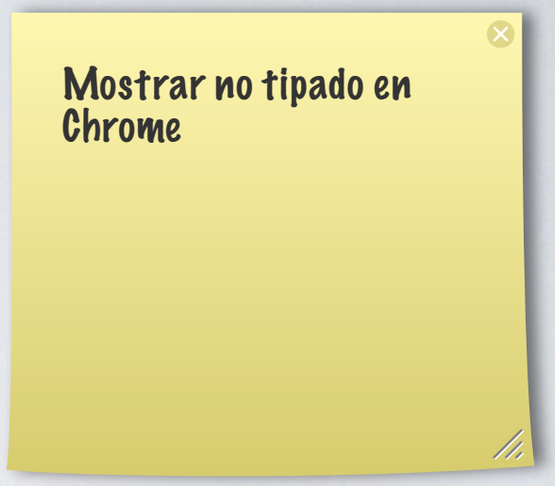
No es tipado

Es interpretado (no compilado)

Es orientado a prototipos

Es funcional (first-class objects)

JAVASCRIPT



No es tipado

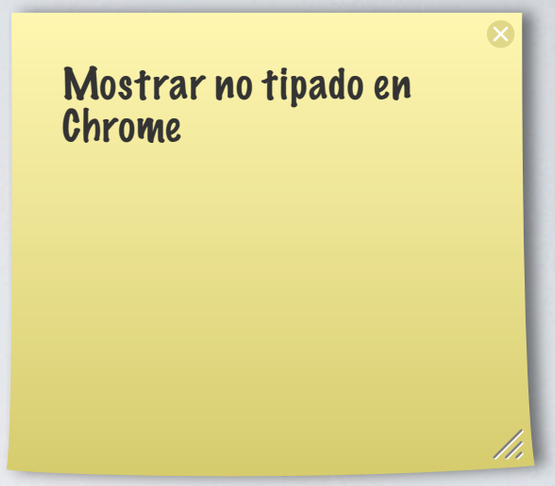
Es interpretado (no compilado)

Es orientado a prototipos

Es funcional (first-class objects)

No hay clases, ni propiedades privadas *

JAVASCRIPT



No es tipado

Es interpretado (no compilado)

Es orientado a prototipos

Es funcional (first-class objects)

No hay clases, ni propiedades privadas *

* *Excepto en Harmony*

JAVASCRIPT

functions are first-class objects

Ejemplo first-class-objects

JAVASCRIPT

functions are first-class objects

Ejemplo first-class-objects

- Las funciones son Objetos

JAVASCRIPT

functions are first-class objects

Ejemplo first-class-objects

- Las funciones son Objetos
- Se pueden asignar a variables

JAVASCRIPT

functions are first-class objects

Ejemplo first-class-objects

- Las funciones son Objetos
- Se pueden asignar a variables
- Pueden ser pasadas como parámetros de otra función.

JAVASCRIPT

functions are first-class objects

Ejemplo first-class-objects

- Las funciones son Objetos
- Se pueden asignar a variables
- Pueden ser pasadas como parámetros de otra función.
- Pueden ser retornadas por otra función.

JAVASCRIPT

JAVASCRIPT

- Patrón de Callbacks -

JAVASCRIPT

- Patrón de Callbacks -

- Patrón Observador -

JAVASCRIPT

- Patrón de Callbacks -

- Patrón Observador -

- Event Driven Development -

JAVASCRIPT

- Patrón de Callbacks -

- Patrón Observador -

- Event Driven Development -

(aka: Desarrollo Basado en Eventos)

PROTOTYPE

- Todos los objetos heredan funcionalidades y atributos por medio del `prototype`
- `prototype` es sencillamente un objeto

PROTOTYPE

- String, Array, Number, RegExp... heredan el prototipo de Object
- Cambios en el prototipo pueden ser usados inmediatamente en todas las instancias que lo heredan

SCOPE

Matrioska Dolls

FUNCIONES

tipos de funciones en JavaScript

FUNCIONES

tipos de funciones en JavaScript

funciones declaradas (recomendable)

FUNCIONES

tipos de funciones en JavaScript

funciones declaradas (recomendable)

```
function a(){}
a()
```

FUNCIONES

tipos de funciones en JavaScript

funciones declaradas (recomendable)

```
function a(){}
a()
```

anónimas asignadas a variables (funciones como expresión)

FUNCIONES

tipos de funciones en JavaScript

funciones declaradas (recomendable)

```
function a(){}
a()
```

anónimas asignadas a variables (funciones como expresión)

```
var a = function(){}
a()
```

FUNCIONES

tipos de funciones en JavaScript

Explicar muy por encima
closures

FUNCIONES

tipos de funciones en JavaScript

Explicar muy por encima closures

anónimas (normalmente usadas inmediatamente)

FUNCIONES

tipos de funciones en JavaScript

Explicar muy por encima closures

anónimas (normalmente usadas inmediatamente)

```
function(){ ... }
```

FUNCIONES

tipos de funciones en JavaScript

Explicar muy por encima closures

anónimas (normalmente usadas inmediatamente)

```
function(){ ... }
(function(){ ... }())
```

FUNCIONES

tipos de funciones en JavaScript

Explicar muy por encima closures

anónimas (normalmente usadas inmediatamente)

```
function(){ ... }
(function(){ ... }())
```

métodos, funciones asignadas a objetos

FUNCIONES

tipos de funciones en JavaScript

Explicar muy por encima closures

anónimas (normalmente usadas inmediatamente)

```
function(){ ... }
(function(){ ... }())
```

métodos, funciones asignadas a objetos

```
var a = {f: function(){} }
a.f()
```

DOM

Document Object Model

Representación dinámica del HTML, en pocas palabras es la traducción de los tags, a elementos con los cuales puedes interactuar por medio de JavaScript.

AJAX

Asynchronous JavaScript and XML

Peticiones asincrónicas (que no bloquean el navegador) al servidor para la carga de texto, usualmente en formato JSON o XML.

TODOS LOS NAVEGADORES FUNCIONAN DIFERENTE

DIFERENCIAS ENTRE NAVEGADORES

DIFERENCIAS ENTRE NAVEGADORES

Manipulación del DOM

DIFERENCIAS ENTRE NAVEGADORES

Manipulación del DOM

Manejo de Eventos del DOM

DIFERENCIAS ENTRE NAVEGADORES

Manipulación del DOM

Manejo de Eventos del DOM

Manejo de Requests XMLHttpRequest (AJAX)

DIFERENCIAS ENTRE NAVEGADORES

En el caso de hacer un request al servidor para obtener una información, comúnmente usarías:
`XMLHttpRequest`

Pero en InternetExplorer 7- no existe, toca usar
`ActiveXObject`

El cual se usa diferente dependiendo del tipo de datos que se descargue.

DIFERENCIAS ENTRE NAVEGADORE

Para escuchar clicks, drags, teclado, etc

DIFERENCIAS ENTRE NAVEGADORE

Para escuchar clicks, drags, teclado, etc

En todos los navegadores MODERNOS puedes usar

DIFERENCIAS ENTRE NAVEGADORE

Para escuchar clicks, drags, teclado, etc

En todos los navegadores MODERNOS puedes usar

```
el.addEventListener('click'...)
```

DIFERENCIAS ENTRE NAVEGADORE

Para escuchar clicks, drags, teclado, etc

En todos los navegadores MODERNOS puedes usar

`el.addEventListener('click'...)`

para asignar listeners de eventos, pero en IE8-

DIFERENCIAS ENTRE NAVEGADORES

Para escuchar clicks, drags, teclado, etc

En todos los navegadores MODERNOS puedes usar

`el.addEventListener('click'...)`

para asignar listeners de eventos, pero en IE8- tienes que usar

DIFERENCIAS ENTRE NAVEGADORE

Para escuchar clicks, drags, teclado, etc

En todos los navegadores MODERNOS puedes usar

`el.addEventListener('click'...)`

para asignar listeners de eventos, pero en IE8- tienes que usar

`el.attachEvent('onclick'...)`

DIFERENCIAS ENTRE NAVEGADORE

Para escuchar clicks, drags, teclado, etc

En todos los navegadores MODERNOS puedes usar

`el.addEventListener('click'...)`

para asignar listeners de eventos, pero en IE8- tienes que usar

`el.attachEvent('onclick'...)`

Y en navegadores viejos...

DIFERENCIAS ENTRE NAVEGADORE

Para escuchar clicks, drags, teclado, etc

En todos los navegadores MODERNOS puedes usar

`el.addEventListener('click'...)`

para asignar listeners de eventos, pero en IE8- tienes que usar

`el.attachEvent('onclick'...)`

Y en navegadores viejos...

`el.onclick = ...`

!¿QUÉ HACER?!

USAR UN FRAMEWORK

- Librerías de JavaScript al rescate -

USAR UN FRAMEWORK

- Librerías de JavaScript al rescate -

NORMALIZAN

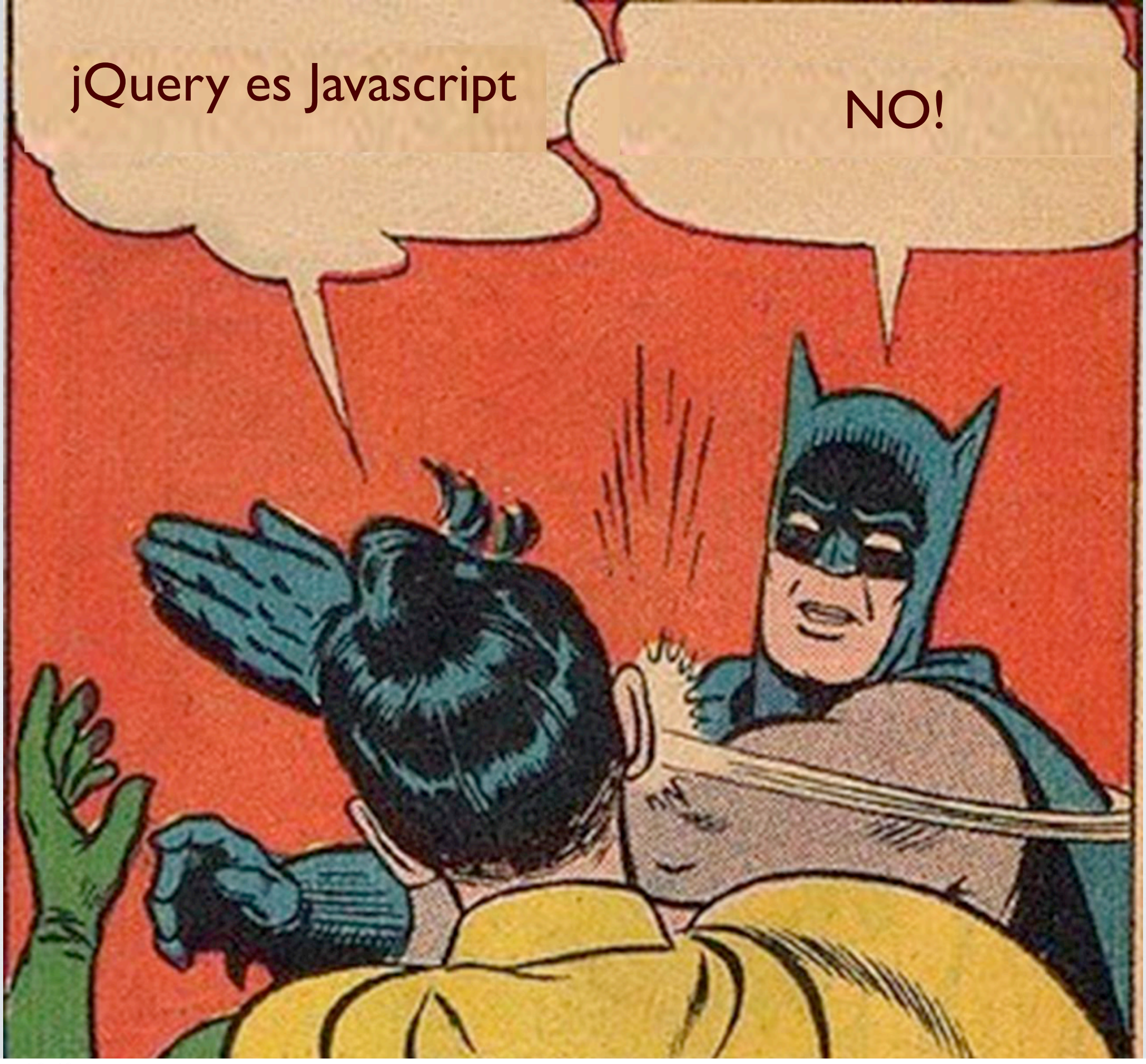
jQuery
write less, do more.

zepto.js

dojo

mootools

jQuery


jQuery es Javascript
NO!

yo desarrollo
en jQuery
NO!

JavaScript

jQuery ≠ JavaScript

jQuery ≠ JavaScript

✓ jQuery funciona sobre JavaScript

≠ JavaScript

- ✓ jQuery funciona sobre JavaScript
- ✓ Todo lo que haces con jQuery lo puedes hacer con JavaScript puro

≠ JavaScript

- ✓ jQuery funciona sobre JavaScript
- ✓ Todo lo que haces con jQuery lo puedes hacer con JavaScript puro
- ✓ JavaScript fue antes de jQuery

≠ JavaScript

- ✓ jQuery funciona sobre JavaScript
- ✓ Todo lo que haces con jQuery lo puedes hacer con JavaScript puro
- ✓ JavaScript fue antes de jQuery
- ✓ Puedes desarrollar web sin jQuery (pero trata de usar un framework o micro-framework)

JQUERY

Navegación y búsqueda en el DOM

Manejo de Eventos

Manipulación del DOM

Ajax

Animaciones

JQUERY

- Un objeto jQuery **$** (es una gran fachada)
- Implementa el Chaining Pattern

JQUERY

uso

```
<!-- jQuery en el proyecto -->
<script src="jquery.js"></script>
<!-- jQuery en el CDN de otro, como Google
-->
<script src="//ajax.googleapis.com/ajax/libs/
    jquery/1.10.2/jquery.min.js"></script>
<!-- Luego si tus scripts -->
<script src="tu-script.js"></script>
```

VANILLA JAVASCRIPT

Termino para referirse a usar JavaScript sin frameworks.

VANILLA JAVASCRIPT

JQUERY

NAVEGACIÓN

vanilla-javascript

```
// Objener todos los links
document.querySelectorAll('a')
document.getElementsByTagName('a')
```

NAVEGACIÓN

jQuery

```
// Objener todos los links
$('a')
```

EVENTOS

vanilla-javascript

```
document
    .getElementById('boton')
    .addEventListener('click', miFuncion)
```

EVENTOS

jQuery

```
$('#boton').on('click', miFuncion)
```

MANIPULACIÓN

vanilla-javascript

```
var el = document.getElementById('elem')
el.className = 'seleccionado'
el.style.fontSize = '2em'
el.addEventListener('click', miFuncion)
```

MANIPULACIÓN

jQuery

```
$('#elem')
    .addClass('seleccionado')
    .css('font-size','2em')
    .on('click', miFuncion)
```

AJAX

vanilla-javascript

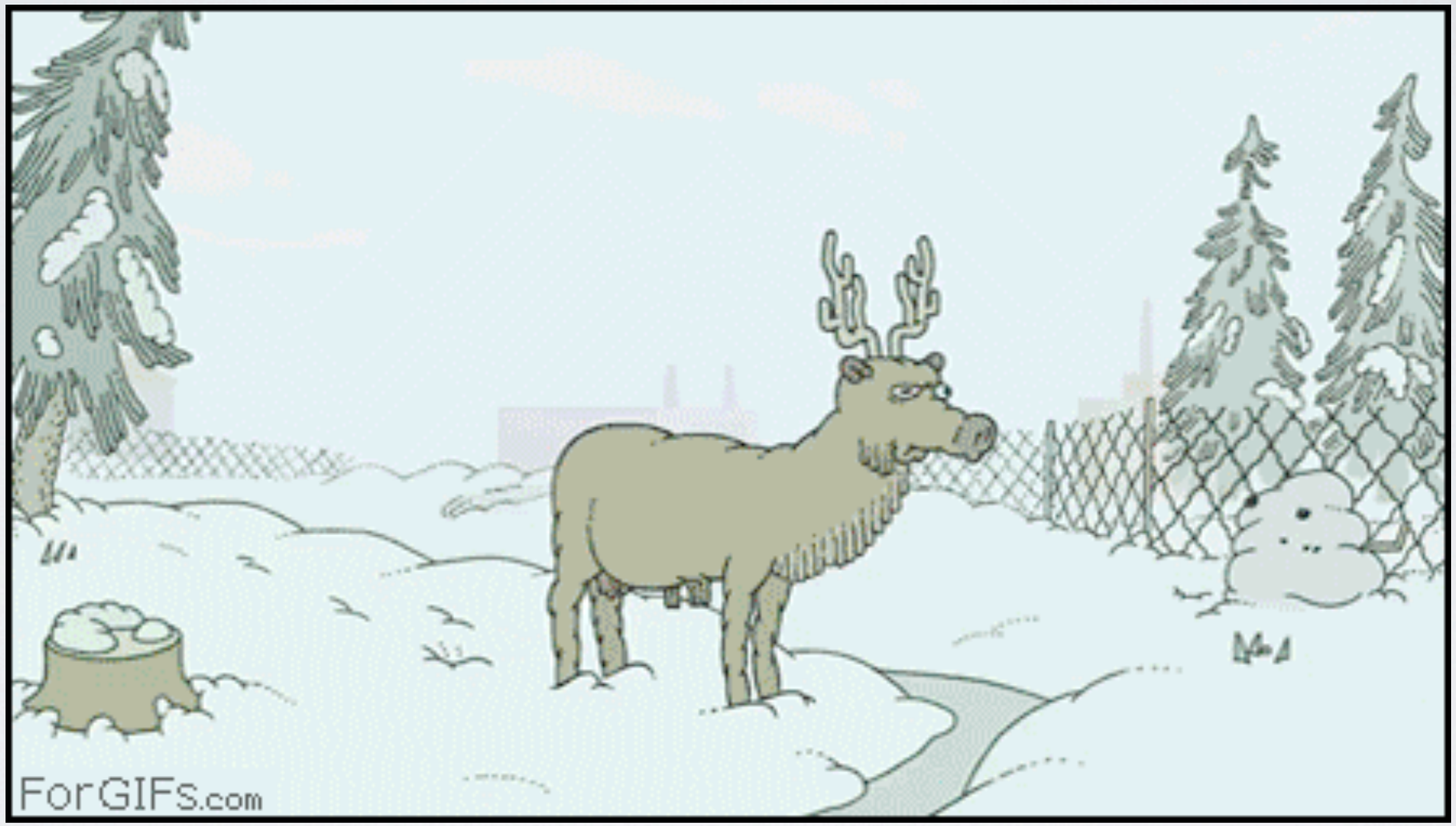
```
var req = new XMLHttpRequest();
req.onload = function(){
    var json = JSON.parse(this.responseText)
    document.getElementById('response')
        .innerHTML = json.data;
};
req.open('get', 'res/data/hola.json', true);
req.send();
```

AJAX

jQuery

```
$.ajax({
    url: 'res/data/hola.json'
}).done(function(res){
    $('#response').html(res.data)
})
```

LETS CODE!!



SVG!!

Scalable Vector Graphics

* HMTL no es basado en XML, XHTML si :)

SVG!!

Scalable Vector Graphics

- Es markup basado en XML (como HTML*)

* HMTL no es basado en XML, XHTML si :)

SVG!!

Scalable Vector Graphics

- Es markup basado en XML (como HTML*)
- Todos los navegadores modernos lo soportan (IE9+)

* HMTL no es basado en XML, XHTML si :)

SVG!!

Scalable Vector Graphics

- Es markup basado en XML (como HTML*)
- Todos los navegadores modernos lo soportan (IE9+)
- Usualmente se generan con ayuda de librerías (Raphaël),

* HMTL no es basado en XML, XHTML si :)

SVG!!

Scalable Vector Graphics

- Es markup basado en XML (como HTML*)
- Todos los navegadores modernos lo soportan (IE9+)
- Usualmente se generan con ayuda de librerías (Raphaël),
- o aplicaciones gráficas (Illustrator).

* HMTL no es basado en XML, XHTML si :)

SVG!!

Scalable Vector Graphics

- Es markup basado en XML (como HTML*)
- Todos los navegadores modernos lo soportan (IE9+)
- Usualmente se generan con ayuda de librerías (Raphaël),
- o aplicaciones gráficas (Illustrator).

Ejemplo:

* HMTL no es basado en XML, XHTML si :)

SVG!!

Scalable Vector Graphics

- Es markup basado en XML (como HTML*)
- Todos los navegadores modernos lo soportan (IE9+)
- Usualmente se generan con ayuda de librerías (Raphaël),
- o aplicaciones gráficas (Illustrator).

Ejemplo:

http://www.arvernes.com/wiki/images/a/a9/Mozilla_Firefox.svg

* HMTL no es basado en XML, XHTML si :)

CANVAS!!

CANVAS!!

- Funciona tal y como un canvas!

CANVAS!!

- Funciona tal y como un canvas!
- No hay tags!

CANVAS!!

- Funciona tal y como un canvas!
- No hay tags!
- Dibujas a nivel de bitmaps (como BMP)

CANVAS!!

- Funciona tal y como un canvas!
- No hay tags!
- Dibujas a nivel de bitmaps (como BMP)
- Si quieres mover algo que dibujaste, toca borrar y redibujar

CANVAS!!

- Funciona tal y como un canvas!
- No hay tags!
- Dibujas a nivel de bitmaps (como BMP)
- Si quieres mover algo que dibujaste, toca borrar y redibujar

Ejemplo:

CANVAS!!

- Funciona tal y como un canvas!
- No hay tags!
- Dibujas a nivel de bitmaps (como BMP)
- Si quieres mover algo que dibujaste, toca borrar y redibujar

Ejemplo:

http://www.anieto2k.com/demo/canvasfirefox.html

SE QUE PUEDE SER MUCHA INFO

Veamos un par de ejemplos sencillos!!

FIN

Gracias!
@davsket / github.com/davsket
#mejorandoweb

PARA APRENDER MÁS

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs
- Español

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs
- Español
  - http://davsket.me/ocho-cosas

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs
- Español
  - http://davsket.me/ocho-cosas
  - http://slid.es/davsket/herramientas-del-front

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs
- Español
  - http://davsket.me/ocho-cosas
  - http://slid.es/davsket/herramientas-del-front
  - http://slid.es/davsket/optimizando-con-raf

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs
- Español
  - http://davsket.me/ocho-cosas
  - http://slid.es/davsket/herramientas-del-front
  - http://slid.es/davsket/optimizando-con-raf
- Inglés

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs
- Español
  - http://davsket.me/ocho-cosas
  - http://slid.es/davsket/herramientas-del-front
  - http://slid.es/davsket/optimizando-con-raf
- Inglés
  - https://github.com/rwldrn/idiomatic.js

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs
- Español
  - http://davsket.me/ocho-cosas
  - http://slid.es/davsket/herramientas-del-front
  - http://slid.es/davsket/optimizando-con-raf
- Inglés
  - https://github.com/rwldrn/idiomatic.js
  - Javascript the Good Parts

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs
- Español
  - http://davsket.me/ocho-cosas
  - http://slid.es/davsket/herramientas-del-front
  - http://slid.es/davsket/optimizando-con-raf
- Inglés
  - https://github.com/rwldrn/idiomatic.js
  - Javascript the Good Parts
  - JavaScript Definitive Guide

PARA APRENDER MÁS

- http://www.meetup.com/BogotaJS/
- http://jsconf.co
- https://github.com/colombiajs
- Español
  - http://davsket.me/ocho-cosas
  - http://slid.es/davsket/herramientas-del-front
  - http://slid.es/davsket/optimizando-con-raf
- Inglés
  - https://github.com/rwldrn/idiomatic.js
  - Javascript the Good Parts
  - JavaScript Definitive Guide
  - http://shichuan.github.io/javascript-patterns/