Introduction to JavaScript

por Ignacio Hernandez y Francisco Bueno

12 / Agosto / 2019

Agenda

Ventajas de JavaScript

Una breve historia de JavaScript

Los fundamentos de JavaScript

Objectivos

Al final del taller, debería poder:

- Escribe un programa básico de JavaScript
- Comprenda los conceptos básicos de JavaScript, que incluyen:
  - Objetos y Primitivos
  - Truthy and Falsy
  - Funciones

Beneficios de JavaScript

- Desarrollo más Rápido y Agradable.
- Menos Codigo.
- Aplicaciones Web, API, de Escritorio, Móviles, IoT, Etc.

¿Qué significa "expresivo" cuando se refiere a lenguajes de programación?
“Expresivo” significa que es fácil escribir código que sea fácil de entender, tanto para el compilador como para un lector humano.

http://stackoverflow.com/questions/638881/what-does-expressive-mean-when-referring-to-programming-languages

Desarrollo Web: Cliente

- ¡JavaScript es el ensamblaje de la web!
- HTML + CSS
- HTTP, HTTPS, HTTP 2.0, TCP / IP

Desarrollo Web: Servidor

- Tiempo de Ejecución
- Servidor Web
- HTTP, HTTPS, HTTP 2.0, TCP / IP

Breve Historia de JavaScript

- 1995 LiveScript and Netscape
- 1996-1997 Primer Estandar ECMA
- 2008 ECMAScript 5
- 2015 ECMAScript 6 (a.k.a. ES6)

HISTORIA

How JavaScript Was Created:
http://speakingjs.com/es5/ch04.html

Temas para comenzar con JavaScript

- Principios de JavaScript
- Primitivos y Objetos
- Funciones
- Truthy and Falsy

Variables

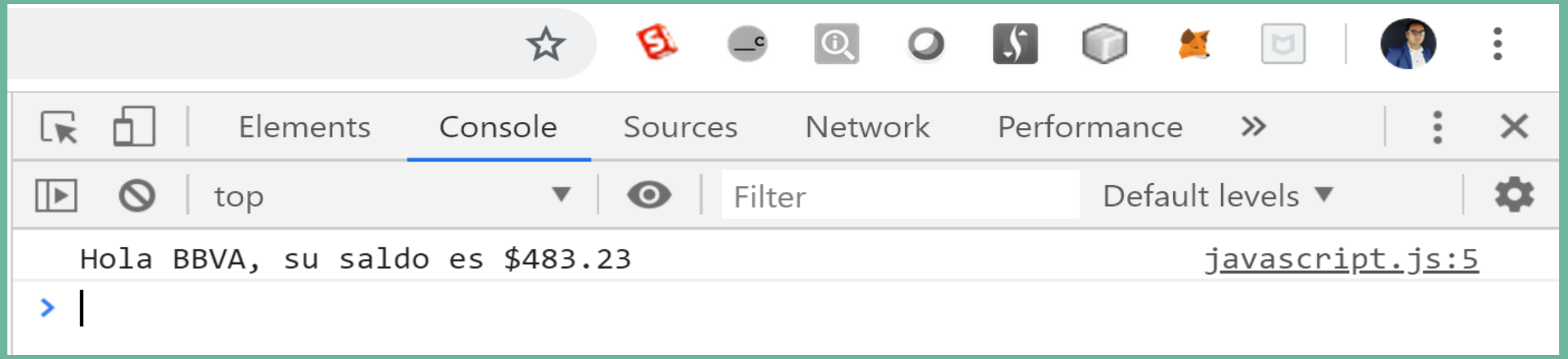
Tipo	Descripcion
Primitivos	Un primitivo (tipo de dato primitivo) son datos que no son un objeto y no tienen métodos.
Objetos	Todo lo demás en JavaScript

```
let userName = “BBVA” // Declaración de Variable de Tipo Cadena de Texto (Primitivo)
let balance = 483.23 // Declaración de Variable de Tipo Numero (Primitivo)
const currencySymbol = “$”

console.log(`Hola ${userName}, su saldo es ${currencySymbol}${balance}`)

// Output Hola BBVA, su saldo es $483.23
```

JS javascript.js ✖ html.html

actividad_2 ▸ JS ▸ JS javascript.js ▸ ...

```
1   let userName = "BBVA" // Declaración de Variable de Tipo Cadena de Texto (Primitivo)
2   let balance = 483.23 // Declaración de Variable de Tipo Numero (Primitivo)
3   const currencySymbol = "$"
4
5   console.log(`Hola ${userName}, su saldo es ${currencySymbol}${balance}`)
6
7
```



Primitivos

En JavaScript, hay 6 tipos de datos primitivos:

- String
- Number
- Boolean
- Null
- Undefined
- Symbol (New in ECMAScript 6)

Cambios realizados desde ES5 a ES6, visite http://es6-features.org/#Constants

Expresiones Aritméticas. Las cuales evalúan números ya sea enteros o flotantes.

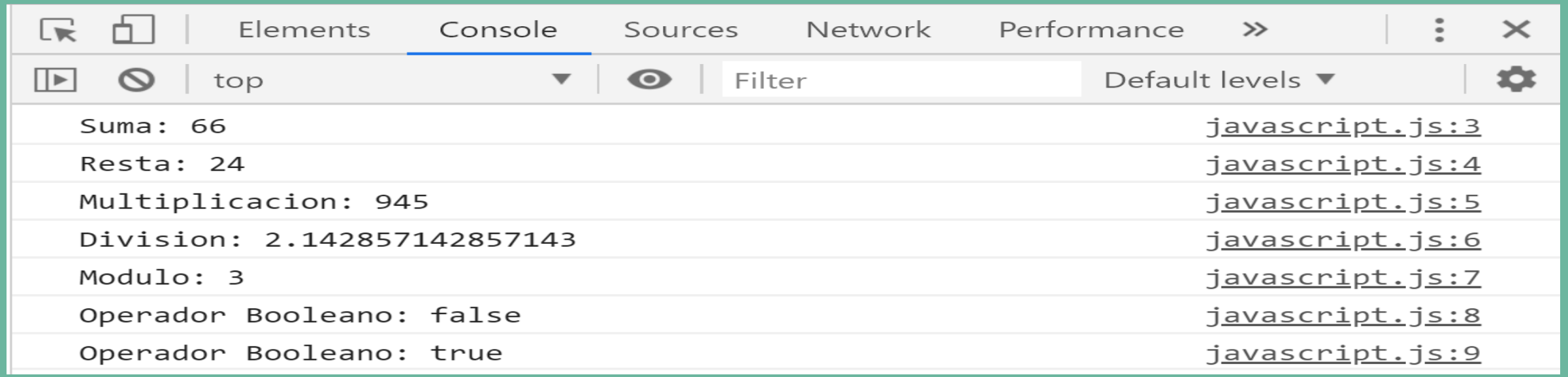
```
num1 = 45;
num2 = 21;
suma = num1 + num2;
```

Expresiones Lógicas. Las cuales evalúan si algo es verdadero o falso.

```
num1 > num2;
num1 < num2;
```

javascript.js

```
JS > JS javascript.js
  1    num1 = 45;
  2    num2 = 21;
  3    console.log("Suma: " + (suma = num1 + num2));
  4    console.log("Resta: " + (resta = num1 - num2));
  5    console.log("Multiplicacion: " + (multiplicacion = num1 * num2));
  6    console.log("Division: " + (division = num1 / num2));
  7    console.log("Modulo: " + (modulo = num1 % num2));
  8    console.log("Operador Booleano: " + (num1 < num2));
  9    console.log("Operador Booleano: " + (num1 > num2));
 10
```



Wrappers Primitivos

A excepción de nulo e indefinido, todos los valores primitivos tienen equivalentes de objeto que envuelven los valores primitivos:

- Cadena para la Cadena Primitiva
- Número para el Número Primitivo
- Booleano para la Primitiva Booleana
- Símbolo para el Símbolo Primitivo (ES6)

Todo lo que no es primitivo es un objeto.

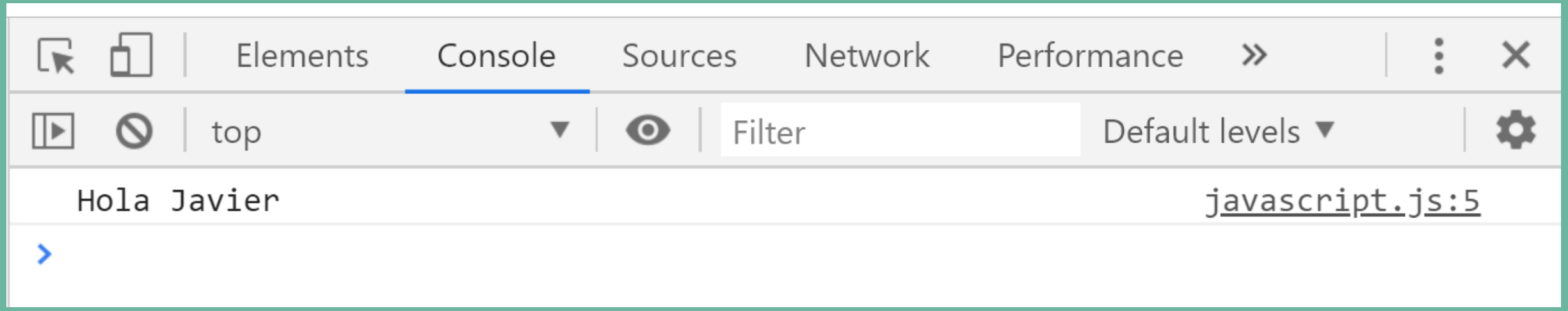
- Objeto
- Array
- Funcion
- Fecha

```
let userName = "javier" // Declaración de Variable de Tipo Cadena de Texto (Primitivo)


userName = userName[0].toUpperCase() + userName.slice(1) // Mayuscula el primer caracter


console.log(`Hola ${userName}`)


// Output Hola Javier
```

javascript.js ● html.html

actividad_2 ▸ JS ▸ javascript.js ▸ ...

```
1   let userName = "javier" // Declaración de Variable de Tipo Cadena de Texto (Primitivo)
2
3   userName = userName[0].toUpperCase() + userName.slice(1) // Capitaliza el primer caracter
4
5   console.log(`Hola ${userName}`)  // Output Hola Javier
6
7
```



```
let a = 1  // Declaración de Variable de Tipo Numero (Primitivo)
let b = a*2  // Declaración de Variable de Tipo Numero (Primitivo)
console.log(b)
if (b % 2 == 0) {  // Realizar la operación de Modulo
console.log(`${b} es un número par`)  // Imprime Resultado
} else {
  console.log(`${b} es un número impar`)
}
```

JS javascript.js ✕ | html.html

actividad_2 ▸ JS ▸ JS javascript.js ▸ ...

```
1   let a = 2  // Declaración de Variable de Tipo Numero (Primitivo)
2   let b = a*2  // Declaración de Variable de Tipo Numero (Primitivo)
3   console.log(b)
4   if (b % 2 == 0) {  // Realizar la operación de Modulo
5   console.log(`${b} es un número par`)  // Imprime Resultado
6   } else {
7     console.log(`${b} es un número impar`)
8   }
9
```

Elements | Console | Sources | Network | Performance | » | ⋮ | ✕

top ▾ | Filter | Default levels ▾ | ⚙

```
4                                          javascript.js:3
4 es un número par                         javascript.js:5
>
```

```
var f = function(str, int, arr) { // string, int o array
  return false;
}

function f() { // Funcion Hoisted
  return true;
}

var f = function f () // Hoisted and Referenciado
  return 1;
}
```

JS javascript.js ● html.html

actividad_2 ▸ JS ▸ JS javascript.js ▸ ...

```
1   function nombrebanco(banco) {
2     console.log("Bienvenido a " + banco);
3   }
4
5   nombrebanco("BBVA");
6
7   nombrecliente("Javier")
8
9   function nombrecliente(cliente) {
10    console.log("Es un gusto saludarlo " + cliente);
11  }
12
```

JavaScript permite las declaraciones de funciones antes de ejecutar cualquier otro segmento de código lo cual permite utilizar una función antes de declararla en el código.

```
var f = function(options) {
  var value = ...
  return value; // retorna un valor
}
```

Función vs Objeto

La función es un objeto que se puede invocar, es decir, las funciones pueden tener constructores, las funciones pueden tener propiedades.

```
var f = function(){
}

f.a = 1

console.log(f.a)
```

```
//Java:
System.out.println("Step: 1");
System.out.println("Step: 2");
Thread.sleep(1000);
System.out.println("Step: 3");

//JavaScript:
console.log('Step: 1')
setTimeout(function () {
  console.log('Step: 2')
}, 1000)
console.log('Step: 3')
```

Escritura Dinámica

JavaScript es un lenguaje poco tipado o "dinámico"

```
var bbva = 42
console.log( typeof bbva ) // returns "number"
var bbva = "banco"
var bbva = true
```

“En JavaScript, las variables no tienen tipos, los valores tienen tipos".

Kyle Simpson

Kyle Simpson es el autor de la serie de No Sabes JavaScript

Numeros

No hay tipos separados para diferentes números como long, int, double!

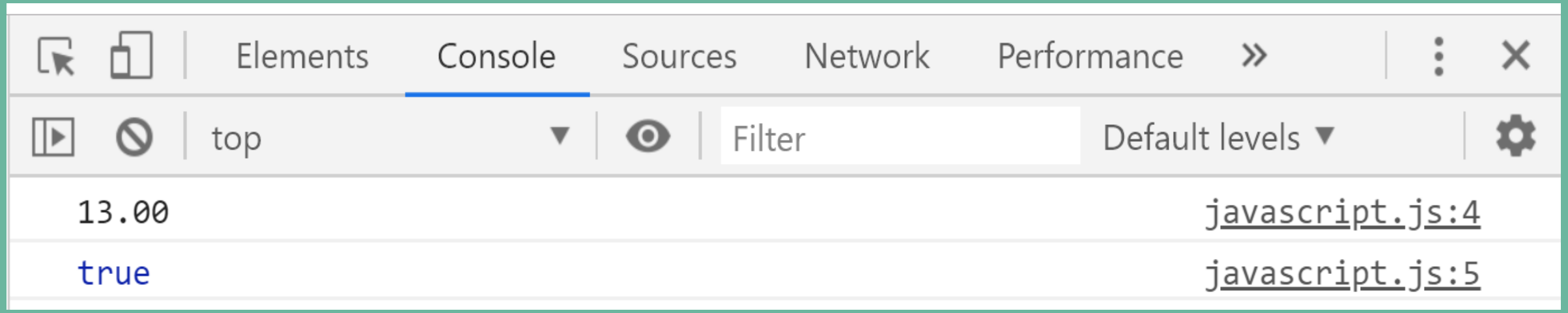
- Todos los números tienen formato binario de doble precisión de 64 bits.
- Para números más largos, convertir a una cadena.

```
let lucky = 13
let answer = new Number(42)
let old = Number(13)

lucky.toFixed(2)
lucky == 13.00
```

JS javascript.js

JS > JS javascript.js > ...

```
1  let lucky = 13
2  let answer = new Number(42)
3  let old = Number(13)
4  console.log(lucky.toFixed(2))
5  console.log(lucky == 13.00)
6
```

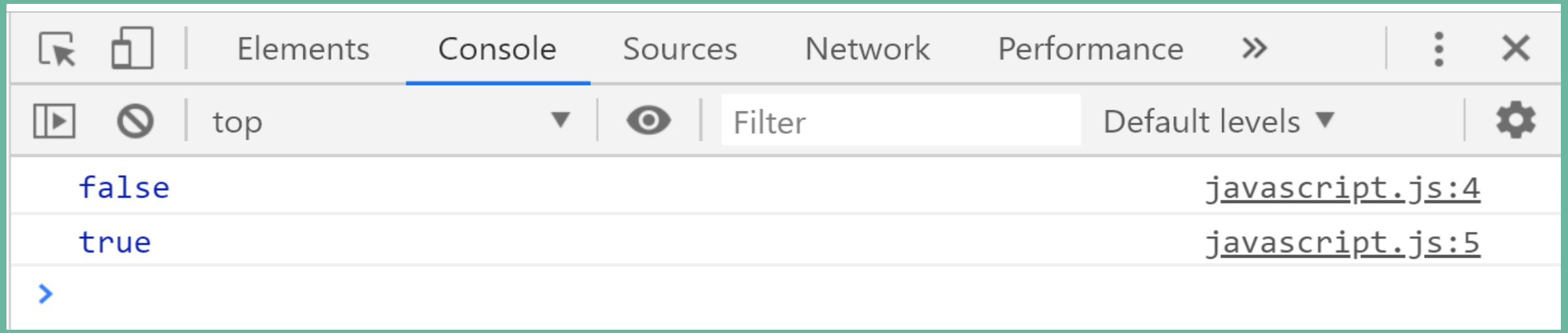


```
true !== false

var ok = true
var obj = new Boolean(true)
ok === obj // false
ok == obj  // true
```

JS javascript.js

JS > JS javascript.js > ...

```
1   true !== false
2   var ok = true
3   var obj = new Boolean(true)
4   console.log(ok === obj) // false
5   console.log(ok == obj)  // true
6
```



Falsy Valores

- false
- null
- undefined
- 0
- NaN
- '' (empty string)

```
(false)? console.log('truthy') : console.log('falsy') // falsy
(null)? console.log('truthy') : console.log('falsy') // falsy
(undefined)? console.log('truthy') : console.log('falsy') // falsy
(0)? console.log('truthy') : console.log('falsy') // falsy
(NaN)? console.log('truthy') : console.log('falsy') // falsy
('')? console.log('truthy') : console.log('falsy') // falsy
```

Truthy

Todos los valores que no son falsos son verdaderos.

```
(1)? console.log('truthy') : console.log('falsy') // truthy
([])? console.log('truthy') : console.log('falsy') // truthy
({})? console.log('truthy') : console.log('falsy') // truthy
(' ')? console.log('truthy') : console.log('falsy') // truthy
```

Secuencias de Escape

Secuencias de Escape	JS
Nueva Línea	/n
Apóstrofe	\’
Retorno Carriage	\r
Barra Invertida	\\
Pestaña Horizontal	\t
Hex	\xdd
Unicode	\udddd

Ejemplo:

```
console.log( "BBVA\'s México\nBienvenido al Banco!" )
```

Cadena Multilínea

- ES5:

```
var str = 'foo \
bar \
xyz'
```

- Es una práctica común usar simplemente la concatenación:

```
var str = 'foo ' +
  'bar '+
  'xyz'
```

- ES6/ES2015:

```
var str = `foo
  bar
  xyz`
```

```
var arr = [element0, element1, ..., elementN]
arr = new Array(element0, element1[, ...[, elementN]])
arr = new Array(arrayLength)

var areasBBVA = ["Salesforce", "Cells", "APX"];
var detalleareaBBVA = [areasBBVA[1], "15 Empleados", "Parques BBVA"]
console.log(areasBBVA);
console.log(detalleareaBBVA);
console.log(areasBBVA.length);
console.log(detalleareaBBVA[2]);
console.log(areasBBVA[areasBBVA.length - 1]) // Ultimo Elemento
```

Arrays: Propiedades y Métodos Útiles

Propiedades

- Array.length

Métodos

- pop(), push()
- forEach()
- some(), every(), map()
- reduce(), filter()
- reverse(), sort()
- slice(), concat(), join()
- indexOf()

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

- Notación Objeto

- Acceso a las Propiedades del Objeto

```
var celular = {
  marca: “Apple”,
  modelo: “iPhone XR”,
  color: “Gris”
};

celular.marca  // Apple
celular.modelo  // iPhone XR
celular.color  // Gris

celular[‘marca’]
celular[‘modelo’]
celular[‘color’]
celular.marca = ‘color’
celular[celular.marca]  // Gris
```

JS javascript.js

JS > JS javascript.js > ...

```
1   var celular = {
2       marca: 'Apple',
3       modelo: 'iPhone XR',
4       color: 'Gris'
5   };
6
7   console.log("Marca: " + celular.marca) // Apple
8   console.log("Modelo: " + celular.modelo)  // iPhone XR
9   console.log("Color: " + celular.color)  // Gris
10
11  console.log(celular["marca"])
12  console.log(celular["modelo"])
13  console.log(celular["color"])
14  celular.marca = "color"
15  console.log(celular[celular.marca])  // Gris
```

Elements Console Sources Network Performance

top Filter Default levels

```
Marca: Apple                    javascript.js:7
Modelo: iPhone XR               javascript.js:8
Color: Gris                     javascript.js:9
Apple                           javascript.js:11
iPhone XR                       javascript.js:12
Gris                            javascript.js:13
Gris                            javascript.js:15
```

Evidencia Taller

1. El programa genera un número aleatorio entre 1 y 100.
2. El jugador adivina el número seleccionando una serie de números.
    1. Se considera que cuando el usuario introduce un valor, éste es siempre válido.
    2. Con cada intento del jugador, el programa le dice si el número secreto es mayor o menor.
3. El jugador tiene 6 intentos para adivinar el número.

Summary

En este taller, has aprendido sobre

- Arrays
- Truthy and Falsy
- Primitivos
- Objetos
- Funciones