

X



Tipos de datos

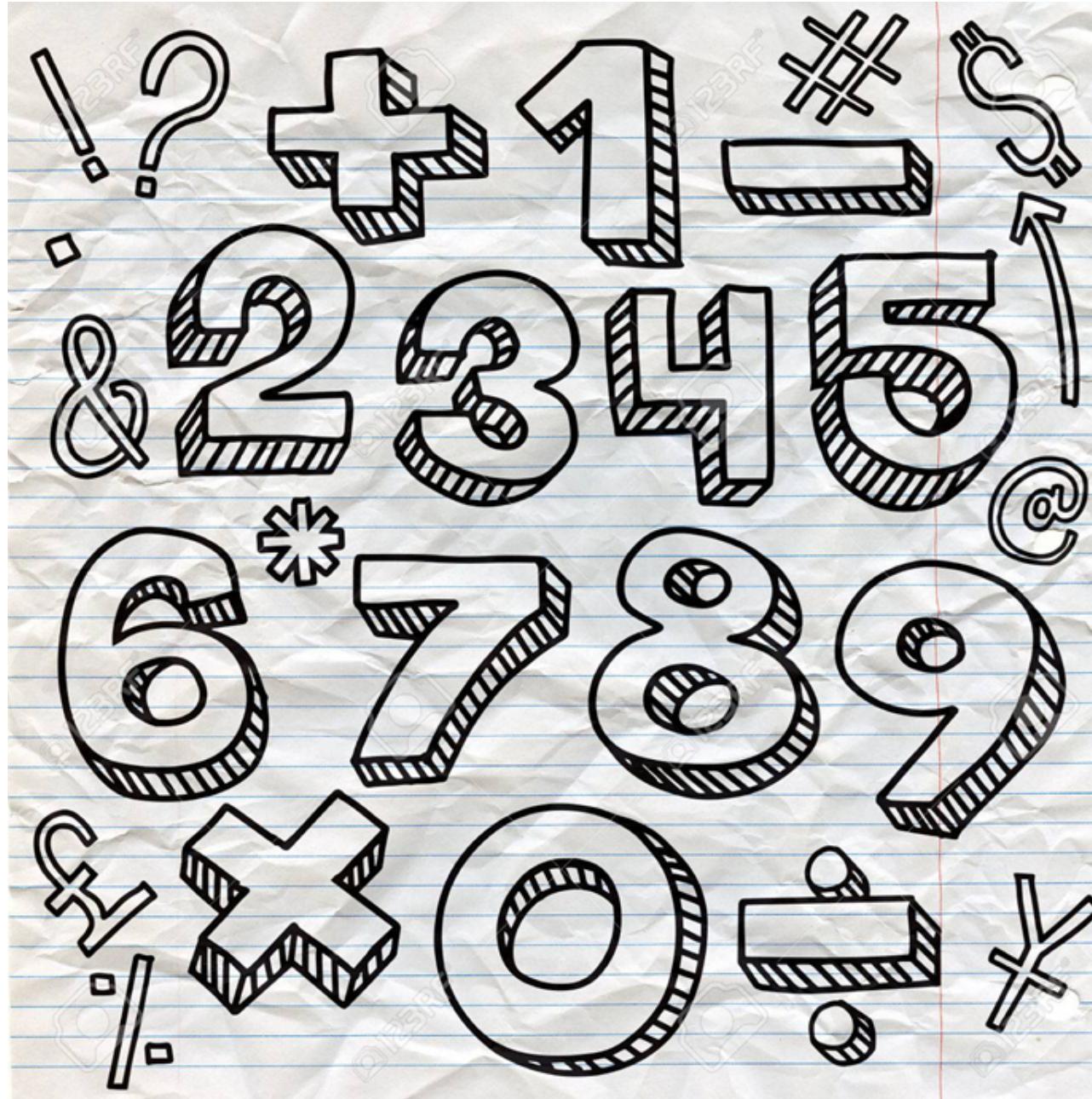
#ElDesafíoEsContigo

Cadenas de texto

! "#\$%& ' () *+, - ./
0123456789: ;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_`
`abcdefghijklmno
pqrstuvwxyz{|}~

```
JS app.js •  
Users > oscar.velandia > Desktop > JS app.js  
1 // Comillas dobles  
2 " Esto en un texto entre comillas dobles"  
3  
4 // Comillas simples  
5 ' Esto en un texto entre comillas simples'  
6  
7 // Backtick  
8 `Lorem Ipsum is simply dummy text of the printing and  
9 typesetting industry. Lorem Ipsum has been the industry's  
10 standard dummy text ever since the 1500s,  
11  
12
```

Números



```
JS app.js      X
Users > oscar.velandia > Desktop > JS app.js
1
2
3
4
5
6
7
      5
    7653739
0.14161825
-2.87
```

Valores lógicos

True

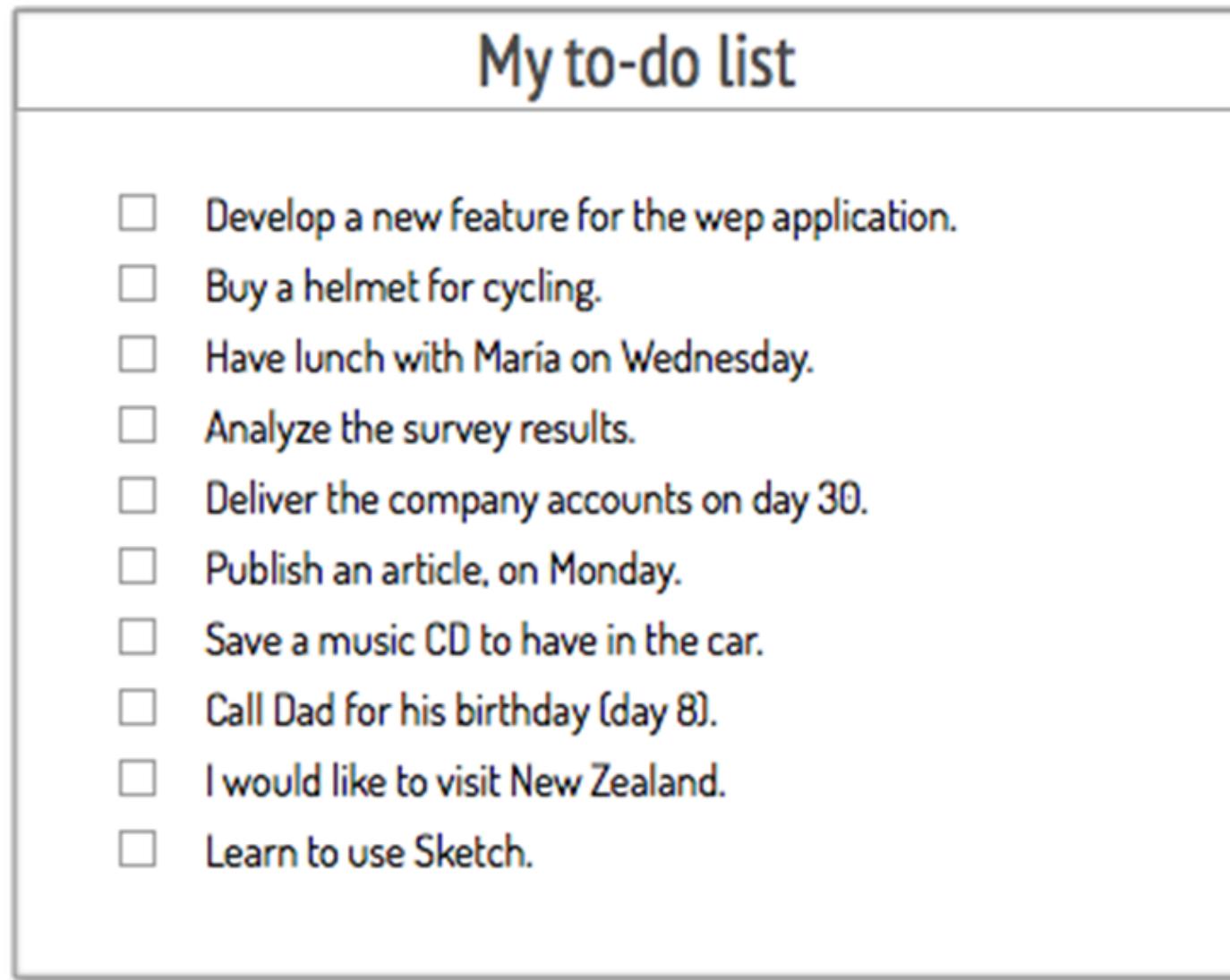


False



```
JS app.js
Users > oscar.velandia > Desktop > JS app.js
1
2
3 true
4
5 false
6
7
8 |
```

Arreglos de datos



```
JS app.js ●  
Users > oscar.velandia > Desktop > JS app.js  
1  
2  
3 [ 'mi tarea numero 1',  
4 1000,  
5 true  
6 ]  
7  
8  
9  
10
```

Objetos



```
1
2
3     {
4         nombre: "Oscar velandia",
5         numeroPasaporte: 1234567
6     }
7
8
9 |
```

X



Tipos de variables

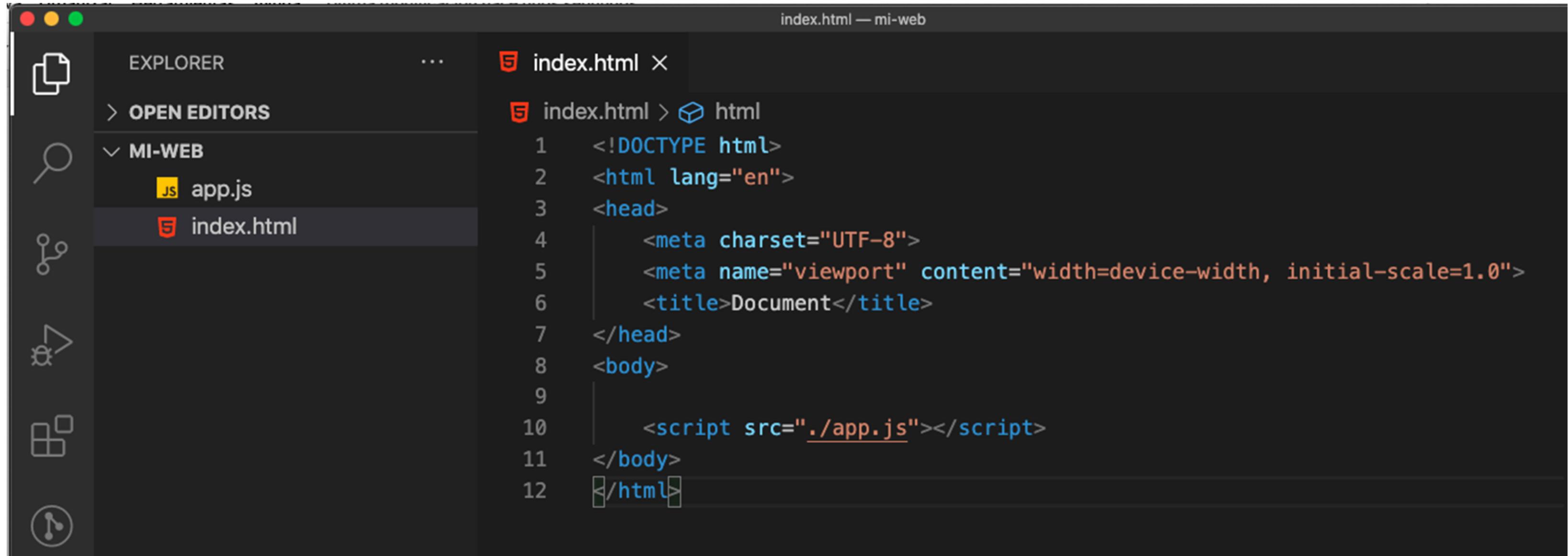
#ElDesafíoEsContigo

Tipos de variables



```
JS Untitled-1 •  
1  
2  
3 const miVariable = 'Soy el valor de una variable'  
4  
5 let colores = ['amarillo', 'negro', 'blanco']  
6  
7 const $ = 1000;  
8  
9 let _persona = {nombre: 'juan', edad: 22}
```

Usar javaScript desde HTML



The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there are icons for file operations, search, symbols, and other development tools. The 'OPEN EDITORS' section lists 'index.html' and 'app.js'. The 'MI-WEB' folder contains both files. The 'index.html' editor is active, displaying the following code:

```
index.html — mi-web
index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9
10  <script src=".//app.js"></script>
11 </body>
12 </html>
```

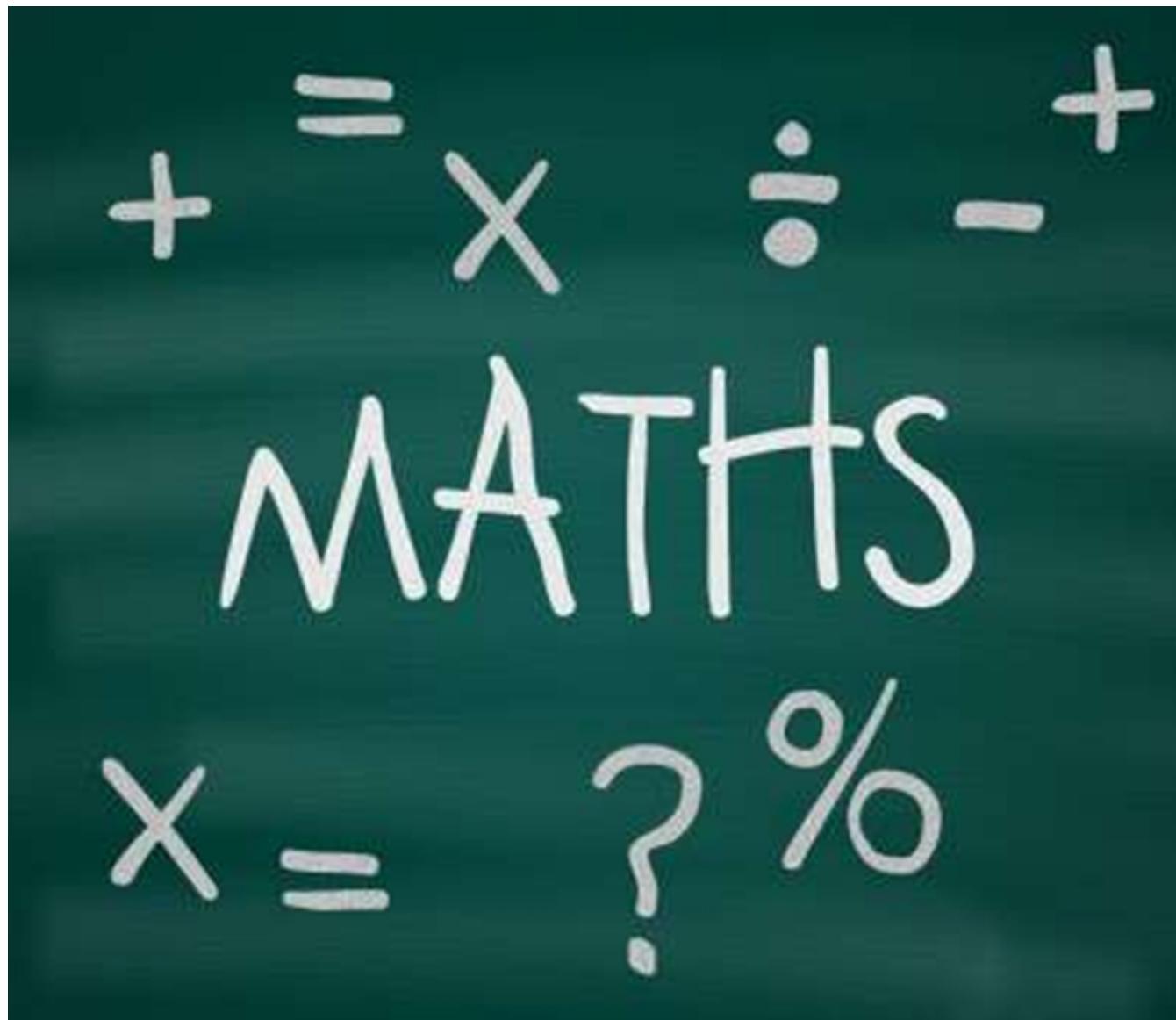
X



Operadores y
comparaciones

#ElDesafíoEsContigo

Operadores aritméticos



```
JS app.js
1
2
3 + Close (⌘W) Suma
4 - Resta
5 * Multiplicacion
6 / Division
7 %
8 ** Modulo
9
10
11
```

Operadores de comparación

True



False



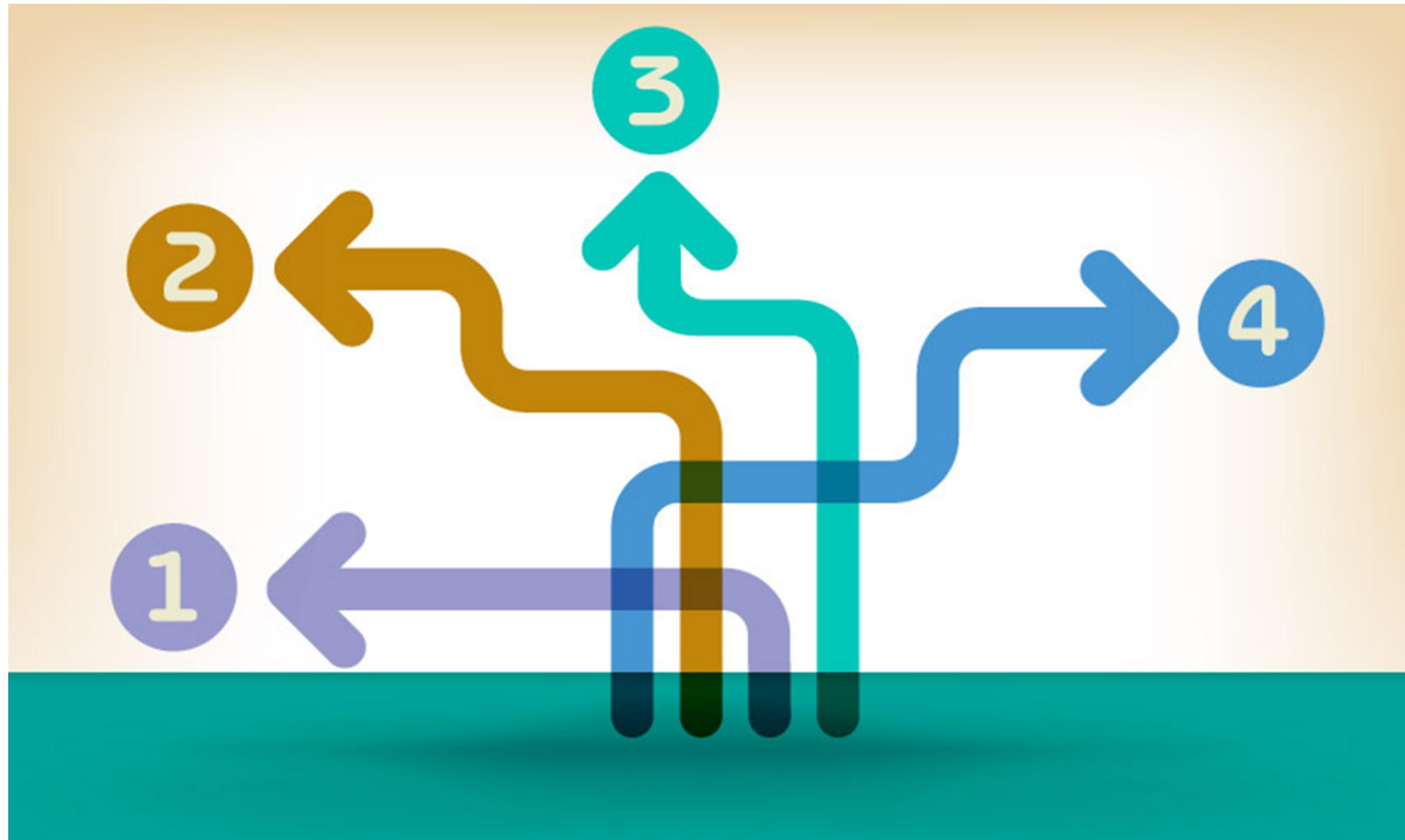
JS	app1.js	
1		
2		
3	>	Mayor que
4	>=	Mayor o igual que
5	<	Menor que
6	<=	Menor o igual que
7	==	Igualdad por valor
8	===	Igualdad tipo de dato
9	!=	Diferente de
10		



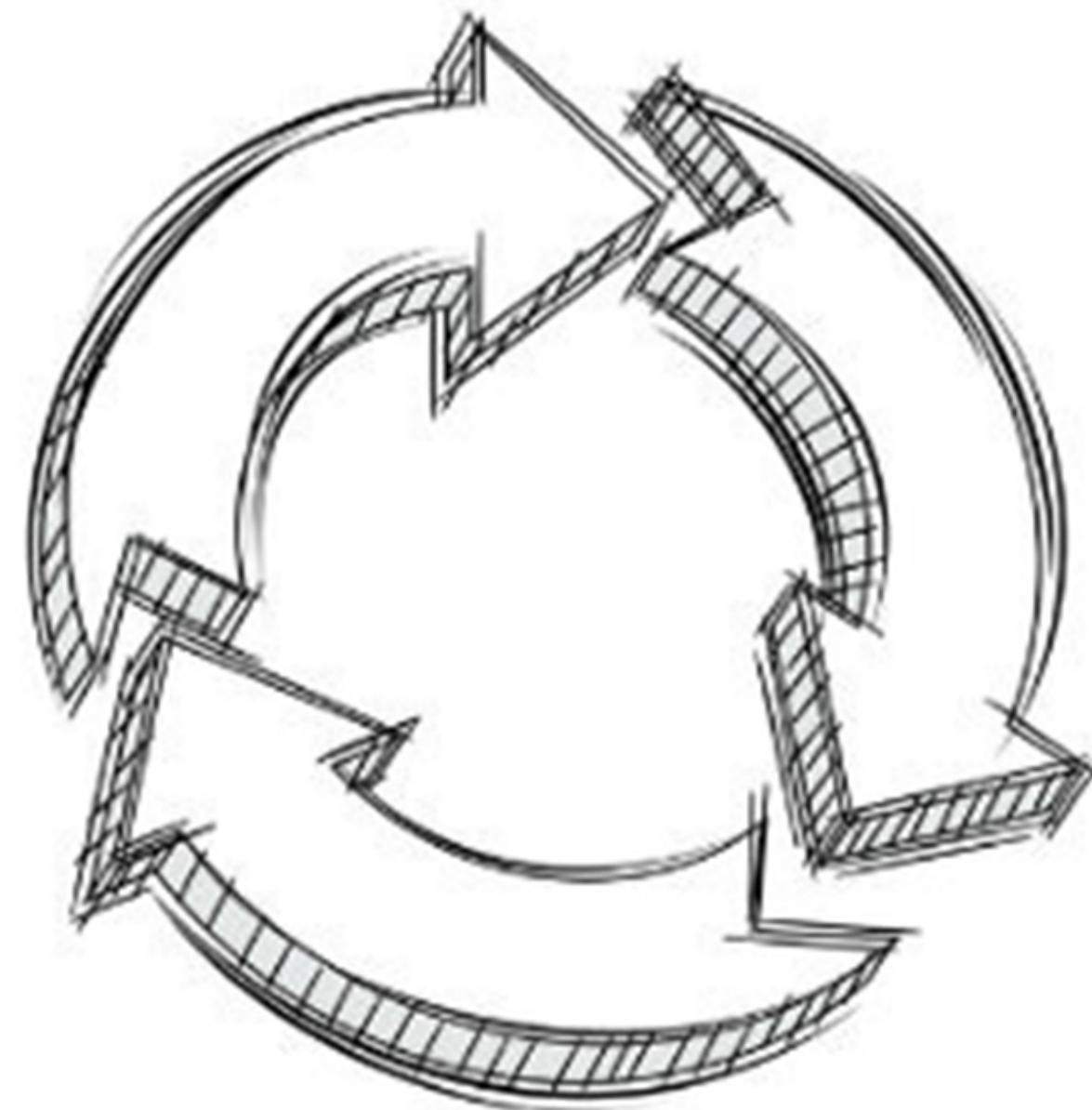
Ciclos y
condicionales

#ElDesafíoEsContigo

Acción por decisión



Procesos repetitivos



X



Objetos y
arreglos

#ElDesafíoEsContigo

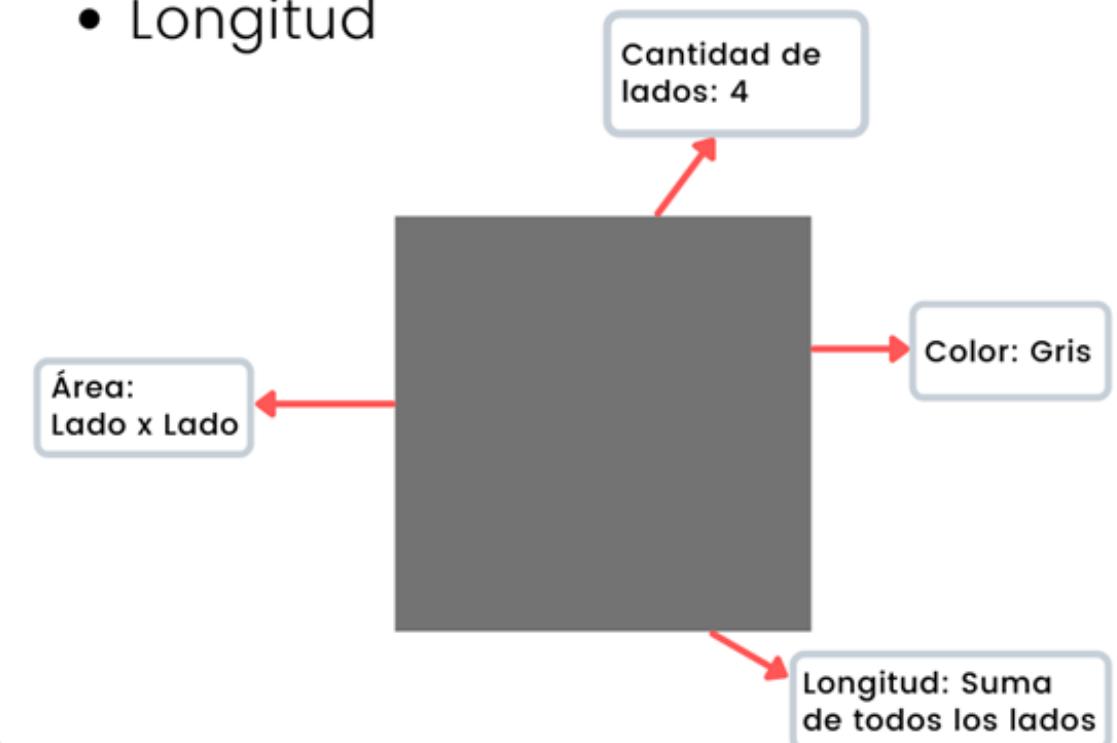
¿Qué es un objeto?

Los objetos son una estructura de datos que contiene unas propiedades con las que representa algo de la vida real.

Objeto figura

Propiedades:

- Cantidad de lados
- Color
- Área
- Longitud



¿Cómo crear un objeto?

```
const cuadrado = new Object();

cuadrado.cantidadLados = 4;
cuadrado.color = 'Gris';
cuadrado.area = lado => lado * lado;
cuadrado.longitud = lado => lado * 4;
```

```
const cuadrado = {
  cantidadLados: 4,
  color: 'Gris',
  area: lado => lado * lado,
  longitud: lado => lado * 4,
};
```

¿Cómo acceder a las propiedades?

```
// Notación de puntos  
console.log(cuadrado.cantidadLados);  
console.log(cuadrado.area(10));  
  
// Notación de corchetes  
console.log(cuadrado['color']);  
console.log(cuadrado['longitud'](10));
```

Operaciones con objetos

- `Object.getOwnPropertyNames()` : Es una función que nos retorna el nombre de las propiedades de un objeto dentro de un arreglo.
- `for... in` : Es una forma de recorrer el objeto tal y como si fuera un arreglo.

```
console.log(Object.getOwnPropertyNames(cuadrado));
// [ 'cantidadLados', 'color', 'area', 'longitud' ]

console.log(Object.keys(cuadrado));
// [ 'cantidadLados', 'color', 'area', 'longitud' ]

for (const key in cuadrado) {
  console.log(cuadrado[key]);
}
// 4 (cantidadLados)
// 'Gris' (color)
// [Function: area] (area)
// [Function: longitud] (longitud)
```

¿Qué es un arreglo?

Los arreglos son un tipo de dato estructurado, que tiene dentro de sí, todos los comportamientos que se le pueden dar a un conjunto de datos.

[**Arrays**]

JS

¿Cómo definir un arreglo?

```
const vestuario = ['Gorro', 'Camisa', 'Pantalón', 'Medias'];
```

¿Cómo acceder a sus valores?

```
// Indicar la posición exacta
console.log(vestuario[0]);

// El clásico for
for (let index = 0; index < vestuario.length; index++) {
    console.log(vestuario[index]);
}

// El foreach
vestuario.forEach((element, index) => console.log(element, index));
```

Operaciones con arreglos

```
// Agregar un elemento al final del arreglo  
vestuario.push('Guantes');

// Eliminar el último elemento del arreglo  
vestuario.pop();

// Añadir un elemento al inicio del arreglo  
vestuario.unshift('Bufanda');

// Eliminar el primer elemento del arreglo  
vestuario.shift();

//Encontrar el índice de un elemento del arreglo  
vestuario.indexOf('Camisa'); // 1
```

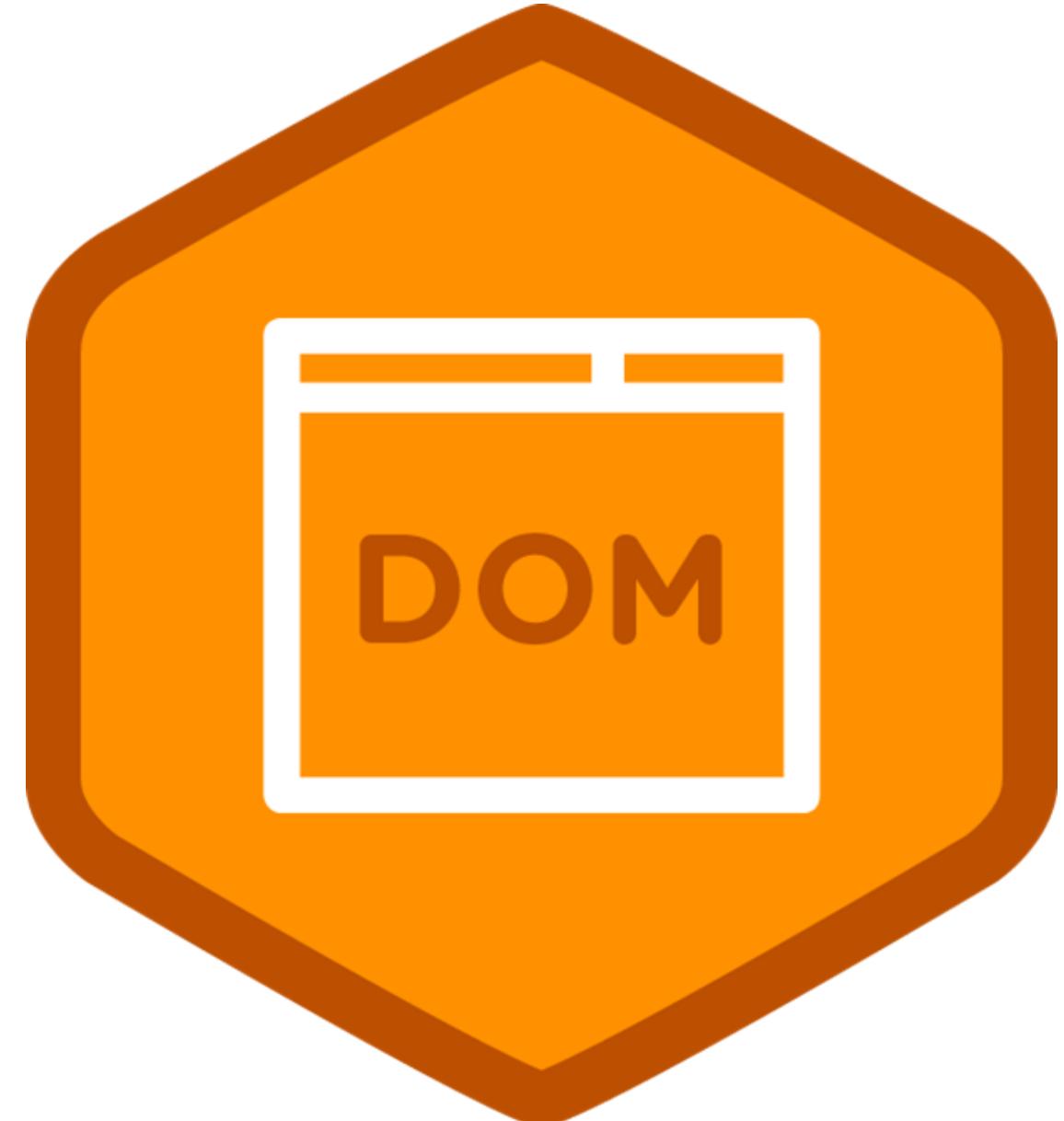


Eventos del DOM

#ElDesafíoEsContigo

Tipos de eventos

- eventos del ratón (MouseEvent):
mousedown, mouseup, click, dblclick,
mousemove, mouseover, mousewheel,
mouseout, contextmenu
- eventos táctiles (TouchEvent): touchstart,
touchmove, touchend, touchcancel
- eventos del teclado (KeyboardEvent):
keydown, keypress, keyup
- eventos de formularios: focus, blur,
change, submit
- eventos de la ventana: scroll, resize,
hashchange, load, unload



Capturar evento mousedown

```
document.getElementById('mousedown').  
addEventListener('mousedown', mousedown);  
  
function mousedown(e) {  
  console.log('Hiciste mousedown');  
}
```

Capturar evento mouseup

```
document.getElementById('mouseup').  
addEventListener('mouseup', mouseup);  
  
function mouseup(e) {  
  console.log('Hiciste mouseup');  
}
```

Capturar evento keydown

```
addEventListener('keydown', keydown);

function keydown(e) {
  console.log('Presionaste la tecla: ' + e.code);
}
```

Capturar evento click

```
document.getElementById('click').  
addEventListener('click', click);  
  
function click(e) {  
  console.log('Diste click');  
}
```



ECMAScript

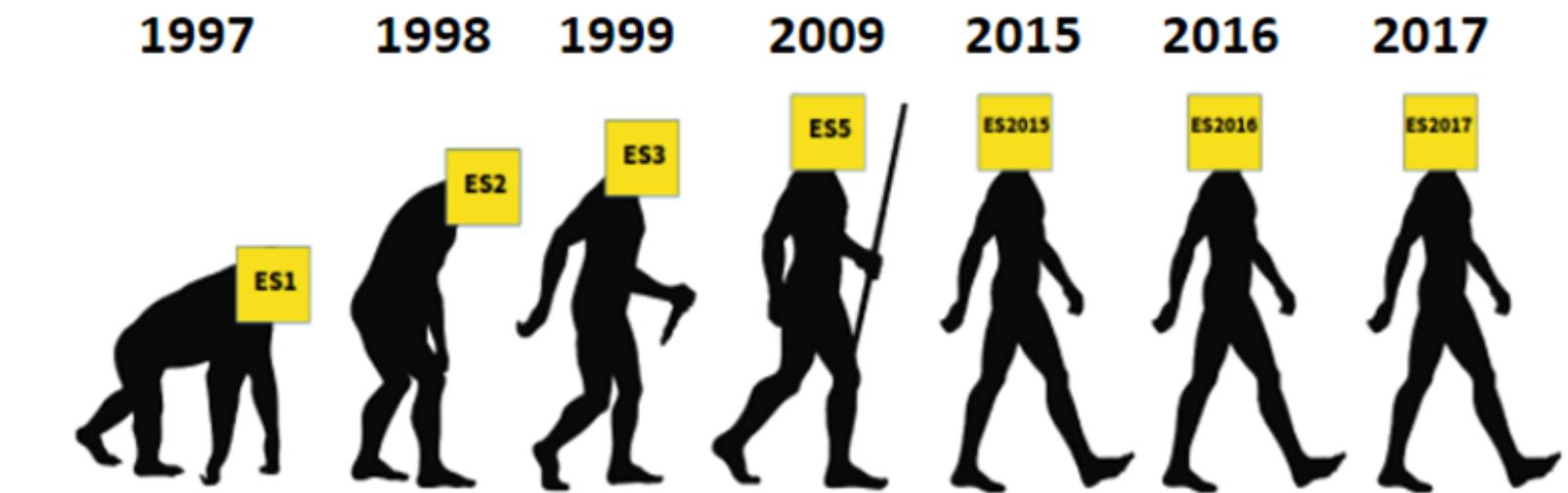
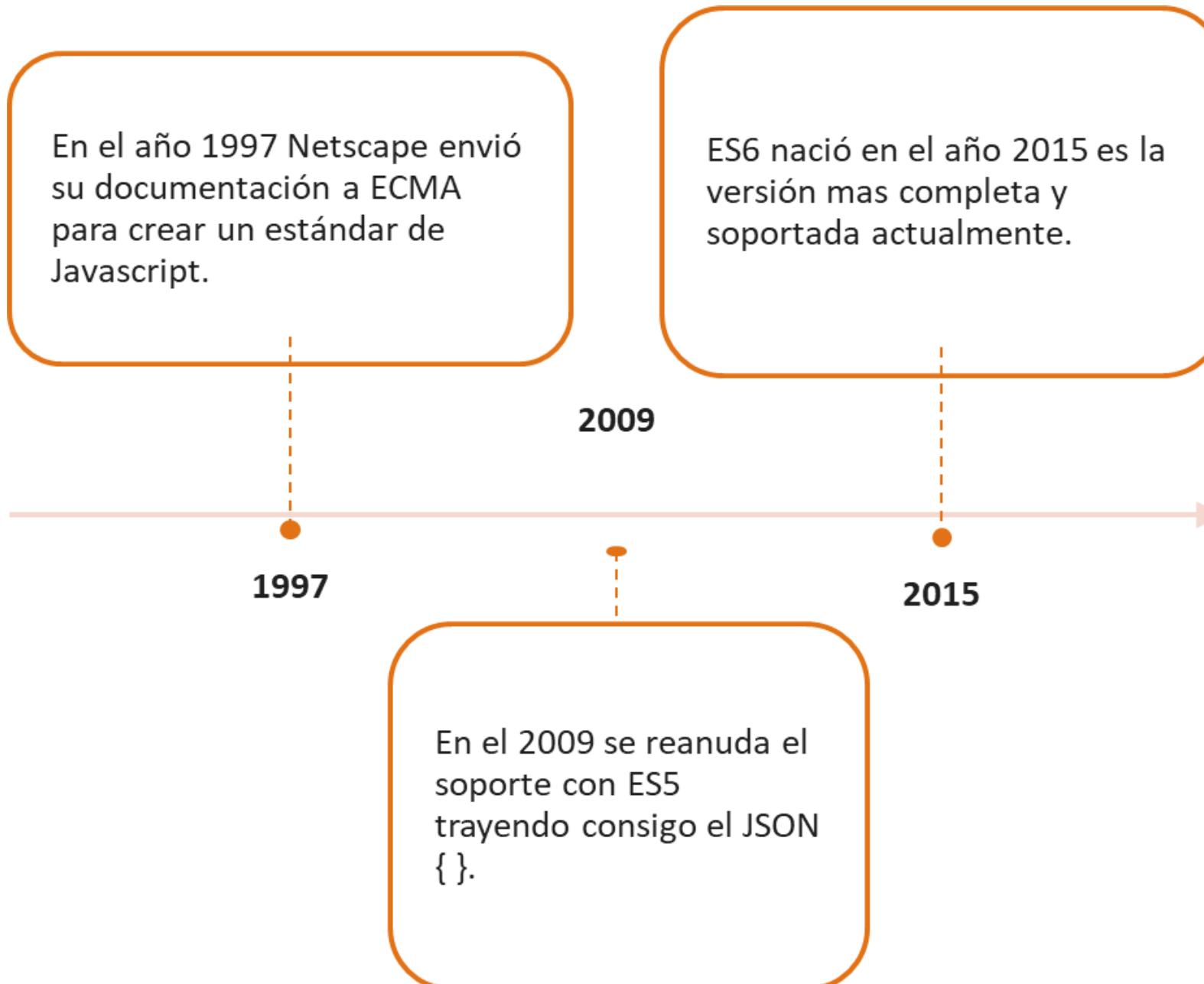
#ElDesafíoEsContigo

¿Qué es ECMAScript?

ECMAScript es el estándar que determina cómo emplear el lenguaje Javascript, permitiendo a los fabricantes de software desarrollar las herramientas adecuadas para interpretarlo correctamente.



ECMAScript



ECMAScript 6 (ES6)

Simplicidad

Funciones flecha y clases mas simples y elegantes.

Nuevos tipos de variables

Variables de bloque 'let' y verdaderas constantes 'const'.

Nuevas estructuras

Strings mas interesantes 'Template strings' e ir más allá usando Maps y Sets



Funciones flecha

La palabra 'function' ya no es necesaria y podemos en su lugar utilizar una flecha '`=>`'.

En caso de que la función solo reciba un argumento no necesitamos envolverlo en paréntesis.

En caso de que nuestra función tenga un valor de retorno en una sola línea las `{ }` y la palabra 'return' sobran.



```
// ES5  
let suma = function (a,b){  
    return a+b;  
};  
  
// ES6  
let suma = (a,b) => a+b;
```

Template strings ‘ ’

Son un tipo especial de cadena con un formato estructurado de acuerdo a unos marcadores o variables que permiten que esta cadena sea dinámica y se pueda reutilizar en las diferentes situaciones que pueda tener nuestro código.

Estructura:

- Uso de comillas invertidas.
- Inclusión de marcador o variables por medio del signo de dólar (\$) y llaves ({}).

```
var objeto={nombre:"cecilio"};
var plantilla= `hola que tal estas ${objeto.nombre}`;
console.log(plantilla);
```

Maps

Un Map es una estructura que permite asociar claves (keys) con valores (values) los cuales se pueden iterar en el orden en que fueron insertados.

Es una estructura muy parecida a los objetos, sin embargo, tienen algunas ventajas que hacen que en ocasiones sean mejor su uso.

Maps

- ✓ Las claves de un Mapa puede ser de cualquier tipo.
- ✓ Es mas fácil obtener el tamaño de un mapa por medio de su propiedad size.
- ✓ Conservan el orden de inserción.

Objetos

- ✓ Las claves de un objeto solo pueden ser cadenas.
- ✓ El tamaño de un objeto se realiza de forma manual.
- ✓ No conserva un orden de inserción

Sets

Un Set es una estructura que permite almacenar valores únicos de cualquier tipo incluyendo referencias a objetos y valores primitivos tales como string, int o boolean.

Métodos

✓ **Set()**: Creación de una nueva instancia

✓ **Add()** : Nos permite agregar un elemento al Set

✓ **Has()**: Retorna un valor boolean de acuerdo a la existencia o no de un elemento

✓ **Delete()**: Eliminar un item.

```
const mySet = new Set();
mySet.add(1);
mySet.add(5);
mySet.add('some text');
mySet.add({a: 1, b: 2});

mySet.has(1); // true
mySet.has(3); // false, 3 no existe

mySet.delete(5); // Elimina 5 del Set
mySet.has(5); // false, 5 fue eliminado
```



Fundación ROFÉ
- Impulsamos ideas de impacto social -



LEANSOLUTIONS
GROUP

PDU
PAYSANDÚ
TIERRA HEROICA



INTENDENCIA
DEPARTAMENTAL
DE PAYSANDÚ

sofka ▶
we make IT simple

EMpower

sumemos
Familias

Alcaldía de Envigado

Juntos
SUMAMOS
por Envigado

30
AÑOS

CÁMARA DE COMERCIO
ABURRÁ SUR

Fundación
Bolívar
Davivienda

ALCALDÍA DE
BARRANQUILLA

ALCALDÍA DE
SANTIAGO DE CALI



Alcaldía de Medellín

X



¡Gracias!

#ElDesafíoEsContigo