Fundamentos-de-JavaScript.pdf

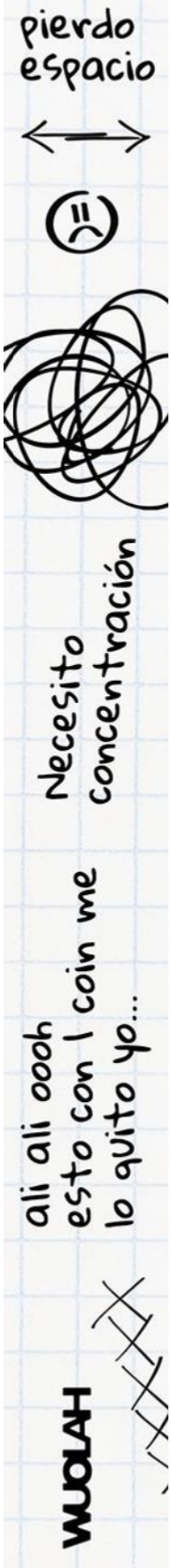
ShadowedWarrior

Lenguajes de marcas y sistemas de gestión de información

1º Desarrollo de Aplicaciones Multiplataforma

Estudios España

Accede al documento original



Fundamentos de JavaScript

```
/* VARIABLES */
let variable = 1;
let presentYear = "2024";
const minAge = 18; // Variable INMUTABLE --> NO PUEDE CAMBIARSE
// minAge = 12; // --> Al probar esto, se mostrará un error

// Ejemplo Hoisting ⬇
console.log(variableHoisting1);

var variableHoisting1; // Estas variables tienen reservadas un espacio en memoria al ejecutar el código y, si las leemos antes de llegar a su inicialización, se mostrará un "undefined" en vez de un error. Se debe evitar el uso de variables "var"

// console.log(variableHoisting2); --> Al probar esto, se soltará un error debido a que variableHoisting2 está con "let"

let variableHoisting2; // A diferencia de con "var", con "let" el intérprete suelta un error ya que no ha encontrado esa variable previamente declarada

/* ÁMBITOS */

/*  Tenemos varios motores que interpretan JavaScript:
    - V8 (Google)
    - Chakra (Microsoft)
    - Spider Monkey (Mozilla)
    - Core Webkit (Apple)

    JavaScript se aplica en estas situaciones:
    - Web (Cliente o Servidor)
    - Escritorio (Frameworks)
    - Movilidad (Frameworks que transpilan)
    - Videojuegos (Frameworks y librerías)

    Diferencia entre undefined (tipo de dato) y null (objeto):

- undefined: La variable está declarada pero no se le ha asignado un valor
- null: Se usa para indicar que la variable no tiene / deja de tener un valor */
```

```
/* TIPADO DINÁMICO */

/*  Las variables pueden cambiar de tipo;
    se determina en tiempo de ejecución

    Podemos saber el tipo haciendo uso de "typeof"

    Se pueden hacer castings de esta forma:
    Number(variable) --> Transforma a número o a NaN
    Boolean(variable) --> Si hay contenido true / Sino, false (si hay
cadena o hay un nº diferente a 0)
    String(variable) --> Transforma el contenido a cadena
    variable = null --> Deja de haber valor
    variable = undefined
*/

let address = "Carril de las Castañetas, 36";
let price = 2.33;
let active = false;
let sample;
let sampleNull = null;

/*
console.log(typeof address);
console.log(typeof price);
console.log(typeof active);
console.log(typeof sample);
console.log(typeof sampleNull); */

let booleanAddress = Boolean(address);
let numberAddress = Number(address);

console.log(booleanAddress);
console.log(numberAddress);

/* FUNCIONES - También provocan hoisting como var */

function printGreet() {
    console.log("Hello, World!");
}

printGreet();
```

```
function getGreet() {
    let greet = "Hello, World!";
    return greet;
}

console.log(getGreet());

function sum(num1 = 0, num2 = 0) { // 0 <-- Le damos un valor por
defecto si no se pasa valor
    let sum = num1 + num2;
    return sum;
}

/* Tipos de funciones */
/* Anónimas --> Se usan cuando no vamos a reutilizar el código pero
queremos encapsular */
let func = function (parametroUno) {
    return parametroUno + ": ";
}

console.log(func("Opción") + 1);
// Estas se pueden pasar como parámetros (callback) y estas provocan un
comportamiento diferente en la función que las contiene
function getCopyRight(name, year, callback) {
    return callback(name, year); // llama a la función que se llame
como el callback pasado y devuelve el resultado de esta
}

let formatWithPipe = function(name, year) {
    return name + " | " + year;
}

getCopyRight("David", 2024, formatWithPipe); // Podríamos usar
cualquier función anónima que trabaje con los parámetros
// También se pueden declarar estas funciones anónimas en la llamada a
la función

/* Funciones autoinvocadas */
(function(name, year){
    console.log(name, year);
}
)("David", 2024); // Estas funciones se ejecutan de manera inmediata
                  // Pueden no tener parámetros
```


```
/* Condicionales */

let hasPizza; // --> Es undefined si no le damos valor

if (hasPizza == true) {
    console.log("Tengo pizza");
} else if (hasPizza == false) {
    console.log("No tengo pizza");
} else {
    console.log("Tengo que mirar si hay pizza en la nevera");
}


/* Ámbitos */

// Tenemos ámbito global, de función y de bloque
let global;

function comprobacionAmbito() {
    console.log(global);
    let funcion;
    if (global === undefined) {
        let bloque;
    }
}

// Comentarios --> Tenemos 3 tipos

// En línea
/* En bloque */
// #! <-- Al inicio del fichero e indica dónde se encuentre el
intérprete de JS

// Interfaces del Web API

/*
    Tenemos 4:
    - Windows (ventana que contiene DOM)
    - Document (el mismo DOM)
    - Event (evento en el DOM)
    - Element (nodo del DOM)
*/

// Eventos
```

```
let elemento = document.querySelector("input");

elemento.addEventListener("click", function (event) {
    console.log("Has hecho click")
})

window.addEventListener("copy", function (event) {
    event.preventDefault();

    console.warn("Intento de copia de imagen, será notificado");
});

// Podemos añadir el evento directamente -->

document.querySelector(".test").addEventListener("click",
function(event) {
    alert("Se ha hecho un click sobre el botón")
});

document.addEventListener("contextmenu", function(event) {
    event.preventDefault();
});

// También podemos evitar que el evento se propague hacia los padres
document.querySelector(".test").addEventListener("contextmenu",
function(event) {
    event.stopPropagation(); // <-- Usando esta función
});
```