UD2 – Básicos JavaScript

DAW2-DWEC

Datos curiosos

No se llama JavaScript. Su nombre real es ECMASCript o ECMA-262

Versiones nueva casi todos los años.

Épocas importantes.

- Primeras versiones.
- ECMAScript 5th edit.(ES5) -> ‘strict mode’, mejoras en arrays y objetos.
- ECMAScript 2015 (ES6) -> Versión de referencia actual, código moderno.
- Versiones posteriores -> Amplían el lenguaje poco a poco. Soporte mediante “Polyfill”.

Características

Interpretado.

Basado en prototipos. Los objetos se definen en base a plantillas.

Case Sensitive.

Débilmente tipado y gestión de tipos dinámico.

Monohílo y asíncrono.

Etiqueta <script>

Permite incluir código JavaScript en el HTML

Se puede insertar dentro del código o en la cabecera.

- Nos gustan los archivos externos de código JS.
- Atributo opcional “defer”

Siempre debe incluir etiqueta de cierre. => PUM!!!! `<script src="app.js" />`

```
<body>
    Hola clase!
    <script>
        console.log('Hola mundo!');
    </script>
</body>
```

```
<head>
    <meta charset="UTF-8">
    <title>Demos JavaScript</title>
    <script src="app.js" defer ></script>
</head>
```

Comentarios y variable

Comentarios

- Línea -> //
- Bloque -> /* */
- Formateados VS Code -> /** */

Declaración de variables

- var -> NO USAR
- let
- const

Recuerda el valor “**undefined**”, el tipado débil y dinámico.

Tipos de datos

Tipos de datos primitivos

- String -> mejor "", '', ``
- Boolean -> mejor true y false
- Number -> mejor 42 o 3.14159
- BigInt
- Symbol()
- null
- undefined

Tipos de datos no primitivos

- Object

Valores especiales

- Nan
- Infinity, -Infinity

Cadenas de texto

Son un array de caracteres.

3 formatos

- Comillas dobles -> NO NOS GUSTA
- Comilla simple o apostrofe
- Comilla invertida (plantillas de texto) -> admiten expresiones ${código JS}

```
elementoHtml.innerHTML = '<p class="destacado">Piensa en todo
dobles</p>';
```

```
const hostCliente = 'www.example.com';
let productoId = 23;
let direccion = `${hostCliente}/productos?productoId=${productoId}`;
```

Operadores

Aritméticos

- +, -, *, /, %, **
- ++variable, --variable, variable++, variable--

Asignación

- =, +=, -=, *=, /=, **=

Comparación

- >, >=, <, <=, ==, !=
- ===, !== -> ESTOS NOS GUSTAN

Operadores - II

Lógicos

- &&, ||, !

Short-circuit

- expresion1 && expresion2. Ejecuta de izquierda a derecha y devuelve la última expresión “truthy”.
- expresion1 || expresion2. Ejecuta de izquierda a derecha y devuelve la primera expresión “truthy”.
- expresion1 ?? valorPorDefecto.

Valores “falsy”

Lista de valores “falsy”

- null
- undefined
- false
- NaN
- 0, 0.0, 0x0
- -0, -0.0, -0x0
- 0n. NOTA: no existe -0n.
- ‘’. Cadena vacía.
- document.all

RECUERDA: Todo lo que no es “falsy” es “truthy”.

Estructura de control - if

Si se cumple la condición se ejecuta el código

Permite concatenar condicionales mediante “else” y “else if”

```
if (condicion) {
// Código a ejecutar si la condición es verdadera
}
```

Estructura de control - switch

Para cuando if else se nos va de las manos…

```
switch (a) {
   case 1:
       break;
   case 2:
      break;
   default:
}
```

Operadores ? Y ??

Operador ternario ?

```
let resultado = condicion ? expresionSiVerdadero : expresionSiFalso;
```

Operador coalescencia nula ??

- Valida que el valor propuesto no sea null o undefined

```
let resultado = valorPropuestro ?? valorPorDefecto;
```

Bucles for, for...of, for...in

for – cuando queremos controlar el número de iteraciones.

```
for (inicialización; condición; actualización) {…}
```

for...of – cuando queremos recorrer todos los valores de una colección.

```
for (variable of iterable) { …}
```

for...in – cuando queremos recorrer las claves de una colección clave-valor.

```
for (let llave in objeto) { … objeto[llave] … }
```

for...in – cuando queremos recorrer los índices de un array.

```
for (let index in array) { … array[index]}
```

Bucles while y do … while

while - Permite iterar mientras se cumple una condición (0:n veces).

```
while (condición) {
…
}
```

do…while – Itera hasta que no se cumple una condición (1:n veces).

```
do {
…
} while (condición);
```

Instrucciones break y continue

break – salta fuera del bucle

continue – salta a la siguiente iteración del bucle

```
for (let i = 0; i < 10; i++) {
    if (i === 5) break; // Detiene el bucle cuando i es 5
    if (i === 3) continue; //Salta a la siguiente iteración cuando i es 3
    console.log(i);
}
```

Arrays y JSON

Arrays

- Se definen con: [] o new Array(tamaño) o [lista valores]
- Se accede a un elemento a través del índice: colección[índice]
- Propiedad .length devuelve el tamaño del array

Objetos JSON

- Forma sencilla de tener una colección clave-valor (está también Map)
- Puedo acceder a las propiedades mediante notación punto (.) notación corchetes ([clave])
- El operador “delete()” permite borrar una propiedad

Funciones

Disponemos de varios tipos.

Función con nombre. **function miFuncion(){}**

- Parámetros por defecto. p1=valor, p2=valor
- Variable “arguments”
- Parámetro …rest

Función anónima. **function(){}**

- Ideal para callbacks

Arrow functions. **()=>{}**

- CUIDADIN CUIDADIN, se come “this” en POO

Otros tipos de funciones

Closures

- Función dentro de otra función.
- Permite tener un estado global a la función interna.

Función autoinvocada IIFE

- Se invoca automáticamente al definirse
- Codo su código interno queda encapsulado

```
function crearContador() {
    let contador = 0;
    return function () {
        contador++;
        return contador;
    };
}
```

```
(function () {
    // Código dentro de la función
    console.log("Esta función se autoinvoca.");
})();
```

Modo estricto – ‘use strict’

El modo estricto evita que cometamos errores comunes de programación.

Se puede definir a nivel script o a nivel de función.

Algunos ajustes de este modo son:

- No poder usar variables sin declararlas.
- No poder usar la instrucción “delete”.
- No poder usar palabras reservadas del lenguaje como nombres de variable.
- No poder definir dos parámetros con el mismo nombre “fn(p1, p1)”.
- En POO no escribir en propiedades de sólo lectura ni leer en propiedades de sólo escritura (getters y setters).
- La función “eval()” no puede definir variables por seguridad.
- El objeto “this” cuando se usa en funciones ahora es “undefined”.

Preguntas