# UD3

## INTRODUCCIÓN A JAVASCRIPT

MP_0373
Lenguaje de marcas y Sistemas de gestión de la información

3.1 JS

# Introducción

Un programa no deja de ser una colección de instrucciones que ejecuta un dispositivo como un ordenador o un teléfono móvil una detrás de otra. Por muy complicado que sea un programa, al final se reduce a operaciones simples como sumar números, escribir un valor en memoria o recuperarlo.

Un **lenguaje de programación** es ese idioma en el que se escriben dichas instrucciones: el programador se comunica con el dispositivo utilizando un lenguaje, en este caso de programación. JavaScript es un lenguaje de programación orientado a web, antiguamente se ejecutaba únicamente en navegadores, pero poco a poco ha conseguido ser full stack, es decir, se puede utilizar JavaScript como lenguaje de programación en todas las capas del desarrollo web.

## Añadir JavaScript a un documento HTML

Normalmente si un navegador encuentra código JavaScript lo ejecutará nada más encontrarlo (con determinadas técnicas avanzadas este comportamiento se puede alterar). Para que un navegador encuentre código JavaScript este debe estar añadido a un documento HTML.

Las maneras de añadir código JavaScript son muy similares a las maneras de añadir código CSS a un documento HTML:

### Código online

Se puede añadir código JavaScript directamente a atributos de elementos HTML, esta manera de añadir código JavaScript no es nada recomendable:

```
<!-- el navegador mostrará un aviso al pulsar sobre el enlace -->
<a href="#" onclick="alert('Hola Mundo')">Pulsar</a>
```

### Dentro de una etiqueta script

HTML dispone de la etiqueta script que tiene la posibilidad de uso de contener JavaScript dentro de ella. En cuanto el navegador encuentre esa etiqueta, si encuentra código dentro de ella empezará a ejecutarlo.

Esta etiqueta puede ser añadida dentro de la etiqueta head o dentro de la etiqueta body. Si se añade dentro del head seguro que el código JavaScript es ejecutado antes de que se empiece a

mostrar la página en el navegador (lo cual puede ser interesante o no dependiendo de las necesidades de ese código JavaScript).

```
<body>
  <h1>...</h1>
  <script>
    alert("hola");
  </script>
</body>
```

**Enlazado desde un fichero .js**

Enlazar el código JavaScript desde un fichero externo es la forma recomendada de añadir código JavaScript a un documento HTML. De esta manera se separa el código JavaScript del contenido del documento HTML y puede ser reutilizado en diferentes páginas.

Un archivo JavaScript (al igual que uno HTML o CSS) es simplemente un fichero de texto plano con extensión .js que contiene código JavaScript. Para enlazar un archivo JavaScript se utiliza la etiqueta script:

```
// fichero index.js
alert("Hola Mundo");

<body>
  <h1>...</h1>
  <script src="index.js"></script>
</body>
```

## Sentencias

Cada una de las instrucciones que forman un programa se llama *sentencia*. Un programa será entonces una lista de sentencias.

Cada sentencia en JavaScript debe ser terminada con punto y coma y por mejorar la legibilidad del código (salvo excepciones) debe escribirse una sentencia por línea.

```
  const a; // Sentencia de declaración de variable a
  const b; // Sentencia de declaraciónd de variable b
  a = 3; // Sentencia de asignación del valor 3 en la variable a
  b = 2; // Sentencia de asignación del valor 2 en la variable b
  alert(a + b); // Sentencia de función alert que mostrará una alerta con el contenido '5'
```

## Funciones de salida

JavaScript por sí mismo puede mostrar datos cuando se ejecuta en un navegador de las siguientes maneras:

### Consola del navegador

Una de las maneras más utilizadas para mostrar datos directamente desde JavaScript con propósito de desarrollo. Los navegadores en sus herramientas de desarrollo incluyen la consola JavaScript mediante la cual se pueden mostrar datos desde código a través de la función console.log.

```
<body>
  <script>
    /* En la consola del navegador aparecerá el valor 5 */
    console.log(2 + 3);
  </script>
</body>
```

### innerHTML

Gracias a esta función que tiene todos los elementos HTML se puede escribir dentro de ellos a través de JavaScript.

```
<body>
    <h1>...</h1>
    <p id="salida"></p>
    <script>
      /*
```

```
    Este código selecciona el elemento HTML con id salida
    y dentro le escribe el resultado de 2 + 3
    */
    document.getElementById("salida").innerHTML = 2 + 3;
  </script>
</body>
```

### Document.write()

Únicamente recomendado para desarrollo, con esta función JavaScript se puede escribir directamente en el documento, lo que se escriba a través de esta función aparecerá después del contenido que tenga body.

```
<body>
  <h1>...</h1>
  <script>
    document.write(2 + 3);
  </script>
</body>
```

### Window.alert()

Esta función dispara una ventana emergente a través del navegador. Habitualmente los navegadores muestran una alerta mostrando la información y con un botón para que el usuario pueda cerrar dicha alerta.

```
<body>
  <h1>...</h1>
  <script>
    window.alert(2 + 3);
  </script>
</body>
```

Se puede utilizar directamente alert() en lugar de window.alert() ya que en los navegadores el objeto window corresponde con el ámbito global.

## Comentarios

Como cualquier lenguaje de programación, JavaScript incluye la posibilidad de añadir comentarios a su código.

Los comentarios son caracteres que no se ejecutarán y son útiles para documentar el código, explicar funcionalidades a futuros desarrolladores u ocultar al intérprete de código partes del programa mientras se está desarrollando. El código debe ser los suficientemente claro para que se pueda explicar por sí mismo, si esto no es posible se debe recurrir a los comentarios.

JavaScript tiene dos maneras de escribir comentarios: comentarios de **una línea** que empiezan con // o comentarios de **bloque** que empiezan con /* y acaban con */.
Siempre que sea posible conviene utilizar los comentarios de línea, a no ser que realmente se esté comentando un bloque grande de código y/o texto que se utilizarán los comentarios de bloque para ello.

```
let b; // Declaración de variable b
let a = 3; // Asignación del valor 3 en la variable a /*
Este código muestra la suma de dos números que se han declarado antes en una alerta
del navegador.
*/
a = 3;
b = 2;
alert(a + b);
```

## Tipos de datos

El tipo de dato es un concepto muy común en programación, como su propio nombre indica, son las diferentes clases de datos que JavaScript puede manejar. Cada tipo de dato tiene sus características y operaciones permitidas, por ejemplo, se pueden multiplicar dos datos que sean números, sin embargo, no se pueden multiplicar cadenas de texto.

JavaScript maneja los siguientes tipos de datos:

- Cadenas de texto.
- Números.
- Booleano.
- Array.
- Objetos.
- Indefinido (undefined).

```
const string = "Hola mundo!";
const number = 3;
const boolean = true;
const array = [1, 2, 3];
const object = {
  name: "Gerardo",
  lasname: "Fernández"
};
let indefinido;
```

Gracias a la palabra reservada typeof se puede preguntar a JavaScript por el tipo de dato de una variable.

```
typeof "Hola Mundo!"; // Devuelve 'string'
typeof 3;             // Devuelve 'number`
```

El tipo de datos especial undefined significa que esa variable todavía no alberga nada o lo que albergaba se ha borrado pero la variable sigue definida:

```
let a;
typeof a; // undefined
a = 3;
typeof a; // number
a = undefined;
typeof a // undefined
```

### Tipado débil

Comprobar el tipo de variable puede ser útil en determinados casos ya que **JavaScript es un lenguaje de programación de tipado débil**, esto quiere decir que para declarar una variable no hace falta indicar el tipo de dato que va a albergar y en la vida de esa variable el tipo de dato puede cambiar:

```
  // Simplemente se declara la variable a, no hace falta indicar qué tipo de dato va a albergar, en otros lenguajes de programación si hace falta
  let a;
  typeof a; // undefined
  a = 3;
  typeof a; // number
  a = "3";
  typeof a; // string
```

Este tipado débil puede acarrear problemas y comportamientos inesperados ya que JavaScript al poder cambiar el tipo de una variable sin consultar con el programador, puede dar lugar a resultados que no son intuitivos:

```
  const a = 3;
  const b = 2;
  console.log(a + b); // 5
  typeof a + b; // number

  const c = "3";
  console.log(c + b); // 32
  typeof c + b; // string
  /* En este caso JavaScript ha hecho lo posible por sumar un número y una cadena de texto, como eso no es posible lo que ha hecho ha sido transformar en cadena de texto el número y concatenar las 2 cadenas de texto, dando resultado a otra cadena de texto */

  console.log(
 + true + true) // 2
  /* JavaScript no puede sumar booleanos, ha decidido convertirlos en números de tal manera que
 = 0 y true = 1, por lo que 0 + 1 + 1 = 2 */
```

Conocer estos comportamientos de JavaScript es necesario para entender el lenguaje, pero no es recomendable utilizarlos a propósito ya hacen que el código sea difícil de entender y de peor mantenimiento. Una buena práctica es el chequeo del tipo de variable cuando se hacen operaciones que puedan provocar estos comportamientos.

## Variables

Una variable permite almacenar valores para poder ser leídos en otro momento. Una variable en JavaScript se declara con la palabra reservada: let y el identificador de la variable.

```
let a; // Declaración de variable
  a = 3; // Asignación de valor en variable
  console.log(a); // la consola mostrará: 3
  a = 2; // Reasignación de valor en variable
  console.log(a); // la consola mostrará: 2
```

El identificador debe ser único, es decir, no puede haber dos variables con el mismo identificador/nombre. Conviene utilizar identificadores cortos, pero a la vez descriptivos para hacer la labor de desarrollo más sencilla. Al elegir un identificador, además se debe saber que:

- Se puede utilizar cualquier combinación de letras, números y el guion bajo.
- El primer carácter de un identificador no puede ser un número.
- JavaScript es case-sensitive, es decir, distingue entre minúsculas y mayúsculas por lo que la variable nombre y Nombre son dos variables distintas al tener distinto identificador.
- No se pueden utilizar palabras reservadas del lenguaje como identificador de variable, es decir, aquellas palabras que el lenguaje utiliza para sí mismo como this, let, const, if, etc.

Se puede declarar una variable y asignarle un valor en la misma sentencia:

```
let a = 3;
```

Si el valor de la variable no va a cambiar a lo largo de la ejecución, es conveniente utilizar la palabra reservada const para declararla.

```
const pi = 3.14;
pi = 33; // Esta sentencia dará un error ya que no puede reasignarse una variable declarada utilizando const
```

## Operadores

Los operadores permiten hacer diferentes operaciones con variables.

### Operadores aritméticos

Estos operadores permiten realizar operaciones aritméticas básicas, algunos de ellos son:

- Suma: +
- Resta: -
- Multiplicación: *
- División: /
- Resto: %: Devuelve el resto de una división entera.

```
const a = 3;
const b = 2;
let result;

result = a + b;
console.log(result); // 5
result = result - a;
console.log(result); // 2
result = result * b;
console.log(result); // 4
result = result / b;
console.log(result); // 2
```

### Operadores comparación

Estos operadores permiten comparar dos expresiones. JavaScript hará lo posible por compararlas y devolver un valor booleano: true o false.

- == igual.
- === igual estricto (compara tipo de la variable).

- != distinto.
- !== distinto estricto (compara tipo de la variable).
- > mayor que.
- < menor que.
- >= mayor o igual.
- <= menor o igual.

```
const a = 3;
const b = 100;

console.log(a == b); //

console.log(a != b); // true
console.log(a > b); //

console.log(a < b); // true

const c = 3;
const d = "3"; // Tipo cadena de texto, no número

console.log(a == c); // true
console.log(a === c); // true
console.log(a == d); // true
console.log(a === d); //
```

### Operadores lógicos

Estos operadores permiten realizar operaciones lógicas, JavaScript hará lo posible por comparar los valores y realizar la operación lógica entre ellos.

- Operador lógico and: &&.
- Operador lógico or: ||.
- Operador lógico not: !.

```
const a = true;
const b = false;
const c = true;
```

```
console.log(a && b); //
console.log(a || b); // true
console.log(!c); // true
```

### Operadores asignación

Con un operador de asignación se asigna el valor de la derecha en el operador de la izquierda. Por ejemplo: x = 3 se asigna el valor 3 a x.

Algunos operadores de asignación:

- =: Asignación simple: x = 3.
- +=: Asignación de suma: x += 3 es lo mismo que: x = x + 3.
- -=: Asignación de resta: x -=3 es lo mismo que: x = x - 3.

### Operadores cadena

El operador + si es utilizado con una cadena de texto no realizará una suma (ya que no es posible sumar cadenas de texto), si no que realizará una concatenación:

```
const hello = "Hola";
const world = "Mundo";
const greeting = hello + " " + world + "!";
console.log(greeting); // "Hola Mundo!"

const anumber = 3;
const helloNanumber = hello + anumber;
console.log(helloNanumber); // "Hola3
```