«Talento Tech»

Front-End JS

Clase 09

Buenos Aires Ciudad

Clase N° 09: Introducción a JavaScript

1. **Introducción a JavaScript**
    - ¿Qué es JavaScript?
    - Características de JavaScript
    - Versiones de ECMAScript
2. **Primeros Pasos con JavaScript**
    - Incorporación de JavaScript en HTML
    - Uso de la Consola del Navegador
    - Comentarios en JavaScript
3. **Variables y Constantes en JavaScript**
    - Declaración de Variables
    - Reglas para Nombrar Variables
    - Modificación de Variables
    - Declaración de Constantes
4. **Tipos de Datos en JavaScript**
    - Tipos de Datos Primitivos
    - El Objeto Number
    - Conversión Numérica
5. **Operadores**
    - Operadores Aritméticos y de Asignación
    - Concatenación de Cadenas

Objetivo de la clase:

En esta clase vas a iniciarte en el lenguaje JavaScript, aprendiendo qué es, sus principales características y cómo incorporarlo en una página web. Comenzarás a declarar variables y constantes, respetando las reglas de nomenclatura y conociendo los tipos de datos primitivos. Además, utilizarás operadores básicos y concatenación de cadenas para realizar operaciones simples, practicando desde la consola del navegador.

1. Introducción a JavaScript

¿Qué es JavaScript?

JavaScript, o, para los amigos, JS, es lo que le da vida a una página web.

Si HTML es el esqueleto y CSS la apariencia, **JavaScript es el alma** que permite que las cosas interactúen. Gracias a este lenguaje podés hacer que un botón muestre una alerta, que un formulario envíe datos sin recargar la página, o que cambie el contenido cuando movés el mouse.

Principales Características de JavaScript

- **Lenguaje del lado del cliente**: se ejecuta directamente en el navegador del usuario.
- **Orientado a objetos**: aunque no sea 100% orientado a objetos como otros lenguajes, JS te permite trabajar con objetos.
- **Tipado débil**: no tenés que especificar el tipo de dato (número, string, etc.) al declarar variables.
- **Interpretado**: no necesita compilación; el navegador se encarga de ejecutarlo al vuelo.

Versiones de ECMAScript

JavaScript evoluciona a través de algo llamado ECMAScript (o ES). Cada versión trae nuevas funcionalidades, más fácil de usar y más potente.

- **ES5 (2009)**: añadió cosas útiles como métodos de arrays y objetos, y el famoso "use strict".
- **ES6/ES2015**: esta es la "revolución" de JS. Añadió cosas como `let`, `const`, funciones flecha y clases.
- **ES2020/ES2021**: trajo nuevas herramientas como **BigInt** (números gigantes), promesas más sencillas, y operadores lógicos más eficientes.

2. Primeros Pasos con JavaScript

Incorporación de JavaScript en HTML

Podés agregar JavaScript a una página de varias maneras:

- **Directamente en el head** (aunque no se usa mucho porque bloquea el renderizado):

```
<script>

    console.log("Hola desde el head!");
```

```
</script>
```

- **En el body**, antes del cierre de `</body>`:

```
<body>
   <p>Ejemplo de texto.</p>
   <script>
     console.log("Hola desde el body!");
   </script>
 </body>
```

- **Usando un archivo externo**, **la mejor práctica**:

```
<script src="mi-script.js"></script>
```

🛑Este último método es el más usado porque separa el código JS del HTML, manteniendo todo más organizado.

2. Uso de la Consola del Navegador

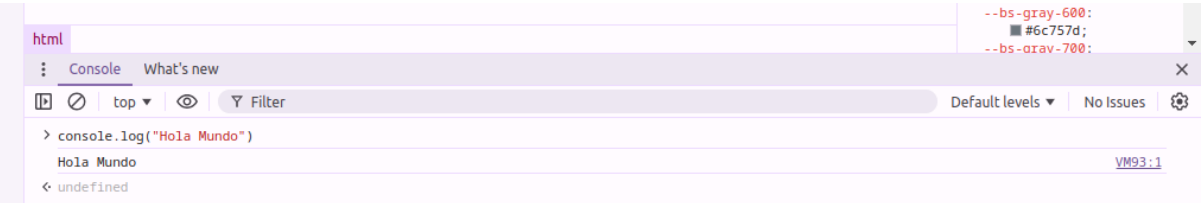
La consola del navegador es tu mejor amiga cuando estás trabajando con JS. Ahí podés ver lo que pasa en tu código, errores o directamente escribir y ejecutar JS.

Para abrir la consola:

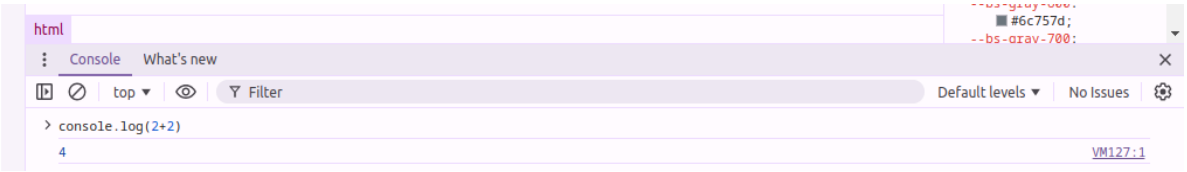
- **Windows/Linux**: Ctrl + Shift + J
- **Mac**: Cmd + Option + J

Ejemplo:

```
console.log("¡Hola Mundo!");
```



```
console.log(2 + 2); // Imprime 4
```



Comentarios en JavaScript

Para hacer tu código más entendible o desactivar partes mientras probás, podés usar **comentarios**.

- Comentario de una sola línea:

```
// Esto es un comentario
```

- Comentario de varias líneas:

```
/*

   Esto es un comentario

   de varias líneas

*/
```

3. Variables y Constantes en JavaScript

Declaración de Variables

En JS tenés tres maneras de declarar variables: **var**, **let** y **const**.

- **var**: la vieja confiable, pero con algunos problemas de alcance (scope). No la uses mucho si no es necesario.
- **let**: la opción moderna y segura.
- **const**: como su nombre indica, es para valores que no van a cambiar.

🛑En la práctica moderna, **se recomienda usar `let` y `const` casi exclusivamente**, y **evitar `var`**. Pero veamos cada una

Ejemplos:

```
var nombre = "Juan";

let edad = 30;

const PI = 3.1416;
```

Reglas para Nombrar Variables

1. Las variables **empiezan** con una letra, guion bajo (_) o signo de dólar ($).
2. Son **case sensitive**, o sea que `nombre` y `Nombre` son diferentes.
3. Se usa **camelCase** para variables compuestas (`primerNombre`).

Modificación de Variables

Podés cambiar el valor de las variables, a menos que sean `const`.

- Se usa cuando sabés que la variable puede cambiar durante la ejecución.

```
let iva = 21;

iva = 10.5;

console.log(iva);  // Imprime 10.5
```

Cuándo usarlo:

- Contadores, flags, valores que evolucionan.
- En bucles (`for`, `while`) y condiciones.

Declaración de Constantes

const es para cuando querés que algo nunca cambie.

```
const IVA = 21;

IVA = 10.5; // Te va a dar Error porque intentas asignarle otro valor y le habías dicho que era una constante.
```

Cuándo usarlo:

- Datos fijos (como configuraciones).
- Arrays u objetos que no vas a reemplazar (pero sí podés modificar su contenido interno).

4. Tipos de Datos en JavaScript

En JS, tenés varios tipos de datos:

- `undefined` = Variable declarada pero sin valor asignado.
    - `let x;`
- `boolean` = Valor lógico: verdadero o falso.
    - `let activo = true;`
- `number` = Números (enteros o decimales).
    - `let edad = 30;`
    - `let temp = 22.5;`
- `string` = Texto entre comillas.
    - `let nombre = "Ana";`
- `null` = Valor nulo intencional.
    - `let resultado = null;`
- `BigInt` = Números extremadamente grandes.
    - `let big = 123456789n;`
- `Symbol` = Valor único e inmutable.
    - `let id = Symbol("clave");`

El Objeto Number

Podés trabajar con números de dos maneras:

- **Como literales**:

```
let numero = 123;
```

- **Usando el constructor Number() (poco usado en la práctica)**:

```
let numObjeto = new Number(123);
```

Conversión Numérica

Podés convertir strings en números con `parseInt()` o `parseFloat()`.

```
let numero = parseInt("42");

console.log(numero); // Imprime 42

let numeroDecimal = parseFloat("42.5");

console.log(numeroDecimal); // Imprime 42.5
```

🛑`parseFloat()` convierte un string a número **permitiendo decimales;** mientras que `parseInt()` convierte solo la parte entera.

5. Operadores

Operadores Aritméticos y de Asignación

JS te permite hacer todas las operaciones matemáticas básicas:

- **Suma (+)**: suma dos números o concatena cadenas.
- **Resta (-)**: resta dos números.
- **Multiplicación (*)**: multiplica dos números.
- **División (/)**: divide dos números.
- **Módulo (%)**: da el resto de una división.
- **Incremento (++)** y **Decremento (--)**: suma o resta 1.

```
let x = 10;

x += 5;  // x ahora es 15

x++;     // x ahora es 16
```

Concatenación de Cadenas

El operador + también une cadenas de texto:

```
let saludo = "Hola, " + "Mundo!";

console.log(saludo);  // Imprime "Hola, Mundo!"
```

🛑Con esta base de JavaScript, podés empezar a darle interactividad a tus proyectos. Recordá que cuanto más practiques, más sentido te va a hacer todo.

💭**Storytelling**

¡Bienvenidos a una nueva etapa en Talento Lab!

Tomás (Desarrollador Senior)

¡Genial que sigamos avanzando! Ahora vamos a dar los primeros pasos en JavaScript, para que tu página no solo muestre contenido, sino que pueda interactuar con el usuario y manejar datos.

Vamos a trabajar con variables, tipos de datos, operaciones y validaciones básicas para que tu código sea robusto y funcional.

Ejercicio práctico #1:

Operaciones con Variables y Tipos de Datos

Lucía (Product Owner)

Queremos que el usuario pueda ingresar dos números y ver qué operaciones podemos hacer con ellos: sumar, restar, multiplicar, dividir y más. También queremos evitar errores si el usuario ingresa algo que no es un número.

Qué tenés que hacer:

- Pedir al usuario dos números usando `prompt()`.
- Convertir esos datos a números con `parseFloat()`.
- Mostrar en la consola los resultados de suma, resta, multiplicación, división y módulo.
- Verificar con `isNaN()` si los datos ingresados son válidos para evitar errores.

Tips para tu código:

- Usá `console.log()` para mostrar mensajes claros con el resultado de cada operación.
- Probá ingresar valores no numéricos y usá `isNaN()` para detectar esos casos.

```
let num1 = prompt("Ingresá el primer número:");
let num2 = prompt("Ingresá el segundo número:");

num1 = parseFloat(num1);
num2 = parseFloat(num2);

console.log("Suma: " + (num1 + num2));
console.log("Resta: " + (num1 - num2));
console.log("Multiplicación: " + (num1 * num2));
console.log("División: " + (num1 / num2));
console.log("Módulo: " + (num1 % num2));
```

🛑**NaN** significa *Not a Number* (no es un número).
En JavaScript, aparece cuando intentás realizar una operación matemática con un valor que no es numérico.
El método `isNaN()` sirve justamente para verificar si un valor **no** es un número válido, y así evitar errores en cálculos.

Ejercicio práctico #2:

Concatenación y Conversión de Tipos de Datos

Lucía (Product Owner)

Queremos mostrar un mensaje que combine el nombre y la edad del usuario, y además saber si es mayor de edad para adaptar el contenido si hace falta.

Qué tenés que hacer:

- Pedir nombre y edad con `prompt()`.
- Validar que la edad sea un número válido con `isNaN()`.
- Convertir la edad de texto a número con `parseInt()` o `Number()`.
- Concatenar los datos para formar un mensaje descriptivo y mostrarlo en consola.
- Comparar la edad para decir si la persona es mayor de edad o no.

Tips para tu código:

- Asegurate que los mensajes en consola sean claros y fáciles de entender.
- Probá distintos valores para asegurarte de que tu programa no falla con entradas inesperadas.

Tomás (Desarrollador Senior)

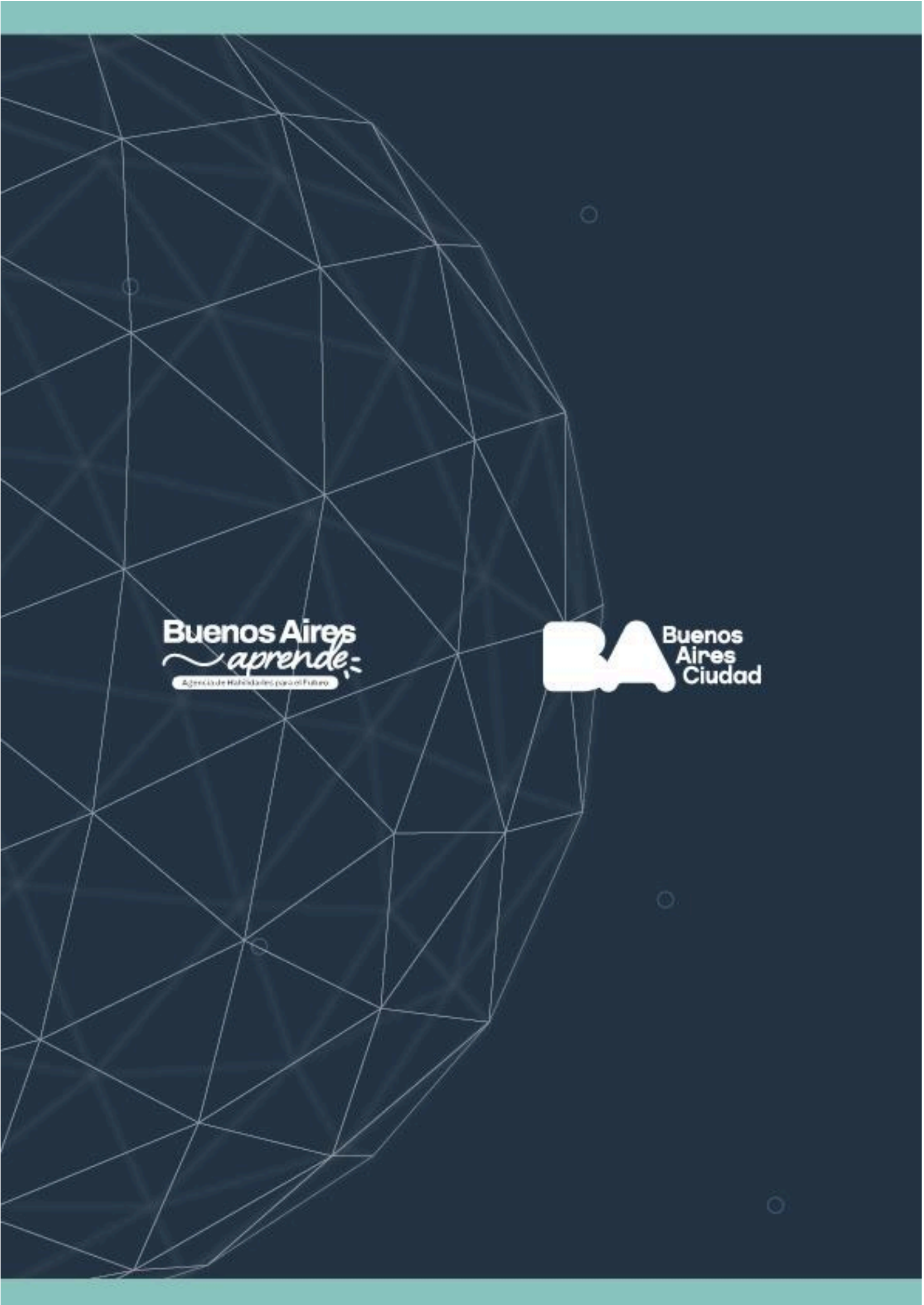
Tomás te recomienda:

No te quedes solo con el código que funciona, ¡probá ingresar cualquier cosa! Eso te va a ayudar a encontrar errores y pensar en cómo mejorar tu programa para que sea más robusto.

🛑La función `isNaN(valor)` **verifica si el valor NO es un número**.
Devuelve `true` → si el valor no se puede interpretar como número.
Devuelve `false` → si el valor sí es un número válido.

```
let edad = parseInt(prompt("Ingresá tu edad:"));

// Verificamos si la edad es un número válido
console.log(isNaN(edad)); // true si NO es número, false si es número
```


Buenos Aires
aprende
Buenos
Aires
Ciudad