CLASE 7 · VIRTUAL

JavaScript: primeras decisiones en una página web

Introducción, variables, tipos de datos, operadores y condicionales.

IFTS 18 20/05/2026 FRONT-END

CONTEXTO

Dónde estamos en la cursada

- HTML: estructura y contenido.
- CSS: diseño visual, layout y responsive.
- Ahora empezamos con JavaScript.
- HTML dice qué hay. CSS dice cómo se ve. JavaScript dice qué pasa.

OBJETIVO

Qué deberían poder hacer al terminar

- Entender para qué sirve JavaScript.
- Declarar variables.
- Reconocer tipos de datos básicos.
- Usar operadores.
- Escribir condicionales simples.
- Tomar decisiones básicas dentro de una página.

PROBLEMA

Qué problema resuelve JavaScript

Reacción

Una página puede responder a una acción del usuario.

Decisión

El código puede elegir qué camino seguir según una condición.

Validación

Podemos revisar datos antes de avanzar.

Interactividad

La interfaz deja de ser estática y empieza a comportarse como una aplicación.

HTML SOLO

Ejemplo simple: sin JavaScript

```
<h1>Bienvenido</h1>
<p>Esta página siempre muestra el mismo contenido.</p>
<button>Ingresar</button>
```

- El botón existe visualmente.
- Pero no tiene comportamiento asociado.
- HTML define estructura, no lógica.

HTML + JS

Ejemplo simple: con JavaScript

```
<h1>Bienvenido</h1>
<p id="mensaje">Presioná el botón para continuar.</p>

<button onclick="saludar()">Ingresar</button>

<script>
  function saludar() {
    document.getElementById("mensaje").textContent = "Hola, bienvenido al sitio.";
  }
</script>
```

- El botón llama a una función.
- La función busca un elemento por su id.
- Luego cambia el texto mostrado en pantalla.

DEFINICIÓN

Qué es JavaScript

JavaScript es un lenguaje de programación que se ejecuta en el navegador.

Sirve para agregar comportamiento a una página web.

No reemplaza a HTML ni a CSS: los complementa.

CAPAS

Tres capas del Front-End

HTML
qué hay

→

CSS
cómo se ve

→

JavaScript
qué pasa

```
<h1>Producto</h1>
<button>Comprar</button>
```

```
button {
  background-color: blue;
  color: white;
}
```

UBICACIÓN

Dónde se escribe JavaScript

```
<!-- Dentro del HTML -->
<script>
  console.log("Hola desde JavaScript");
</script>

<!-- En archivo separado -->
<script src="app.js"></script>
```

- Para probar, puede estar dentro del HTML.
- Para proyectos reales, conviene separarlo en un archivo .js.
- Separar ayuda a mantener el código más ordenado.

CONSOLA

Primer contacto: console.log

```
console.log("Hola mundo");
```

- Muestra información en la consola del navegador.
- Sirve para probar, revisar errores y entender qué está pasando.
- No se muestra directamente dentro de la página.

COMENTARIOS

Comentarios en JavaScript

```
// Comentario de una línea

/*
Comentario
de varias líneas
*/
```

- Ayudan a explicar una intención.
- No se ejecutan.
- No conviene usarlos para tapar código confuso.

VARIABLES

Qué es una variable

```
let nombre = "Eduardo";
let edad = 35;
let estaActivo = true;
```

- Una variable guarda un dato.
- Ese dato puede reutilizarse más adelante.
- Es la base para escribir lógica.

LET

Variables con let

```
let contador = 0;

contador = contador + 1;

console.log(contador);
```

- let se usa cuando el valor puede cambiar.
- Ejemplos típicos: cantidad de clics, intentos, puntos o errores.

Valor modificable

CONST

Variables con const

```
const nombreCurso = "Desarrollo Front-End";
const anio = 2026;
```

- const se usa cuando el valor no debería cambiar.
- No significa que sea más importante.
- Significa que no vamos a reasignar ese valor.

BUENAS PRÁCTICAS

var: por qué no lo usamos al inicio

Antes

```
var nombre = "Ana";
```

Ahora

```
let nombre = "Ana";
const curso = "Front-End";
```

Preferimos let y const

REGLA PRÁCTICA

Cuándo usar const y cuándo usar let

```
const precio = 1000;
let cantidad = 2;

let total = precio * cantidad;
```

- Usar const por defecto.
- Usar let solamente si el valor va a cambiar.
- Esto reduce errores y hace más clara la intención del código.

TIPOS

Tipos de datos básicos

string	Texto: nombres, mensajes, direcciones, títulos.
number	Números: cantidades, precios, edades, promedios.
boolean	Verdadero o falso: activo, aprobado, visible, logueado.
undefined	Variable declarada, pero sin valor asignado.
null	Ausencia de valor decidida intencionalmente.

STRING

Texto en JavaScript

```
const nombre = "Lucía";
const materia = "Front-End";
const mensaje = "Bienvenida a la clase";

const ciudad = 'Buenos Aires';
```

- Un string representa texto.
- Puede escribirse con comillas dobles o simples.
- Lo importante es abrir y cerrar con el mismo tipo de comilla.

NUMBER

Números en JavaScript

```
const edad = 22;
const precio = 1500;
const promedio = 7.5;
```

- JavaScript usa number para enteros y decimales.
- Sirve para cálculos, comparaciones y validaciones.

BOOLEAN

Verdadero o falso

```
const estaLogueado = true;
const tieneDescuento = false;
const formularioCompleto = true;
```

- Un boolean representa verdadero o falso.
- Es clave para tomar decisiones.
- Normalmente se usa dentro de condiciones.

UNDEFINED

Cuando todavía no hay valor

```
let telefono;

console.log(telefono);
```

- La variable existe.
- Pero todavía no recibió ningún valor.
- El resultado será undefined.

NULL

Vacío intencional

```
let usuarioSeleccionado = null;
```

- null representa ausencia de valor de forma intencional.
- undefined significa que todavía no se asignó.
- null significa que se decidió dejar vacío.

TYPEOF

Saber el tipo de un dato

```
console.log(typeof "Hola");
console.log(typeof 25);
console.log(typeof true);
```

- typeof permite inspeccionar el tipo de dato.
- Es útil cuando algo no se comporta como esperamos.

CONCATENACIÓN

Unir textos con +

```
const nombre = "Martina";
const saludo = "Hola " + nombre;

console.log(saludo);
```

- Concatenar es unir textos.
- El signo + puede sumar números o unir strings según el caso.

TEMPLATE STRINGS

Texto más cómodo con variables

```
const nombre = "Martina";
const edad = 21;

const mensaje = `Hola ${nombre}, tenés ${edad} años.`;

console.log(mensaje);
```

- Se usan comillas invertidas.
- Permiten insertar variables dentro del texto con ${ }.
- Suelen ser más legibles que concatenar con +.

OPERADORES

Operadores aritméticos

```
const suma = 10 + 5;
const resta = 10 - 5;
const multiplicacion = 10 * 5;
const division = 10 / 5;
const resto = 10 % 3;
```

- Permiten hacer cálculos.
- El operador % devuelve el resto de una división.
- Son la base para calcular totales, descuentos, promedios y cantidades.

EJEMPLO

Calcular total

```
const precioProducto = 2500;
const cantidad = 3;

const total = precioProducto * cantidad;

console.log(`Total a pagar: $${total}`);
```

- Se guardan datos en variables.
- Se calcula un resultado.
- Se muestra un mensaje final en consola.

COMPARACIÓN

Operadores de comparación

```
console.log(10 > 5);
console.log(10 < 5);
console.log(10 >= 10);
console.log(10 <= 8);
console.log(10 === 10);
console.log(10 !== 5);
```

- Las comparaciones devuelven true o false.
- Sirven para decidir si algo se cumple o no.

IGUALDAD

Igualdad estricta

Evitar

```
10 == "10"
```

Usar

```
10 === 10
10 !== 5
```

Comparar valor y tipo

AND

Operador lógico &&

```
const edad = 20;
const tieneEntrada = true;

console.log(edad >= 18 && tieneEntrada === true);
```

- && significa “y”.
- Todas las condiciones deben cumplirse.
- Si una falla, el resultado es false.

OR

Operador lógico ||

```
const esAdmin = false;
const esProfesor = true;

console.log(esAdmin || esProfesor);
```

- || significa “o”.
- Alcanza con que una condición se cumpla.
- Es útil cuando hay caminos alternativos válidos.

NOT

Operador lógico !

```
const estaLogueado = false;

console.log(!estaLogueado);
```

- ! invierte el valor.
- Si era false, pasa a true.
- Si era true, pasa a false.

IF

Condicional if

```
const edad = 18;

if (edad >= 18) {
  console.log("Puede ingresar");
}
```

- El bloque se ejecuta solo si la condición es verdadera.
- La condición va entre paréntesis.
- El código a ejecutar va entre llaves.

IF / ELSE

Dos caminos posibles

```
const edad = 16;

if (edad >= 18) {
  console.log("Puede ingresar");
} else {
  console.log("No puede ingresar");
}
```

- Si la condición se cumple, entra al if.
- Si no se cumple, entra al else.
- Sirve para resolver decisiones binarias.

ELSE IF

Más de dos caminos

```
const nota = 8;

if (nota >= 7) {
  console.log("Promociona");
} else if (nota >= 4) {
  console.log("Aprueba, pero no promociona");
} else {
  console.log("Debe recuperar");
}
```

- else if permite evaluar condiciones intermedias.
- El orden importa.
- La primera condición verdadera corta el resto.

CURSADA

Ejemplo aplicado: nota de cursada

```
const asistencia = 80;
const promedio = 7.5;
const trabajoAprobado = true;

if (asistencia >= 75 && promedio >= 7 && trabajoAprobado) {
  console.log("Promociona la materia");
} else {
  console.log("No promociona");
}
```

- Combina variables, operadores y condicionales.
- El ejemplo se parece a reglas reales de cursada.
- Muestra por qué los booleanos son útiles.

LOGIN

Ejemplo aplicado: login básico

```
const email = "alumno@ifts.edu.ar";
const password = "123456";

if (email === "alumno@ifts.edu.ar" && password === "123456") {
  console.log("Acceso permitido");
} else {
  console.log("Email o contraseña incorrectos");
}
```

- No es seguridad real.
- Es un ejemplo lógico para entender condiciones.
- La validación real requiere servidor y otras medidas.

CARRITO

Ejemplo aplicado: carrito de compras

```
const totalCompra = 12000;
let envioGratis = false;

if (totalCompra >= 10000) {
  envioGratis = true;
}

console.log(envioGratis);
```

- La variable cambia según una condición.
- Primero se define un valor inicial.
- Después el if puede modificarlo.

DESCUENTO

Ejemplo aplicado: descuento

```
const precio = 20000;
const esEstudiante = true;

let descuento = 0;

if (esEstudiante) {
  descuento = precio * 0.20;
}

const precioFinal = precio - descuento;

console.log(`Precio final: $${precioFinal}`);
```

- Se calcula un descuento condicional.
- Si no corresponde descuento, queda en cero.
- El precio final se calcula siempre al final.

HTML + JS

Ejemplo aplicado a una página

```
<h2>Calculadora de descuento</h2>

<script>
  const precio = 10000;
  const tieneCupon = true;

  if (tieneCupon) {
    console.log("Se aplica descuento");
  } else {
    console.log("No tiene descuento");
  }
</script>
```

- Todavía no modificamos el HTML.
- Por ahora usamos consola para entender la lógica.
- Más adelante lo conectamos con el DOM.

ERROR COMÚN

Confundir = con ===

Incorrecto

```
if (edad = 18) {
  console.log("Puede ingresar");
}
```

Correcto

```
if (edad === 18) {
  console.log("Tiene 18 años");
}
```

= asigna, === compara

ERROR COMÚN

Comparar textos mal escritos

```
const rol = "Admin";

if (rol === "admin") {
  console.log("Acceso de administrador");
}
```

- JavaScript distingue mayúsculas y minúsculas.
- "Admin" no es igual a "admin".
- Este tipo de error es muy común en validaciones.

LEGIBILIDAD

Condiciones demasiado largas

Poco claro

```
if (edad >= 18 && tieneDni && pagoRealizado && !estaBloqueado && cupoDisponible) {
  console.log("Puede ingresar");
}
```

Más claro

```
const puedeIngresar = edad >= 18 && tieneDni && pagoRealizado && !estaBloqueado && cu

if (puedeIngresar) {
  console.log("Puede ingresar");
}
```

EJERCICIO GUIADO 1

Producto con precio, descuento y stock

Consigna

Crear variables para un producto y resolver una lógica mínima de compra.

1. Crear nombre, precio, cantidad, tieneDescuento y stockDisponible.
2. Calcular subtotal.
3. Calcular descuento si corresponde.
4. Calcular total final.
5. Mostrar un mensaje según disponibilidad de stock.

Trabajo en clase: 10 a 15 minutos

RESOLUCIÓN

Producto con precio, descuento y stock

```
const nombreProducto = "Auriculares";
const precio = 15000;
const cantidad = 2;
const tieneDescuento = true;
const stockDisponible = 5;

const subtotal = precio * cantidad;
let descuento = 0;

if (tieneDescuento) {
  descuento = subtotal * 0.10;
}

const total = subtotal - descuento;

if (cantidad <= stockDisponible) {
```

- La lógica está dividida en pasos.
- Primero datos, luego cálculos, luego decisión.
- Ese orden ayuda a leer el programa.

EJERCICIO GUIADO 2

Sistema simple de evaluación

Consigna

Crear una lógica de evaluación similar a una cursada.

1. Crear nombre del alumno.
2. Crear nota.
3. Crear asistencia.
4. Crear trabajo práctico aprobado.
5. Mostrar “Promociona”, “Regulariza” o “Debe recuperar”.

Trabajo en clase: 10 minutos

RESOLUCIÓN

Sistema simple de evaluación

```
const alumno = "Sofía";
const nota = 8;
const asistencia = 85;
const tpAprobado = true;

if (nota >= 7 && asistencia >= 75 && tpAprobado) {
  console.log(`${alumno} promociona`);
} else if (nota >= 4 && asistencia >= 75 && tpAprobado) {
  console.log(`${alumno} regulariza`);
} else {
  console.log(`${alumno} debe recuperar`);
}
```

- El primer if evalúa promoción.
- El else if evalúa regularidad.
- El else cubre los casos restantes.

EJERCICIO RÁPIDO

Agregar JavaScript a la página anterior

Consigna

Tomar la página realizada en clases anteriores y agregar un archivo app.js.

1. Crear variables con datos de la página.
2. Mostrar mensajes en consola.
3. Usar al menos un if.
4. Usar al menos un operador lógico.
5. Usar un template string.

Entrega o revisión rápida en clase

INTEGRACIÓN

Integración mínima HTML + JS

index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Clase 7 JS</title>
</head>
<body>
  <h1>Primer JavaScript</h1>

  <script src="app.js"></script>
</body>
</html>
```

app.js

```
const usuario = "Alumno";
const edad = 20;

if (edad >= 18) {
  console.log(`${usuario} puede continuar`);
} else {
  console.log(`${usuario} no puede continuar`);
}
```

CIERRE

Qué nos tiene que quedar

Hoy vimos

- Qué es JavaScript.
- Variables con **let** y **const**.
- Tipos de datos básicos.
- Operadores aritméticos, comparativos y lógicos.
- Condicionales con **if**, **else if** y **else**.
- Ejemplos aplicados a situaciones reales.

JavaScript empieza cuando la página tiene que tomar una decisión.

Próxima clase: integración práctica de HTML + CSS + JavaScript.