DESARROLLO DE SISTEMAS WEB — FRONT-END

Clase 8
Integración HTML + CSS + JavaScript

Cómo una página pasa de ser contenido estático a una interfaz que responde a lo que hace el usuario.

IFTS 18 | 27/05/2026 | PRESENCIAL

PROPÓSITO

Qué problema resolvemos hoy

Ya vimos piezas separadas. El problema real aparece cuando hay que conectarlas.

- HTML crea elementos.
- CSS les da forma visual.
- JavaScript lee datos, toma decisiones y actualiza la pantalla.

La clase no busca memorizar líneas: busca entender el recorrido completo de una interacción.

OBJETIVOS

Al terminar la clase deberían poder…

Leer

Reconocer qué parte del código pertenece a HTML, CSS o JavaScript.

Seguir

Explicar qué ocurre desde que el usuario completa el formulario hasta que aparece el mensaje.

Corregir

Detectar errores simples de conexión, lógica y nombres de variables o id.

MAPA

Las tres capas de una página

HTML

Define qué elementos existen: títulos, formularios, botones, campos y secciones.

CSS

Define cómo se ven: colores, espaciado, tamaños, disposición y responsive.

JavaScript

Define qué pasa: leer datos, validar, decidir y cambiar contenido.

Cuando algo falla, conviene preguntar primero: ¿falló la estructura, el diseño o el comportamiento?

FLUJO

La interacción como recorrido

1. Completa

El usuario ingresa datos.

2. Envía

Presiona el botón.

3. Lee

JS toma valores del HTML.

4. Decide

Evalúa condiciones.

5. Responde

Cambia el mensaje visible.

Este recorrido es más importante que cualquier línea aislada.

CASO GUÍA

Simulador de inscripción a curso

Vamos a trabajar sobre una interfaz simple que recibe datos de una persona interesada en un curso.

- Nombre.
- Edad.
- Curso elegido.
- Modalidad.
- Conocimientos previos.



ENTRADA Y SALIDA

Qué datos entran y qué respuesta sale

Dato de entrada	Uso en el programa	Ejemplo
Nombre	Personalizar el saludo	Laura
Edad	Validar edad mínima	20
Curso	Mostrar el curso elegido	JavaScript inicial
Conocimientos previos	Elegir recomendación	No

La salida es un mensaje armado por JavaScript, no un texto fijo escrito en el HTML.

ORGANIZACIÓN

Tres archivos, tres responsabilidades

index.html

Estructura del documento y elementos que el usuario ve o completa.

styles.css

Reglas visuales: fondo, tarjeta, formulario, botón y resultado.

script.js

Lógica: eventos, lectura de datos, validaciones y mensajes.

Separar archivos ayuda a pensar mejor el problema y a corregir más rápido.

HTML

Estructura principal

```
<header class="encabezado">
  <h1>Inscripción a curso Front-End</h1>
  <p>Completá tus datos para recibir una recomendación.</p>
</header>

<main class="contenedor">
  <section class="tarjeta">
    <h2>Datos del alumno</h2>
    <!-- Formulario -->
  </section>
</main>
```

Acá todavía no hay lógica.

- Se define la jerarquía visual.
- Se organiza el contenido.
- Se prepara el lugar donde irá el formulario.

HTML responde: **¿qué elementos existen?**

HTML

Formulario: el punto de entrada

```
<form id="formularioInscripcion">
  <label for="nombre">Nombre</label>
  <input type="text" id="nombre">

  <label for="edad">Edad</label>
  <input type="number" id="edad">

  <button type="submit">Generar recomendación</button>
</form>
```

El formulario es la entrada de datos.

- El usuario escribe.
- El navegador guarda esos valores en los campos.
- JavaScript después los puede leer.

HTML

Select, checkbox e id

```
<select id="curso">
  <option value="">Seleccionar curso</option>
  <option value="html">HTML inicial</option>
  <option value="css">CSS aplicado</option>
  <option value="js">JavaScript inicial</option>
</select>

<input type="checkbox" id="conocimientos">
```

- **`id`** identifica un elemento concreto.
- **`value`** es el valor que lee JavaScript.
- El texto visible puede ser distinto del valor interno.
- El checkbox se lee con **`checked`** .

HTML

Área de resultado

```
<div id="resultado" class="resultado">
  Completá el formulario para ver la recomendación.
</div>
```

Este elemento es el puente entre HTML y JavaScript.

- Al principio muestra un texto inicial.
- Después JavaScript reemplaza ese texto.
- El usuario ve ahí la respuesta del sistema.

CSS

Qué aporta CSS en este ejemplo

CSS no cambia la lógica, pero cambia la experiencia.

Jerarquía

Ayuda a distinguir encabezado, tarjeta, formulario y resultado.

Legibilidad

Mejora tamaños, espaciados y separación entre campos.

Confianza

Una interfaz clara reduce dudas y errores del usuario.

CSS

Diseño base y tarjeta

```css
body {
  margin: 0;
  font-family: Arial, sans-serif;
  background: #f3f6fb;
  color: #1f2937;
}

.tarjeta {
  background: white;
  padding: 28px;
  border-radius: 16px;
}
```

- El **body** define la base general.
- La tarjeta agrupa el formulario.
- El borde redondeado y el espaciado hacen más amable la lectura.

CSS

Formulario, botón y resultado

```css
form {
  display: flex;
  flex-direction: column;
  gap: 12px;
}

button {
  padding: 14px;
  background: #1d4ed8;
  color: white;
}

.resultado {
  margin-top: 20px;
  padding: 16px;
  background: #e0f2fe;
}
```

La interfaz guía la acción:

- Formulario vertical.
- Botón visible.
- Resultado separado del formulario.

JAVASCRIPT

Qué tiene que hacer el script

Escuchar	Detectar cuándo el usuario envía el formulario.
Leer	Tomar los valores escritos o seleccionados.
Validar	Revisar si falta algo o si la edad no corresponde.
Decidir	Elegir una recomendación según las condiciones.
Mostrar	Actualizar el contenido visible en la página.

JAVASCRIPT

Seleccionar elementos

```
const formulario = document.getElementById("formularioInscripcion");
const resultado = document.getElementById("resultado");
```

JavaScript no trabaja “con la página entera” de forma mágica.

- Primero busca elementos concretos.
- Para encontrarlos usa el **`id`**.
- Si el id no coincide, el script falla.

JAVASCRIPT

Escuchar el envío del formulario

```
formulario.addEventListener("submit", function(evento) {
  evento.preventDefault();

  resultado.textContent = "Formulario enviado.";
});
```

- **`submit`** ocurre cuando se envía el formulario.
- **`preventDefault()`** evita la recarga.
- **`textContent`** cambia el texto visible.

JAVASCRIPT

Leer valores del formulario

```
const nombre = document.getElementById("nombre").value;
const edad = document.getElementById("edad").value;
const curso = document.getElementById("curso").value;
const modalidad = document.getElementById("modalidad").value;
const tieneConocimientos = document.getElementById("conocimientos").checked;
```

Hay dos lecturas distintas:

- **`.value`** : texto, número o selección.
- **`.checked`** : verdadero o falso en un checkbox.

JAVASCRIPT

Validar antes de decidir

```
if (nombre === "" || edad === "" || curso === "" || modalidad === "") {
  resultado.textContent = "Por favor, completá todos los campos obligatorios.";
  return;
}

const edadNumero = Number(edad);

if (edadNumero < 16) {
  resultado.textContent = "Este curso está recomendado para mayores de 16 años.";
  return;
}
```

El orden importa:

1. Primero se revisan campos vacíos.
2. Después se convierte la edad.
3. Después se compara la edad.
4. Si hay error, **`return`** corta la ejecución.

JAVASCRIPT

Elegir recomendación

```
let recomendacion = "";

if (tieneConocimientos) {
  recomendacion = "Podés avanzar con ejercicios de integración.";
} else {
  recomendacion = "Te recomendamos repasar primero HTML y CSS.";
}
```

Acá aparece la lógica condicional.

- Un mismo formulario puede dar respuestas distintas.
- La respuesta depende de los datos del usuario.
- Esto convierte una página estática en una interfaz interactiva.

JAVASCRIPT

Mostrar el mensaje final

```
resultado.textContent =
  "Hola " + nombre + ". Elegiste " + nombreCurso +
  " en modalidad " + modalidad + ". " + recomendacion;
```

El mensaje final combina:

- Datos ingresados.
- Opciones seleccionadas.
- Resultado de las condiciones.

La página responde con información personalizada.

LÓGICA

Tabla de decisión simplificada

Condición	Qué revisa	Resultado
Falta un dato	Nombre, edad, curso o modalidad vacío	Mensaje de campos obligatorios
Edad menor a 16	La edad convertida a número	Mensaje de edad mínima
Con conocimientos	Checkbox marcado	Avanzar con integración
Sin conocimientos	Checkbox sin marcar	Repasar HTML y CSS

Esta tabla ayuda a entender el código sin depender todavía de escribirlo perfecto.

CÓDIGO COMPLETO

Lectura general del script

```
const formulario = document.getElementById("formularioInscripcion");
const resultado = document.getElementById("resultado");

formulario.addEventListener("submit", function(evento) {
  evento.preventDefault();

  const nombre = document.getElementById("nombre").value;
  const edad = document.getElementById("edad").value;
  const curso = document.getElementById("curso").value;
  const modalidad = document.getElementById("modalidad").value;
  const tieneConocimientos = document.getElementById("conocimientos").checked;

  // Validaciones y mensaje final...
});
```

No hace falta memorizarlo. Hay que ubicar bloques:

1. Selección de elementos.
2. Evento.
3. Lectura de datos.
4. Validaciones.
5. Mensaje final.

ERROR TÍPICO

Cuando el id no coincide

HTML

```
<input id="nombre">
```

JavaScript incorrecto

```
document.getElementById("nombreAlumno")
```

Si JavaScript busca un id que no existe, no puede leer el dato. La conexión se rompe.

ERROR TÍPICO

Asignar no es comparar

Incorrecto

```
if (nombre = "") {
  // ...
}
```

Un solo igual asigna un valor.

Correcto

```
if (nombre === "") {
  // ...
}
```

Tres iguales comparan valor y tipo.

ERROR TÍPICO

value, checked y Number

.value

Sirve para leer textos, números escritos y opciones seleccionadas.

.checked

Sirve para saber si un checkbox está marcado: true o false.

Number()

Convierte texto numérico en número para poder comparar bien.

Muchos errores no son “JavaScript difícil”: son datos leídos con la herramienta equivocada.

ACTIVIDAD 1

Clasificar responsabilidades

Completar qué tecnología tiene la responsabilidad principal en cada caso.

Tarea	HTML	CSS	JS
Crear el campo nombre			
Dar color al botón			
Leer la edad ingresada			
Mostrar un mensaje personalizado			
Separar visualmente el resultado			

ACTIVIDAD 2

Seguir el recorrido del programa

Consigna

Para cada caso, seguir las condiciones en orden y escribir qué mensaje debería aparecer.

1. ¿Falta algún dato obligatorio?
2. ¿La edad es menor a 16?
3. ¿Tiene conocimientos previos?
4. ¿Qué mensaje final corresponde?

No se trata de adivinar: se trata de seguir el recorrido lógico del programa.

ACTIVIDAD 2

Casos para resolver

Caso	Nombre	Edad	Curso	Modalidad	Conoc.	Mensaje
A	Laura	20	JS	Virtual	No	
B	Martín	15	HTML	Presencial	Sí	
C		22	CSS	Virtual	Sí	
D	Sofía	28	CSS	Presencial	Sí	

El caso C fuerza a ver que la primera validación corta el flujo antes de revisar edad o conocimientos.

ACTIVIDAD 3

Detectar y corregir errores

Código con errores

```
const nombre = document.getElementById("nombreAlumno").value;

if (nombre = "") {
  resultado.textContent = "Falta completar el nombre";
}
```

Preguntas guía

- ¿El id existe en el HTML?
- ¿La condición compara o asigna?
- ¿Qué línea corregirían primero?
- ¿Qué error rompería el programa?

ACTIVIDAD 3

Corrección esperada

```
const nombre = document.getElementById("nombre").value;

if (nombre === "") {
  resultado.textContent = "Falta completar el nombre";
}
```

Correcciones:

- **`nombreAlumno`** no coincidía con el HTML.
- **`=`** asignaba un valor.
- **`===`** compara correctamente.

ACTIVIDAD 4

Mejorar el sistema

Consigna de mejora

Agregar una condición nueva al comportamiento del formulario.

1. Si el curso elegido es JavaScript.
2. Y no tiene conocimientos previos.
3. Entonces recomendar repasar HTML y CSS antes de avanzar.

```
if (curso === "js" && tieneConocimientos === false) {
  recomendacion = "Te recomendamos repasar HTML y CSS antes de avanzar con JavaScript.";
}
```

CRITERIOS

Cómo sabemos si está bien resuelto

Responsabilidad	Identifica correctamente si el problema pertenece a HTML, CSS o JavaScript.
Flujo	Sigue el orden de validaciones sin saltearse condiciones.
Precisión	Usa bien id, value, checked, Number, return y ===.
Explicación	Puede justificar por qué aparece un mensaje y no otro.

PUESTA EN COMÚN

Preguntas para cerrar la actividad

1. ¿Qué error apareció más fácil de detectar?
2. ¿Qué parte costó más: leer HTML, CSS o JavaScript?
3. ¿Por qué el orden de las validaciones cambia el resultado?
4. ¿Qué pasaría si quitamos `preventDefault()` ?
5. ¿Dónde empieza realmente la interacción?

PUENTE

Lo que prepara para la clase siguiente

Funciones

Para separar tareas y no tener todo el código mezclado.

Eventos

Para responder a clics, envíos, cambios y acciones del usuario.

DOM

Para modificar la página de forma más precisa y flexible.

Hoy vimos el recorrido completo. Después vamos a ordenar mejor cada parte.

CIERRE

Idea que tiene que quedar

Resumen

- HTML crea la estructura.
- CSS construye la presentación.
- JavaScript agrega comportamiento.
- La interacción ocurre cuando una acción del usuario cambia algo visible.

Una interfaz no es solo lo que se ve: también es lo que responde.

Próxima clase: funciones, eventos e interacción con el usuario.