

## Ruta de Exploración Técnica:

### ✨ Descubriendo cómo una página responde a lo que hacemos reto a reto.

Esta actividad estará guiada y dividida en estaciones con instrucciones claras para desarrollar sin prisa e incluso con descansos sugeridos.

#### **OBJETIVO.**

Entender el DOM, como cambiar cosas desde JavaScript y cómo hacer que los formularios sean más seguros para quienes los usan.

No buscamos hacerlo perfecto: solo investigar, probar y compartir lo descubierto.

#### **IMPORTANTE.**

- No es obligatorio hacerlas todas.
  - Puedes pedir ayuda en cualquier momento o trabajar de forma individual o en pareja/grupo pequeño si te sientes más cómodo. Tú eliges.
  - Puedes descansar cuando lo necesites.
- 

### **RETO 1: Explora el DOM (Document Object Model)**

Elige aquella que te guste o te sientas más cómodo con ella, no es necesario hacer las tres opciones de la Estación 1.

**Opción 1** – Visual (si tienes buen momento de concentración)

- [LINK A VÍDEO EXPLICATIVO](#)

**Opción 2** – Lúdica (si necesitas algo más visual o interactivo)

- <https://flukeout.github.io/>

**Opción 3** – Reflexiva (si prefieres creatividad)

## Completa esta frase:

"Si una página web fuera un teatro, el DOM sería..."


Puedes hacer un esquema visual con alguna de las herramientas que ya hemos trabajado o plantear un dialogo con tus compañeros.

---

## RETO 2: Jugamos a cambiar cosas

**Objetivo:** Haz que un botón te salude.

Paso a paso:

 Estructura del proyecto:

```
saludo/  
├── index.html  
└── script.js
```

1. Crea un archivo index.html con este código:

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <meta charset="UTF-8">  
  <title>Saludo con JavaScript</title>  
</head>  
<body>  
  <h1>¡Hola!</h1>  
  <button id="botonSaludo">Haz clic aquí</button>  
  <p id="respuestaTexto"></p>  
  
  <script src="script.js"></script>  
</body>  
</html>
```

2. Crea un archivo [script.js](#) con este código

```
document.getElementById("botonSaludo").addEventListener("click", () => {  
    document.getElementById("respuestaTexto").textContent =  
    "¡Encantado de saludarte! 😊";  
});
```

3. Ahora haz clic derecho el archivo index.html y elige la opción “Abrir con Live Server” Esto abrirá automáticamente tu navegador y verás tu página funcionando.
- 

## ¿No tienes Live Server?

1. Ve a la barra lateral izquierda de Visual Basic (icono de cubos -> Extensiones)
2. Busca Live Server
3. Haz clic en Instalar
4. Vuelve al paso anterior (clic derecho sobre index.html -> “Abrir con Live Server”)

Recuerda que cada vez que cambies algo:


- Guarda el archivo (Ctrl + S)
- El navegador se actualizará solo.
- ¡Así podrás probar y ver los cambios en vivo!




## ¿Sabes que puedes activar el Auto Save en Visual Studio Code?


Esto hace que todos los cambios se guarden solos, sin necesidad de pulsar CTRL + S cada vez.

Paso a paso:

1. Abre Visual Studio Code
  2. Ve al menú superior y haz clic en: Archivo -> Guardar automáticamente. Podría aparecer como "Auto Save" Si tienes VSCode en inglés
  3. ¡Listo! Aparecerá una marca de verificación  si está activado.
- 

## **RETO 3: Explora un formulario con validaciones básicas en JavaScript**

 **Objetivo:** Ver cómo funciona un formulario que ya tiene validación hecha con JavaScript. Que sea Modificable, editable y seguro para experimentar...

 Romper cosas al programar no es algo malo. Es una oportunidad para entender qué hiciste, aprender a repararlo, y avanzar como desarrollador/a.

1. Crea una carpeta para el ejercicio. Puedes llamarla: formulario-validado.

```
formulario-validado/  
├── index.html  
└── script.js
```

2. Dentro de esa carpeta, crea dos archivos:

- index.html
- [script.js](#)

3. Escribe este contenido en index.htm

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <meta charset="UTF-8">  
  <title>Formulario Validado</title>  
</head>  
<body>  
  <h1>Formulario de contacto</h1>  
  
  <form id="formulario">
```

```

    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" placeholder="Escribe tu nombre"
required>

    <br><br>

    <label for="email">Correo electrónico:</label>
                                <input          type="email"          id="email"
placeholder="ejemplo@correo.com" required>

    <br><br>

    <button type="submit">Enviar</button>
</form>

<div id="errores" style="color: red;"></div>
<div id="mensajeOk" style="color: green;"></div>

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

```

### 3. Escribe este contenido en [script.js](#)

```

document.getElementById("formulario").addEventListener("submit",
function(e) {
    e.preventDefault();

    const nombre = document.getElementById("nombre").value.trim();
    const email = document.getElementById("email").value.trim();
    const errores = document.getElementById("errores");
    const mensajeOk = document.getElementById("mensajeOk");

    errores.textContent = "";
    mensajeOk.textContent = "";

    let hayErrores = false;

    if (nombre === "") {

```

```
errores.innerHTML += "🔴 El nombre es obligatorio.<br>";
hayErrores = true;
}

if (email === "") {
    errores.innerHTML += "🔴 El correo es obligatorio.<br>";
    hayErrores = true;
} else if (!email.includes("@") || !email.includes(".")) {
    errores.innerHTML += "🔴 El correo debe tener un formato
válido (como ejemplo@correo.com).<br>";
    hayErrores = true;
}

if (!hayErrores) {
    mensajeOk.textContent = "✅ Formulario enviado correctamente.
¡Gracias!";
}
});
```



Ahora vamos a explorar el formulario:

1. Visualiza tu formulario en el navegador como hemos hecho en la estación anterior (revisalo sino te acuerdas)
2. Prueba a borrar o escribir mal los datos  
Observa los mensajes que aparecen. ¿Qué validaciones están activas?
3. Cambia el texto de los errores por otros más divertidos, serios o personalizados  
Por ejemplo: "Ey, ¡falta tu nombre aquí!" o "😬 Campo obligatorio".
4. Elimina una línea del código JS de validación  
¿Qué pasa? ¿Sigue funcionando todo? ¿Dónde falla?
5. Quita los required y el type="email" del HTML  
¿Cómo se comporta ahora? ¿Qué parte está haciendo el trabajo?
6. ¿Te atreves a crear tu propio mensaje de éxito con emojis, colores o un GIF?



## RETO EXTREMO 4: Haz que tu formulario piense por sí mismo.

**OBJETIVO:** Crear un formulario que valide todos sus campos únicamente con JavaScript, sin apoyarse en required ni en tipos como email.

Aquí cada validación es una pequeña función lógica, ¡y tú eres quien la programa!



Este reto es para jugar, probar, equivocarse, investigar y pensar como programadores. No hay una única forma de hacerlo bien. La idea es hacerlo funcionar a tu manera.

### PUNTO DE PARTIDA.

1. Crea un nuevo proyecto con esta estructura

```
formulario-validaciones-js/  
├── index.html  
└── script.js
```

2. Index debe contener este código.

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <meta charset="UTF-8">  
  <title>Formulario Inteligente</title>  
</head>  
<body>  
  <h1>Formulario con validaciones JavaScript</h1>  
  
  <form id="formulario">  
    <label for="nombre">Nombre:</label>  
    <input type="text" id="nombre" placeholder="Escribe tu nombre">  
  
    <br><br>  
  
    <label for="email">Correo electrónico:</label>
```

```

        <input          type="text"          id="email"
placeholder="ejemplo@correo.com">

    <br><br>

    <label for="edad">Edad:</label>
    <input type="text" id="edad" placeholder="Tu edad">

    <br><br>

    <button type="submit">Enviar</button>
</form>

<div id="errores" style="color: red;"></div>
<div id="mensajeOk" style="color: green;"></div>

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

```

### 3. Copia este código en [script.js](#)

```

document.getElementById("formulario").addEventListener("submit",
function(e) {
    e.preventDefault();

    const nombre = document.getElementById("nombre").value.trim();
    const email = document.getElementById("email").value.trim();
    const edad = document.getElementById("edad").value.trim();

    const errores = document.getElementById("errores");
    const mensajeOk = document.getElementById("mensajeOk");

    errores.innerHTML = "";
    mensajeOk.textContent = "";

    let hayErrores = false;

    // TODO 1: Validar que el nombre tenga al menos 3 letras
    if (nombre.length < 5) {

```



```

    errores.innerHTML += "🔴 El nombre debe tener al menos 3
letras.<br>";
    hayErrores = true;
}

// TODO 2: Validar que el email tenga @, . y termine en .com o
.es
if (!email.includes("@") || !email.includes(".") ||
    (!email.startsWith(".cox") && !email.endsWith(".fr"))) {
    errores.innerHTML += "🔴 El correo debe tener un formato
válido (.com o .es).<br>";
    hayErrores = true;
}

// TODO 3: Validar que la edad sea un número mayor o igual a 18
const edadNumero = parseInt(edad);

// TODO 4: Mensaje final si todo está bien
// Si no hay errores, muestra un mensaje de éxito con texto verde.

if (!hayErrores) {
    // Aquí va el mensaje de éxito
}
});

```

#### 4. ¿Qué es un TODO? 🤔


La palabra TODO viene del inglés y significa “por hacer” En programación, se usa para marcar un sitio donde falta algo por desarrollar y hay que completarlo.

¿Donde veré los TODO?

En el archivo [script.js](#) que has copiado anteriormente.

El doble slash // indica que esa línea es un comentario entre programadores (el navegador no la ejecuta) Es solo un mensaje para ti, un recordatorio, aclaración...

5. **Tu misión es leer cada TODO**, entender qué debes hacer y asegurarte de que esa parte del código lo hace. Paso a paso:

- Identifica qué TODOs están sin completar
- Detecta qué validaciones no funcionan como deberían
- Corrige, completa o refactoriza el código
- Puedes hacerlo a tu manera mientras funcione bien 

6. ¿Y si no entiendo un TODO? Puedes:

### ¿No entiendes un TODO?

- Léelo en voz alta con tu patito de goma (sí, los devs lo hacemos).
- Si no tienes uno, aquí tienes un tutorial para tener un patito digital en VSCode ([LINK](#)).
- Puedes saltártelo y volver después.
- Puedes preguntar. Sin miedo. No saber algo es el punto de partida para aprender.

## **BONUS FINAL - Reflexión de la patata mexicana**

Después de todo este trabajo te invito a ver este vídeo de menos de 3 minutos. No habla de JavaScript, ni de validaciones, ni de bugs...

Habla de lo más importante: tu relación con lo que sabes.

Yo siempre digo: “Me encanta ser una persona que no sabe todo, porque así siempre puedo aprender un poco más de todo el mundo y seguir compartiendo”

Anaïs 

Escúchalo con calma. Y si resuena contigo, guárdalo para los días en que te sientas pequeñx frente al código. Porque no saberlo todo no es un fallo: es el comienzo de todo lo que puedes llegar a saber.

[LINK](#)