

Universidad Tecnológica Metropolitana

Aplicaciones web

Práctica 1

Mstra. Martínez Dominguez Ruth Betzaida

Alumno. González Cen Cristian Alexander

Parcial 2

3°A

Turno matutino

Fecha de entrega: 12/06/2024

Link del repositorio de GitHub: https://github.com/CristianCenxd/Pr-ctica_1_DOM_C-digo.git

Índice

Contenido

Ejercicio 1	7
inciso a)	7
Instrucciones	7
Solución	7
Inciso b)	7
Instrucciones	7
Solución	7
Inciso c)	8
Instrucciones	8
Solución	8
Inciso d)	9
Instrucciones	9
Solución	9
Ejercicio 2	10
Inciso a)	10
Instrucciones	10
Solución	10
Inciso b)	11
Instrucciones	11
Solución	11
Ejercicio 3	12
Inciso a)	12
Instrucciones	12
Solución	13
Ejercicio 4	13
Inciso a)	13
Instrucciones	13
Solución	14
Inciso b)	14

Instrucciones	14
Solución	15
Inciso c)	15
Instrucciones	15
Solución	15
Inciso d)	16
Instrucciones	16
Solución	16
Inciso e)	17
Instrucciones	17
Solución	17
Inciso f)	18
Instrucciones	18
Solución	18
Ejercicio 5	19
Inciso a)	19
Instrucciones	19
Solución	19
Ejercicio 6	20
Inciso a)	20
Instrucciones	20
Solución	20
Ejercicio 7	21
Inciso a)	21
Instrucciones	21
Solución	21
Ejercicio 8	23
Inciso a)	23
Instrucciones	23
Solución	23
Ejercicio 9	24
Inciso a)	24
Instrucciones	24

Solución	24
Ejercicio 10	26
Inciso a)	26
Instrucciones.....	26
Solución	26

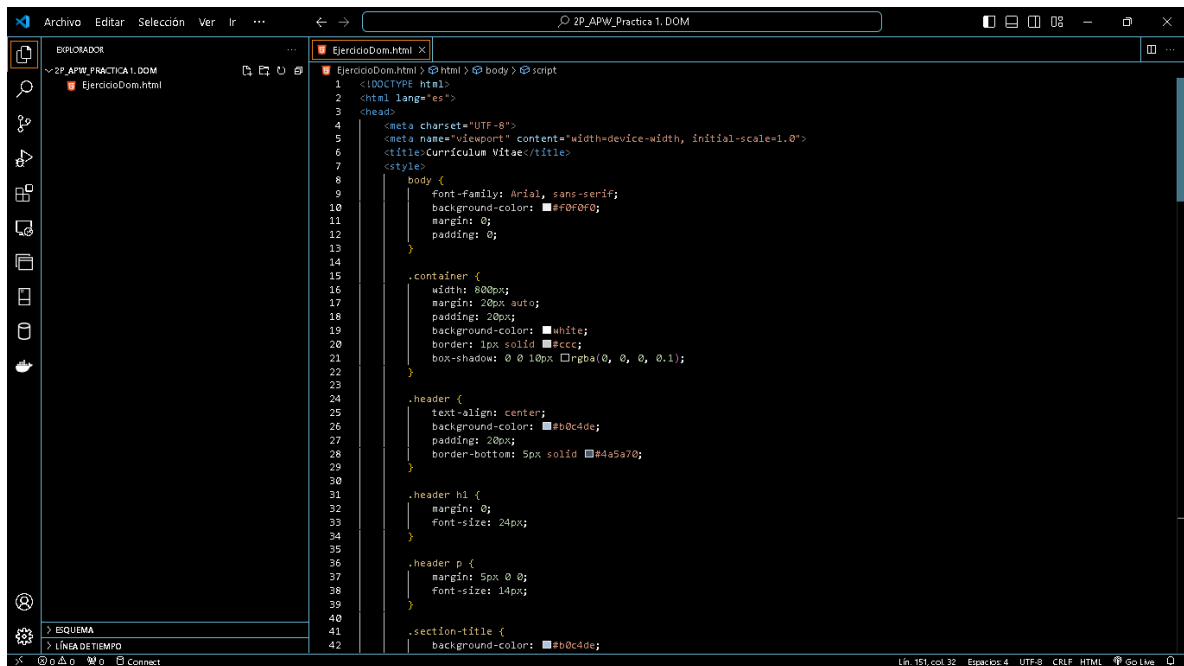
Introducción

En esta práctica de la materia Aplicaciones de Desarrollo Web, nos adentraremos en la manipulación del Document Object Model (DOM) utilizando JavaScript.

El objetivo es entender y aplicar diversas técnicas para interactuar y modificar elementos HTML de manera dinámica. Aprenderemos a seleccionar elementos por su ID, clase y nombre de etiqueta, además de explorar cómo cambiar su contenido, atributos y estilos.

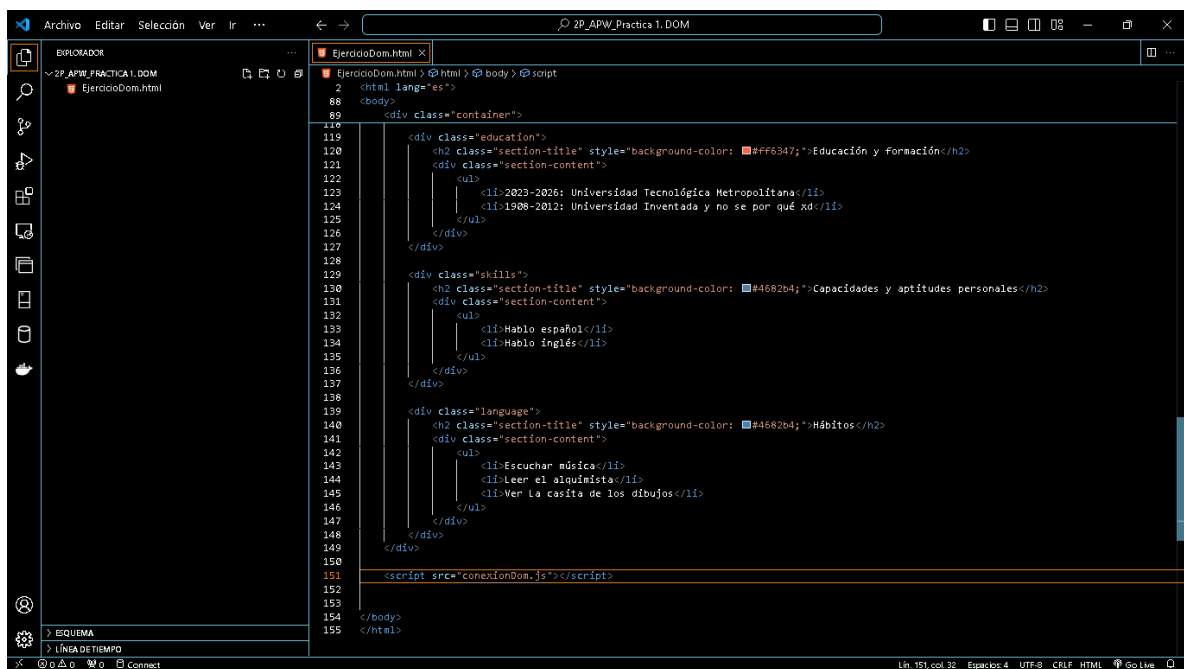
También veremos cómo agregar nuevos elementos al DOM, manejar eventos de usuario, y realizar validaciones en formularios. Esta práctica nos proporcionará las habilidades fundamentales necesarias para desarrollar aplicaciones web interactivas y responsivas, mejorando significativamente la experiencia del usuario.

1. Lo que hacemos como primer paso es elegir un html para poder trabajar con el. En mi caso yo escogí el curriculum.html y le cambié el nombre por EjercicioDom.html.



The screenshot shows the VS Code editor with the file 'EjercicioDom.html' open. The code is a CSS reset and basic styling for a curriculum page. The left sidebar shows the file explorer with '2P_APW_PRACTICA1.DOM' and 'EjercicioDom.html'. The bottom status bar indicates 'Ln 151, col 12'.

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Curriculum Vitae</title>
7   <style>
8     body {
9       font-family: Arial, sans-serif;
10      background-color: #f0f0f0;
11      margin: 0;
12      padding: 0;
13    }
14
15    .container {
16      width: 800px;
17      margin: 20px auto;
18      padding: 20px;
19      background-color: #fff;
20      border: 1px solid #ccc;
21      box-shadow: 0 0 10px #444;
22    }
23
24    .header {
25      text-align: center;
26      background-color: #e0e0e0;
27      padding: 20px;
28      border-bottom: 5px solid #444;
29    }
30
31    .header h1 {
32      margin: 0;
33      font-size: 24px;
34    }
35
36    .header p {
37      margin: 0;
38      font-size: 14px;
39    }
40
41    .section-title {
42      background-color: #e0e0e0;
```



The screenshot shows the VS Code editor with the file 'EjercicioDom.html' open. The code is the HTML structure for the curriculum page, including sections for education, skills, and language. The left sidebar shows the file explorer with '2P_APW_PRACTICA1.DOM' and 'EjercicioDom.html'. The bottom status bar indicates 'Ln 151, col 32'.

```
2 <html lang="es">
88 <body>
89   <div class="container">
90     <div class="education">
91       <h2 class="section-title" style="background-color: #fff6347;">Educación y formación</h2>
92       <div class="section-content">
93         <ul>
94           <li>2023-2026: Universidad Tecnológica Metropolitana</li>
95           <li>1908-2012: Universidad Inventada y no se por qué xd</li>
96         </ul>
97       </div>
98     </div>
99
100     <div class="skills">
101       <h2 class="section-title" style="background-color: #4682b4;">Capacidades y aptitudes personales</h2>
102       <div class="section-content">
103         <ul>
104           <li>Hablo español</li>
105           <li>Hablo inglés</li>
106         </ul>
107       </div>
108     </div>
109
110     <div class="language">
111       <h2 class="section-title" style="background-color: #4682b4;">Hábitos</h2>
112       <div class="section-content">
113         <ul>
114           <li>Escuchar música</li>
115           <li>Leer el alquímista</li>
116           <li>Ver La casita de los dibujos</li>
117         </ul>
118       </div>
119     </div>
120   </div>
121
122   <script src="conexionDom.js"></script>
123
124 </body>
125 </html>
```

Ejercicio 1

inciso a)

Instrucciones

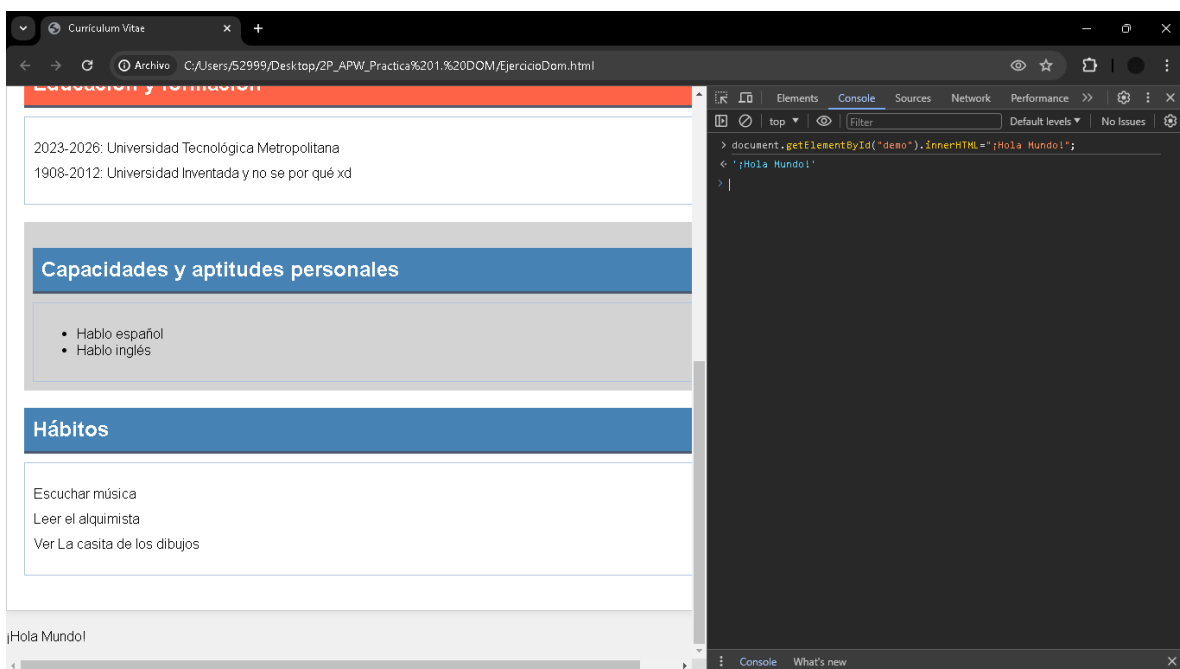
Uso del método `getElementById` y de la propiedad `innerHTML`.

Cambia el contenido (el `innerHTML`) del elemento `<p>` con `id="demo"`

Solución

En este HTML, hay un elemento `<p>` con el ID "demo" y cuando ejecutas este script en la consola:
`document.getElementById("demo").innerHTML = "¡Hola Mundo!"`;

El contenido se cambia por ¡Hola Mundo!



Inciso b)

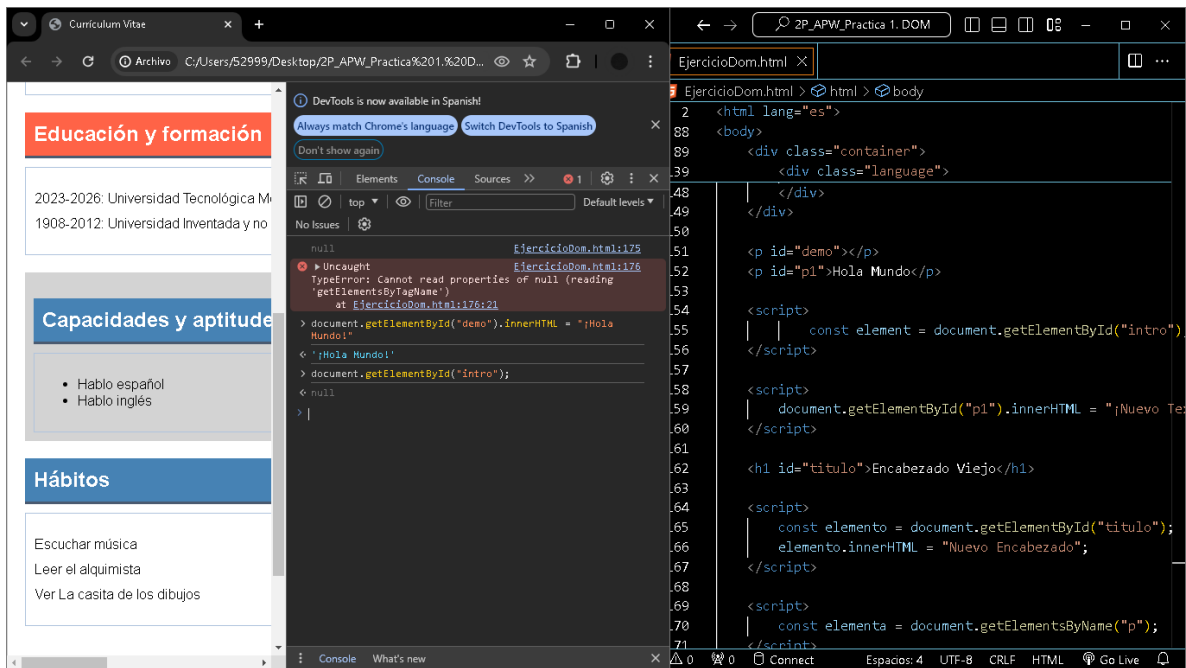
Instrucciones

Encuentra el elemento con `id="intro"`

`const element = document.getElementById("intro");`

Solución

Este método busca y devuelve el primer elemento del documento que tiene el ID especificado, en este caso, "intro"



Inciso c)

Instrucciones

Cambia el elemento de una etiqueta <p>

<p id = "p1">Hola Mundo</p>

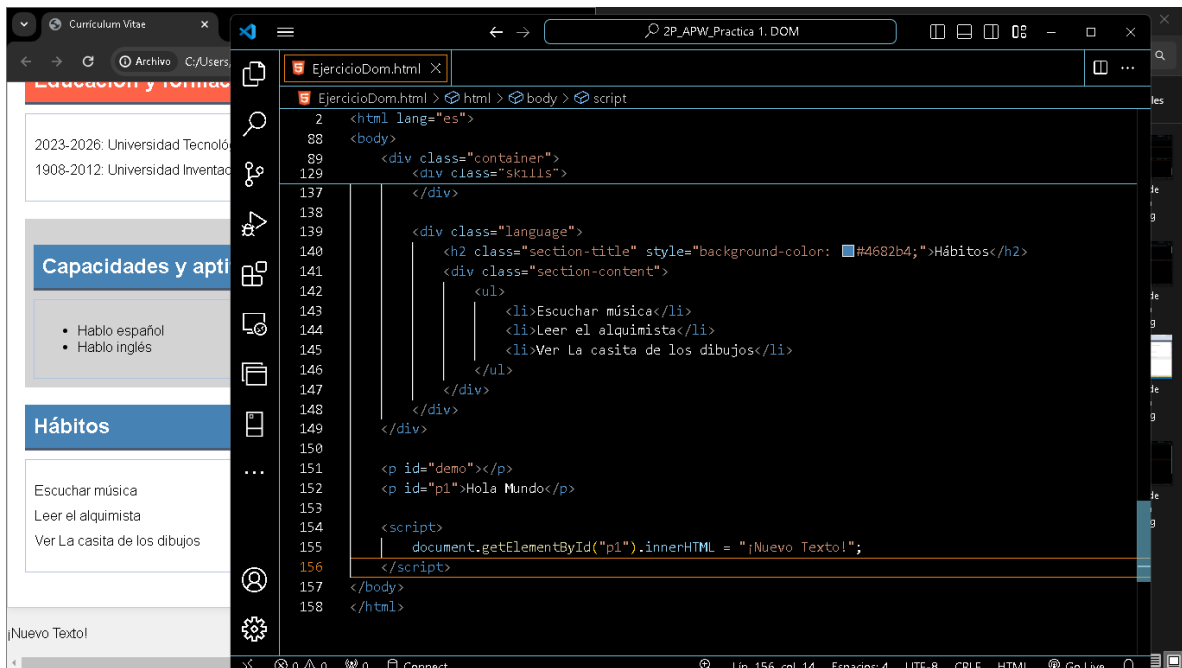
</script>

document.getElementById("p1").innerHTML=iNuevo Texto!;

</script>

Solución

El comando `document.getElementById("p1").innerHTML = "¡Nuevo Texto!"`; sirve para cambiar el contenido de un elemento HTML identificado por su ID. Este es un párrafo `<p>` con el ID "p1" que inicialmente contiene el texto "Hola Mundo" y luego cambia su contenido interno (innerHTML) a "¡Nuevo Texto!".



Inciso d)

Instrucciones

Cambia el contenido de un elemento <h1>.

<h1 id="titulo">Encabezado Viejo</h1>

<script>

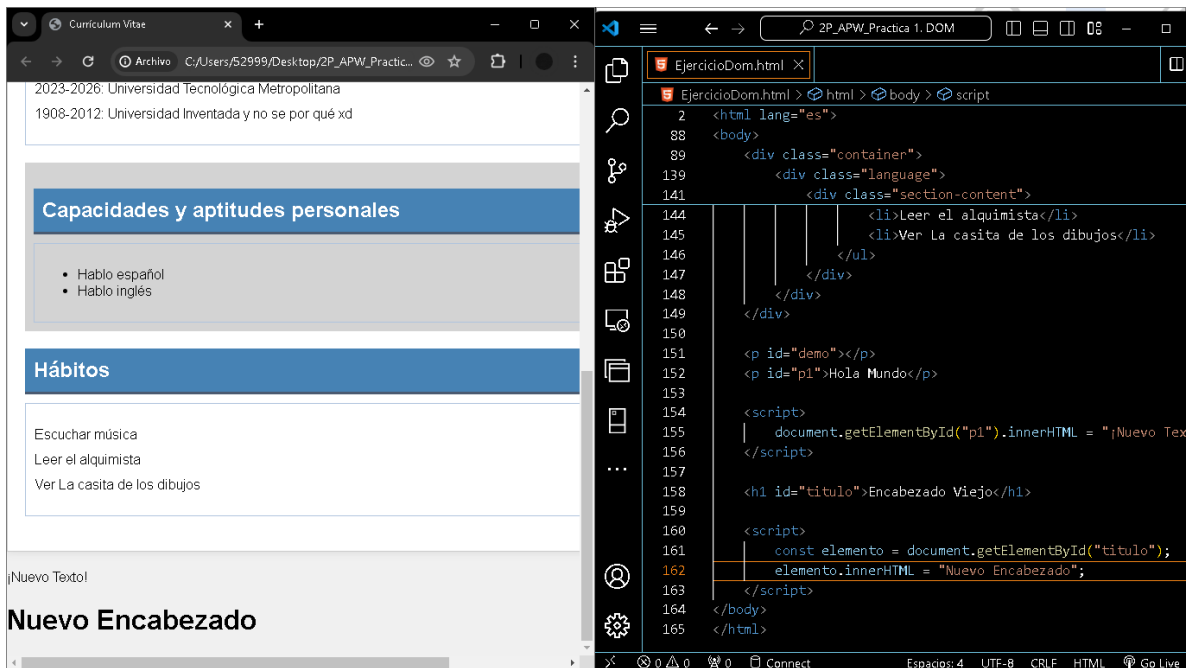
const elemento = document.getElementById("titulo");

elemento.innerHTML="Nuevo Encabezado";

</script>

Solución

El comando `document.getElementById("titulo").innerHTML = "Nuevo Encabezado";` se utiliza para cambiar el contenido de un elemento `<h1>` identificado por su ID. Primero selecciona el elemento `<h1>` con el ID "titulo" y luego Cambia su contenido interno a "Nuevo Encabezado".



Ejercicio 2

Inciso a)

Instrucciones

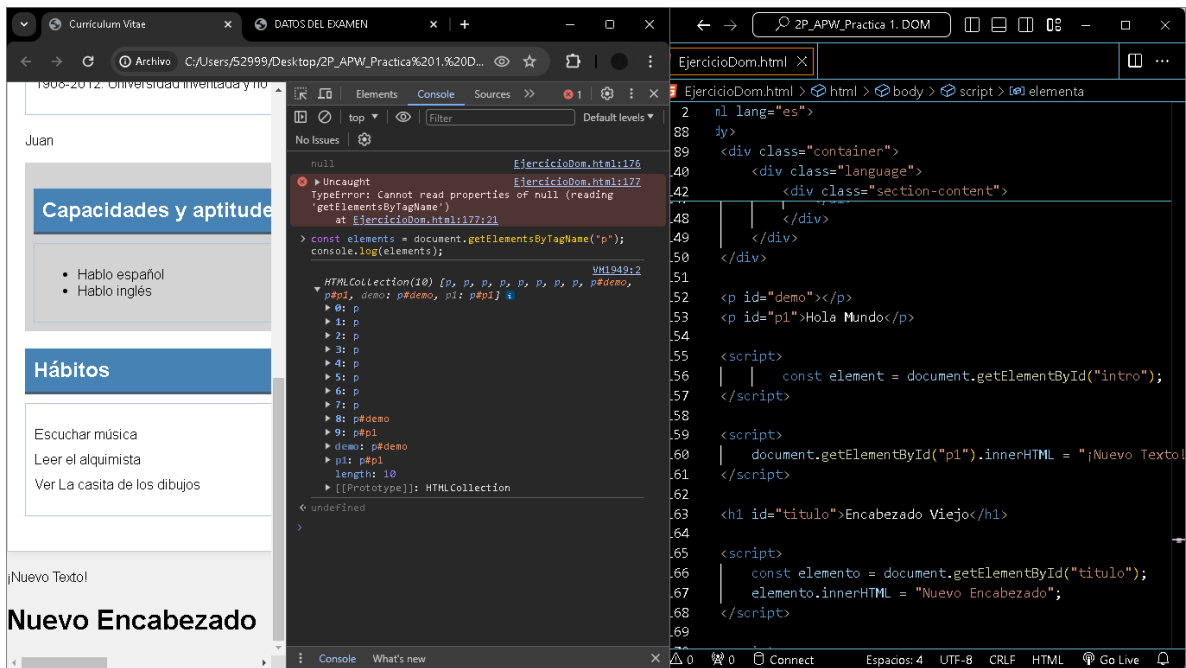
Búsqueda de elementos HTML por nombre de etiqueta.

En este ejemplo encuentra todos los elementos <p>de uno de tus proyectos .HTML

const element = document.getElementsByTagName("p");

Solución

El comando `const element = document.getElementsByTagName("p");` se utiliza para seleccionar todos los elementos HTML que tienen un atributo `name` con el valor `"p"` y almacenarlos en una variable.



Inciso b)

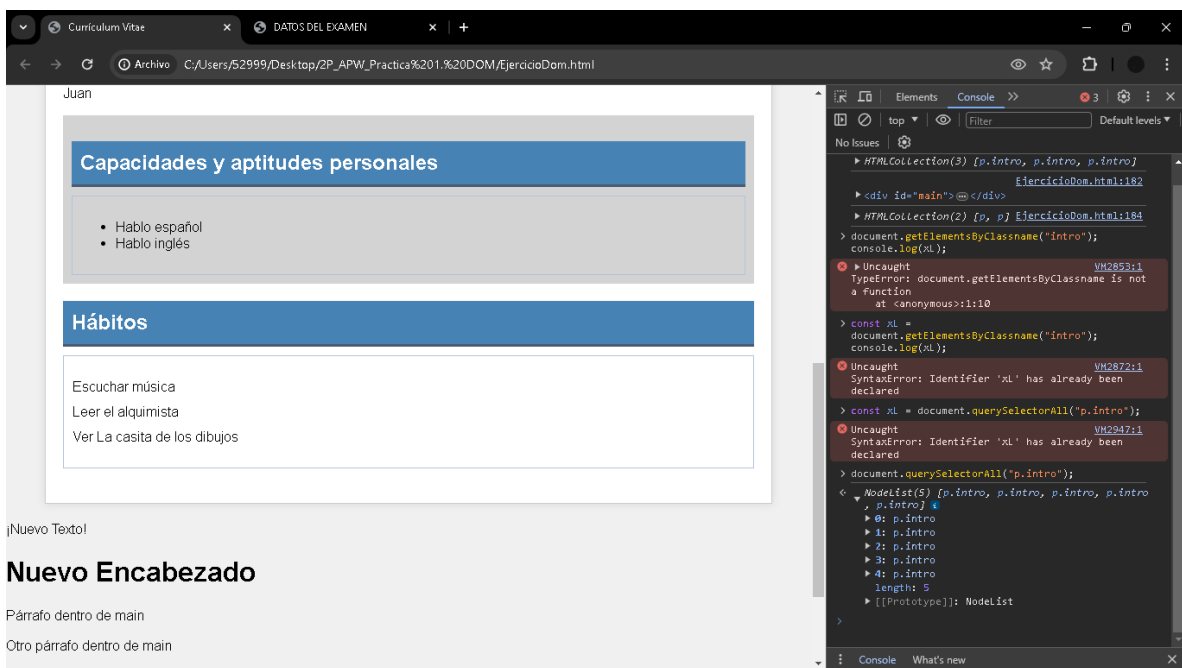
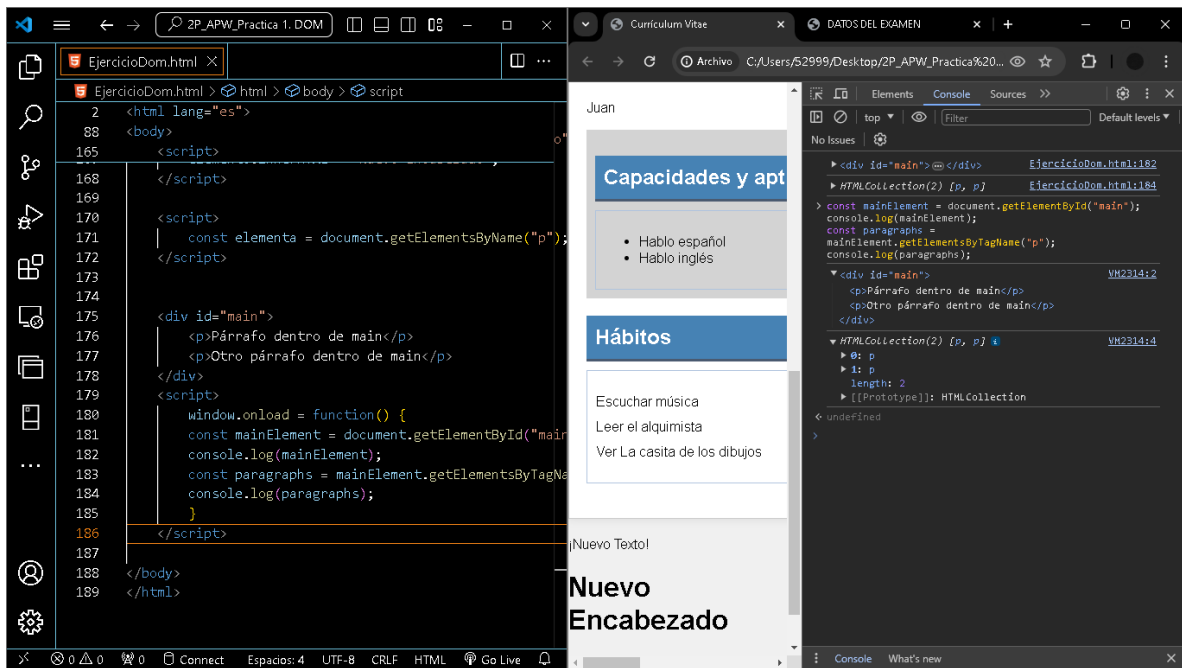
Instrucciones

Encuentra el elemento con `id="main"` y luego encuentra todos los elementos `<p>` dentro del main.

```
const x = document.getElementById("main");
console.log(x);
const y = x.getElementsByTagName("p");
console.log(y);
```

Solución

El comando `const x = document.getElementById("main");` se utiliza para seleccionar un elemento HTML con el ID "main" y almacenarlo en la variable x. Luego, `x.getElementsByTagName("p");` selecciona todos los elementos `<p>` que son descendientes del elemento almacenado en x y los almacena en la variable y. Finalmente, `console.log(x);` y `console.log(y);` muestran en la consola los elementos seleccionados, respectivamente.



Ejercicio 3

Inciso a)

Instrucciones

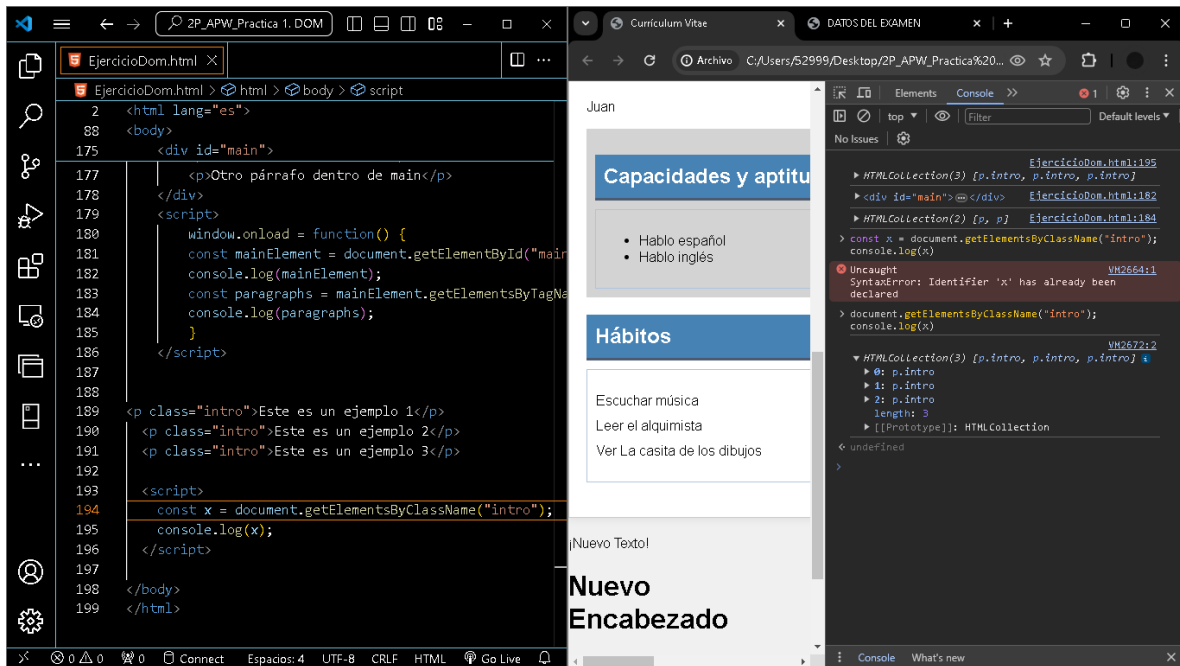
Si desea encontrar todos los elementos con el mismo nombre de clase, use:

`getElementsByClassName()`.

```
const x = document.getElementsByClassName("intro")
console.log(x);
```

Solución

El comando `document.getElementsByClassName("intro")` se utiliza para seleccionar todos los elementos HTML que tienen la clase CSS "intro" y almacenarlos en una colección de nodos. Luego, `console.log(x)`; muestra esta colección en la consola del navegador.



Ejercicio 4

Inciso a)

Instrucciones

Búsqueda de elementos HTML mediante selectores de CSS

Devuelve una lista de todos los elementos `<p>` con `class = "intro"`.

```
<p class="intro">este es un ejemplo</p>
```

```
<p class="intro">esto es otro ejemplo</p>
```

```
<script>
```

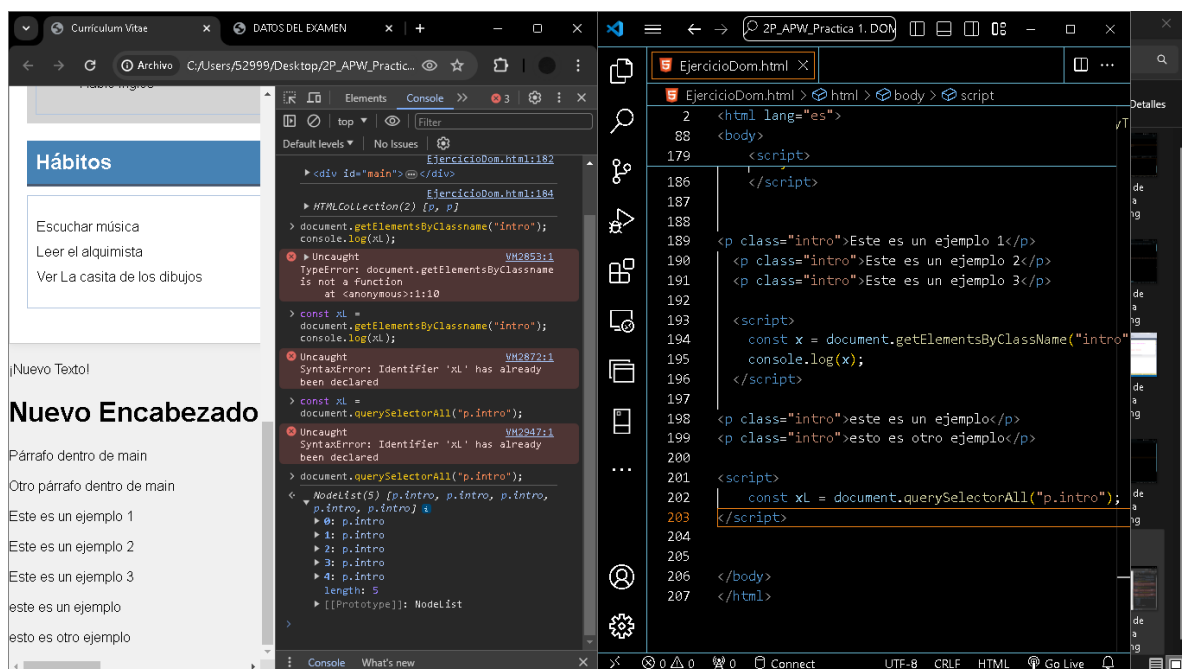
```
const x = document.querySelectorAll("p.intro");
</script>
```

Solución

El comando `document.querySelectorAll("p.intro")` selecciona todos los elementos `<p>` que tienen la clase "intro" y devuelve una `NodeList` que contiene esos elementos y `const x`: Declara una constante llamada `x` y le asigna la `NodeList` de elementos seleccionados. Después de ejecutar el script, la variable `x` contendrá una `NodeList` con todos los párrafos `<p>` que tienen la clase "intro". En este caso, `x` contendrá dos elementos:

```
<p class="intro">este es un ejemplo</p>
```

```
<p class="intro">esto es otro ejemplo</p>
```



Inciso b)

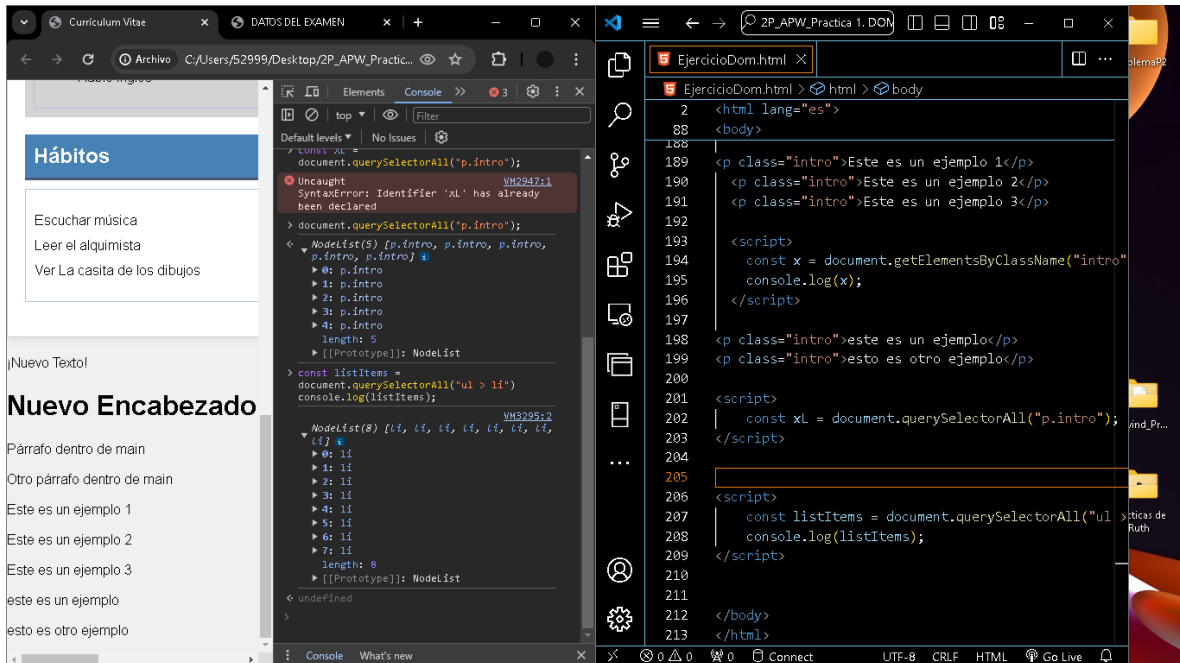
Instrucciones

Encuentra los elementos `` hijos de ``, para ello debes crear una lista desordenada con al menos 5 elementos dentro de la lista.

```
const listItems = document.querySelectorAll("ul > li")
console.log(listItems);
```

Solución

Utilizando el comando `const listItems = document.querySelectorAll("ul > li");` seleccionamos todos los elementos `` que son hijos directos de cualquier elemento ``. Luego, `console.log(listItems);` muestra esta colección de elementos en la consola del navegador.



Inciso c)

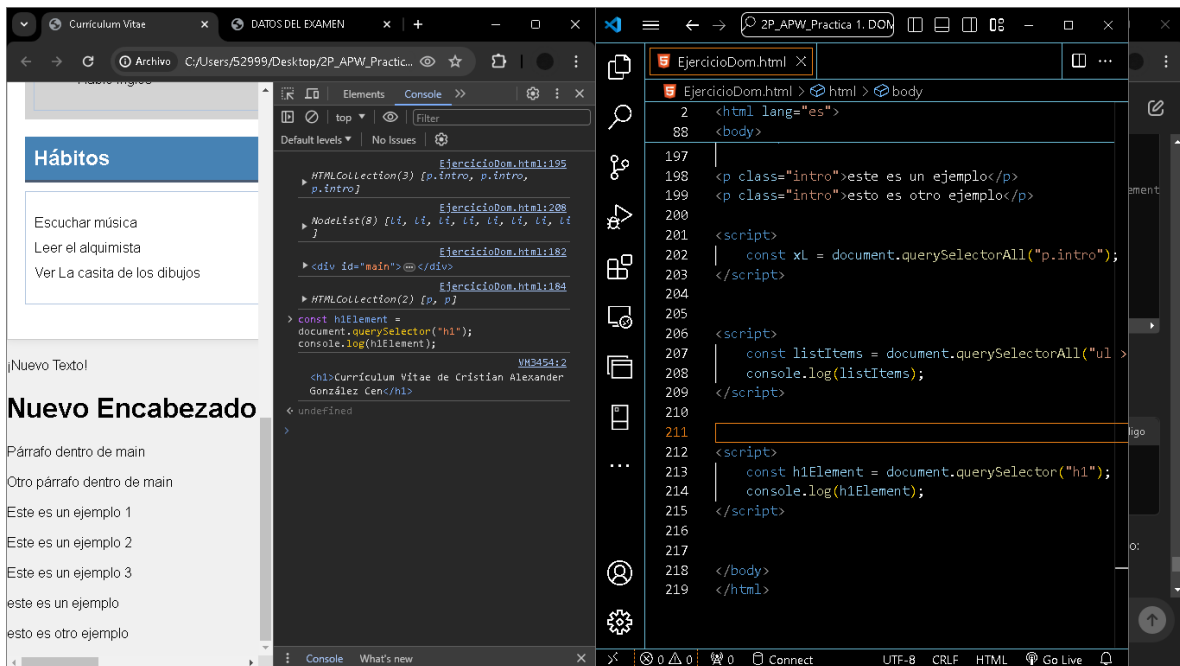
Instrucciones

Encontrar en la consola e imprimir el elemento `<h1>`.

```
const h1Element = document.querySelector("h1");
console.log(h1Element);
```

Solución

Con el comando `document.querySelector("h1")` lo utilizo para seleccionar el primer elemento `<h1>` en el documento HTML. Luego, `console.log(h1Element);` muestra este elemento en la consola del navegador.



Inciso d)

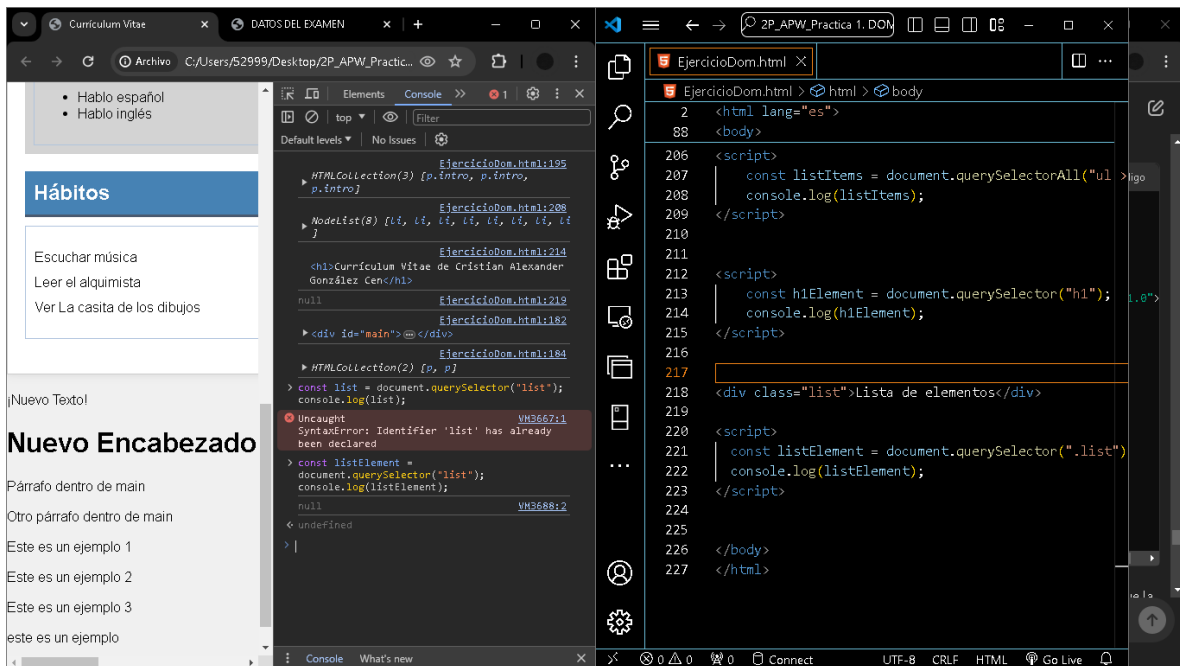
Instrucciones

Encontrar la clase `list` usando `querySelector()`.

```
const list = document.querySelector(".list");  
console.log(list);
```

Solución

Este comando lo utilice para seleccionar el primer elemento con la clase "list" y luego mostrarlo en la consola.



Inciso e)

Instrucciones

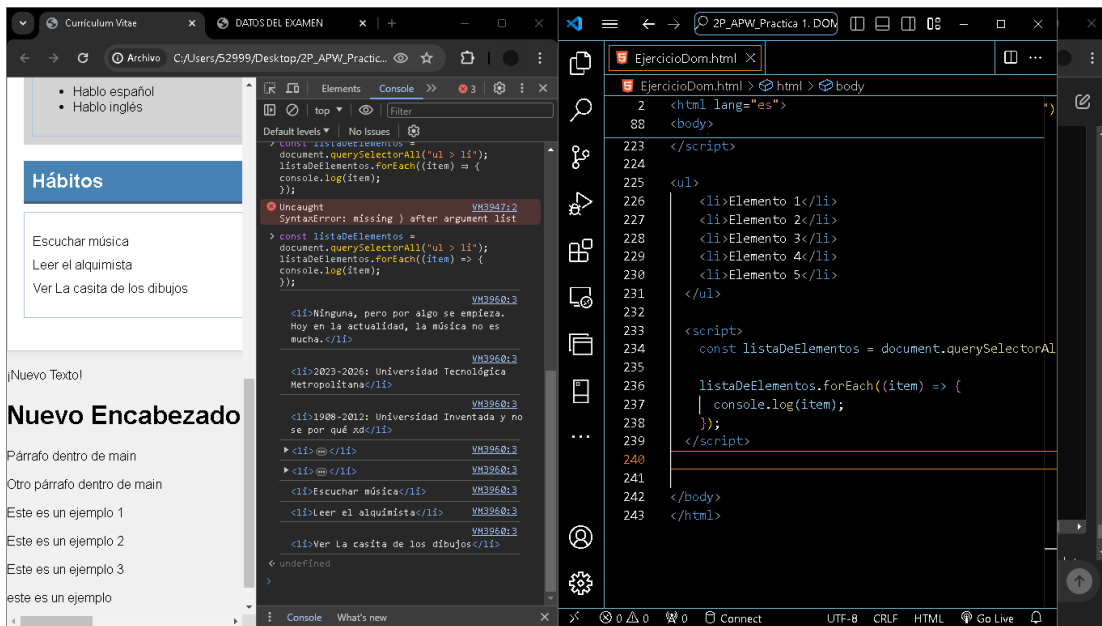
Imprimir los elementos `` haciendo uso del ciclo `forEach()` para iterar sobre la `NodeList` e imprimir cada uno de los elementos.

`const listaDeElementos = document.querySelectorAll("ul > li");`

*`listaDeElementos.forEach((item) => {`
 *`console.log(item);`
*`});`***

Solución

El comando `document.querySelectorAll("ul > li")`: Selecciona todos los elementos `` que son hijos directos de cualquier elemento ``, devolviendo una `NodeList` (una colección de nodos), luego el comando `const listaDeElementos`: Declara una constante llamada `listaDeElementos` y le asigna la `NodeList` de elementos seleccionados. Y por último con el comando `console.log(item)`: Muestro el resultado de cada elemento `` en la consola.



Inciso f)

Instrucciones

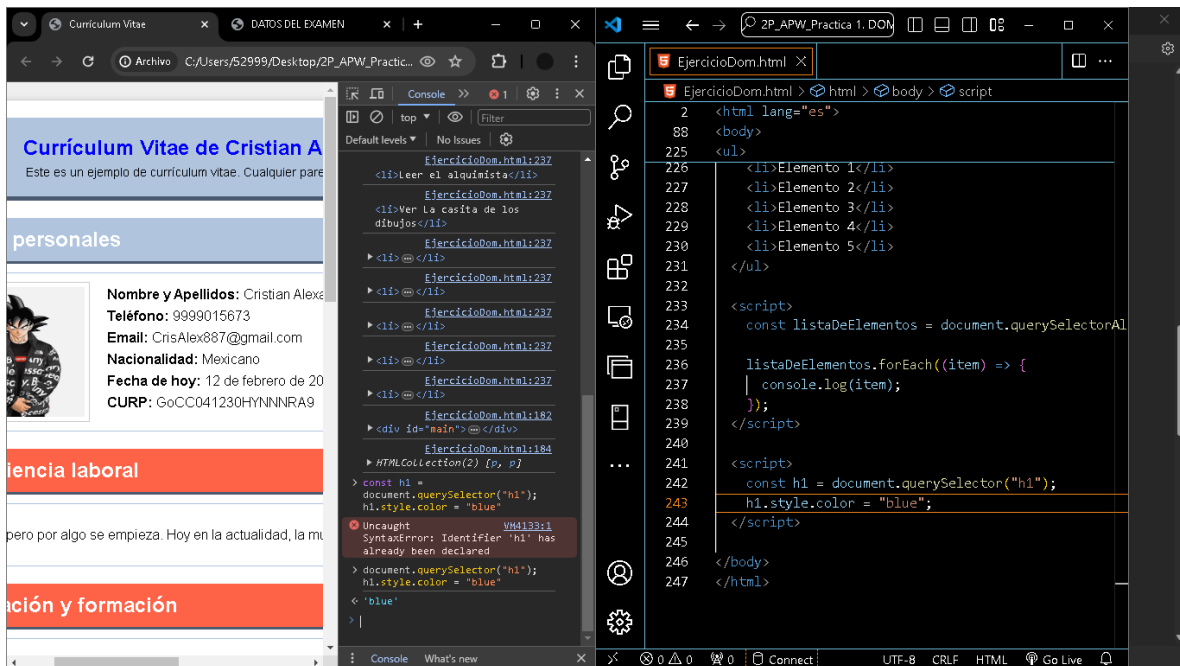
Uso de la propiedad style para cambiar estilos en línea CSS.

```
const h1 = document.querySelector("h1");
```

```
h1.style.color = "blue";
```

Solución

El comando `document.querySelector("h1")` se utiliza para seleccionar el primer elemento `<h1>` en el documento HTML. Luego, `h1.style.color = "blue";` cambia el color del texto de ese elemento a azul.



Ejercicio 5

Inciso a)

Instrucciones

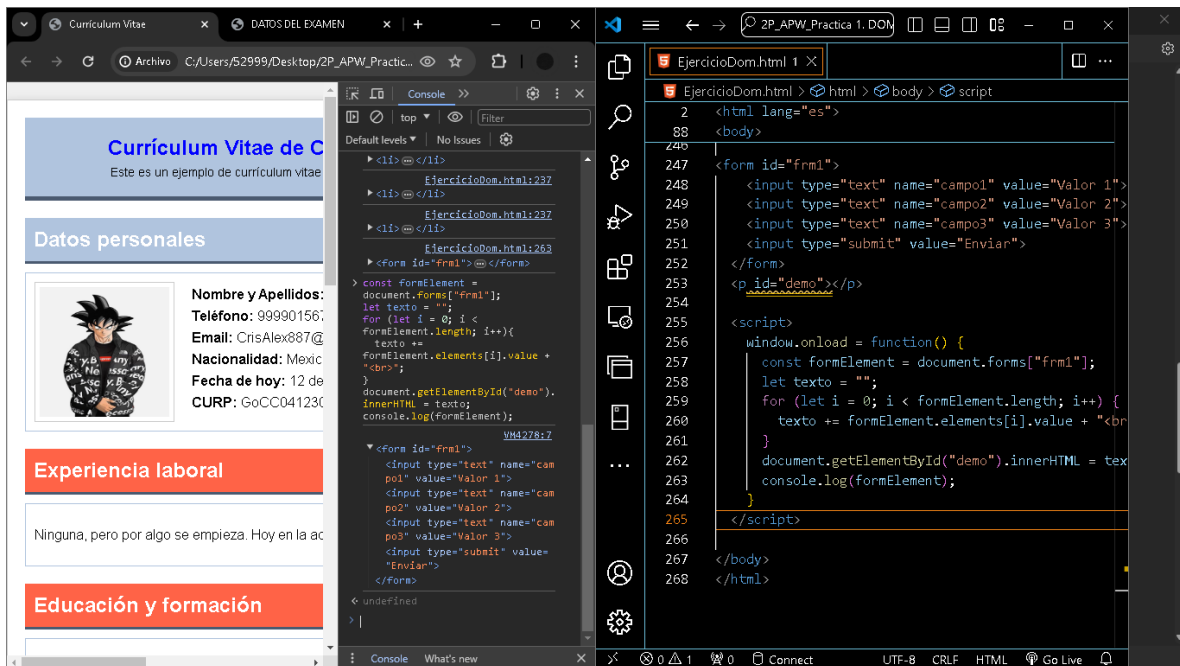
Búsqueda de elementos HTML por colecciones de objetos HTML.

a) Encuentra el elemento de formulario con id="frm1", en la colección de formularios, y muestra todos los valores de los elementos.

```
const x = document.forms["form1"];
let texto = "";
for (let i = 0; i < x.length; i++){
    texto += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = texto;
console.log(x);
```

Solución

Este código selecciona el formulario con el ID "frm1", itera sobre todos sus elementos, y recoge sus valores en una cadena. Luego, muestra estos valores en el elemento con el ID "demo"



Ejercicio 6

Inciso a)

Instrucciones

Agregar nuevos elementos al documento HTML.

Agregar elementos al árbol del DOM usando los métodos `document.createElement()`, `appendChild()` y haciendo uso de la propiedad `textContent`.

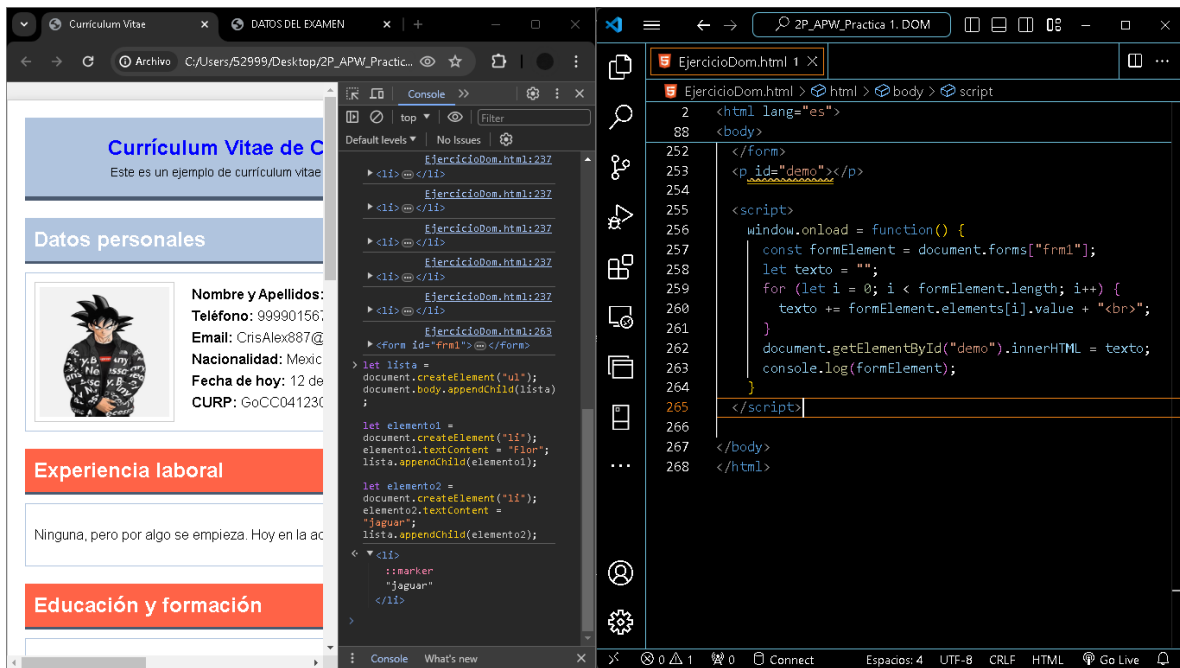
```
let lista = document.createElement("ul");
document.body.appendChild(lista);
```

```
let elemento1 = document.createElement("li");
elemento1.textContent = "Flor";
lista.appendChild(elemento1);
```

```
let elemento2 = document.createElement("li");
elemento2.textContent = "jaguar";
lista.appendChild(elemento2);
```

Solución

En resumen, este código crea una nueva lista desordenada y le agrega dos elementos con los textos "Flor" y "jaguar".



Ejercicio 7

Inciso a)

Instrucciones

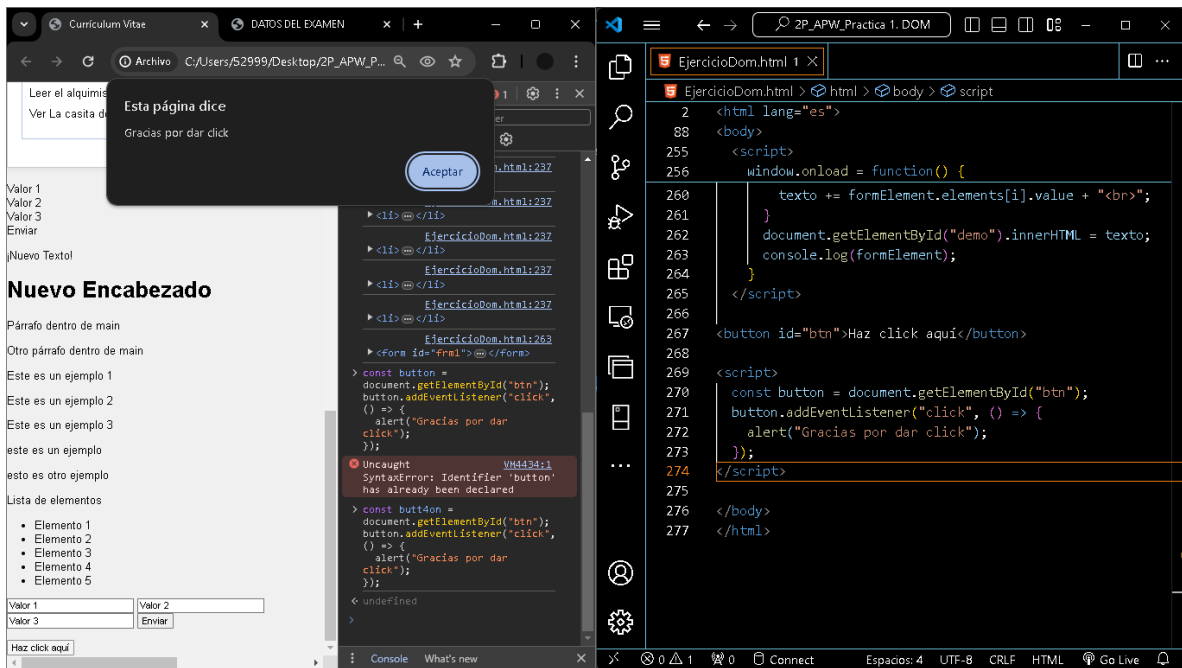
Usando el método `addEventListener()` para escuchar eventos en la página.

Crea un botón en el documento html, donde al hacer click aparezca un mensaje de alerta.

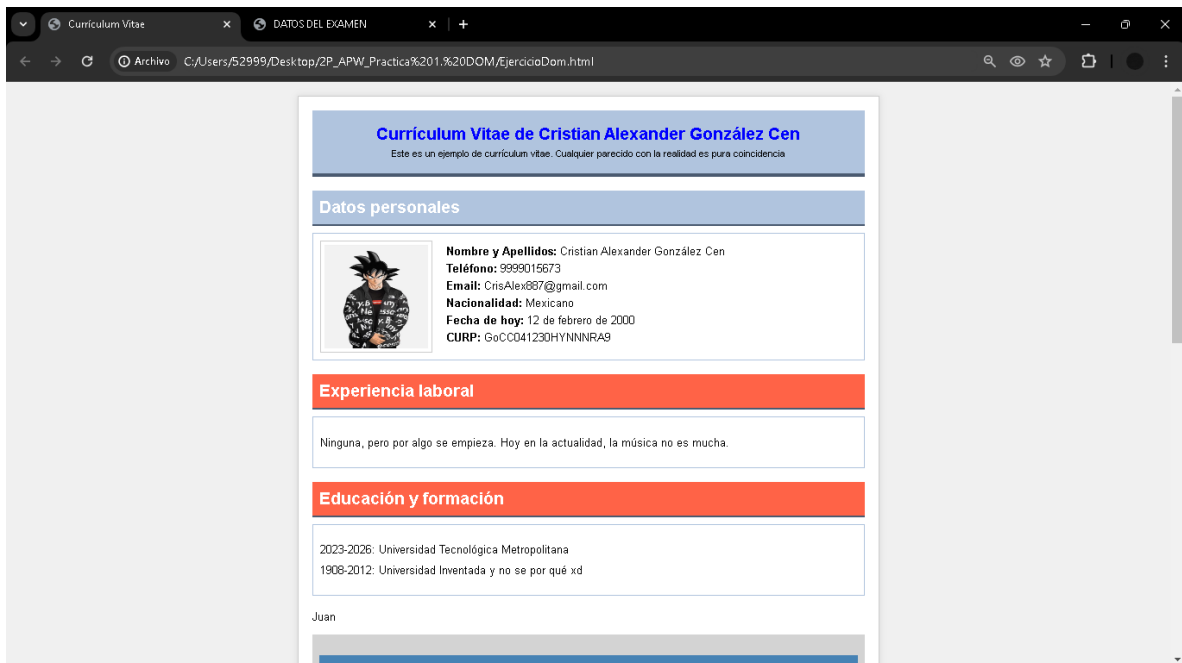
```
const button = document.getElementById("btn");
button.addEventListener("click", () => {
  alert("Gracias por dar click");
});
```

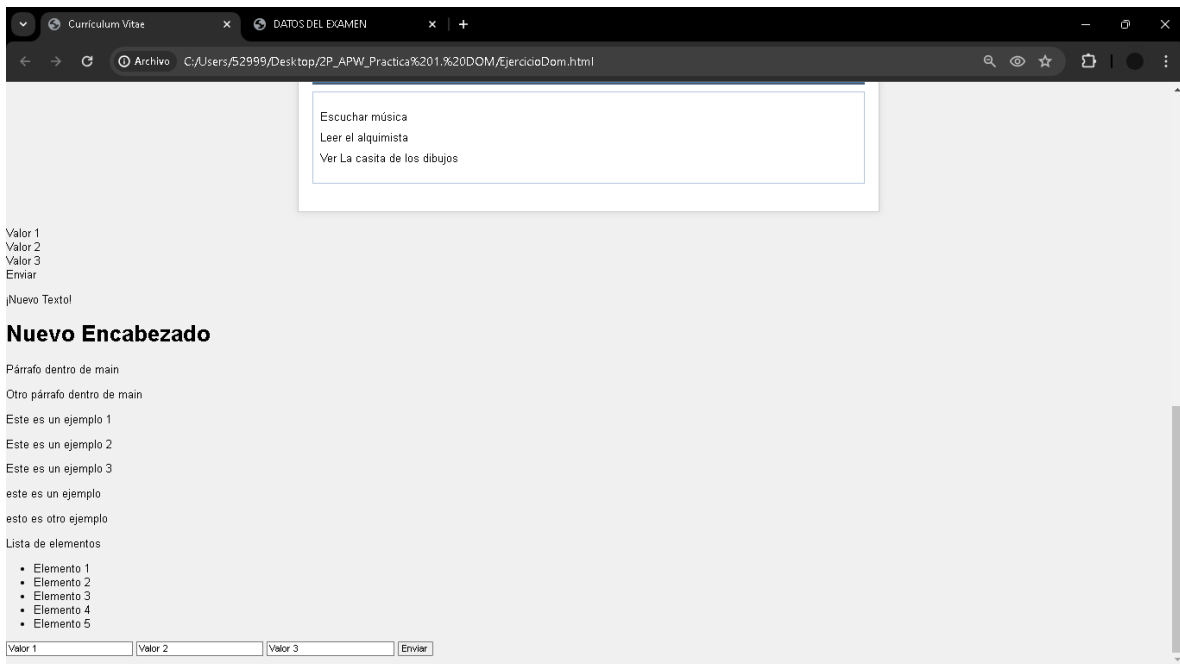
Solución

Este código selecciona un botón con el ID "btn" y le añade un evento de escucha para el evento de clic. Cuando el botón es clicado, se ejecuta una función que muestra una alerta con el mensaje "Gracias por dar click".



Hasta ahorita se ve de esta forma





Ejercicio 8

Inciso a)

Instrucciones

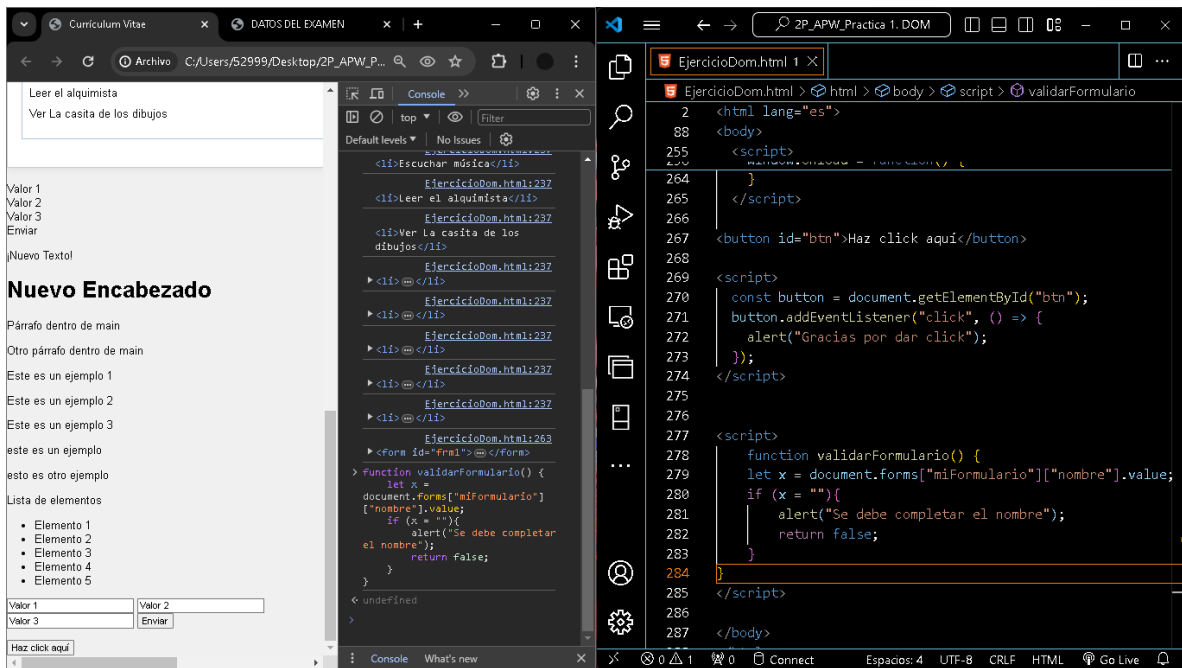
Validación de formulario.

Si un campo de formulario (name) esta vacío, se muestra una alerta con un mensaje y devuelve falso para evitar que se envíe y se redirigione a otra página.

```
function validarFormulario() {
    let x = document.forms["miFormulario"]["nombre"].value;
    if (x == ""){
        alert("Se debe completar el nombre");
        return false;
    }
}
```

Solución

Esta función validarFormulario se utiliza para asegurarse de que el campo "nombre" de un formulario no esté vacío antes de permitir el envío del formulario. Si el campo está vacío, se muestra una alerta al usuario indicando que debe completar el nombre, y se evita el envío del formulario devolviendo false. En mi caso no devolvió nada porque no tenía un formulario en mi html



Ejercicio 9

Inciso a)

Instrucciones

Cambiar el valor de un atributo

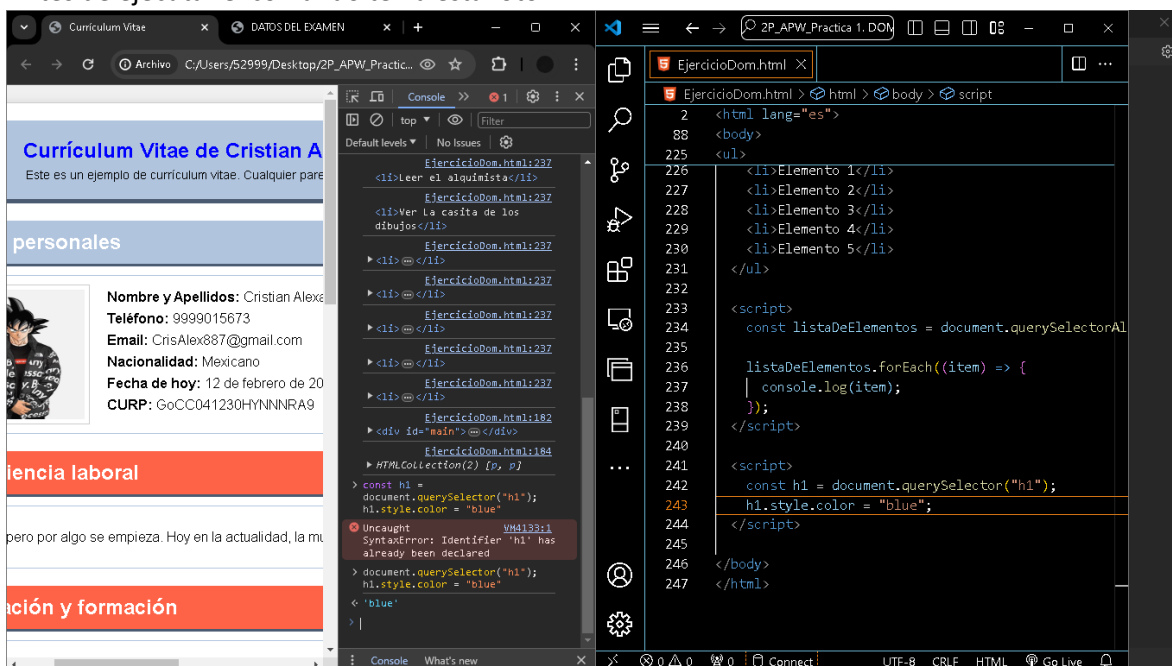
Cambiar el valor de un atributo src de un elemento .

`document.getElementById("milimagen").src = "cat.jpg";`

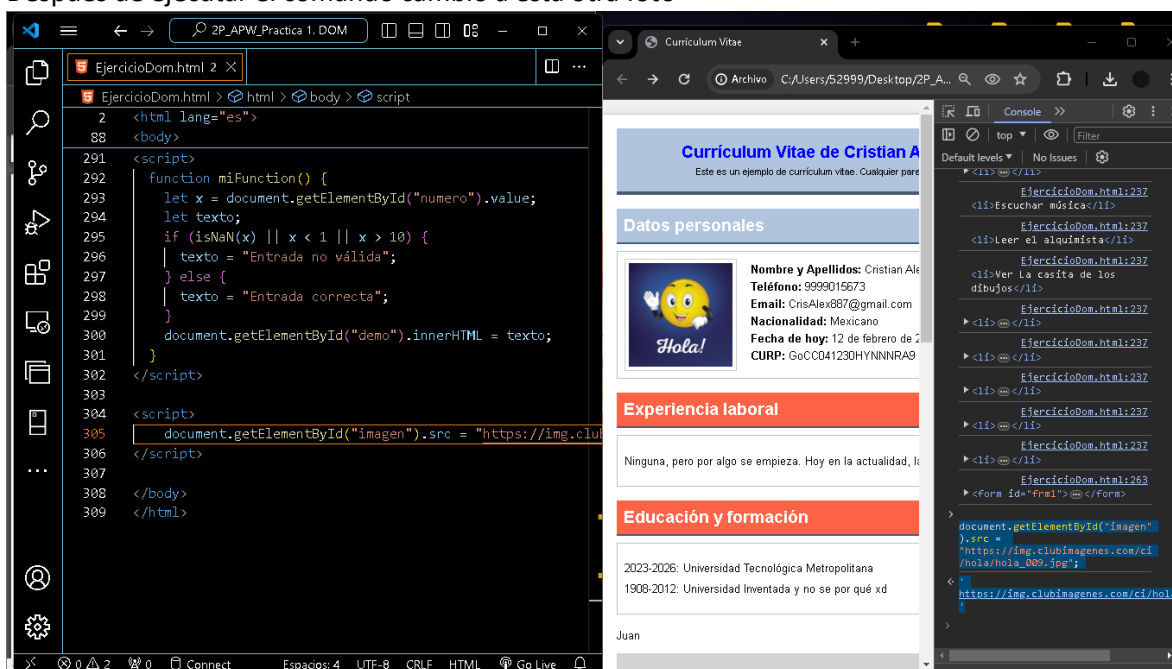
Solución

El comando `document.getElementById("milimagen").src = "cat.jpg";` me sirvió para cambiar la fuente (source) de una imagen en un documento HTML

Antes de ejecutar el comando tenía esta foto:



Despues de ejecutar el comando cambio a esta otra foto



Ejercicio 10

Inciso a)

Instrucciones

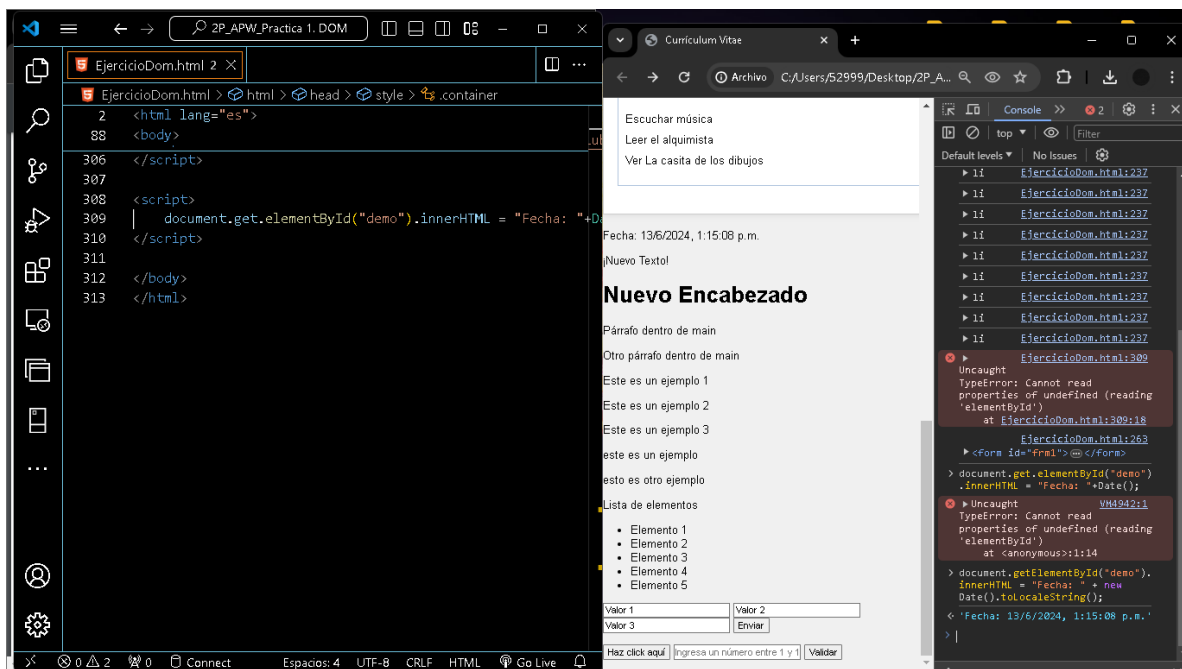
Contenido dinámico

Agregar la hora actual a una etiqueta con `id="demo"`.

```
document.getElementById("demo").innerHTML = "Fecha: "+Date();
```

Solución

El comando `document.getElementById("demo").innerHTML = "Fecha: " + Date();` lo utilice para mostrar la fecha y hora actual en mi documento HTML.



Conclusión

En esta práctica sobre la manipulación del DOM con JavaScript, aprendimos a interactuar de manera dinámica con elementos HTML, comenzando por cambiar el contenido usando `getElementById` e `innerHTML`, lo cual es fundamental para actualizar la información en una página web sin recargarla.

Exploramos la búsqueda de elementos por etiqueta, clase y mediante selectores CSS avanzados, lo que nos permitió hacer modificaciones masivas y precisas. Además, comprendimos cómo manejar colecciones de objetos HTML como formularios, creando y añadiendo nuevos elementos al DOM para nuestras páginas.

Implementamos `addEventListener` para responder a eventos de usuario, esencial para interfaces interactivas, y desarrollamos funciones de validación de formularios para asegurar la integridad de los datos. También manipulamos atributos de elementos HTML y actualizamos contenido dinámico como la fecha y hora actual. Esta práctica consolidó nuestro entendimiento de cómo gestionar elementos HTML de manera eficiente, preparándonos para crear aplicaciones web dinámicas y mejorando la experiencia del usuario.

Link del repositorio de GitHub: https://github.com/CristianCenxd/Pr-ctica_1_DOM_C-digo.git