

**Universidad Tecnológica Metropolitana**

Aplicaciones web

Práctica 1

Mstra. Martínez Dominguez Ruth Betzaida

**Almno.** 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](#_Toc169190468)

[inciso a) 7](#_Toc169190469)

[Instrucciones 7](#_Toc169190470)

[Solución 7](#_Toc169190471)

[Inciso b) 7](#_Toc169190472)

[Instrucciones 7](#_Toc169190473)

[Solución 7](#_Toc169190474)

[Inciso c) 8](#_Toc169190475)

[Instruciones 8](#_Toc169190476)

[Solución 8](#_Toc169190477)

[Inciso d) 9](#_Toc169190478)

[Instrucciones 9](#_Toc169190479)

[Solución 9](#_Toc169190480)

[Ejercicio 2 10](#_Toc169190481)

[Inciso a) 10](#_Toc169190482)

[Instrucciones 10](#_Toc169190483)

[Solución 10](#_Toc169190484)

[Inciso b) 11](#_Toc169190485)

[Instrucciones 11](#_Toc169190486)

[Solución 11](#_Toc169190487)

[Ejercicio 3 12](#_Toc169190488)

[Inciso a) 12](#_Toc169190489)

[Instrucciones 12](#_Toc169190490)

[Solución 13](#_Toc169190491)

[Ejercicio 4 13](#_Toc169190492)

[Inciso a) 13](#_Toc169190493)

[Instrucciones 13](#_Toc169190494)

[Solución 14](#_Toc169190495)

[Inciso b) 14](#_Toc169190496)

[Instrucciones 14](#_Toc169190497)

[Solución 15](#_Toc169190498)

[Inciso c) 15](#_Toc169190499)

[Instrucciones 15](#_Toc169190500)

[Solución 15](#_Toc169190501)

[Inciso d) 16](#_Toc169190502)

[Instrucciones 16](#_Toc169190503)

[Solución 16](#_Toc169190504)

[Inciso e) 17](#_Toc169190505)

[Instrucciones 17](#_Toc169190506)

[Solución 17](#_Toc169190507)

[Inciso f) 18](#_Toc169190508)

[Instrucciones 18](#_Toc169190509)

[Solución 18](#_Toc169190510)

[Ejercicio 5 19](#_Toc169190511)

[Inciso a) 19](#_Toc169190512)

[Instrucciones 19](#_Toc169190513)

[Solución 19](#_Toc169190514)

[Ejercicio 6 20](#_Toc169190515)

[Inciso a) 20](#_Toc169190516)

[Instrucciones 20](#_Toc169190517)

[Solución 20](#_Toc169190518)

[Ejercicio 7 21](#_Toc169190519)

[Inciso a) 21](#_Toc169190520)

[Instrucciones 21](#_Toc169190521)

[Solución 21](#_Toc169190522)

[Ejercicio 8 23](#_Toc169190523)

[Inciso a) 23](#_Toc169190524)

[Instrucciones 23](#_Toc169190525)

[Solución 23](#_Toc169190526)

[Ejercicio 9 24](#_Toc169190527)

[Inciso a) 24](#_Toc169190528)

[Instrucciones 24](#_Toc169190529)

[Solución 24](#_Toc169190530)

[Ejercicio 10 26](#_Toc169190531)

[Inciso a) 26](#_Toc169190532)

[Instrucciones 26](#_Toc169190533)

[Solución 26](#_Toc169190534)

**Introducción**

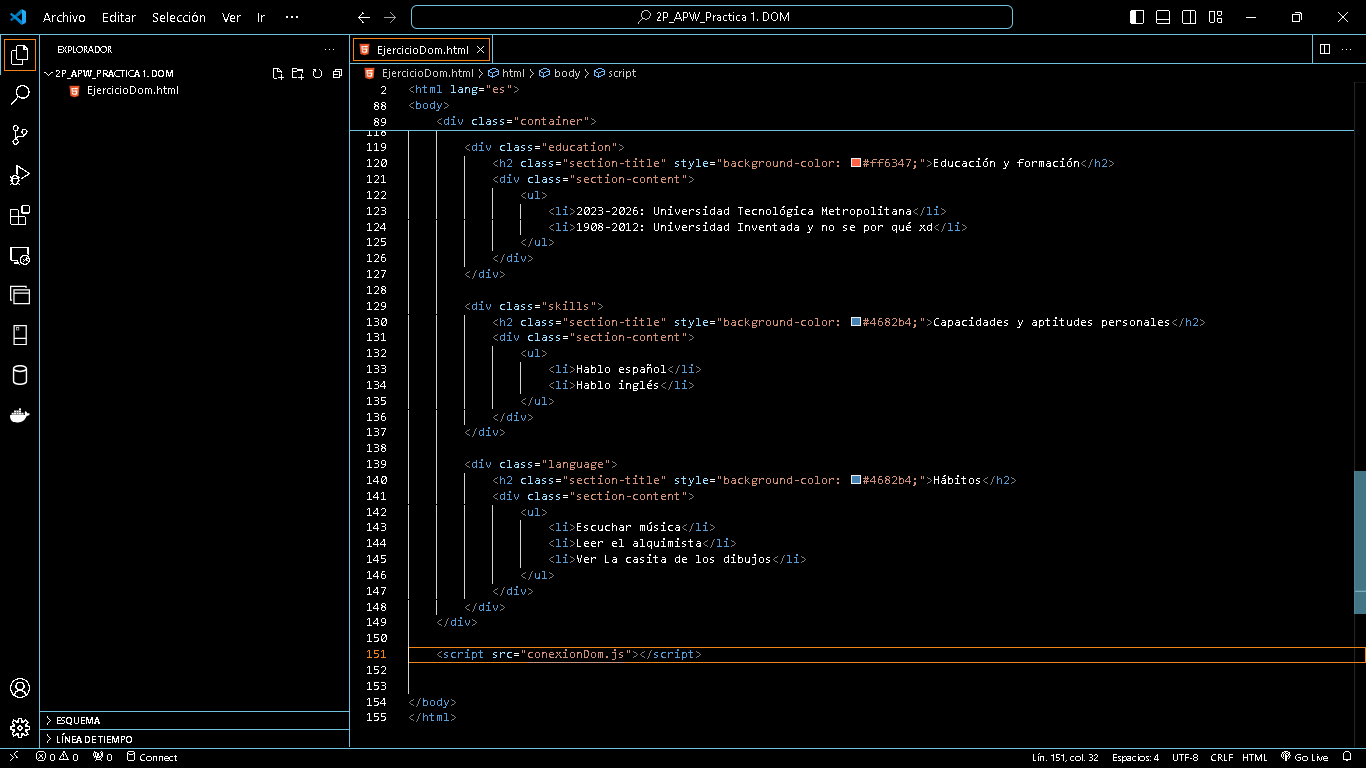
En esta práctica de la materia Aplicaciones de Desarrollo Web, nos adentraremos en la manipulación del Document Object Model (DOM) utilizando JavaScript.

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 trabjar con el. En mi caso yo escogí el curriculum.html y le cambié el nombre por EjercicioDom.html.





# Ejercicio 1

## inciso a)

### Instrucciones

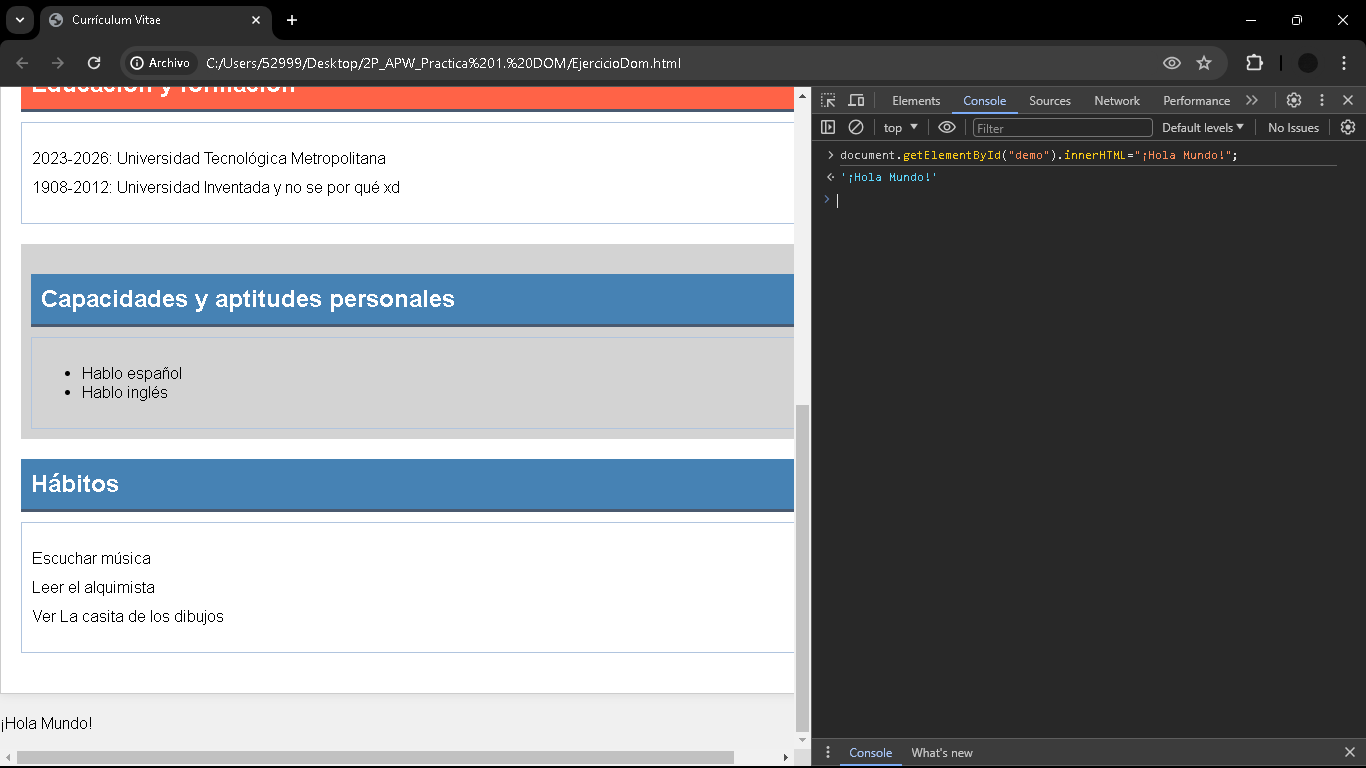
*Uso del método getElementById y de la propiedad innerHTML.*

*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)

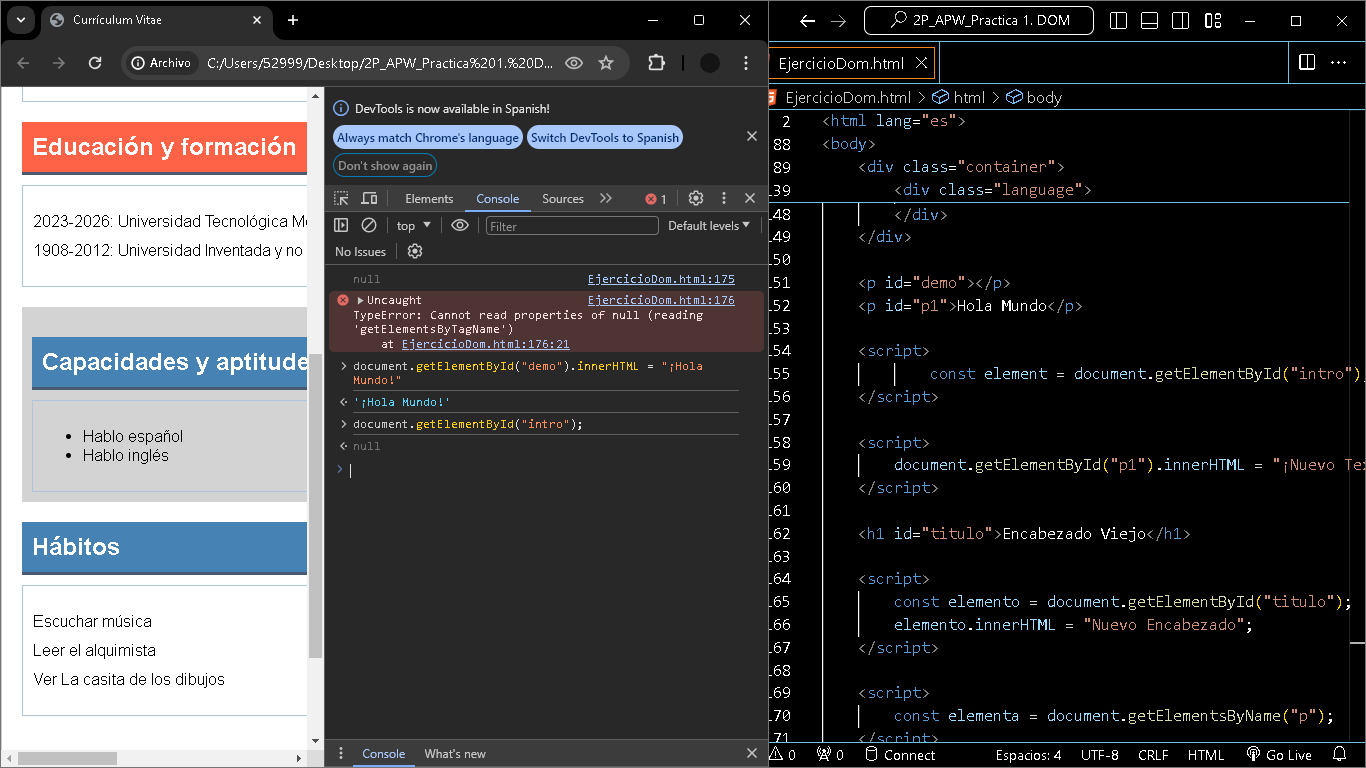
### Instrucciones

*Encuentra el elemento con id=”intro”*

*const elment = 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)

### Instruciones

*Cambia el elemento de una etiqueta <p>*

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

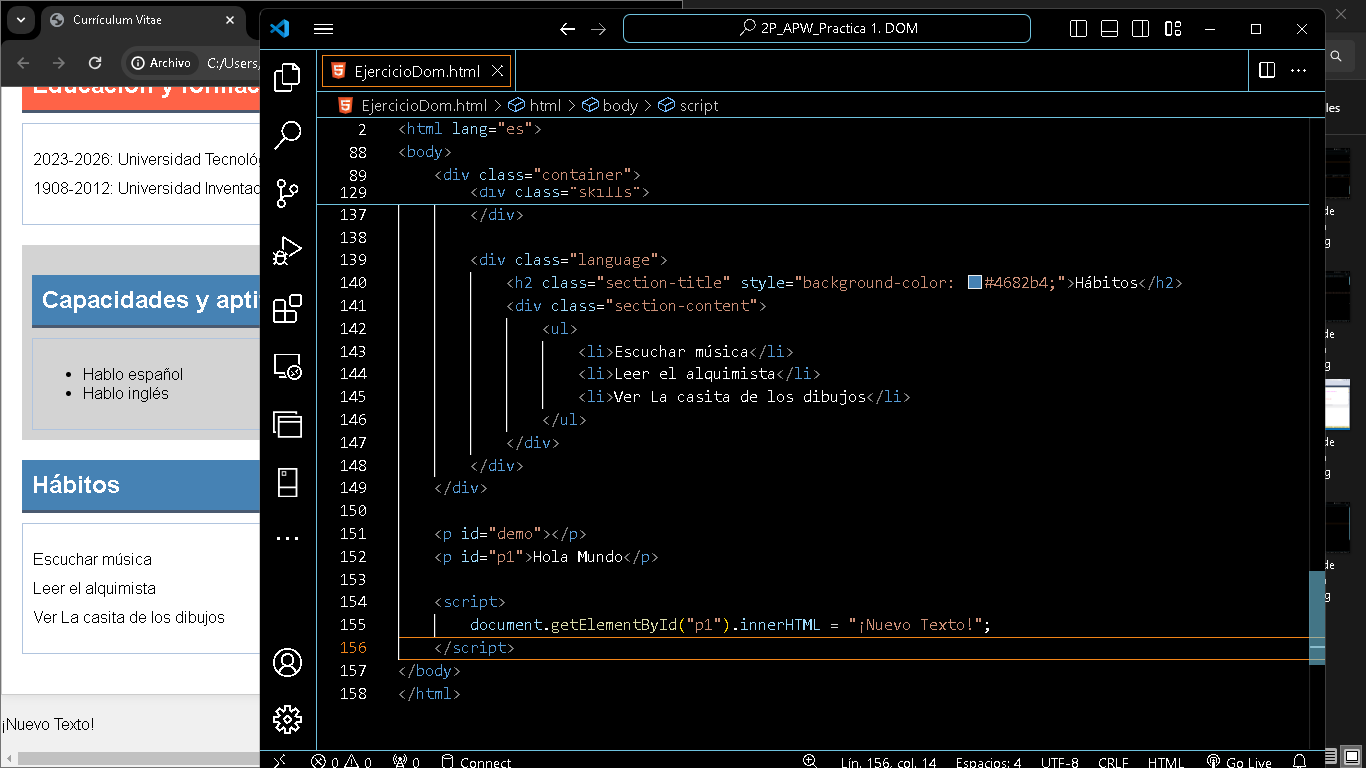
*<script>*

*document.getElementById("p1").innerHTML=¡Nuevo 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>*

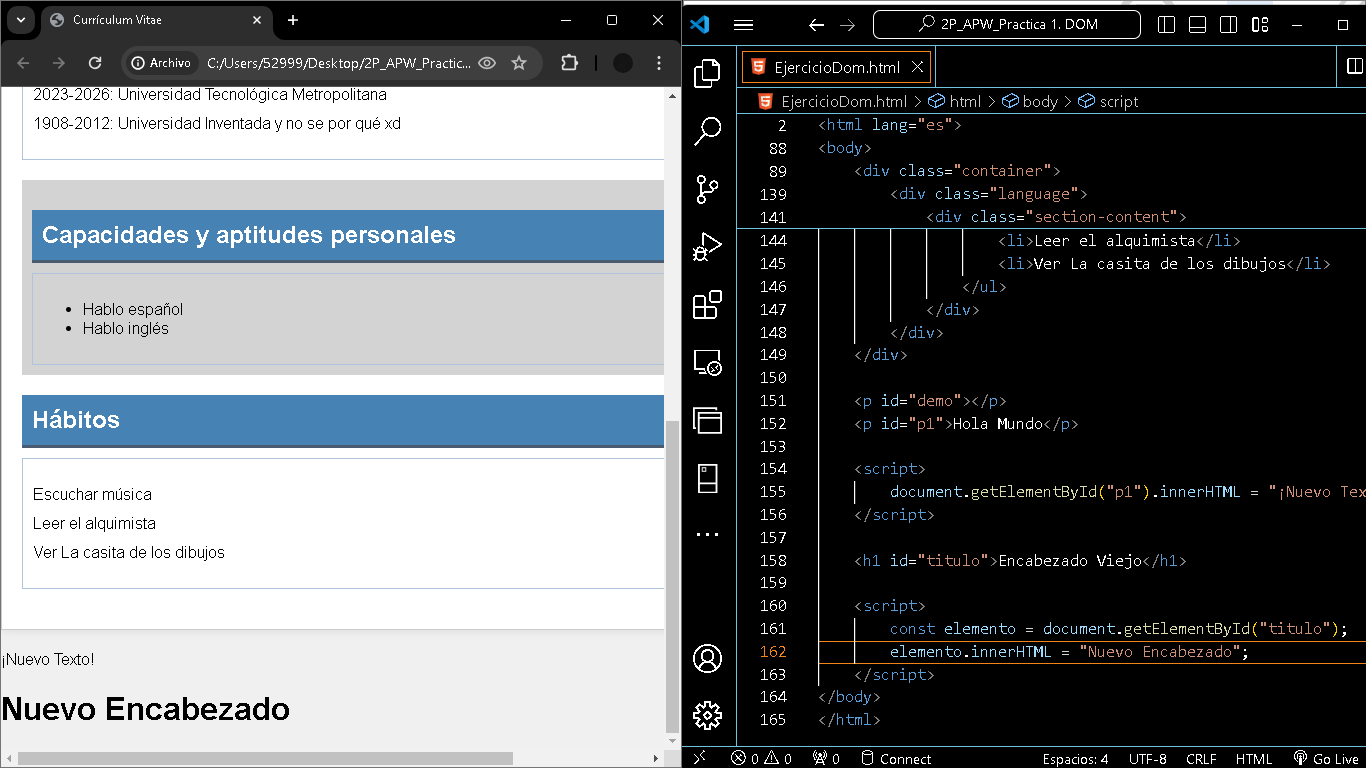
*const elemento = document.getElementById("titulo");*

*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

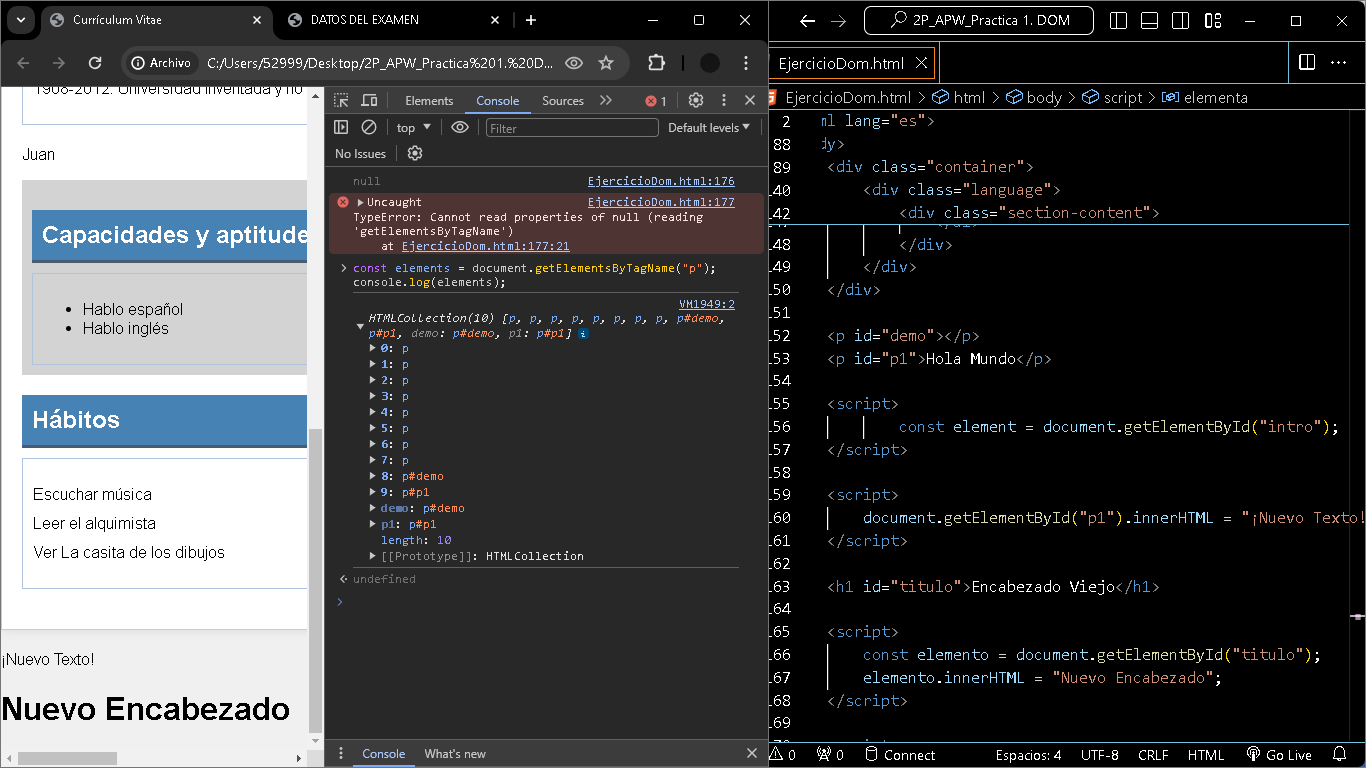
*Búsqueda de elementos HTML por nombre de etiqueta.*

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

*const element = document.getElementsByName("p");*

### Solución

El comando const element = document.getElementsByName("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");*

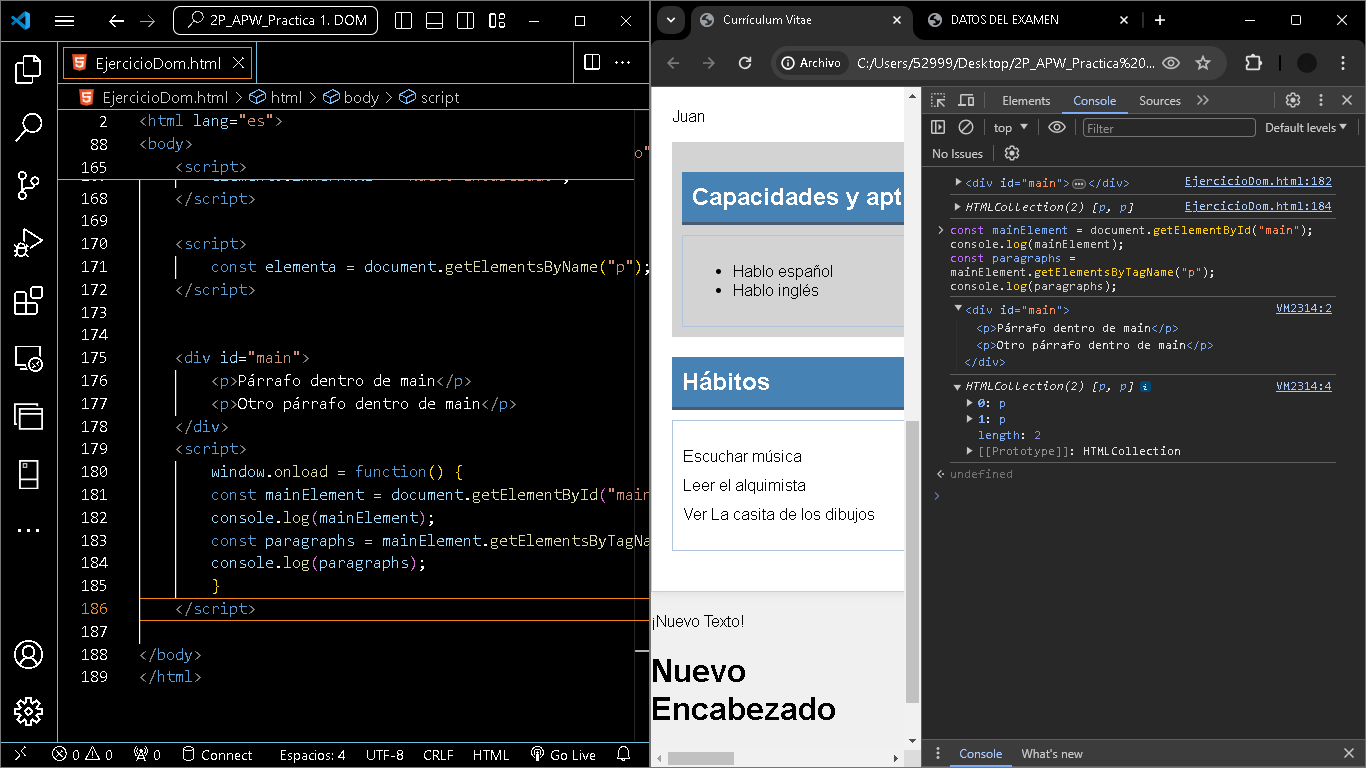
*console.log(x);*

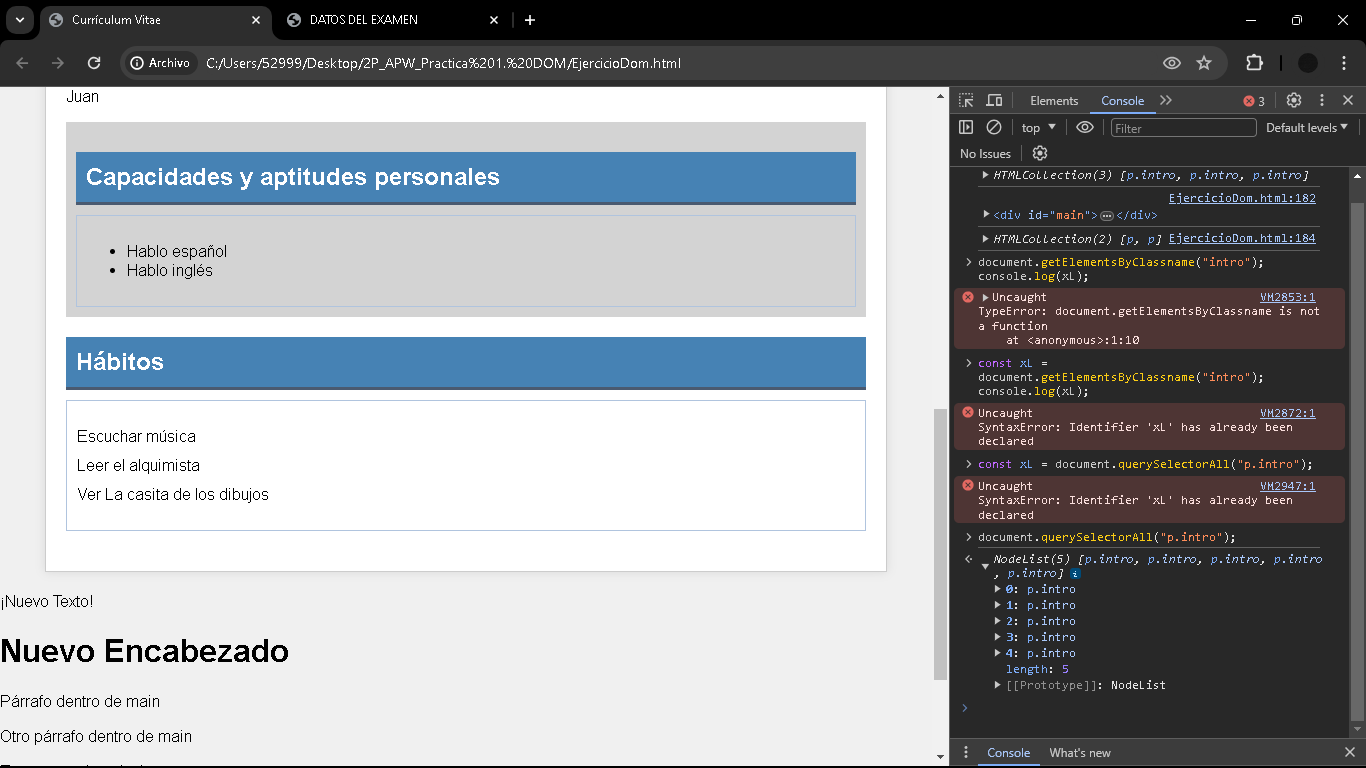
*const y = x.getElementsByTagName("p");*

*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:*

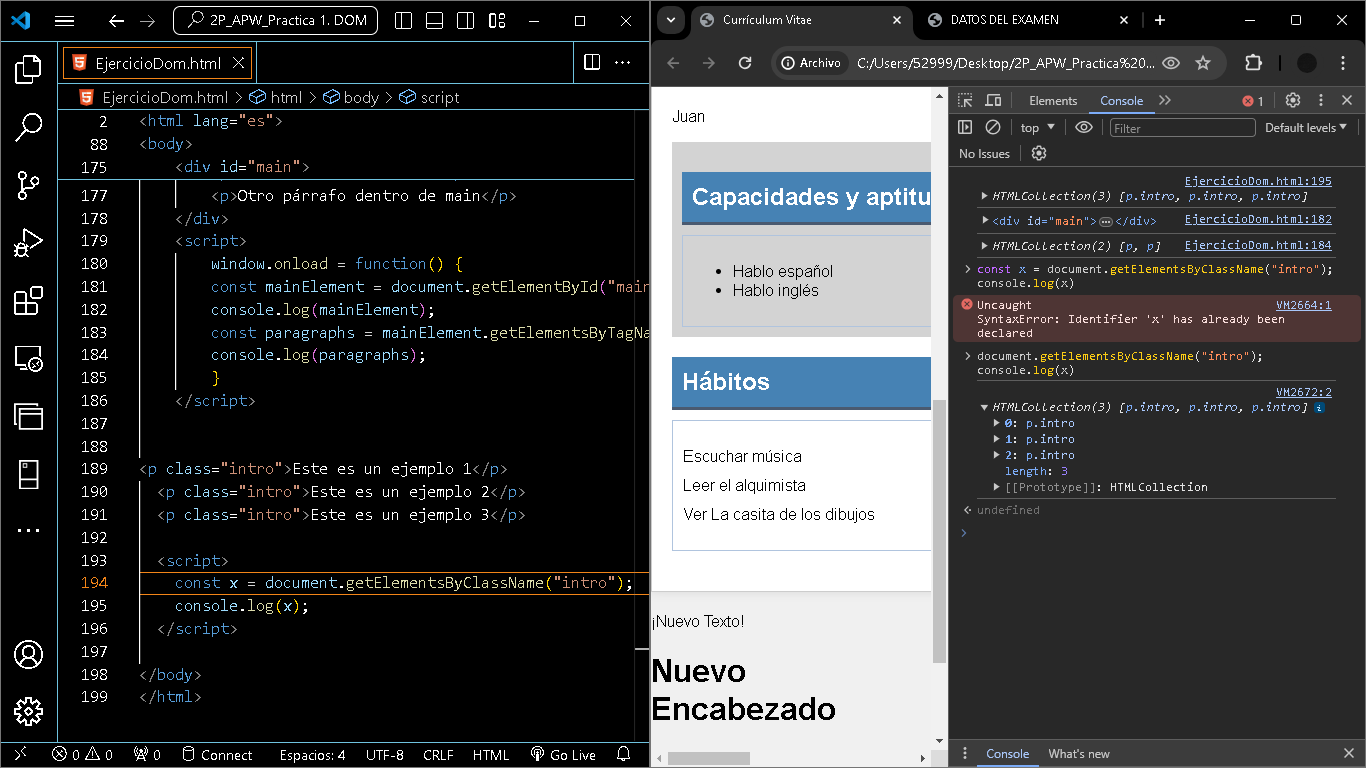
*getElementsByClassName().*

*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");*

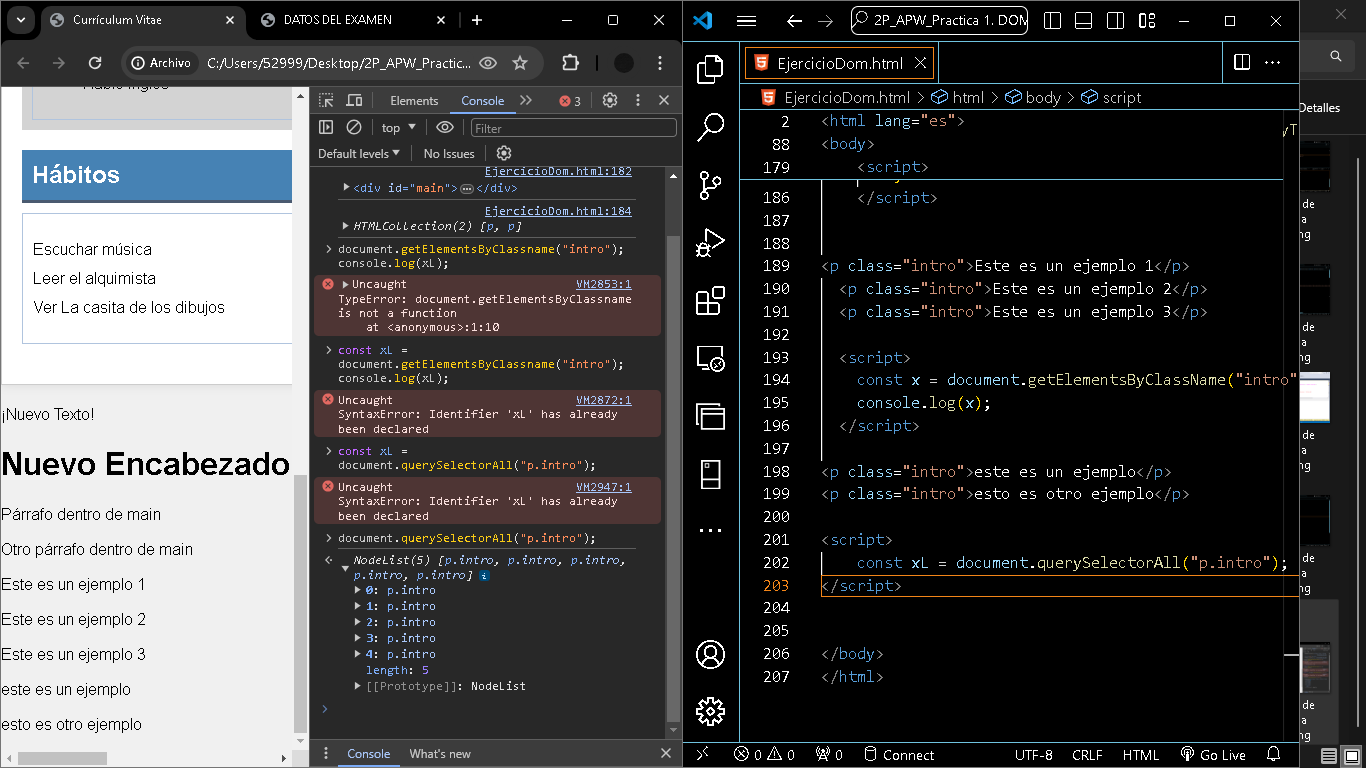
*</script>*

### 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

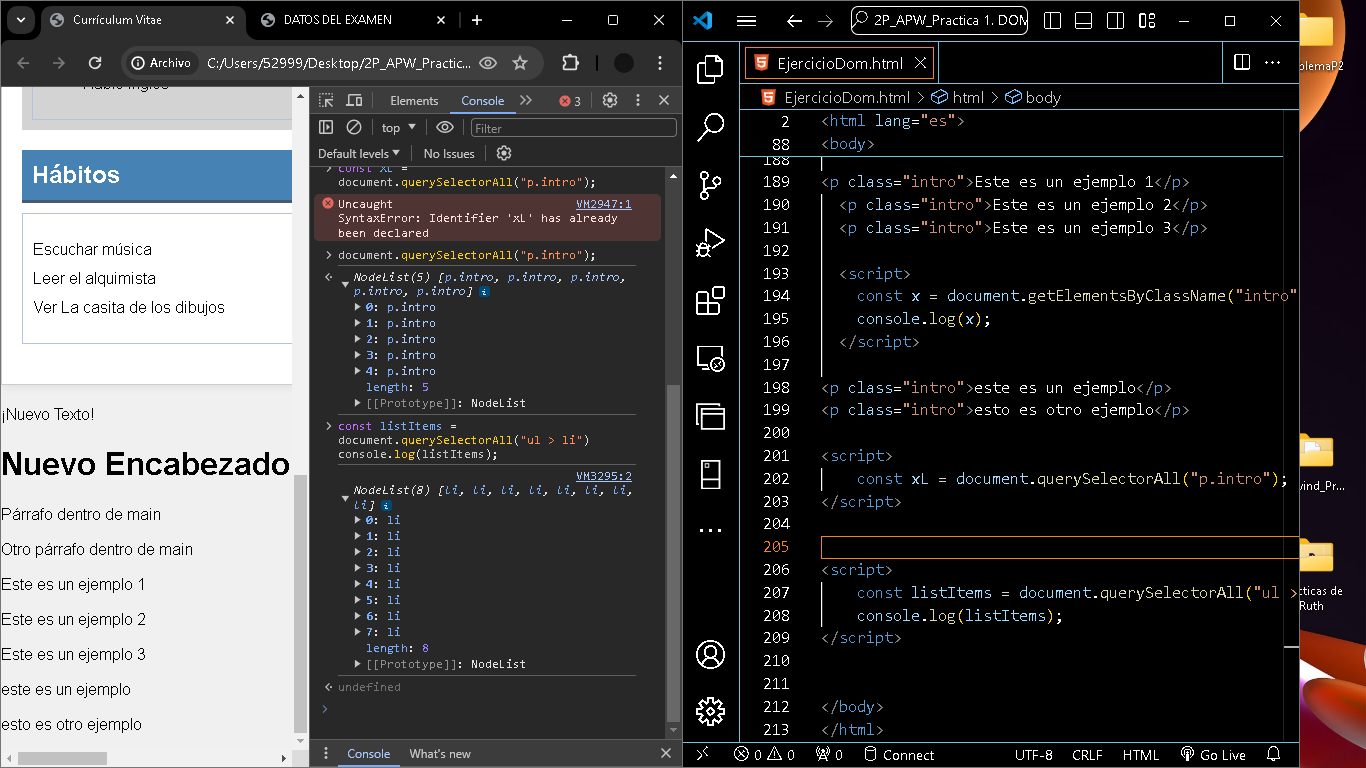
*Encuentra los elementos <li>hijos de <ul>, para ello debes crear una lista desordenada con al menos 5 elementos dentro de la lista.*

*const listItems = document.querySelectorAll("ul > li")*

*console.log(listItems);*

### Solución

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



## Inciso c)

### Instrucciones

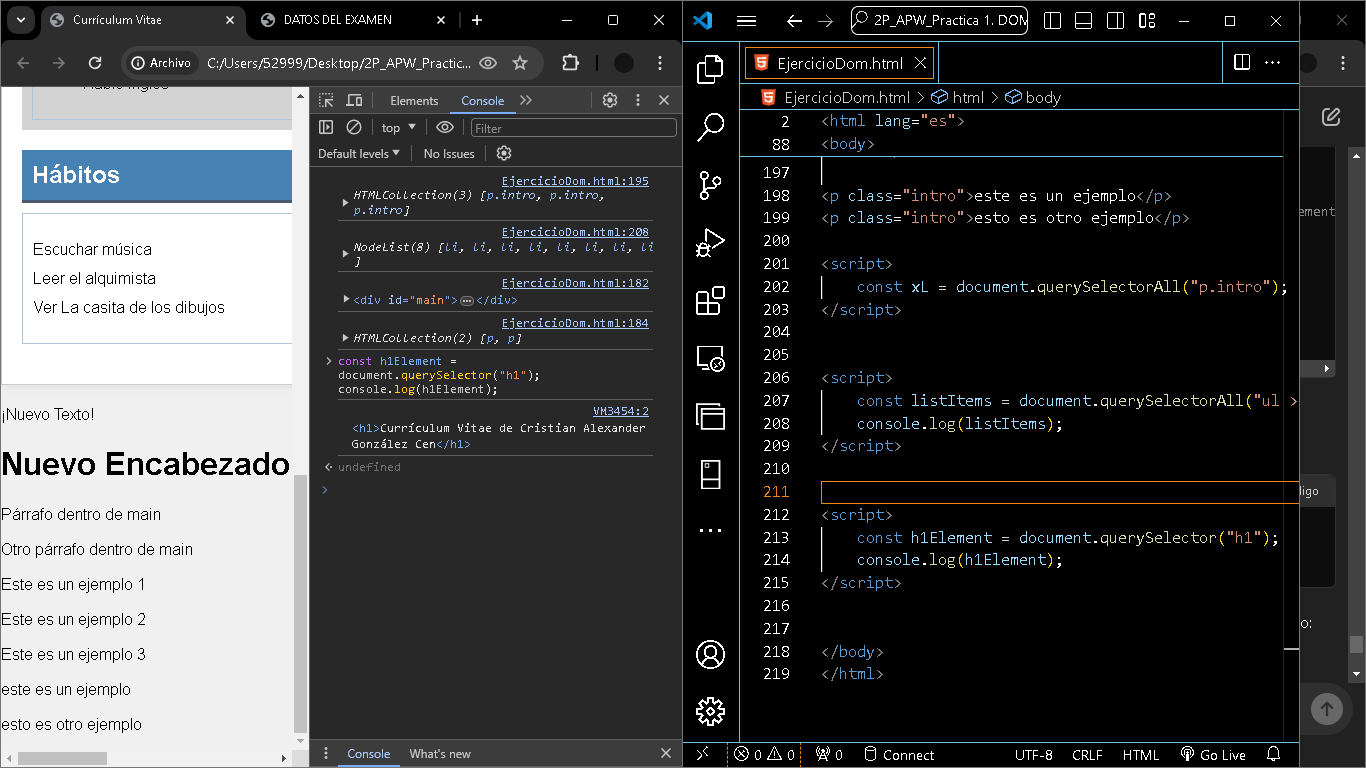
*Encontrar en la consola e imprimir el elemento<h1>.*

*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

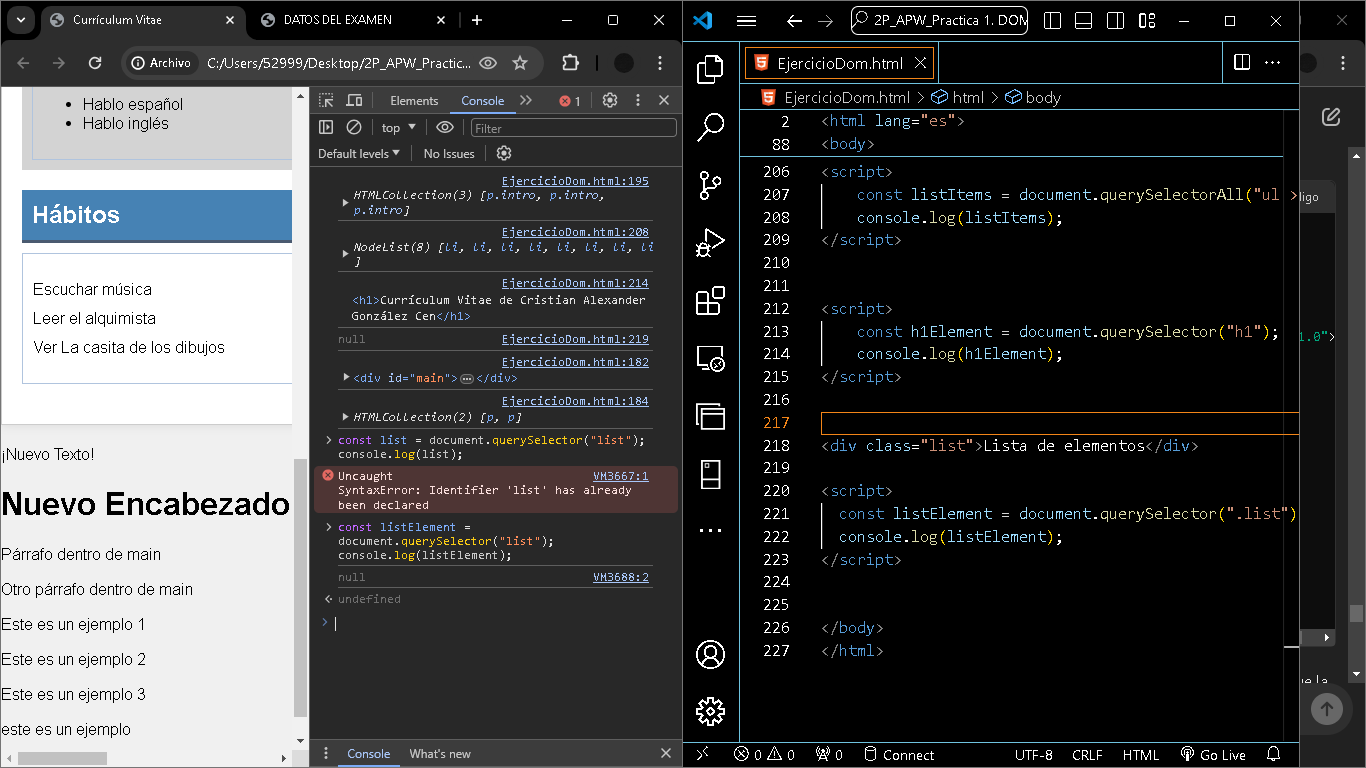
*Encontrar la clase listusando querySelector().*

*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 <li>haciendo uso del ciclo forEach()para iterar sobre la NodeList e imprimir cada uno de los elementos.*

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

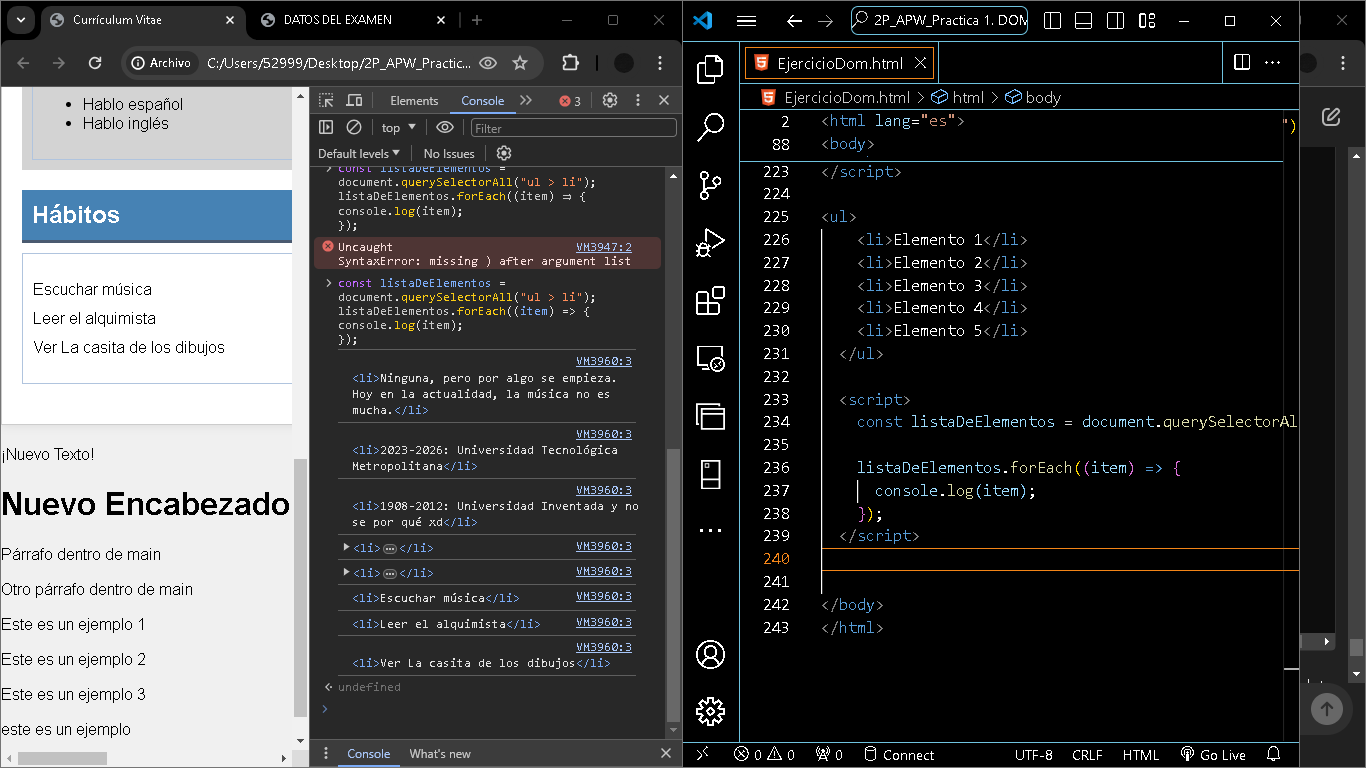
*listaDeElementos.forEach((item) ⇒ {*

*console.log(item);*

*});*

### Solución

El comando document.querySelectorAll("ul > li"): Selecciona todos los elementos <li> que son hijos directos de cualquier elemento <ul>, 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): Mustro el resultado de cada elemento <li> en la consola.



## Inciso f)

### Instrucciones

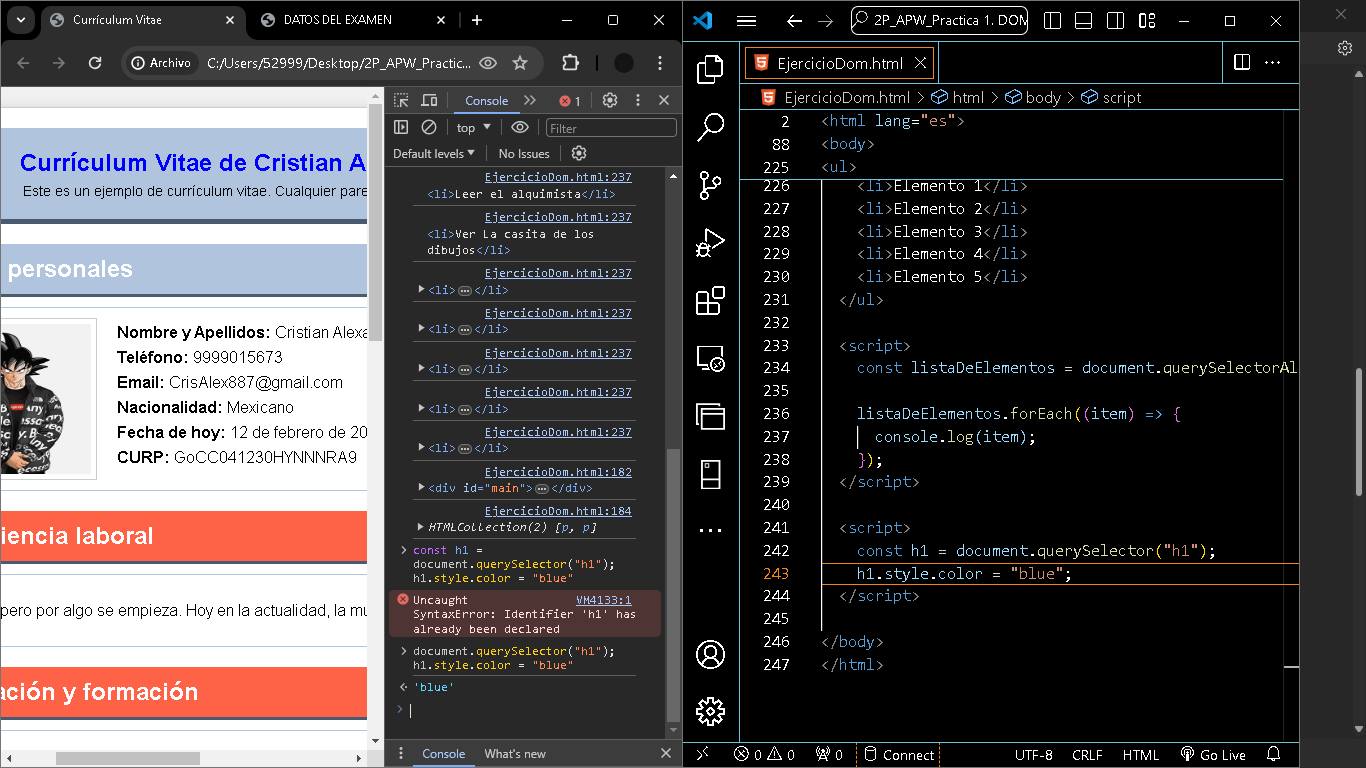
*Uso de la propiedad style para cambiar estilos en línea CSS.*

*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>";*

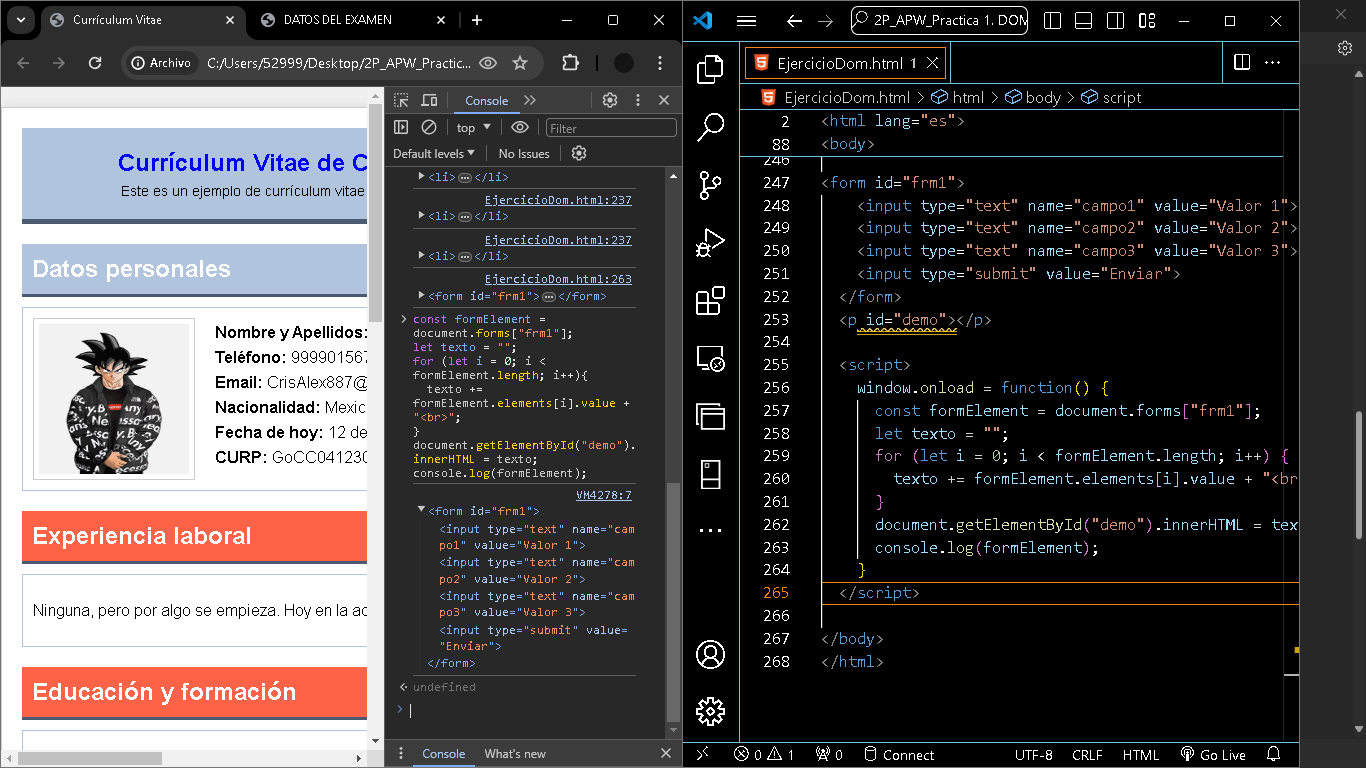
*}*

*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 losmétodosdocument.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);*

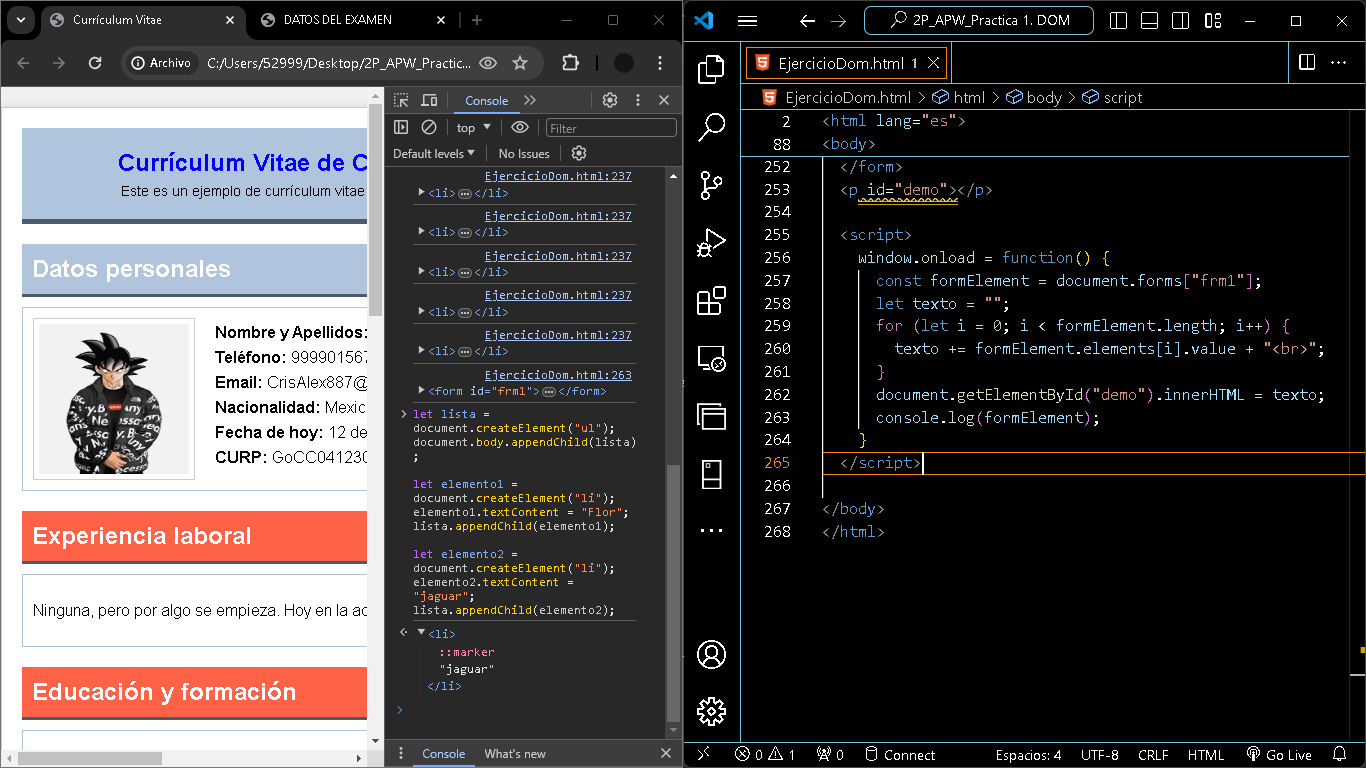
*let elemento2 = document.createElement("li");*

*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");*

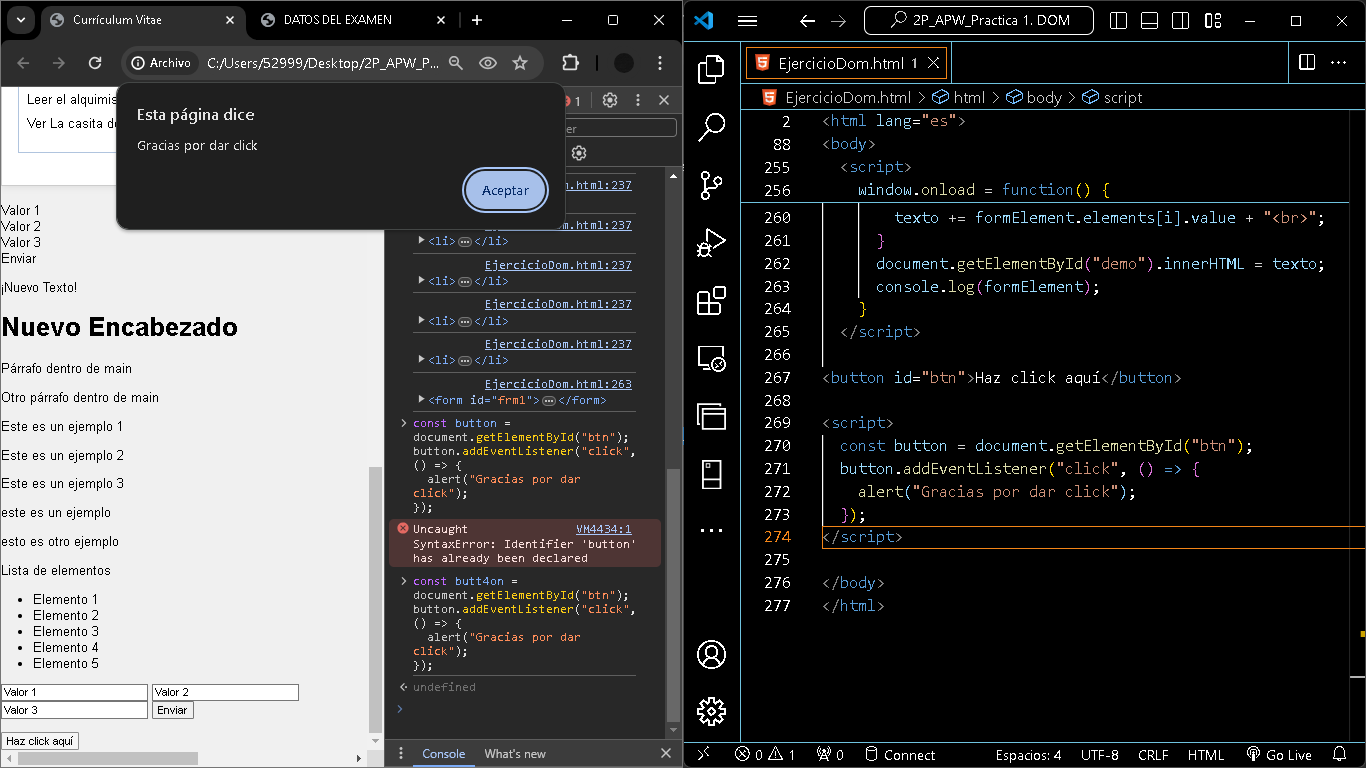
*button.addEventListener("click", () ⇒ {*

*alert("Gracias por dar click");*

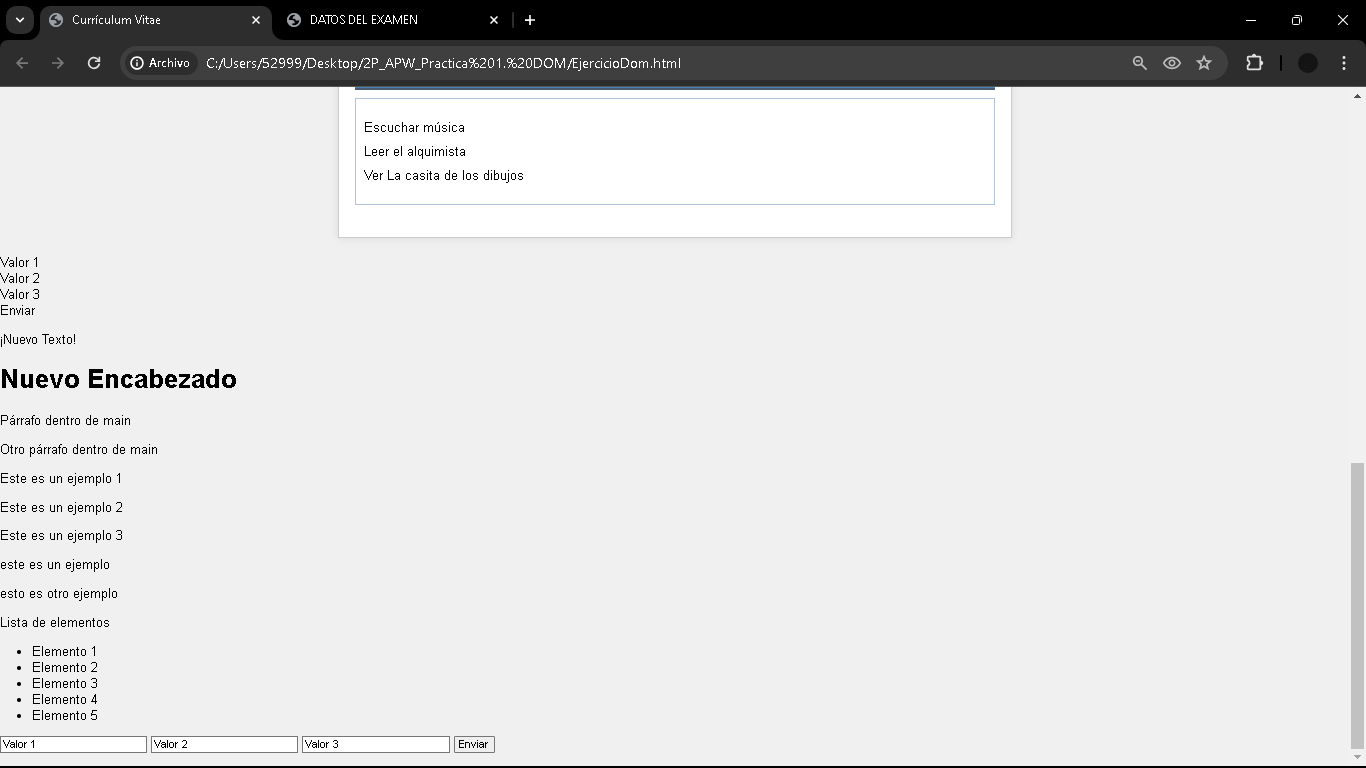
*});*

### 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 redireccione a otra página.*

*function validarFormulario() {*

*let x = document.forms["miFormulario"]["nombre"].value;*

*if (x = ""){*

*alert("Se debe completar el nombre");*

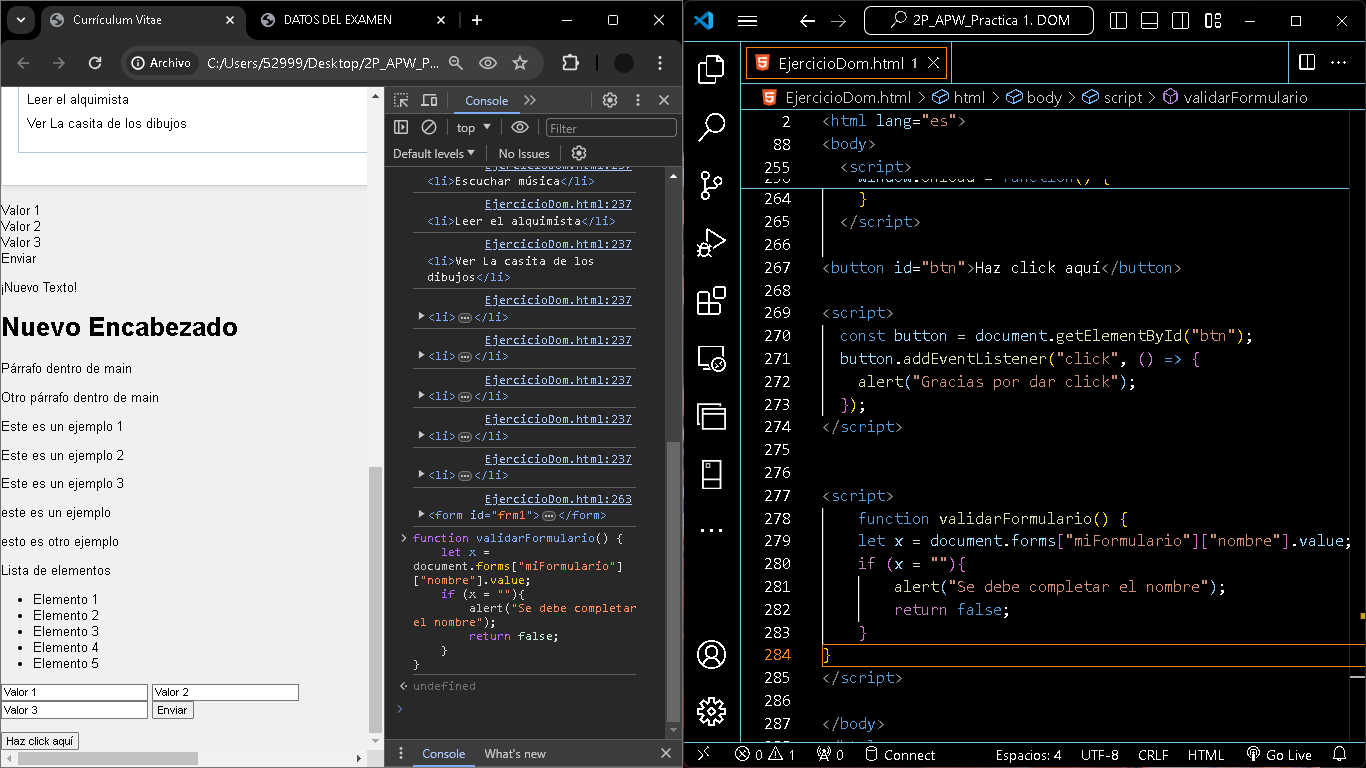
*return false;*

*}*

*}*

### 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

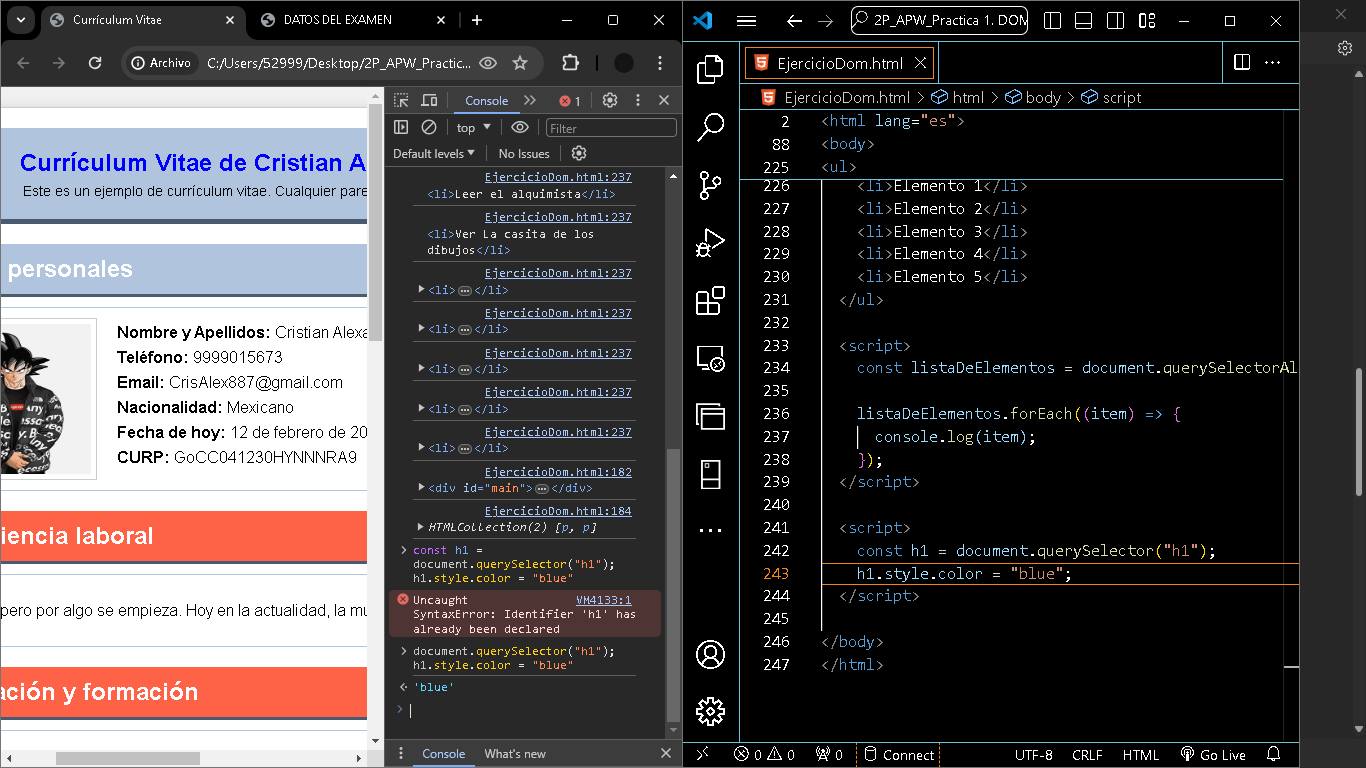
*Cambiar el valor de un atributo*

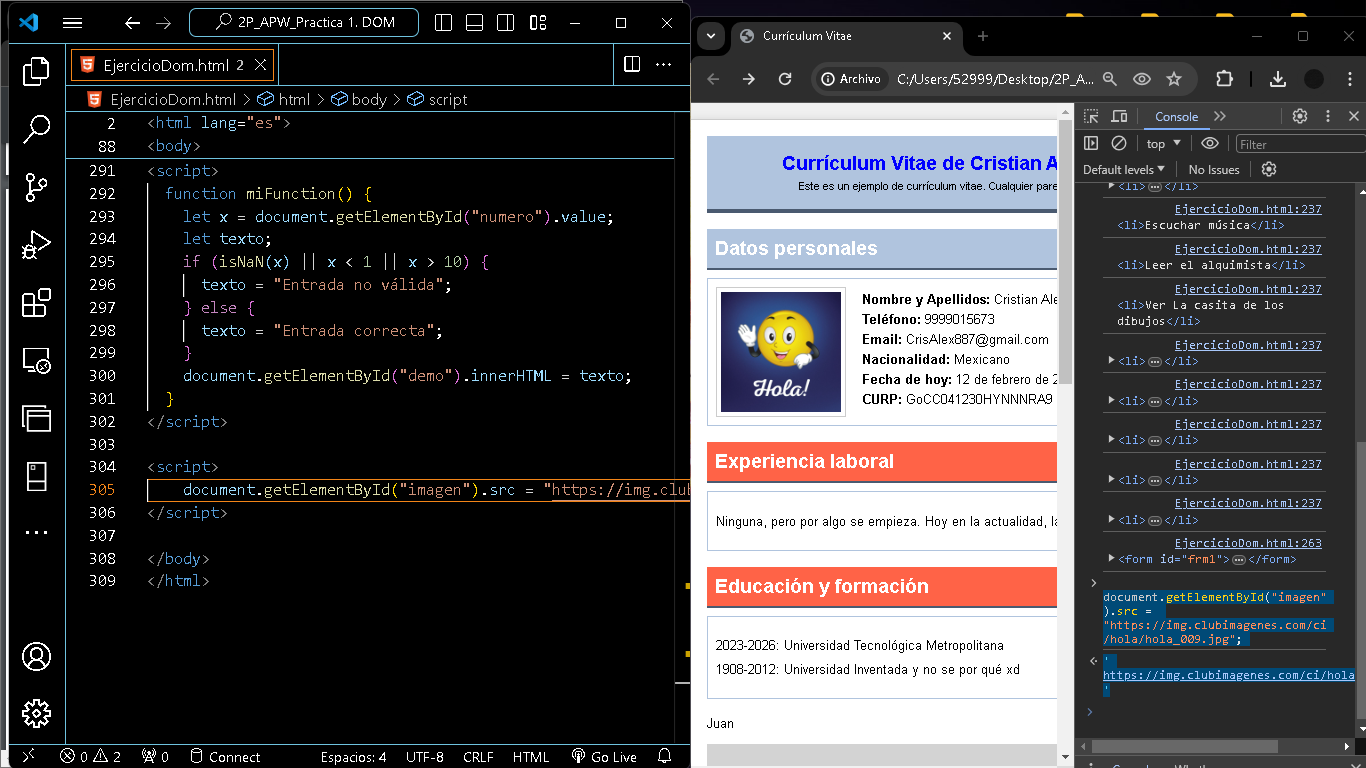
*Cambiar el valor de un atributo src de un elemento <img>.*

*document.getElementById(""miImagen).src = "cat.jpg";*

### Solución

El comando document.getElementById("miImagen").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

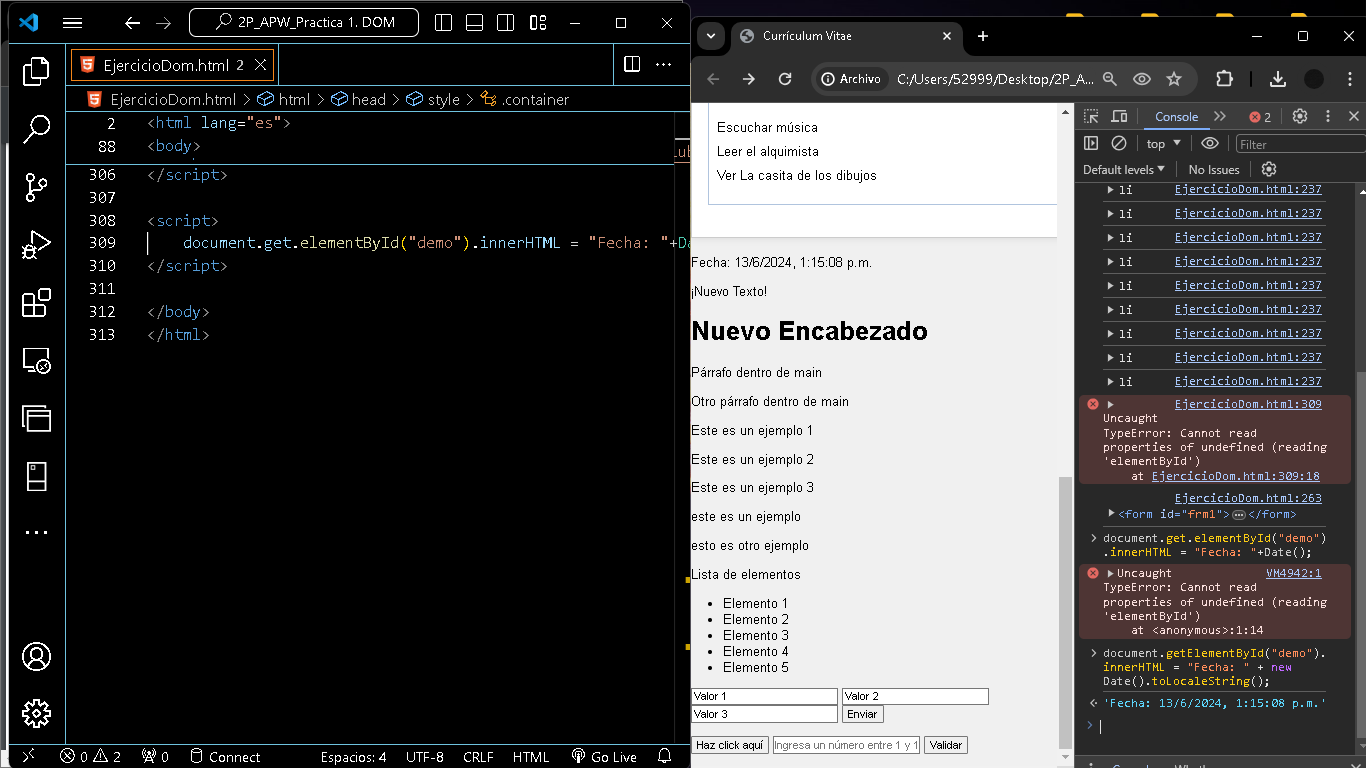
*Contenido dinámico*

*Agregar la hora actual a una etiqueta con id=”demo”.*

*document.get.elementById("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>