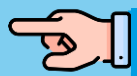
Funciones y Ciclos

DOM

{desafío} latam_

Codificar un programa utilizando funciones para la reutilización de código acorde al lenguaje Javascript.

- Unidad 1:
  Introducción al lenguaje JavaScript
- Unidad 2:
  Funciones y Ciclos
- Unidad 3:
  Arrays y Objetos
- Unidad 4:
  APIs



Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Ejecutar funciones creadas de JavaScript para interactuar con los elementos del DOM y manipular su comportamiento.*
- *Codificar un script que permita la selección y manipulación de elementos del DOM aplicando listener.*

{desafío} latam_

De acuerdo a lo aprendido anteriormente ¿Para qué sirve la palabra *continue* dentro de un for?

De acuerdo a lo aprendido ¿Qué es un ciclo anidado?

/* Document Object Model (DOM) */

El concepto DOM

- El DOM es un estándar adoptado por los navegadores web, creado por el W3C (World Wide Web Consortium), con el fin de ser un medio que permita a los programas y scripts, acceder y gestionar el contenido.

- El estándar DOM, contempla 3 tipos de documentos:

Core DOM	XML DOM	HTML DOM
Modelo estándar para todos los tipos de documentos acogidos.	Modelo estándar para los tipos de documentos XML (usado ampliamente en servicios web)	Modelo estándar para los documentos HTML (utilizado en sitios web)

Document Object Model (DOM)

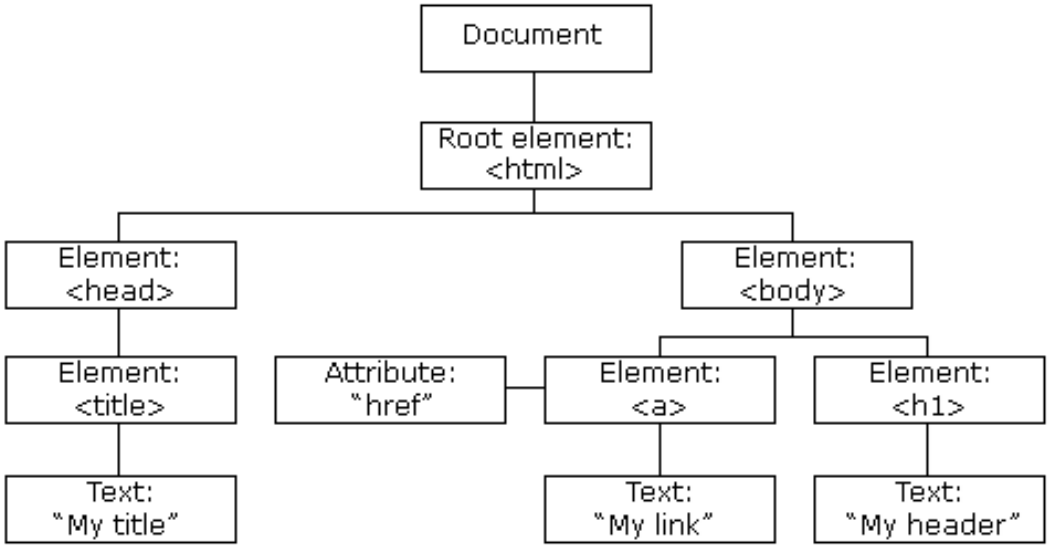
HTML DOM

- El estándar HTML DOM está definido principalmente por un modelo de objeto e interfaz de programación destinado a la gestión de elementos HTML, el cual se compone de los siguientes elementos:

    - **Objetos** que representan elementos HTML.
    - **Propiedades** que pueden ser valores manipulables de los elementos HTML.
    - **Métodos** para gestionar elementos mediante acciones programadas.
    - **Eventos** que reaccionan a alguna acción.

Document Object Model (DOM)

Estructura del DOM en HTML



/* Métodos de uso común para manipular el DOM */

Métodos de uso común para manipular el DOM

Maneras de acceder a elementos del DOM

- **Id:** El acceso mediante el atributo id es uno de los más utilizados y de fácil uso, puesto que es un identificador único de un elemento en el DOM.

```
<div id="contenedor">
    <p id="parrafo">Hola soy un párrafo.</p>
</div>
```

```
var parrafo = document.getElementById("parrafo");
```

- **Tag:** El acceso por tag se realiza especificando el tipo de elemento HTML. Por ejemplo, si deseamos buscar todos los elementos de tipo párrafo en el DOM (<p>), podríamos escribir el siguiente código:

```
<div id="contenedor">
  <p>Hola soy un párrafo.</p>
</div>
```

- Usando **getElementsByTagName**, seleccionamos el elemento p.

```
var parrafos = document.getElementsByTagName("p");
```

Métodos de uso común para manipular el DOM

Maneras de acceder a elementos del DOM

- **Clase:** También podemos buscar un elemento mediante su atributo class en el DOM. Por ejemplo, si deseamos buscar un elemento que contenga la clase botón, podríamos escribir el siguiente código:

```
<div id="contenedor">
  <button type="button" class="boton">Soy un
botón</button>
</div>
```

```
var boton =
document.getElementsByClassName("boton
");
```

- **Selectores CSS:** Es posible buscar elementos mediante selectores CSS (de la misma forma que se realiza en el código CSS). Por ejemplo, si deseamos buscar todos los elementos <p> con la clase párrafo, podríamos escribir el siguiente código:

```
<div id="contenedor">
    <p class="parrafo">Hola soy un párrafo.</p>
    <p class="parrafo">Hola soy otro párrafo.</p>
</div>
```

```
var parrafos =
document.querySelectorAll(".parrafo");
```

Demostración - “Acceso a elementos de DOM”

Ejercicio guiado

Acceso a elementos del DOM

- Se solicita seleccionar los siguiente elementos mediante el uso del DOM:
  1. El “id” con valor “conteneder”.
  2. El ul mediante la instrucción getElementsByTagName.
  3. La clase con el valor “menu” mediante la instrucción getElementsByClassName.
  4. La clase con el valor “item” mediante la instrucción querySelectorAll.

El html podrás encontrarlo en Ejercicios - DOM, en el LMS

{desafío} latam_

```
<!DOCTYPE html>
<html lang="en">
<head>
          <meta charset="UTF-8">
          <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
          <title>Document</title>
</head>
<body>
   <div id="contenedor">
     <div class="menu">
       <ul>
         <li class="item">Item 1</li>
         <li class="item">Item 2</li>
         <li class="item">Item 3</li>
       </ul>
     </div>
   </div>
   <script src="script.js"></script>
</body>
</html>
```

Ejercicio guiado

Acceso a elementos del DOM

- **Paso 1:** Crea una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js. En el index.html debes escribir el código proporcionado al inicio del ejercicio.

- **Paso 2:** En el archivo script.js, que ya se encuentra enlazado desde el index.html, agregaremos la instrucción necesaria para poder seleccionar el elemento indicado en el HTML con el id igual a contenedor (punto 1) mediante las instrucciones "document.getElementById()" y almacenamos este resultado en una variable para futuras operaciones y procesos que veremos más adelante, seguidamente mostramos el resultado de la variable en un console.log.

```
var idContenedor = document.getElementById("contenedor");
console.log(idContenedor);
```

{desafío} latam_

Ejercicio guiado

Acceso a elementos del DOM

- **Paso 3**: Al ejecutar el código anterior, el resultado sería:

```
<div id="contenedor">...</div>
```

- **Paso 4:** En el archivo script.js, agregaremos la instrucción necesaria para poder seleccionar el elemento indicado en el HTML con la etiqueta <ul> mediante las instrucciones “document.getElementsByTagName()” (punto 2) y almacenamos este resultado en una variable, seguidamente mostramos el resultado por consola

```
var elementoUl = document.getElementsByTagName("ul");
console.log(elementoUl);
```

Ejercicio guiado

Acceso a elementos del DOM

- **Paso 5:** Al ejecutar el código anterior, el resultado sería:

```
HTMLCollection { 0: ul, length: 1 }
```

- **Paso 6:** En el archivo script.js, agregaremos la instrucción necesaria para poder seleccionar el elemento indicado en el HTML con la clase “menu” mediante las instrucción “document.getElementsByClassName()” (punto 3) y almacenamos este resultado en una variable.

```
var menuClase = document.getElementsByClassName("menu");
console.log(menuClase);
```

Ejercicio guiado

Acceso a elementos del DOM

- **Paso 7:** Al ejecutar el código anterior, el resultado sería.

```
HTMLCollection { 0: div.menu, length: 1 }
```

- **Paso 8:** En el archivo script.js, agregaremos la instrucción necesaria para poder seleccionar los elemento indicado en el HTML con la clase “item” mediante la instrucción “document.querySelectorAll()” (punto 4) y almacenamos este resultado en una variable, seguidamente mostramos el resultado por consola:

```
var items = document.querySelectorAll(".item");
console.log(items);
```

```
NodeList(3) [ li.item, li.item, li.item ]
```

{desafío} latam_

/* Cambiando elementos del DOM */

Cambiando elementos del DOM

Para elementos donde el texto se define entre llaves

- Para obtener el contenido de un elemento, podemos hacer uso de la propiedad innerHTML a través del acceso de un objeto.

- Por ejemplo, vamos a cambiar el texto “Hola a todos” por “Soy un párrafo”, para esto hacemos lo siguiente

```
<p id="parrafo">Hola a todos</p>
```

```
var parrafo = document.getElementById("parrafo").innerHTML;
console.log(parrafo);
```

```
document.getElementById("parrafo").innerHTML = "Soy un
párrafo";
```

Cambiando elementos del DOM

Para elementos donde el texto está presente en el atributo

- Para elementos como botones, donde el texto visible está presente en un atributo, el procedimiento es distinto. Por ejemplo, en el siguiente botón:

```
<input type="button" value="Accion" id="btnAccion">
```

```
document.getElementById("btnAccion").value = "Boton";
```

- Así como manipulamos el texto del botón, podemos manipular los diferentes atributos de los elementos HTML que quisiéramos. Por ejemplo, podríamos cambiar el id del botón y cambiar el botón por una caja de texto de la siguiente forma:

```
document.getElementById("btnAccion").id = "nuevoId";
```

```
document.getElementById("btnAccion").type = "text";
```

Cambiando elementos del DOM

Para elementos donde el texto está presente en el atributo

- Ahora, disponemos de un método específico para configurar atributos de los elementos del DOM: **setAttribute,** el cual recibe como parámetro el nombre del atributo que se desea configurar junto con el valor que se le asociará.

- En el caso del botón con el atributo **id** y valor **btnAccion,** podríamos configurar su propiedad **style** para, por ejemplo, cambiar su color de fondo:

```
document.getElementById('btnAccion').setAttribute('style
', 'background-color: blue');
```

Demostración - “Cambiando elementos del DOM”

{desafío} latam_

Ejercicio guiado

Cambiando elementos del DOM

- Se requiere modificar el texto asociado al elemento <p> que posea un “id” con el nombre “textoSaludo” a “Hola, este párrafo fue modificado”. Igualmente, modificar los valores del elemento <input> con “id” igual a “entradaUno”, por el “value” igual a “Clic Aqui”, el id por “clicUno” y el tipo “type” por “button”. Además, agregar el atributo “style” con la propiedad “color: red; background-color: green”. Implementar las instrucciones necesarias para acceder al DOM y modificar los valores mencionados desde el siguiente documento HTML:

El html podrás encontrarlo en Ejercicios - DOM, en el LMS

```
<!DOCTYPE html>
<html lang="en">
<head>
            <meta charset="UTF-8">
            <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
            <title>Document</title>
</head>
<body>
   <div id="contenedor">
     <div class="menu">
       <p id="texto">Soy un párrafo en un
documento HTML</p>
       <p id="textoSaludo">Soy un párrafo en
un documento HTML</p>
       <input value="Correo Electrónico"
id="entradaUno" type="email"/>
       <input value="Entrada de datos"
id="entradaDos" type="email"/>
     </div>
   </div>
   <script src="script.js"></script>
</body>
</html>
```

Ejercicio guiado

Cambiando elementos del DOM

- **Paso 1:** Para seleccionar el elemento <p> con el id de nombre “textoSaludo”, debemos utilizar el document.getElementById(), para modificar el texto asociado dentro de las etiquetas <p>, se debe utilizar el “innerHTML”, quien permitirá modificar ese valor:

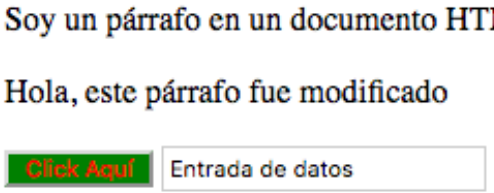
```
document.getElementById("textoSaludo").innerHTML = "Hola, este párrafo fue modificado";
```

Ejercicio guiado

Cambiando elementos del DOM

- **Paso 2:** Modificaremos el elemento input con el id denominado “entradaUno”, en este caso se deben cambiar distintos atributos de este elemento, como el caso del value, id y type. Se debe utilizar la instrucción document.getElementById(), más las propiedades de id, value y type para cambiar el valor existente por el nuevo valor:

```
document.getElementById("entradaUno").value = "Click Aquí";
document.getElementById("entradaUno").type = "button";
document.getElementById("entradaUno").setAttribute('style',
'color: red; background-color: green');
```



/* Eventos */

Eventos

Listener

- El estándar HTML DOM incluye eventos que nos permiten reaccionar mediante JavaScript, a eventos HTML ya definidos. Esto funciona como una suerte de observador, el cual está pendiente de la ejecución de eventos HTML.

- Estos observadores son conocidos como listeners. Para agregar un listener a un elemento del DOM, se debe hacer uso del método addEventListener, que puede recibir como parámetros:

EVENTO | **FUNCIÓN** | **PROPAGACIÓN DEL EVENTO**

```
element.addEventListener(evento, funcion, propagacion);
```

{desafío} latam_

Demostración - “Rescatar el valor del input”

Ejercicio guiado

Rescatar el valor del input

- Se posee un sitio web donde existe un buscador, con ayuda de listener vamos a rescatar el valor que escriba el usuario en un campo creado con un elemento input. Además, se contará con un botón donde el usuario podrá hacer clic y mostrar el texto escrito en el input dentro de una etiqueta <p> con la clase denominada “resultado”. Cuentas con el documento HTML

El html podrás encontrarlo en Ejercicios - DOM, en el LMS

```
<!DOCTYPE html>
<html lang="en">
<head>
          <meta charset="UTF-8">
          <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
          <title>Document</title>
</head>
<body>
   <div class="contenedor">
      <div class="buscador">
        <input class="input-a-buscar"
type="text" name="buscador">
        <button id="boton"
type="button">Buscar</button>
      </div>
      <p class="resultado"></p>
   </div>
   <script src="script.js"></script>
</body>
</html>
```

Ejercicio guiado

Rescatar el valor del input

- **Paso 1:** Lo primero que se necesita es acceder al elemento button para luego hacer uso del listener addEventListener. Entonces se debe guardar el elemento que trae la instrucción getElementById en una variable, para poder aplicar el addEventListener directamente a esa variable con el evento del tipo “click”.

```
var miBtn = document.getElementById("boton");
miBtn.addEventListener('click',function(){
 //cuerpo de la función anónima que se ejecuta al hacer click
});
```

Ejercicio guiado

Rescatar el valor del input

- **Paso 2:** Ahora una vez que el usuario haga click en el botón, debemos tomar el valor ingresado en el elemento input y agregarlo a nuestro párrafo con clase resultado al final de la estructura HTML, para esto hacemos uso de los selectores querySelector, el cual, permite buscar mediante los selectores de CSS el elemento donde debemos agregar y mostrar el texto ingresado por el usuario.

```
var texto = document.querySelector(".input-a-buscar");
document.querySelector(".resultado").innerHTML = "Estas
buscando: " + texto.value;
```

Ejercicio guiado

Rescatar el valor del input

- **Paso 3:** Ahora si unimos todo el código, quedaría el script de esta manera:

```
var miBtn = document.getElementById("boton");
miBtn.addEventListener('click',function(){
 var texto = document.querySelector(".input-a-buscar");
 document.querySelector(".resultado").innerHTML = "Estas buscando: " +
texto.value;
});
```

- **Paso 4:** Al ejecutar el código anterior en nuestro navegador, el resultado obtenido al ingresar perros en el campo, sería:

Estas buscando: perros

{desafío} latam_

/* Ejecutar funciones creadas de JavaScript al enviar un formulario */

Demostración - “Login”

Ejercicio guiado

Login

En el sitio web se encuentra un formulario que solicita a los usuarios ingresar el correo electrónico y contraseña.

Con ayuda de listener, vamos a rescatar los valores que escriba el usuario dentro del formulario. Además, se contará con un botón donde el usuario podrá hacer clic para procesar los datos del login y mostrar el texto escrito en el input dentro de una etiqueta <p> con la clase denominada resultado. Indicar en el mensaje, por ejemplo: “Bienvenido usuario@usuario.com”.

- ***El html podrás encontrarlo en Ejercicios - DOM, en el LMS***

Ejercicio guiado

Login

- **Paso 1:** Acceder al elemento formulario primeramente y luego le agregamos un listener, en este caso, usaremos el evento "submit". Los submit son eventos propios de los formularios y se utilizan para enviar la información que es introducida en estos, igualmente que en el ejemplo anterior, activaremos una función externa que llamaremos “login”.

```
let form = document.getElementById( "formulario" );
form.addEventListener( "submit", login);
```

Ejercicio guiado

Login

- **Paso 2:** Ahora en la función login, debemos leer los datos ingresados por el usuario y dar la bienvenida como usuario registrado:

```
function login(){
   var email = document.querySelector(".email");
   var password = document.querySelector(".password");
   document.querySelector(".resultado").innerHTML = `Bienvenido
${email.value}`;
};
```

Ejercicio guiado

Login

- **Paso 3:** Unir todos los trozos de código en uno solo:

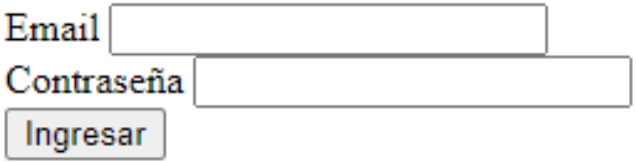
```
function login(){
   var email = document.querySelector(".email");
   var password = document.querySelector(".password");
   document.querySelector(".resultado").innerHTML = `Bienvenido
${email.value}`;
};

let form = document.getElementById( "formulario" );
form.addEventListener( "submit", login);
```

Ejercicio guiado

Login

- **Paso 4:** Al ejecutar el código anterior, agregando un correo, una contraseña y haciendo un click en el botón de enviar, el cambio ocurre muy rápidamente, debido a que la página se recarga nuevamente ella sola, limpiando los campos con los valores ingresados y desapareciendo el mensaje mostrado en la parte inferior del documento. Esto se debe al comportamiento por defecto de los formularios con el evento submit, que recargan la página donde se encuentran automáticamente.



Demostración - “Cancelar el comportamiento por defecto de un objeto”

{desafío} latam_

Ejercicio guiado

Cancelar el comportamiento por defecto de un objeto

Como vimos en el Ejercicio guiado: Login, existen cambios que se ejecutan por defecto y que hacen que la página tenga un comportamiento distinto al que esperamos. Por ende, veamos cómo cancelar este comportamiento:

- **Paso 1:** Partiendo del ejercicio anterior en el paso 3, realizamos unas pequeñas modificaciones para poder apreciar mejor el evento “submit” del formulario, agregando un alert después de recibir los datos para mostrar el correo electrónico:

```
function login(){
   var email = document.querySelector(".email");
   var password = document.querySelector(".password");
   alert("Ingreso exitoso: "+email.value);
};

let form = document.getElementById( "formulario" );
form.addEventListener( "submit", login);
```

Ejercicio guiado

Cancelar el comportamiento por defecto de un objeto

- Paso 2: Ejecuta nuevamente el código anterior en el navegador web ingresando como correo electrónico “usuario@usuario.com” y el resultado en pantalla sería:

Ejercicio guiado

Cancelar el comportamiento por defecto de un objeto

- Paso 3: Aplicamos ahora el **preventDefault()** al paso anterior dentro de la función “login”, agregando el parámetro “event” a la función para que pueda ser leído y utilizado en conjunto con el preventDefault, quedando.

```
function login(event){
   event.preventDefault();
   var email = document.querySelector(".email");
   var password = document.querySelector(".password");
   alert("Ingreso exitoso: "+email);
};

let form = document.getElementById( "formulario" );
form.addEventListener( "submit", login);
```

Demostración - “Validar un formulario utilizando JavaScript”

{desafío} latam_

Ejercicio guiado

Validar un formulario utilizando JavaScript

Un uso común que vemos día a día en la web son las validaciones que se aplican a los inputs de un formulario. Por ejemplo, cuando te quieres registrar en un sitio web y la contraseña debe cumplir una serie de reglas o cuando en un buscador no ingresas nada y aprietas el botón “buscar”, aparece una venta con un error por no ingresar datos.

Para ejemplificar este caso, realizaremos un ejercicio partiendo del código HTML del **Ejercicio guiado: Rescatar el valor del input.**

Ejercicio guiado

Validar un formulario utilizando JavaScript

- **Paso 1:** En el index.html, se agrega la estructura del documento mencionado.

- **Paso 2:** Revisar el código de JavaScript que nos permite recoger el valor ingresado en el formulario. Lo primero que se necesita es acceder al elemento button para luego hacer uso del listener addEventListener. Entonces se debe guardar el elemento que trae la instrucción getElementById en una variable, para poder aplicar el addEventListener directamente a esa variable con el evento del tipo “click”:

```
var miBtn = document.getElementById("boton");
miBtn.addEventListener('click',function(){});
```

Ejercicio guiado

Validar un formulario utilizando JavaScript

- **Paso 3:** Ahora una vez que el usuario haga click en el botón, debemos tomar el valor ingresado en el elemento input y agregarlo a nuestro párrafo con clase resultado al final de la estructura HTML, para esto hacemos uso del selector querySelector.

```
var texto = document.querySelector(".input-a-buscar");
document.querySelector(".resultado").innerHTML = "Estas buscando: " +
texto.value;
```

Ejercicio guiado

Validar un formulario utilizando JavaScript

- **Paso 4:** Ahora si unimos todo el código, quedaría el script de esta manera:

```
var miBtn = document.getElementById("boton");

miBtn.addEventListener('click',function(){
 var texto = document.querySelector(".input-a-buscar");
 document.querySelector(".resultado").innerHTML = "Estas
buscando: " + texto.value;
});
```

Ejercicio guiado

Validar un formulario utilizando JavaScript

- **Paso 5:** Este código cumple con el objetivo, pero si uno presiona el botón sin antes introducir el texto este sigue funcionando y muestra un resultado vacío. Para prevenir esta situación, validamos que para mostrar el mensaje debe haber ingresado cualquier texto en el input

```
var miBtn = document.getElementById("boton");

miBtn.addEventListener('click',function(){
  var texto = document.querySelector(".input-a-buscar");

  if(texto.value !== "") {
     document.querySelector(".resultado").innerHTML = "Estas
buscando: " + texto.value;
  };
});
```

Ejercicio guiado

Validar un formulario utilizando JavaScript

- **Paso 6:** La validación está correcta, pero ahora se siente como que el botón no hace nada, siempre que estamos validando algo es importante indicarle al usuario lo que está pasando. Para mostrar un mensaje de error, modificaremos nuestro HTML agregando un

```
<div class="contenedor">
  <div class="buscador">
    <input class="input-a-buscar" type="text" name="buscador">
     <button id="boton" type="button">Buscar</button>
  </div>
  <p class="error" style="color: red"></p>
  <p class="resultado"></p>
</div>
```

Ejercicio guiado

Validar un formulario utilizando JavaScript

- **Paso 7:** Ahora debemos agregar la lógica para que cuando el usuario no ingrese un texto se le indique y pueda así solventar el error agregando el texto antes de buscar.

```
if (texto.value !== "") {
  document.querySelector(".resultado").innerHTML = "Estas buscando: " +
texto.value;
} else {
  document.querySelector(".error").innerHTML = "Para poder buscar debes
ingresar una palabra";
};
```

Ejercicio guiado

Validar un formulario utilizando JavaScript

- **Paso 8:** Si ejecutas todo el código anterior, podemos observar en pantalla lo siguiente cuando hagas un click en el botón buscar sin ingresar ningún tipo de texto.

Ejercicio guiado

Validar un formulario utilizando JavaScript

- **Paso 9:** El código ahora muestra un mensaje cuando no se ingresa una palabra, pero si el usuario ahora escribe cualquier texto en el campo de búsqueda y hace un click sobre el botón buscar, el mensaje de error se mantiene. Para corregir este detalle, se puede enviar una cadena de texto vacía al innerHTML del elemento cuando la condición se cumpla, de lo contrario se envía el mensaje de error en sí, alternando el mensaje dependiendo del resultado de la condición de validación. Ahora nuestro buscador ha quedado validado

```
if(texto.value !== "") {
  document.querySelector(".resultado").innerHTML = "Estas buscando: " +
texto.value;
  document.querySelector(".error").innerHTML = "";
}else {
  document.querySelector(".resultado").innerHTML = "";
  document.querySelector(".error").innerHTML = "Para poder buscar debes
ingresar una palabra";
}
```

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante

Próxima sesión...

- *Reconocer las principales características de las expresiones regulares para identificar sus ventajas y contextos de uso.*
- *Aplicar expresiones regulares básicas sobre cadenas de texto para validar la estructura de la información recibida.*

{desafío}
latam_

Academia de
talentos digitales