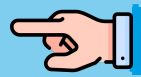
Funciones

{desafío} latam_

Activen las cámaras los que puedan y pasemos asistencia

Crear funciones para reutilizar códigos y separar la lógica de una página web del código HTML.

- Unidad 1: Introducción a JavaScript.
- Unidad 2: Condiciones.
- Unidad 3: Funciones.
- Unidad 4: Arreglos y objetos.
- Unidad 5: Métodos de arreglos.
- Unidad 6: APIs



Te encuentras aquí

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos

- Conceptualización
- Ejercitación

- Reflexión

Inicio

{desafío}
latam_

Objetivos

/* Crear y llamar funciones */

/* Crear funciones que reciban parámetros */

/* Llamar funciones desde el onclick de un elemento */

/* Crear funciones que devuelvan valores */

Activación de conceptos

Contesta la pregunta correctamente y gana un punto

Instrucciones:

- Se realizará una pregunta, el primero en escribir “YO” por el chat, dará su respuesta al resto de la clase.
- El docente validará la respuesta.
- En caso de que no sea correcta, dará la oportunidad a la segunda persona que dijo “Yo”.
- Cada estudiante podrá participar un máximo de 2 veces.
- Al final, el/la docente indicará el 1º, 2º y 3º lugar.

¿Qué se obtiene al comparar 2 === '2' ?

¿Qué se obtiene al comparar true || false?

¿Qué se obtiene al comparar true && false?

¿Qué se obtiene al comparar true && false == false? ¿En qué orden se resuelve?

{desafío} latam_

Activación de conceptos

Primer lugar:
_____

Segundo lugar:
_____

Tercer lugar:
_____

Desarrollo

{desafío}
latam_

/* Introducción a funciones */

{desafío}
latam_

Variables en JavaScript

var, let y const

- **var:** es una palabra reservada que sirve para declarar una variable cuyo alcance será global.
- **let:** similar a var, pero tiene un alcance de bloque.
- **const:** se define como constante, es una forma guardar en memoria datos cuyos valores no cambian.

Veamos ejemplos de cada una.

variables con var

- Como su nombre lo indica, una variable significa que su valor puede cambiar (*es variable*).
- En este caso estamos modificando con el valor de a dentro del bloque del if.

```
<script>
/* Aquí estamos en el scope global */
var a = 5
if (a = 10) {
    /* Aquí estamos en un scope de bloque */
    console.log(a)
}

</script>
```

{desafío} latam_

En la guía de ejercicios se aborda el concepto de scope o alcance de variables.

variables con let

- let también permite modificar los valores de una variable inicialmente declarada, como es el caso con la variable **a**.
- Nótese que dentro del bloque del if estamos declarando una variable b y la estamos llamando fuera del bloque.

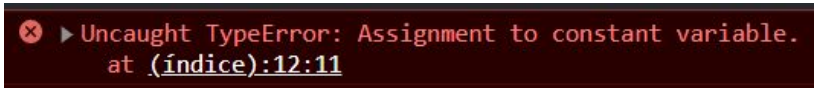
```
<script>
  /* Aquí estamos en el scope global */
  let a = 5
  if (a = 10) {
    /* Aquí estamos en un scope de bloque */
    console.log(a)
    let b = 10
  }
  // Salimos del bloque del if y llamamos a la variable b en el scope global
  console.log(b)
</script>
```

En la guía de ejercicios se aborda el concepto de scope o alcance de variables.

{desafío} latam_

constantes (const)

- A diferencia de const, su valor es constante, si ejecutamos el código de la imagen veremos el siguiente error.




```
<script>
  /* Aquí estamos en el scope global */
  const a = 5
  if (a = 7) {
    /* Aquí estamos en un scope de bloque */
    console.log(a)
  }
</script>
```

En la guía de ejercicios se aborda el concepto de scope o alcance de variables.

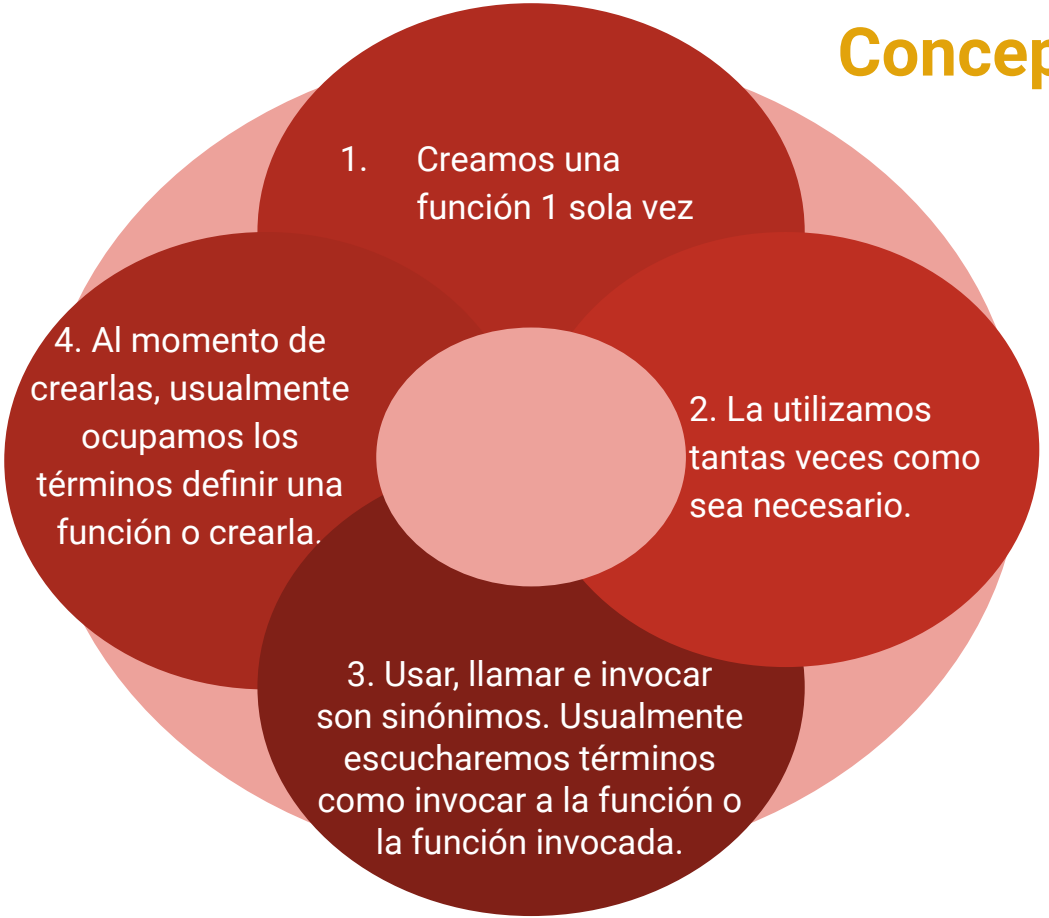
¿Qué es una función?

- Las funciones son conjuntos de instrucciones que podemos programar una vez y utilizarlas cada vez que necesitemos.

- Las funciones tenemos que crearlas y luego usarlas (llamarlas)

```
function aprenderHaciendo(){
 // ¡Manos al código!
}
```

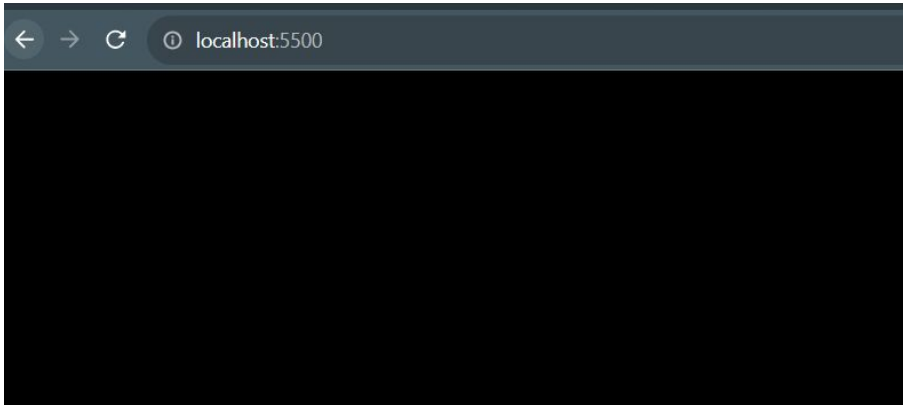
Conceptos básicos



Creando nuestra primera función

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Funciones</title>
</head>
<body>
  <script>
  function pintar_negro() {
    elemento = document.querySelector("body");
    elemento.style.backgroundColor = "black"
  }

  pintar_negro()
  </script>
</body>
</html>
```



{desafío} latam_

Entendamos el código

Definición de la función

```
function pintar_negro(){
  elemento = document.querySelector("body")
  elemento.style.backgroundColor = color
 }
 pintar_negro();
```

Entre llaves se encuentra el **cuerpo** de la función.

El cuerpo de la función contiene toda la lógica que dicha función ejecuta.

Invocación o llamar a la función declarada.

{desafío} latam_

Ejercicio

Dentro de la página web, crear la función pintar_rojo y la función pintar_amarillo y luego llamarlas.

Ejercicio
¡Manos al teclado!

{desafío} latam_

Objetivos

/* Crear y llamar funciones */ 

/* Crear funciones que reciban parámetros */

/* Llamar funciones desde el onclick de un elemento */

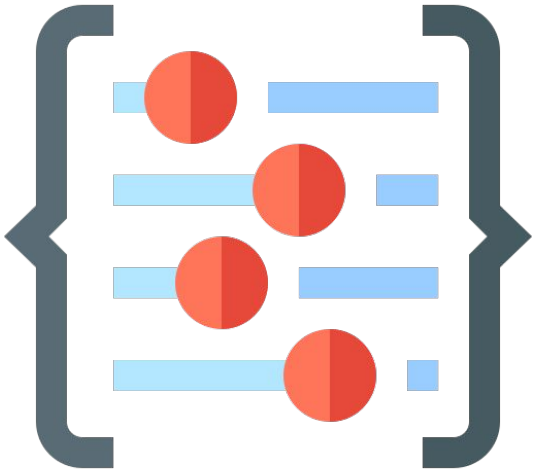
/* Crear funciones que devuelvan valores */

/* Funciones y parámetros */

¿Qué son los parámetros?

- Los parámetros son valores que puede recibir una función que nos permite hacerla mucho más flexible.

- Por ejemplo podemos modificar la función pintar que creamos previamente para que reciba un color y luego utilizar este color para pintar la página.

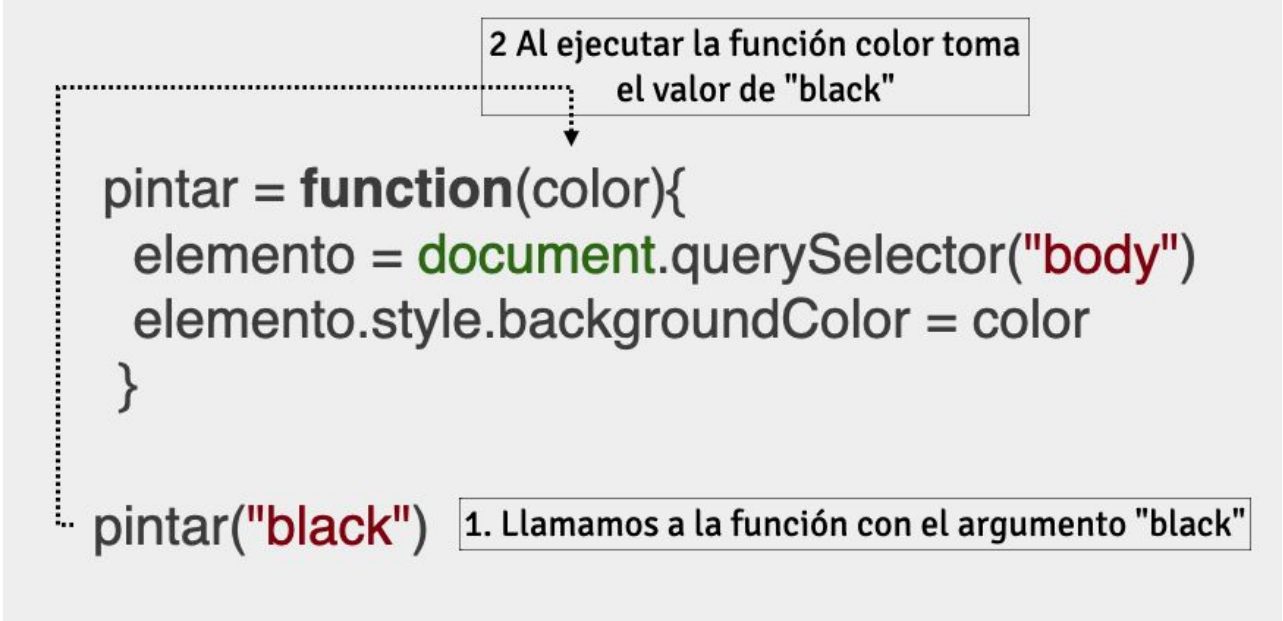


Ejemplo de uso

```
pintar = function(color){
  elemento = document.querySelector("body")
  elemento.style.backgroundColor = color
 }

 pintar("black");
 pintar("red");
 pintar("yellow");
```

Parámetros y argumentos



Funciones y parámetros

Funciones con múltiples parámetros

Las funciones pueden tener más de un parámetro, para lograrlo simplemente tenemos que separarlos con coma al momento de definir la función.

```
funcionMultiplesParametros(par1, par2, par3)
```

Al llamar a la función tenemos que pasar los valores en el mismo orden que están definidos los parámetros.

```
funcionMultiplesParametros("azul", "#id-2", 5)
```

¿Qué se muestra en la consola del inspector de elementos?

```
funcionMultiplesParametros(par1, par2,
par3) {
  console.log(par2)
}

funcionMultiplesParametros(3, 2, 1)
```

¡Manos a la obra!

{desafío} latam_

Funciones y parámetros

Parámetros con valores por defecto

```
pintar = function(color = "black"){
  elemento = document.querySelector("body")
  elemento.style.backgroundColor = color
 }

 pintar();
 pintar("red");
 pintar("yellow");
```

{desafío} latam_

¡Manos a la obra!

¿Qué se muestra en la consola del inspector de elementos?

```
funcionMultiplesParametros(par1, par2 = 2, par3 = 2 )
{
  console.log(par2 + par3)
}

funcionMultiplesParametros(3)
```

{desafío} latam_

Objetivos

/* Crear y llamar funciones */ 

/* Crear funciones que reciban parámetros */ 

/* Llamar funciones desde el onclick de un elemento */

/* Crear funciones que devuelvan valores */

/* Funciones y DOM */

{desafío}
latam_

Funciones

Dentro de onclick llamaremos a la función, la función debe estar definida dentro de script. El script puede estar en el mismo archivo o en otro.

```
<button onclick="pintar('black')"> Pintar negro</button>
<script>
  function pintar(color) {
    elemento = document.querySelector("body")
    elemento.style.backgroundColor = color
  }
</script>
```

Reutilizando funciones

Ahora que onclick llama a la función, se hace sencillo reutilizar la lógica.

```
<button onclick="pintar('black')"> Pintar negro</button>
<button onclick="pintar('red')"> Pintar rojo</button>
<button onclick="pintar('green')"> Pintar verde</button>
<script>
  function pintar(color) {
    elemento = document.querySelector("body");
    elemento.style.backgroundColor = color;
  }
</script>
```

Ejercicio guiado

En una página nueva pondremos 3 imágenes, al hacer click en una de ellas le agregaremos bordes.

Demostración

{desafío} latam_

Solución Ejercicio Guiado

```
<img id="img-1" src="http://placekitten.com/g/300/300" alt="" onclick="agregarBordes('img-1')">
<img id="img-2" src="http://placekitten.com/g/300/300" alt="" onclick="agregarBordes('img-2')">
<img id="img-3" src="http://placekitten.com/g/300/300" alt="" onclick="agregarBordes('img-3')">

<script>
  /* Creamos la función */
  function agregarBordes(elementId) {
    elemento = document.querySelector('#' + elementId);
    elemento.style.border="dashed 3px brown"
  }
</script>
```

Ejercicio

- Modifica la página anterior para que la función adicionalmente al elemento pueda recibir un color.
- Especificar distintos colores al llamar a la función en el onlick.

Ejercicio
¡Manos al teclado!

Objetivos

/* Crear y llamar funciones */ ✅

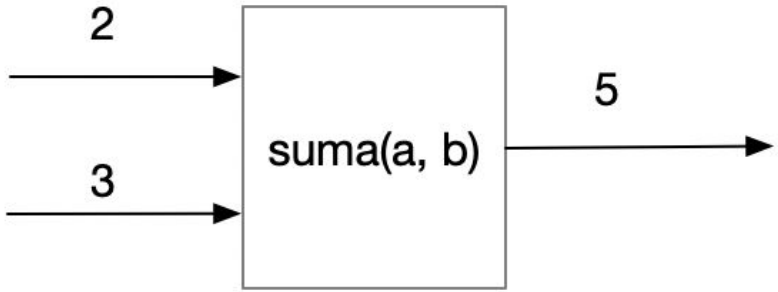
/* Crear funciones que reciban parámetros */ ✅

/* Llamar funciones desde el onclick de un elemento */ ✅

/* Crear funciones que devuelvan valores */

/* El retorno de las funciones */

Valor y funciones



Las funciones reciben valores y pueden además devolver valores

```
function suma(a, b){
  return a + b
}

alert(suma(2,3)) /* Opción 1 */
resultado = suma(2,3) /* Opción 2 */
console.log(resultado)
```

Devolver el valor de funciones

```
» function suma (a, b){
    alert (a + b)
}
suma(2, 3)
← undefined
```

```
function suma (a, b) {
    return a + b
}
suma(2, 2)
```

No todas las funciones devuelven un valor, sino solo aquellas que tienen return

Ejercicio

Nos piden crear una función llamada getBkgColor para obtener el color de fondo de un elemento web. O sea debe buscar un elemento a partir de un selector (#id o .class) entregado y debe devolver el color de fondo del elemento.

- ¿Cuál es/son los parámetros de entrada de la función?
- ¿Qué valor deberíamos devolver?
- Ahora creamos la función

Ejercicio
¡Manos al teclado!

Solución Ejercicio

```
<div id="el-1" class="element" style="background-color: red"></div>
<div id="el-2" class="element" style="background-color: yellow"></div>
<script>
  function getBkgColor(selector){
    ele = document.querySelector(selector)
    return ele.style.backgroundColor
  }
  getBkgColor("#el-1") /* Probamos nuestra función */
</script>
```

Ejercicio

Nos piden crear una función getValue que obtenga el valor de un input a partir de un selector y devuelva un texto que diga "mucho" si el valor es mayor que un parámetro, "exacto" si el valor es igual al parámetro o "muy poco" si el valor es menor al parámetro.

Antes de hacer el ejercicio discutir:

- ¿Cuál/es es/son los parámetros de entrada de la función?
- ¿Qué valor deberíamos devolver?

Ejercicio
¡Manos al teclado!

{desafío} latam_

Objetivos

/* Crear y llamar funciones */ ✅

/* Crear funciones que reciban parámetros */ ✅

/* Llamar funciones desde el onclick de un elemento */ ✅

/* Crear funciones que devuelvan valores */ ✅

Cierre

{desafío}
latam_

Reflexionemos

¿Existe algún concepto que no hayas comprendido? ¡Revisémoslo antes de terminar la clase!

{desafío} latam_

¿Qué sigue?

- Revisar la guía que trabajarán de forma autónoma.
- Revisar en conjunto el desafío.

{desafío} latam_

{desafío} latam_

Academia de talentos digitales