

***RECUERDA PONER A GRABAR LA  
CLASE***





***¿DUDAS DEL ON-BOARDING?***

**MIRALO AQUI**



Clase 04. JAVASCRIPT

# ***PROGRAMACIÓN CON FUNCIONES***



## ***OBJETIVOS DE LA CLASE***

Entender:

- ¿Qué es una función y cómo nos ayuda a escribir menos código?
- ¿Qué son los parámetros de entrada y salida de una función?
- ¿Qué es el Scope global y el Scope local?
- ¿Qué es una función anónima y una función flecha?

# ***GLOSARIO:***

## ***Clase 3***

**Ciclos en JS:** en programación, ciclo se refiere a un conjunto de indicaciones que se repiten bajo ciertas condiciones. Las estructuras de ciclos o cíclicas son las que debemos utilizar cuando necesitamos repetir ciertas operaciones de la misma manera durante N cantidad de veces.

**Sentencia break:** a veces, cuando escribimos una estructura for, necesitamos que bajo cierta condición el ciclo se interrumpa. Para eso se utiliza esta sentencia.

**Sentencia continue:** a veces, cuando escribimos una estructura for, necesitamos que bajo cierta condición, el ciclo saltee esa repetición y siga con la próxima. Para eso se utiliza esta sentencia.

**Estructura while:** permite crear bucles que se ejecutan ninguna o más veces, dependiendo de la condición indicada.

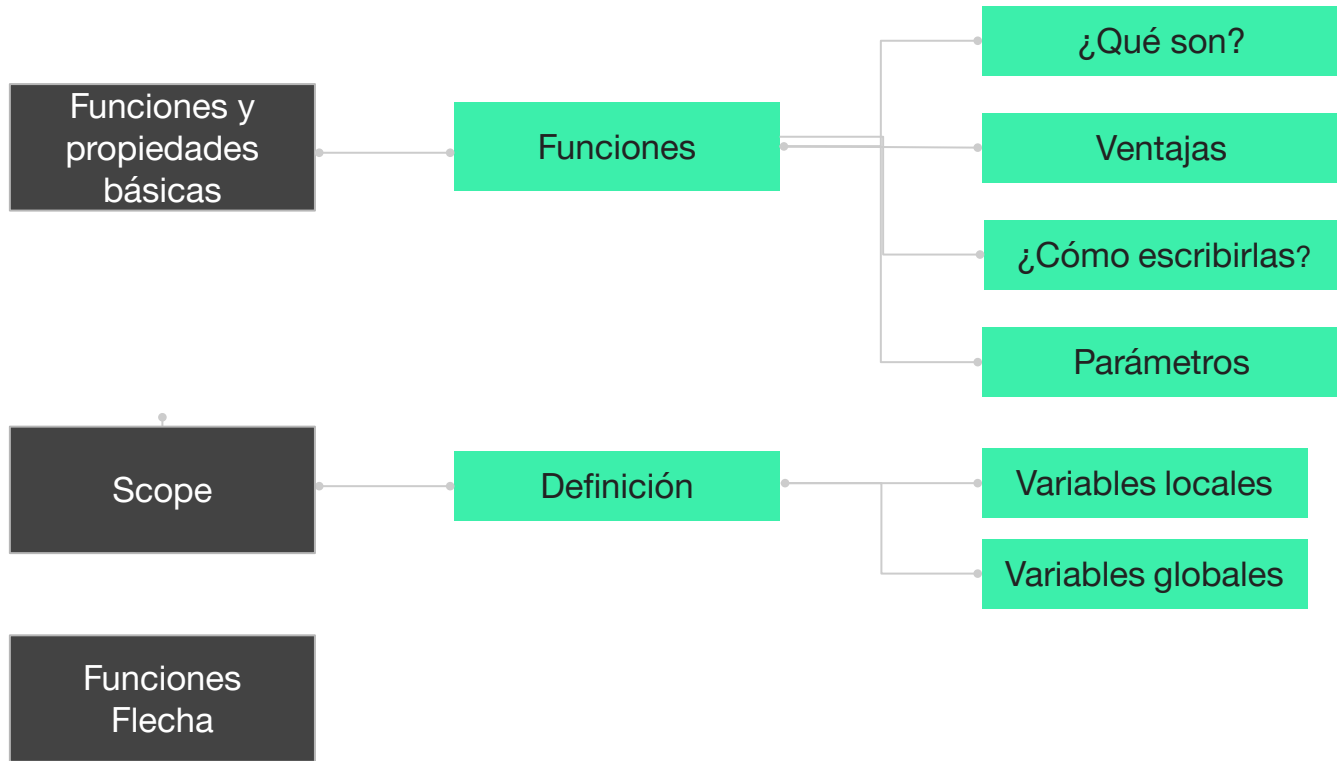
**Declarar función:** se dice declarar cuando uno define una función en el código.

.

# ***MAPA DE CONCEPTOS***

# MAPA DE CONCEPTOS CLASE 4

¡Para  
recordar!



# ***CRONOGRAMA DEL CURSO***

## Clase 3



### **Ciclos/Iteraciones**



EJEMPLOS EN VIVO



CREAR UN ALGORITMO  
UTILIZANDO UN CICLO

## Clase 4



### **Funciones**



EJEMPLO EN VIVO



CREAR UN ALGORITMO  
UTILIZANDO FUNCIÓN



SIMULADOR INTERACTIVO

## Clase 5



### **Objetos**



EJEMPLOS EN VIVO



CREAR UN OBJETO Y  
UTILIZARLO



INCORPORAR OBJETOS



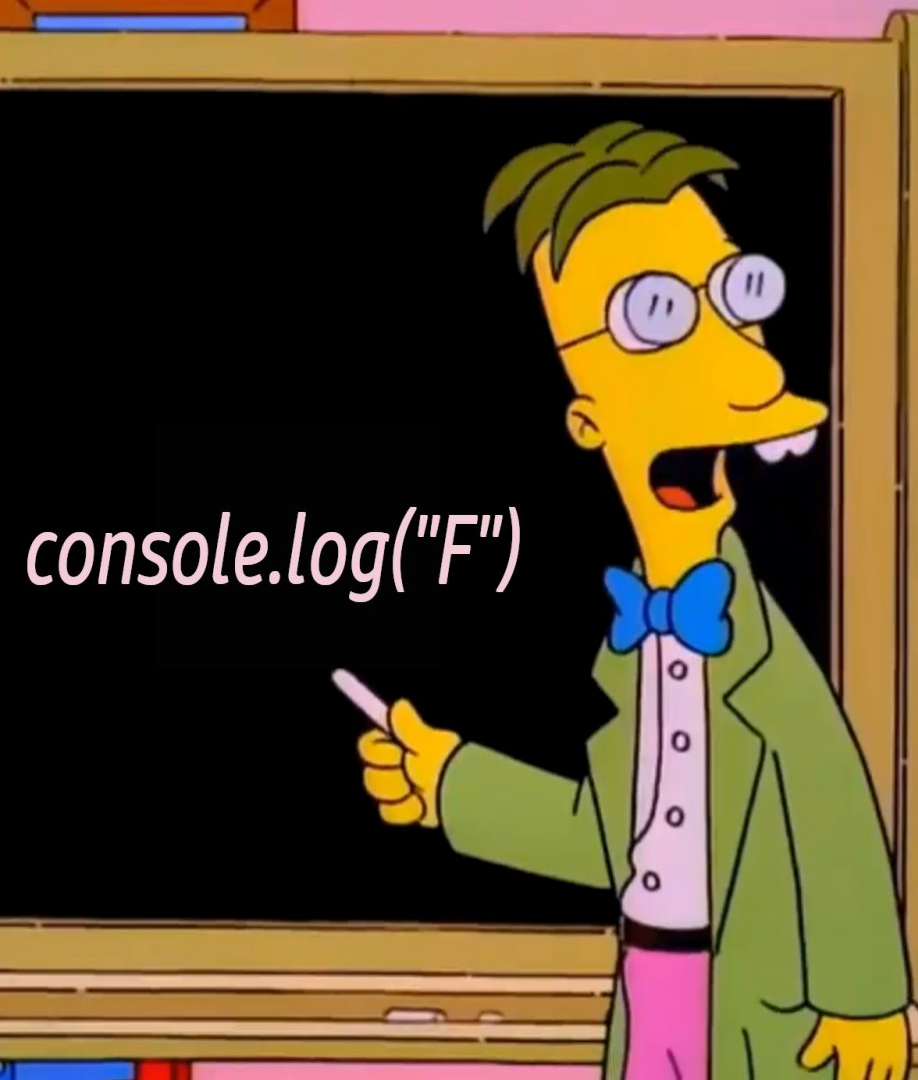


# ***HERRAMIENTAS DE LA CLASE***

*Les compartimos algunos recursos para acompañar la clase*

- Guión de clase N° 4 [aquí](#).
- Quizz de clase N° 4 [aquí](#)
- Booklet de Javascript [aquí](#)
- FAQs de Javascript [aquí](#)

# ***FUNCIONES Y PROPIEDADES BÁSICAS***



# ***FUNCIONES***

Cuando se desarrolla una aplicación o sitio web, es muy habitual utilizar una y otra vez las mismas instrucciones.

En programación, una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta, que luego se pueden reutilizar a lo largo de diferentes instancias del código.

# ¿Y QUÉ VENTAJAS ME DAN LAS FUNCIONES?

Las principales ventajas del uso de funciones son:

- Evitar instrucciones duplicadas ([Principio DRY](#))
- Solucionar un problema complejo usando tareas sencillas ([Principio KISS](#))
- Focalizarse en tareas prioritarias para el programa ([Principio YAGNI](#))
- Aporta ordenamiento y entendimiento al código
- Aporta facilidad y rapidez para hacer modificaciones

# ***¿CÓMO ESCRIBIRLAS?***

Todas las funciones se escriben igual. Deben tener un nombre en minúscula y sin espacios. Deben abrirse y cerrarse con llaves. El contenido de la función se escribe entre las llaves. El nombre de la función no se puede repetir en otra.

```
function saludar() {  
    console.log("¡Hola estudiantes!");  
}
```

# ¿Y AHORA?

Una vez que declaramos la función podemos usarla en cualquier otra parte del código las veces que queramos. Para ejecutar una función sólo hay que escribir su nombre y finalizar la sentencia con (). A esto se lo conoce como *llamada de la función*

```
saludar();
```

# ***EJEMPLO PRÁCTICO***

Si debemos solicitar un nombre al usuario mostrarlo en un alert, normalmente podríamos hacer esto:

```
var nombreIngresado = prompt("Ingresar nombre");  
alert("El nombre ingresado es " + nombreIngresado);
```

Si queremos repetir esto 2 veces más , podemos copiar y pegar el código.

```
var nombreIngresado = prompt("Ingresar nombre");  
alert("El nombre ingresado es " + nombreIngresado);  
var nombreIngresado = prompt("Ingresar nombre");  
alert("El nombre ingresado es " + nombreIngresado);
```

# ***USANDO UNA FUNCIÓN***

Podríamos entonces crear una función que se llame solicitarNombre() para solicitar al usuario la cantidad de veces que necesitamos

```
function solicitarNombre() {  
    let nombreIngresado = prompt("Ingresar nombre");  
    alert("El nombre ingresado es " + nombreIngresado);  
}
```

Para llamar a la función, la invocamos en otra parte del código:

```
solicitarNombre();  
solicitarNombre();  
solicitarNombre();
```



# ***FUNCIONES: PARÁMETROS***

# ***PARÁMETROS***

Una función simple, puede no necesitar ninguna dato para funcionar.

Pero cuando empezamos a codificar funciones más complejas, nos encontramos con la necesidad de recibir cierta información para funcionar.

Cuando enviamos a la función uno o más valores para que ser empleados en sus operaciones, estamos hablando de los **parámetros de la función**.

Los parámetros se envían a la función mediante variables y se colocan entre los paréntesis posteriores al nombre de la función.

# PARÁMETROS

```
function conParametros(parametro1, parametro2) {  
    console.log(parametro1 + " " + parametro2);  
}
```

Dentro de la función, el valor de la variable parametro1 tomará al primer valor que se le pase a la función, y el valor de la variable parametro2 tomará el segundo valor que se le pasa.

# EJEMPLO APLICADO: SUMAR Y MOSTRAR

```
//Declaración de variable para guardar el resultado de la suma
let resultado = 0;
//Función que suma dos números y asigna a resultado
function sumar(primerNumero, segundoNumero) {
    resultado = primerNumero + segundoNumero;
}
//Función que muestra resultado por consola
function mostrar(mensaje) {
    console.log(mensaje);
}
//Llamamos primero a sumar y luego a mostrar
sumar(6, 3);
mostrar(resultado);
```

# ***RESULTADO DE UNA FUNCIÓN***

El ejemplo anterior sumamos dos números a una variable declarada anteriormente. Pero las funciones pueden generar un valor de retorno usando la palabra `return`, obteniendo el valor cuando la función es llamada

```
function sumar(primerNumero, segundoNumero) {  
    return primerNumero + segundoNumero;  
}  
  
let resultado = sumar(5, 8);
```

```
function calculadora(primerNumero, segundoNumero, operacion) {  
  switch (operacion) {  
    case "+":  
      return primerNumero + segundoNumero;  
      break;  
    case "-":  
      return primerNumero - segundoNumero;  
      break;  
    case "*":  
      return primerNumero * segundoNumero;  
      break;  
    case "/":  
      return primerNumero / segundoNumero;  
      break;  
    default:  
      return 0;  
      break;  
  }  
}  
  
console.log(calculadora(10, 5, "*"));
```

# ***EJEMPLO APLICADO: CALCULADORA***

Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***

***CODER HOUSE***



***BREAK***

**¡5/10 MINUTOS Y VOLVEMOS!**



# ***VARIABLES LOCALES Y GLOBALES***

# ***DEFINICIÓN***

El ámbito de una variable (llamado "scope" en inglés), es la zona del programa en la que se define la variable, el contexto al que pertenece la misma dentro de un algoritmo.

JavaScript define dos ámbitos para las variables: global y local.

# VARIABLES LOCALES

Cuando definimos una variable dentro de una función o bloque es una **variable local**, la misma existirá sólo durante la ejecución de esa sección. Si queremos utilizarla por fuera, la variable no existirá para JS.

```
function sumar(primerNumero, segundoNumero) {  
    let resultado = primerNumero + segundoNumero;  
}  
  
//No se puede acceder a la variable resultado fuera del bloque  
console.log(resultado);
```



► Uncaught ReferenceError: resultado is not defined

# ***VARIABLES GLOBALES***

Si una variable se declara fuera de cualquier función o bloque, automáticamente se transforma en variable global, independientemente de si se define utilizando la palabra reservada var, o no.

```
let resultado = 0

function sumar(primerNumero, segundoNumero) {
    resultado = primerNumero + segundoNumero;
}

sumar(5, 6);

//Se puede acceder a la variable resultado porque es global
console.log(resultado);
```

# ***FUNCIONES ANÓNIMAS Y FUNCIONES FLECHA***

# ***FUNCIONES ANÓNIMAS***

Una función anónima es una función que se define sin nombre y se utiliza para ser pasadas como parámetros o asignada a variable. En el caso de asignarla a una variable, pueden llamar usando el identificador de la variable declarada

```
//Generalmente, las funciones anónimas se asignan a variables declaradas como constantes  
const suma = function (a, b) { return a + b };  
const resta = function (a, b) { return a - b };  
console.log(suma(15,20));  
console.log(resta(15,5));
```

# ***FUNCIONES FLECHA***

Identificamos a las funciones flechas como funciones anónimas de sintaxis simplificada. Están disponibles desde la versión ES6 de JavaScript, no usan la palabra **function** pero usamos `=>` (flecha) entre los parámetros y el bloque

```
const suma = (a, b) => { return a + b };  
//Si es una función de una sola línea con retorno podemos evitar escribir el cuerpo.  
const resta = (a, b) => a - b ;  
console.log(suma(15,20));  
console.log(resta(20,5));
```

# ***EJEMPLO APLICADO: CALCULAR PRECIO***

```
const suma = (a,b) => a + b;
const resta = (a,b) => a - b;
//Si una función es una sola línea con retorno y un parámetro puede evitar escribir los ()
const iva = x => x * 0.21;
let precioProducto = 500;
let precioDescuento = 50;
//Calculo el precioProducto + IVA - precioDescuento
let nuevoPrecio = resta(suma(precioProducto, iva(precioProducto)), precioDescuento);
console.log(nuevoPrecio);
```



Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***

***CODER HOUSE***



# ***SIMULADOR INTERACTIVO***

Empieza a armar la estructura inicial de tu proyecto integrador.

# ***SIMULADOR INTERACTIVO***

**Formato:** Página HTML y código fuente en JavaScript. Debe identificar el apellido del alumno/a en el nombre de archivo comprimido por “claseApellido”.

**Sugerencia:** Algunos criterios a tener en cuenta para seleccionar un proceso a simular por primera vez son:

“ELEGIR UN PROCESO BIEN CONOCIDO” : Si conozco una situación que implique adquirir cierta información y estoy bien familiarizado en “cómo se hace” es más fácil traducir la solución a un lenguaje de programación.

“ELEGIR UN PROCESO QUE ME RESULTE INTERESANTE” : Si me siento motivado sobre el tema, es más llevadero enfrentar los retos de desarrollo e interpretación: Antes de programar existe la etapa de relevamiento y análisis que me permite identificar cómo solucionar el proceso.

Desafío  
entregable



# SIMULADOR INTERACTIVO

**>> Consigna:** Con los conocimientos vistos hasta el momento, empezarás a armar la estructura inicial de tu proyecto integrador. A partir de los ejemplos mostrados la primera clase, deberás:

- Pensar el alcance de tu proyecto: ¿usarás un cotizador de seguros? ¿un simulador de simulador personalizado?
- Armar la estructura HTML del proyecto.
- Incorporar al menos un prompt para pedir un dato y luego mostrarlo mediante alert realizando alguna operación.
- Utilizar funciones para realizar esas operaciones.

Desafío  
entregable



## **>>Aspectos a incluir en el entregable:**

Archivo HTML y Archivo JS, referenciado en el HTML por etiqueta `<script src="js/miarchivo.js"></script>`, que incluya la definición de un algoritmo en JavaScript que emplee funciones para resolver el procesamiento principal del simulador

## **>>Ejemplo:**

Calcular costo total de productos y/o servicios seleccionados por el usuario.

Calcular pagos en cuotas sobre un monto determinado.

Calcular valor final de un producto seleccionado en función de impuestos y descuentos.

Calcular tiempo de espera promedio en relación a la cantidad de turnos registrados.

Calcular edad promedio de personas registradas.

Calcular nota final de alumnos ingresados.



# ***FUNCIONES RELACIONADAS***

Define tres funciones cuyas llamadas secuenciales resuelvan un proceso complejo.

# ***FUNCIONES RELACIONADAS***

**Formato:** código fuente en JavaScript, [Sublime Text](#) o [VisualStudio](#).

**Sugerencia:** para llevar adelante esta tarea, te sugerimos pensar un proceso complejo, descomponerlo al menos en tres partes, y realizar una función que se encargue de cada una de ellas.

Desafío  
Complementario



**>> Consigna: codifica al menos tres funciones cuyas instrucciones permitan resolver un problema particular, segmentado en tareas. La información a procesar debe ser ingresada por el usuario, y el resultado del procesamiento visualizado en una salida.**

**>>Aspectos a incluir en el entregable:**

Archivo HTML y archivo JavaScript referenciados, que incluyan la definición y llamada de al menos tres funciones.

**>>Ejemplos:**

- Ejemplo de procesamiento: cálculo de IVA

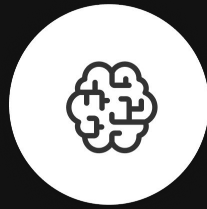
1) Ingresar precio de costo - 2) Sumar IVA - 3) Mostrar precio calculado.

- Ejemplo de procesamiento: determinar si un número es múltiplo

1) Ingresar números a verificar - 2) ¿Es múltiplo? - 3) Mostrar resultado.

***¿PREGUNTAS?***





## ***¡PARA PENSAR!***

*¿Te gustaría comprobar tus conocimientos de la clase?*

Te compartimos a través del chat de zoom  
el enlace a un breve quiz de tarea.

*Para el profesor:*

- Acceder a la carpeta "Quizzes" de la camada
  - Ingresar al formulario de la clase
  - Pulsar el botón "Invitar"
  - Copiar el enlace
- Compartir el enlace a los alumnos a través del chat





# ***RECURSOS:***

Material  
ampliado



- Scope |  
*Te lo explico con gatitos.*
- Documentación |  
*Documentación LET*  
*Documentación CONST*

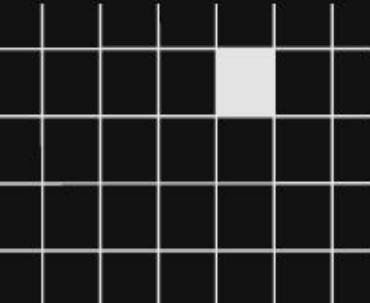
Disponible en [nuestro repositorio](#).

***CODER HOUSE***



# ***¡MUCHAS GRACIAS!***

Resumen de lo visto en clase hoy:

- Parámetros y resultado de una función.
  - Variables locales y globales.
  - Funciones anónimas y flecha
- 



***OPINA Y VALORA ESTA CLASE***

***#DEMOCRATIZANDO LA EDUCACIÓN***

***CODER HOUSE***

***CODER HOUSE***

***¡GRACIAS POR ESTUDIAR CON  
NOSOTROS!***

---