**Declaración de funciones**

Empecemos por lo básico:

Las funciones constan de 4 partes:

1. La *palabra reservada* **function.**
2. El nombre con el que vamos a llamar a la función:  
   function **nombreFuncion**
3. Los paréntesis donde van a ir los *parámetros* (si es que los necesita):  
   function nombreFuncion(**parametro1,parametro2,...,parametroX**)
4. Entre *{} llaves* el código que queremos que ejecute la función:  
   function nombreFuncion(parametro1,...,parametroX)**{ codigo }**

Si queremos que una función nos devuelva un valor, vamos a necesitar la palabra clave **return**. Después de eso va a ir todo lo que queramos devolver.

Con las funciones, podemos hacer infinidad de cosas, pero vamos a empezar por algo fácil. Acá escribimos la función que dobla el valor del número que le pasemos.

function doble(numero) {

return 2 \* numero;

}

Como podemos ver, arriba declaramos una función con la palabra reservada **function** con el nombre *doble*. Entre los paréntesis, pusimos el **parámetro** *numero* el cual va a estar reemplazado por el número que le demos cuando lo llamemos. Para terminar, con la palabra **return** le decimos que lo que queremos es que devuelva 2 veces el *número* que le dimos.

Es muy importante comprender que ahora si bien estamos **declarando** una función no la estamos **ejecutando**.

Para ejecutar la función y que haga lo que queremos, hay que llamarla por el nombre y entre paréntesis ingresar el número que queremos que utilice.

Prueba **declarando** a la función triple, la cual triplique el valor pasado por parámetro y nos devuelva su resultado.

function triple(numero) {

return 3 \* numero;

}

**Ejecutando funciones**

Hasta ahora estuvimos **declarando** funciones. Esto quiere decir que estuvimos escribiendo el código que van a ejecutar, pero nunca las ejecutamos, ¿o si?.

Veamos el siguiente ejemplo.

Acá **declaramos** la función doble.

function doble(numero) {

return 2 \* numero;

}

Con esto ahora contamos con la funcionalidad **doble**, una función que al **ejecutarla (llamarla)**va a devolvernos el doble del número que le pasemos.

Ahora podemos usarla de la siguiente forma:

doble(2);

Esto lo que va a hacer es efectivamente **ejecutar**el código que escribimos antes. Va a llamarse a la función *doble* y se le va a pasar por parámetro el valor 2. La función *doble* va a tomar este valor (2) y va a retornar ese valor \* 2. Lo interesante es que ahora también podemos usar esta función y probar con los valores que querramos.

Si quisiéramos **mostrar por consola (en Playground es la pantalla que está por debajo del botón de *Ejecutar respuesta*y que dice *Terminal*)** ese resultado tendríamos que imprimirlo usando una función propia de JavaScript llamada console.log de la siguiente forma:

console.log(doble(2));

Como ves, console.log

Como veras, ye te damos la función **cuadruple** que recibe un número y lo multiplica por 4. Ahora te pedimos que debajo de la declaración te encargue de **ejecutar**esa función, para ello necesitas darle un valor de reemplazo al parámetro, en este caso te pedimos que el valor sea 5. Además queremos que **imprimas por consola**el resultado de esa ejecución.

function cuadruple(numero) {

return 4\* numero;

}

console.log(cuadruple(5));

**Haciendo cuentas**

Ya vimos los operadores básicos matemáticos +, -, / y \*

Ahora podemos usar las funciones para hacer las cuentas mas fáciles.

function sumar(numero1, numero2) {

return numero1 + numero2;

}

Escribí la función multiplicar que tome dos parámetros, los multiplique y retorne su valor.

function multiplicar(numero1, numero2) {

return numero1 \* numero2;

}

**Funcionception**

Teniendo estas funciones pequeñas, podemos combinarlas para hacer cosas más complejas. Esto se debe a que podemos llamar y ejecutar una función dentro de otra.

Por ejemplo, si queremos sumar dos números y después multiplicarlos por 3, podríamos hacerlo de la siguiente forma:

function sumar(numero1, numero2) {

return numero1 + numero2;

}

function tripleDeLaSuma(numero3, numero4) {

let resultadoDeSuma = sumar(numero3, numero4);

return resultadoDeSuma \* 3;

}

Acá vemos que al ya tener la función **sumar()** definida podemos llamarla dentro de otra función (en este caso dentro de **tripleDeLaSuma()**), guardar el resultado de su ejecución en una nueva variable y hacernos el trabajo más fácil.

Ahora te pedimos que declares una función llamada  tripleDeLaSuma() que reciba dos parámetros. Luego tiene que sumar ambos y devolvernos el triple del valor del resultado de sumar ambos parámetros . Para ello ya contás (aunque no la veas declarada) con la función triple, que recibe **un parámetro** y retorna el valor del mismo multiplicado por tres.

function sumar (numero1, numero2){

return numero1 + numero2;

}

function tripleDeLaSuma(numero3, numero4){

resultadoDeSuma = sumar(numero3, numero4);

return triple (resultadoDeSuma);

}

# Fórmulas más complejas

Ya vimos que podemos hacer cálculos matemáticos simples y de alguna complejidad.

Ahora vayamos a algo más importante.

Queremos hacer funciones que nos permitan sacar el área y el perímetro de un círculo.

Hagamos una función llamada perimetro que nos diga el perímetro de un círculo cuando le damos el radio como parámetro.

Luego hagamos otra función llamada  area que nos dé el área de un círculo cuando recibe el radio como *parámetro*.

function perimetro(radio) {

return 3.14 \* radio \* 2

}

function area(radio) {

return 3.14 \* radio \* radio

}

**Operando Strings**

¿Y que podemos hacer que no sea matemática? ¡Varias cosas! Por ejemplo, podemos trabajar con **Strings:**

*Si te acordás, los****strings****son un tipo de dato. Representan cadenas de texto y lo que se escribe dentro de las comillas("") es literal.*

¿Qué podemos hacer entonces con los strings? Por ejemplo podemos medir qué tan largos son (en caracteres) con el método length:

"biblioteca".length // devuelve 10

"Buenos días".length // devuelve 11, porque el espacio es un caracter también

Javascript nos permite a través de ese método saber, como mencionamos antes, que cantidad de caracteres tiene una cadena de texto, lo único que tenemos que hacer en poner **.length**luego de la cadena de caracteres de la cual necesitemos el dato de su longitud.

También te recordamos que podemos sumar **strings**... Si, ¡sumarlos! Aunque el término correcto es **concatenarlos**, es decir, obtener uno nuevo, juntando dos o más **strings**:

"aa" + "bb" // devuelve "aabb"

"aa" + " " + "bb" // devuelve "aa bb"

Probemos qué tan claro quedó: Escribí una función **longitudNombreCompleto** que tome un nombre y un apellido como parametro y devuelva su longitud total, contando un espacio extra para separar a ambos.

Por ejemplo cuando la ejecutemos:

longitudNombreCompleto("Cosme", "Fulanito")

// devuelve 14

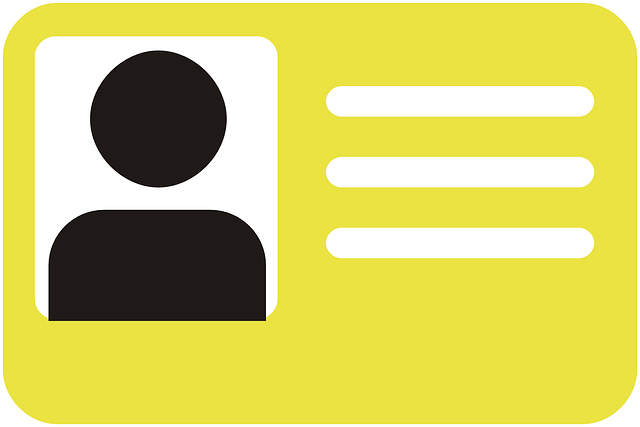
function longitudNombreCompleto(nombre, apellido) {

return nombre.length + " ".length + apellido.length

}

**Cartelitos**

Para una importante conferencia, los organizadores nos pidieron que escribamos cartelitos identificatorios que va a tener cada asistente.



Para eso, tenemos que juntar su nombre, su apellido, y su título (dr., dra., lic., etc) y armar un único string.

Escribí la función escribirCartelito, que tome como parámetros un título, un nombre y un apellido y devuelva un único string como resultado. Por ejemplo:

// Si ejecutamos la función con éstos datos...

escribirCartelito("Dra.", "Ana", "Pérez");

// ...la misma nos devuelve "Dra. Ana Pérez",

function escribirCartelito(titulo, nombre, apellido) {

return titulo + " " + nombre + " " + apellido;

}

**¿Qué tal si no?**

En el ejercicio anterior analizamos el siguiente código:

if ( diaSoleado == true ){

    console.log('¡Qué bien! Salgamos a la plaza');

}

Y comentamos que siempre y cuando el valor de **diaSoleado** sea exactamente igual a true, vamos a ejecutar el bloque de código que se encuentra dentro de las {}.

Muy bien, pero ¿qué pasa si queremos hacer algo en caso de que el día no esté soleado?. Es ahí en donde entra el else, la contraparte del if. Se vería así:

if ( diaSoleado == true ){

    console.log('¡Qué bien! Salgamos a la plaza');

} else {

    console.log('¡Ufff, que mal! Bueno, mejor nos quedamos haciendo código');

}

El condicional else nos permite ejecutar un bloque de código, en el caso que la condición del if no se cumpla, es decir, no sea verdadera. A diferencia del if, la estructura del else solo tiene 2 partes:

1. La palabra reservada else.
2. El bloque de código que se ejecuta en el caso que la condición del if no se cumpla.

Entonces, ahora que sabemos cómo funcionan el if  y  else, es importante que sepas que **¡podemos incorporar condicionales a nuestras funciones!** Veamos un ejemplo de una función que nos retorna true si un número es mayor a 5, o false en caso contrario:

function esMayor(unNumero) {

if ( unNumero > 5){

     return true;

} else {

     return false;

}

}

En el ejemplo anterior, la función se encargará de recibir un número por **parámetro** y utilizando if  y  else, retorna true  o  false  si el mismo es mayor a 5.

Si! Seguro que ya te diste cuenta! Cómo estamos trabajando con funciones, a diferencia del ejercicio anterior, ahora volvemos a utilizar la palabra reservada return .

Definí la función sePoneLaOlla, la cual recibe por parámetro el string del día de la semana. Esta función retorna "¡Pongamos la olla, hoy se comen pastas!" si recibe por parámetro el valor "Domingo", caso contrario retornará "Mejor lo dejamos para el domingo".

function sePoneLaOlla(dia) {

if (dia == "Domingo") {

return "¡Pongamos la olla, hoy se comen pastas!";

} else {

return "Mejor lo dejamos para el domingo"

}

}

# Ejercicio 1: Fin de semana

El objetivo de este ejercicio es imprimir en consola un texto que irá variando según el día que contenga la variable dia. Para esto, ya presentamos un código hecho con if/else que tendremos que modificar y convertirlo a un switch, manteniendo el mismo resultado.

function finDeSemana (dia) {

switch (dia) {

case 'viernes':

return console.log('buen finde');

break;

case 'lunes':

return console.log('buena semana');

break;

default:

return console.log('buen dia');

break;

}

}

# Ejercicio 2: Tengo clases

El objetivo de este ejercicio es imprimir en pantalla un texto que estará condicionado de la siguiente manera. Utilizando switch deberemos evaluar si la variable "día" es lunes, miércoles o viernes; y, en ese caso, debe imprimir el texto "tenés clases". Para cualquier otro caso debe imprimir "no tenés clases".

# Ejercicio 2: Tengo clases

El objetivo de este ejercicio es imprimir en pantalla un texto que estará condicionado de la siguiente manera. Utilizando switch deberemos evaluar si la variable "día" es lunes, miércoles o viernes; y, en ese caso, debe imprimir el texto "tenés clases". Para cualquier otro caso debe imprimir "no tenés clases".

# Ejercicio 2: Tengo clases

El objetivo de este ejercicio es imprimir en pantalla un texto que estará condicionado de la siguiente manera. Utilizando switch deberemos evaluar si la variable "día" es lunes, miércoles o viernes; y, en ese caso, debe imprimir el texto "tenés clases". Para cualquier otro caso debe imprimir "no tenés clases".

let dameCinco = () => [1,2,3,4,5];

let multiplicarPorDos = () => 123 \* 2 ;

let mostrarNombre = () => "Mi nombre es Hernán";

**Arrow function con parametro**

Ahora nuestro trabajo será refactorizar la función saludar() que recibe un parámetro, es decir, escribirla con la sintaxis necesaria para convertirla en una arrow function.

let saludar = nombre => 'Hola, ' + nombre + '!';

# Ahora probemos con más de uno

¡Vamos a pasar nuestra función declarada a una arrow!

Y sí, solo nos falta practicar la sintaxis de una arrow function que reciba más de un parámetro. Tomemos la función saludar(), que esta vez recibe un nombre y un apellido, y veamos de convertirla en una arrow function.

let saludar = (nombre, apellido) => 'Hola, ' + nombre + ' ' + apellido + '!';

**Pasar un callback**

Vamos a crear tres funciones y ver cómo empezar a trabajar con callbacks.

La primera función se llamará doble. Esta recibirá un número y retornará el doble.

La segunda función se llamará triple. Esta recibirá un número y retornará el triple.

La tercera función se llamará aplicarCallback y recibirá por parámetro un numero y una función. Esta retornará el valor de aplicar dicha función al valor que recibió.

Ejemplo:

aplicarCallBack(2, doble); // 4

aplicarCallBack(4, triple); // 12

let doble = numero => numero \* 2;

let triple = numero => numero \* 3;

let aplicarCallback = (numero, callback) => (callback(numero));

# Hagamos una calculadora

Vamos a ver cómo podemos crear una función que nos sirva como calculadora. Empecemos con algo sencillo...

#### **Creemos cuatro funciones que realicen las operaciones principales de una calculadora. Nuestro trabajo será definir la función suma, resta, multiplicacion y division. Cada una de ellas recibirá dos parámetros y deberá realizar la operación matemática según corresponda. Por ejemplo, la función suma deberá retornar la suma de ambos parámetros; la función resta, la resta de ambos parámetros, y así sucesivamente.**

#### **Ahora sí, con esas funciones ya definidas, creemos la función calculadora. Esta función recibirá dos parámetros numéricos y una función, que será nuestro callback.**

let suma = (numero1, numero2) => numero1 + numero2;

let resta = (numero1, numero2) => numero1 - numero2;

let multiplicacion = (numero1, numero2) => numero1 \* numero2;

let division = (numero1, numero2) => numero1 / numero2;

let calculadora = (numero1, numero2, operacion) => operacion(numero1, numero2);