Clase 4 - Programación con funciones

CLASE N°3 - Glosario

* 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. Se puede usar para comenzar o finalizar la clase, según sea más conveniente..

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 puede reutilizar a lo largo de diferentes instancias del código.

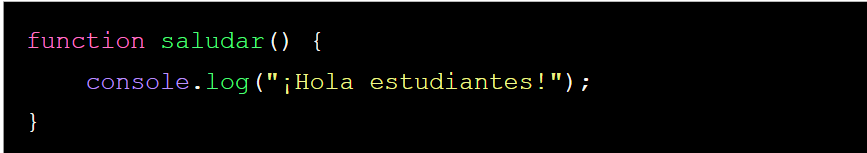
¿Y qué ventajas me dan las funciones?

Las principales ventajas del uso de funciones son:

* Evita instrucciones duplicadas (Principio DRY).
* Soluciona un problema complejo usando tareas sencillas (Principio KISS).
* Focaliza tareas prioritarias para el programa (Principio YAGNI).
* Aporta ordenamiento y entendimiento al código.
* Aporta facilidad y rapidez para hacer modificaciones.

Declaración

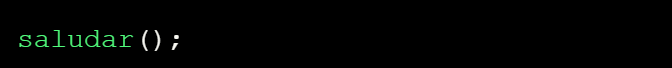
Se declara a través de la palabra reservada function. Deben tener un nombre en minúscula y sin espacios seguidos de los característicos paréntesis (). El contenido de la función se escribe entre las llaves. El nombre de la función no se puede repetir en otra.



Llamado

Una vez que declaramos la función, podemos usarla en cualquier otra parte del código todas las veces que queramos.

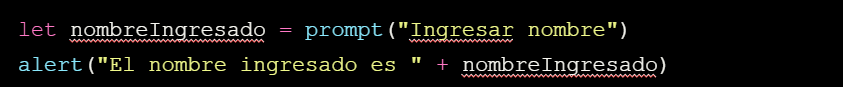
Para ejecutar una función sólo hay que escribir su nombre y finalizar la sentencia con (). A esto se lo conoce como llamado a la función.



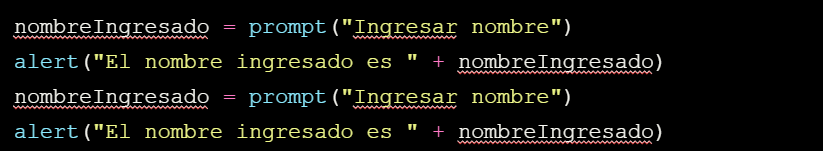
Donde escribamos el llamado, se interpretarán las instrucciones definidas en esa función.

Ejemplo práctico

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

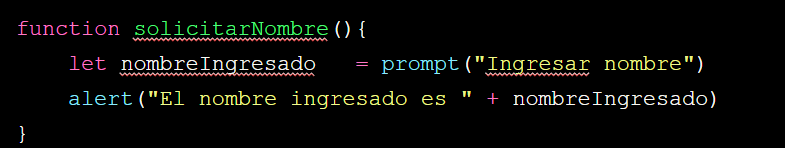


Si queremos repetir esto 2 veces más , podemos copiar y pegar el código.

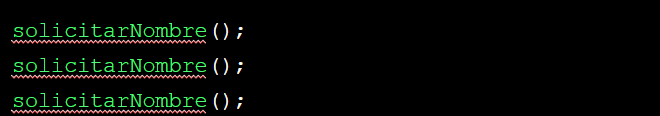


Usando una función

Podríamos entonces crear una función que se llame solicitarNombre() para que se le solicite al usuario la cantidad de veces que necesitemos



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



Funciones

Parámetros

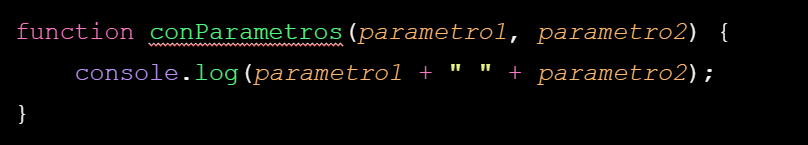
Una función simple, puede no necesitar ningún dato para funcionar.

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

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

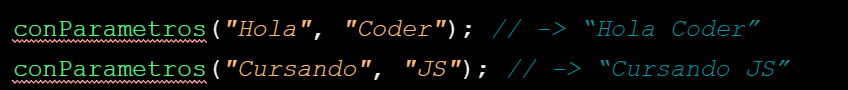
Los parámetros se envían a la función mediante variables y se colocan entre los paréntesis posteriores al nombre de la función.

Los parámetros son variables que se declaran dentro de la función, entre sus paréntesis. Los valores de éstos se definen luego en el llamado.



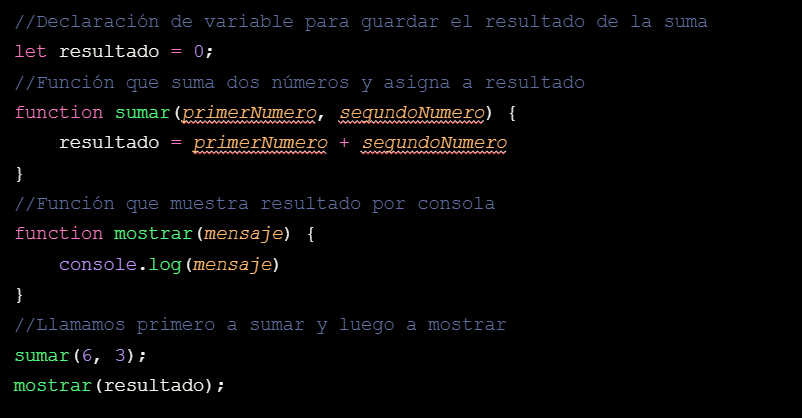
Así, podemos armar funciones dinámicas que, siguiendo la lógica que queramos, pueden generar distintos resultados al recibir diferentes valores.

El valor que toman estos parámetros se definen en el llamado. Cuando llamamos a la función, los valores que pasamos a la función entre paréntesis se asignan posicionalmente a los parámetros correspondientes, generando posibles resultados diferentes:



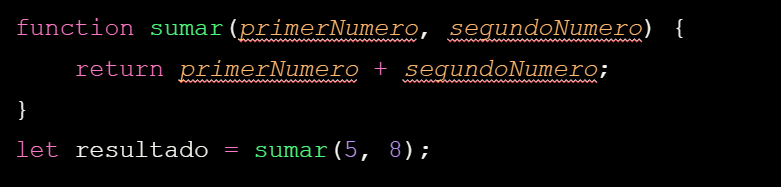
En este caso, el primer string que pasamos se asigna en parametro1, y el segundo string en parametro2; armando las salidas según la lógica definida.

Ejemplo aplicado: Sumar y mostrar



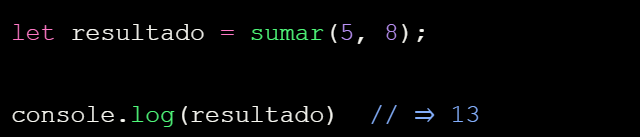
Resultado de una función

En 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.

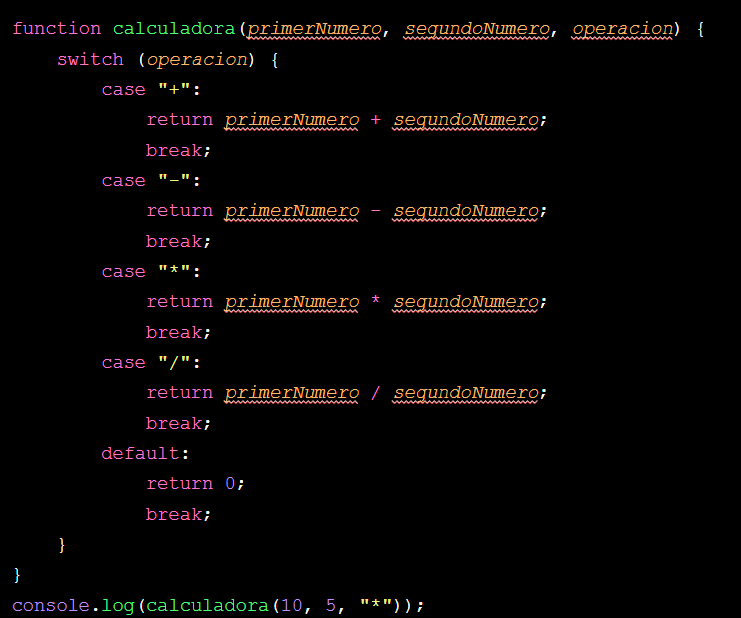


La función puede comportarse como una operación que genera valores (como en las operaciones matemáticas y lógicas previas).

En el espacio donde se llama a la función se genera un nuevo valor: este valor es el definido por el return de la misma.



Ejemplo aplicado: Calculadora



Scope

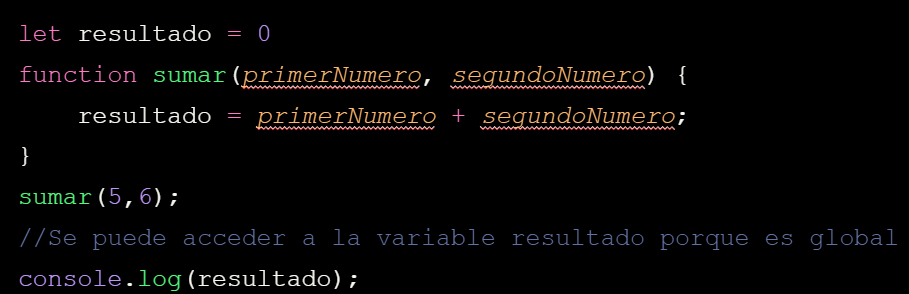
El scope o ámbito de una variable es la zona del programa en la cual se define, el contexto al que pertenece la misma dentro de un algoritmo, restringiendo su uso y alcance.

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

Variables globales y locales

Variables globales

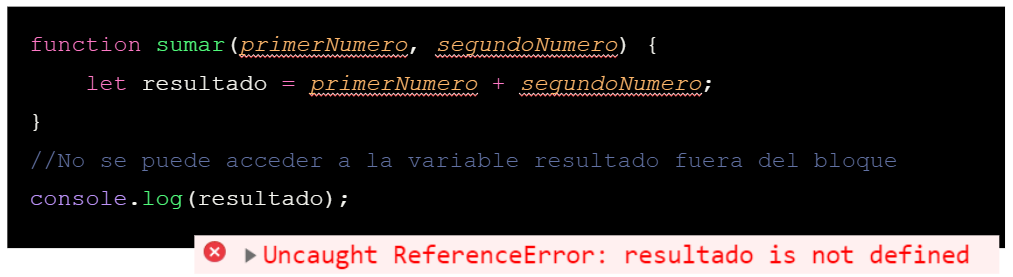
Si una variable se declara fuera de cualquier función o bloque, automáticamente se transforma en variable global.



Puede ser referenciada desde cualquier punto del programa.

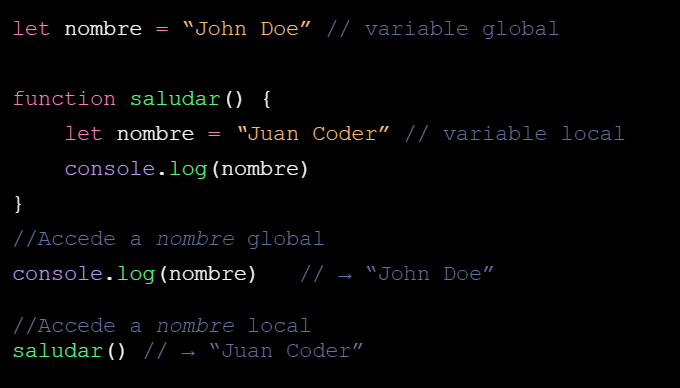
Variables locales

Cuando definimos una variable dentro de una función o bloque es una variable local, y será accesible sólo dentro de ese espacio. Si queremos utilizarla por fuera, la variable no existirá para JS.



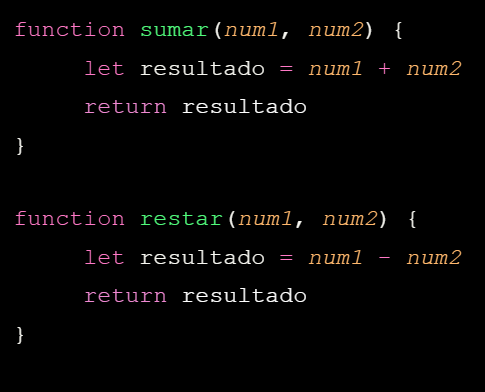
Variables locales y globales

Hay que entender que las variables globales y locales se identifican como diferentes entre sí, y pueden existir en el programa bajo el mismo nombre sin conflicto.



Scope

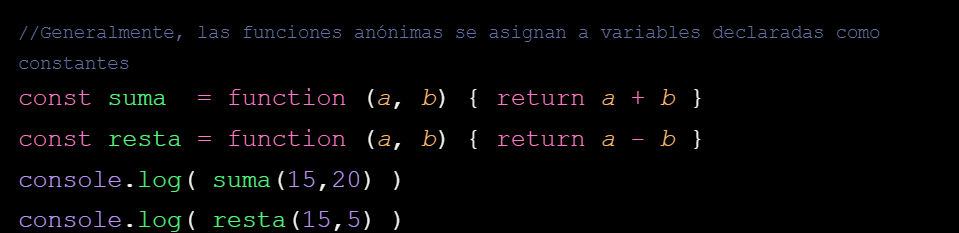
Entender que cada scope local es un espacio cerrado nos permite crear bloques de trabajo bien diferenciados e independientes, sin preocuparnos por repetir nombres de variables, sabiendo que se entienden como diferentes según donde las llamemos.



Funciones anónimas y funciones flecha

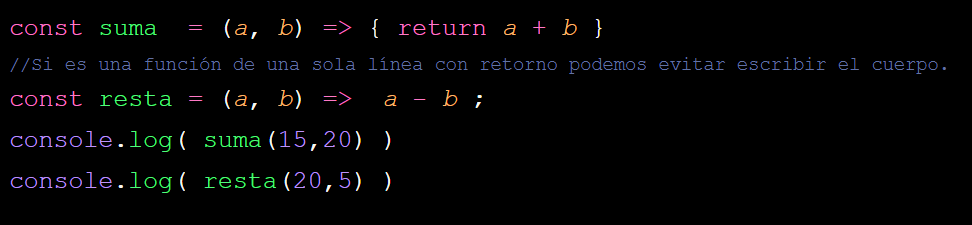
Funciones anónimas

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

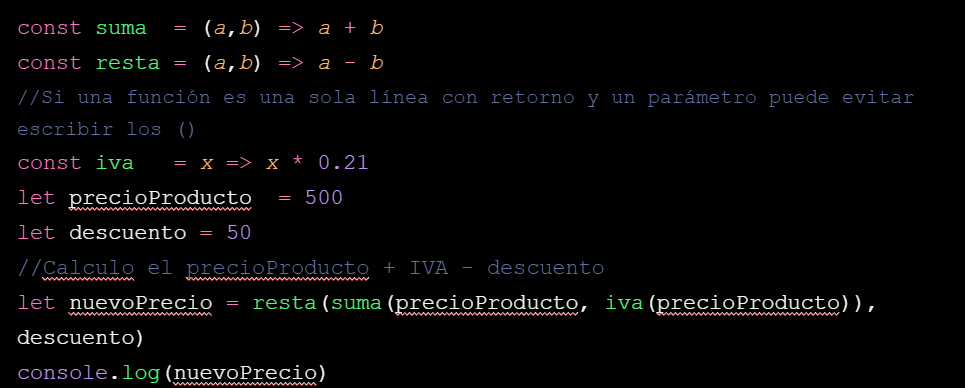


Funciones flecha

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 usa => (flecha) entre los parámetros y el bloque.



Ejemplo aplicado: Calcular precio



MATERIAL AMPLIADO

**Scope**

* [Te lo explico con gatitos](https://teloexplicocongatitos.com/_next/image?url=https%3A%2F%2Fdoomvault.nyc3.digitaloceanspaces.com%2Ftlecg%2Fbig%2Fprog09.jpg&w=1200&q=75)

**Documentación**

* [Documentación LET](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/let)
* [Documentación CONST](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/const)