



Introducción a Javascript

Clase 3



Funciones



Funciones

¿Qué es una función?

- Es una secuencia de órdenes o instrucciones almacenadas en un espacio reservado de la memoria que tiene un nombre de referencia
- Dentro de una función hay código Javascript, común y corriente

Memoria RAM		
nombre	apellido	Saludar
'Julian'	'Pacilio'	<pre>let nombre = prompt('ingrese su nombre'); console.log('Hola' + nombre)</pre>



Funciones

¿Para qué sirven las funciones?

- Las funciones permiten separar en módulos los pasos de un programa más grande.
- Las funciones permiten escribir una sola vez aquellos bloques de código que se usan en más de un lugar, para usarlas todas las veces que sea necesario.
- Las funciones, en principio, realizan una tarea específica dentro una aplicación más grande.
- Las funciones son la ÚNICA manera que existe para generar interacción entre HTML y Javascript.



Funciones / Creación



Funciones: creación

¿Cómo creamos una función?

- Existen tres formas de crear una función, veamos 2
- Como función
- Como variable
- Si se crea una función como variable, sólo se la podrá ejecutar luego de ser declarada... ¿Por qué? Porque no se puede utilizar una variable que aún no fue declarada.



Funciones / Ejecución



Funciones: ejecución

¿Qué es ejecutar una función?

- La ejecución de la función sucede cuando "se llama" para que procese el código que tiene almacenado.

¿Cómo se ejecuta una función?

- Se debe escribir el nombre de la función seguida de sus paréntesis.
 - `NombreDeLaFuncion();`
- Cuando se ejecuta una función no lleva la palabra function



Funciones / Ámbitos



Funciones: ámbito de las variables

Variables dentro y fuera de las funciones

- Dentro de una función se pueden declarar nuevas variables, que SOLAMENTE van a existir dentro de dicha función. Esto significa que si se llama a una de esas variables desde fuera de la función, se producirá un error de indefinición.
- Se presentan entonces dos posibilidades:
- Declarar una variable dentro de una función
 - Solamente existe dentro de la función
- Utilizar una variable declarada por fuera de la función
 - Existe por fuera y dentro de la función
- Esto se conoce como ámbito o scope de las variables



Tipos de variables según su ámbito



Funciones: ámbito de las variables

Variables globales

- Existen en todo el programa (son las que veníamos utilizando)
- Son todas las variables declaradas por fuera de las funciones, en el ámbito global
- Toda variable global puede ser accedida por una función

Variables privadas

- Existen solamente dentro de la función donde son creadas
- Son las variables declaradas en las funciones, en un determinado ámbito o “scope”
- No se puede acceder a ellas por fuera de la función
- Si dentro de una función se declara una variable con el mismo nombre que una global, la función prioriza a la variable privada, “ignorando” a la global



***"Toda variable privada nace, vive y muere
dentro de la función que la creó"***

Un programador anónimo



Funciones: parámetros / argumentos



Funciones: ámbito de las variables

¿Qué son los parámetros / argumentos?

- Cuando se crea una función, así como al ejecutarla, se les pone paréntesis luego del nombre.
- Esos paréntesis están reservados para pasarle datos adicionales a la función.
- Estos datos se llaman parámetros o argumentos, según el siguiente concepto:
 - Argumentos: los datos que se le pasan a la función cuando se la ejecuta.
 - Parámetros: los datos que recibe y guarda la función al ser ejecutada
- Son datos que vienen de afuera de la función y se pueden usar por dentro de la misma, sin depender de una variable global.
- El orden en que se pasan los argumentos al ejecutar una función debe coincidir con el orden de los parámetros que fueron definidos al crear la función.
- Todo parámetro de una función ejecutada, implícitamente está siendo declarado como una variable privada e inicializándose con el argumento que recibe.
- Por consiguiente, cuando una función termina de realizar todas las acciones, los valores recibidos por los parámetros dejan de existir.
- No hay límite en la cantidad de parámetros de una función.
- Los parámetros / argumentos se separan con coma dentro de los paréntesis.



Funciones: ámbito de las variables

Implementación de parámetros / argumentos


- Basándonos en el ejemplo anterior, sería más óptimo si creamos una función con un parámetro, que reciba como argumento la ruta de la imagen.
- Entonces:
 - Al ejecutar la función se le pasa el valor de la ruta de la imagen, como argumento.
 - Al crear la función, se crea un parámetro para recibir ese valor.
- De esta forma, evitamos el uso de la variable global y no tenemos que cambiar su valor antes de cada ejecución de la función.



Funciones: ámbito de las variables

Inicialización por defecto y verificación

- Siendo los parámetros variables, se las puede inicializar, de ser necesario, con un valor por defecto a la hora de crear la función.
- Esto puede ser útil para la verificación de parámetros, siendo ellos:
- Obligatorios: el parámetro necesita un argumento para el correcto funcionamiento.
 - Se lo puede inicializar con un valor de referencia, como ser null.
 - Mediante condicionales se puede verificar si se recibió un valor o si la variable sigue vacía.
- Opcionales: el parámetro no necesita un argumento para el correcto funcionamiento.
 - Se lo puede inicializar con un valor "útil" para el correcto funcionamiento.



***"Todo parámetro de una función es una variable
privada: nace, vive y muere dentro de ella.
Los argumentos son el sentido de su vida"***

Un programador anónimo



Funciones: retorno



Funciones: retorno

Tipos de funciones

- En programación existen dos tipos de funciones:
 - Void: funciones que se ejecutan, hacen todas sus acciones y no devuelven ningún valor.
 - Con retorno: funciones que se ejecutan, hacen todas sus acciones y devuelven un valor.

Retorno

- Dentro de la función existe la palabra reservada return, que cumple 2 tareas:
 - Corta toda acción pendiente de la función.
 - Devuelve un valor hacia afuera de la función. Ese valor será lo que se ponga a continuación de return.
 - Si luego del return hay una variable, devuelve su contenido, NO la variable en sí.
- El return
 - Puede ser único en la función, como la última instrucción a ejecutar.
 - Puede haber varios, distribuidos dentro de condicionales. Obviamente se estará ejecutando uno solo.



Funciones: retorno

Implementación de retorno

- El return siempre devuelve UN VALOR y no una variable
- Cuando se ejecuta una función con retorno, ese valor se puede utilizar al finalizar la ejecución.
- Por ejemplo, prompt, confirm, parseInt y parseFloat tienen retorno
- El valor retornado se puede usar directamente (por ejemplo el prompt dentro un parse) o guardarse en una variable para utilizarlo posteriormente (como generalmente hacemos).
- Una función sin retorno, al ser ejecutada, devolverá siempre undefined.
- Retomando el ejemplo que veníamos utilizando, podría devolver el string de la cadena:

```
const CrearImgOpcionalConRetorno = function(rutaArchivo = 'items/000000.jpg') {  
    return '';  
}
```



***"El retorno es el último mensaje que deja la función
antes de morir"***

Un programador anónimo



Fin de la clase

Este es el espacio para dudas y/o preguntas existenciales