PROGRAMACIÓN BÁSICA EN JAVASCRIPT

Módulo 2

OTEC PLATAFORMA 5 CHILE

MÓDULO 2

CLASE 4

Introducción

En esta clase comenzaremos a trabajar con Funciones, pero ¿qué es una función? Una Función es un bloque de código que nos permite realizar una tarea en particular. Para que la Función se ejecute, "algo" debe invocarla.

Es una herramienta muy útil porque estiliza el código y lo hace más escalable.

Y ahora se preguntará... ¿por qué son útiles? las Funciones nos permiten guardar partes de código en paquetes que podemos volver a usar.

Las Funciones, inicialmente, deben ser definidas y, luego, deben ser llamadas para que se puedan ejecutar.

Teniendo este panorama, seguiremos con las definiciones necesarias...

Declaración y llamado de una Función

Para ejecutar una Función, primero hay que declararla. En este proceso, se escribe el bloque de código que se guardará para, luego, ejecutar.

Sintaxis de una Función:

```
function nombreEnCamelCase() {
 // Los paréntesis deben quedar vacíos y, luego,
 // se abre una llave para alojar las instrucciones.
 console.log("¡Soy una Función!"); // Bloque de código que se ejecutará luego.
} // Se cierra la llave.
```

Ejecución de una Función:

```
nombreEnCamelCase(); // El paréntesis es el operador activador de la Función. Es
decir que sirven para activarla y, por eso, siempre deben quedar vacíos.
```

Imagine que quiere mostrar por consola el Feliz Cumpleaños. Podría hacerlo de esta manera:

```
console.log("Que los cumplas feliz.")
console.log("Que los cumplas feliz.")
console.log("Que los cumplas, Julieta.")
console.log("Que los cumplas feliz.")
```

Ahora, si quisieras volver a mostrarlo, tendrías que escribir ese código nuevamente. En cambio, con una Función, solo tendrías que escribir el código una vez y ejecutarlo cuantas veces lo necesites:

```
function cantarCumple (){
 console.log("Que los cumplas feliz.")
 console.log("Que los cumplas feliz.")
 console.log("Que los cumplas,
Julieta.")
 console.log("Que los cumplas feliz.")
}
//Ahora, ejecuto la Función cuantas
veces necesite:

cantarCumple()
cantarCumple()
cantarCumple()
```

Diferencia entre declarar y ejecutar una Función.

Cuando declaramos una Función escribimos el bloque de código que engloba una funcionalidad determinada. Es decir, son las instrucciones para correr un programa.

Ese bloque se guarda en la memoria para que lo usemos cuantas veces queramos.

Para ejecutar el código de la Función tenemos que llamarla junto al operador de invocación (). Por lo tanto, al declarar una Función estamos generando las instrucciones y, al ejecutarla, la estamos usando.

¿Qué es un *Parámetro*?

Un Parámetro es una Variable que creamos al momento de definir una Función y, cada vez que la ejecutamos, le pasamos un argumento(su valor) como input.

Generalmente los terminos "Parámetros y "Argumentos" se utilizan de manera indistinta para referirse a los datos que le pasamos a la función, sin embargo, su connotación es diferente. Si quiere indagar más sobre las diferencias entre estos términos, haga click acá.

Cuando le pasamos un Parámetro a una Función estamos haciendo un código mucho más escalable. Por ejemplo...

Siguiendo el ejemplo del Feliz Cumpleaños, podríamos variar el nombre del cumpleañero usando un Parámetro. Si bien es factible declarar varias Funciones, una por cada persona (como cantarCumpleClaudia, o cantarCumpleElon, etc.), optimizaríamos el código de esta manera:

```
function cantarCumpleA(nombre) {
 // Cuando declaramos la Función, entre los paréntesis, definimos el nombre del
Parámetro. En este caso: nombre.
 console.log("¡Que los cumplas feliz!");
 console.log("¡Que los cumplas feliz!");
 console.log("¡Que los cumplas " + nombre + "!");
 console.log("¡Que los cumplas feliz!");
}
// Usaremos los Parámetros en el código, aunque aún no sepamos exactamente cuál será su
valor.
cantarCumpleA("Claudia");
cantarCumpleA("Elon");
cantarCumpleA("Jeff");
// El valor del Parámetro lo obtendrán una vez que se ejecute.
```

¿Cómo Manejar Los Errores En Programación?

Los que trabajamos en desarrollo de software estamos acostumbrados a enfrentarnos con errores en el sistema, llamados frecuentemente bug. Los problemas en la lógica de nuestra aplicación, o en casos de uso que no se contemplaron originalmente, ocurren todo el tiempo sin importar la experiencia que tengas. Por eso, es muy importante no perder la calma ni caer en la frustración, ya que te alejaran de su resolución.

Hay muchas técnicas para depurar programas. Todas se basan en identificar el error y corregirlo. Uno de los más comunes es:

`Uncaught TypeError : cannot read property “x” of undefined Para solucionarlo, te recomendamos:`

1. Identificar el problema: En este caso, no se puede leer una propiedad x de un Objeto indefinido.

2. Saber por qué está ocurriendo: Para esto debemos controlar cuál es nuestra Variable en cada momento para saber cuándo quedó indefinida.

3. Corregir el problema en la línea de código donde encontramos el error.

Estos pasos los podés implementar para todos los errores que se muestren en tu consola.

Funciones con *múltiples Parámetros*

Podemos ejecutar las Funciones pasándoles todos los Parámetros que necesitemos. Pero, debemos separar los Parámetros con una coma.

Importante: **El orden de los Parámetros puede afectar el resultado.**

```
function saludarTres(nombre1, nombre2, nombre3) {
 console.log("Hola " + nombre1);
 console.log("Hola " + nombre2);
 console.log("Hola " + nombre3);
}
saludarTres("Ron", "Harry", "Hermione");
saludarTres("Hermione", "Harry", "Ron");
```

¿Qué sucede si dejamos un Parámetro sin definir? Pruebe este ejemplo en su consola:

```
saludarTres("Ron", "Harry")
```

La Keyword `return`

La keyword return se usa al declarar una Función para devolver un valor específico de ella. Esto sucede para guardar ese valor en una Variable o usarlo por fuera del bloque de definición de la Función.

Importante: Al usar esta palabra reservada se da por finalizada la ejecución de la Función, independientemente de la extensión del bloque de código.

```
function cuadrado(numero) {
 return numero * numero;
}
cuadrado(cuadrado(2)) > // En esta
Función estamos reutilizando el
valor que retorna la Función
cuadrado.
 16;
```

El *alcance de las variables* en JavaScript

El alcance (en inglés scope) de una Variable indica cuán disponible estará cuando se la invoque. Las Variables Globales son aquellas que están accesibles en todo el bloque de código de un programa. En cambio, las Variables Locales son accesibles solo en el ámbito de la Función donde fueron declaradas.

```
let nombre = "Juan"; // nombre es una Variable Global

function saludar() {
 let apellido = "Lopez"; // apellido es una Variable Local
 console.log("Hola, " + nombre + " " + apellido);
}
saludar();
```

Cuando ejecutemos la Función saludar, la consola nos devolverá como resultado los valores guardados tanto en la Variable nombre como apellido. Esto sucede porque la Variable nombre es Global y apellido es Local dentro de la Función que estamos ejecutando.

Sin embargo, en caso de que quisiéramos seguir avanzando en nuestro código para crear una Función despedir, que reutilice ambos valores, la consola nos devolverá un error porque la Variable apellido no está definida dentro de la Función despedir:

```
function despedir() {
 console.log("Chau, " + nombre + " " + apellido)
}

despedir()
```

Importante: Para poder reutilizar la Variable apellido deberíamos crearla por fuera de la Función saludar.

Una buena práctica en programación es definir todas las Variables Globales al principio de tu código.

MÓDULO 2

GRACIAS

Aquí finaliza la clase n°4 del módulo 2

OTEC PLATAFORMA 5 CHILE