

Funciones

Las funciones son piezas del código reutilizable que nosotros escribimos y definimos para poder utilizar más de una vez. Esto nos facilita la legibilidad de todo el conjunto del código, así como la agilidad en el desarrollo.

Las funciones deben de ser definidas como las variables, pero estas una vez definidas con cambian, siempre ejecutan la misma porción de código. Para definir una función debemos usar la palabra reservada **function** seguido del nombre que le queremos dar, parentesis abiertos y cerrar y unos corchetes para encapsular todo el código de nuestra función. Veamos un ejemplo sencillo.



```
function Saludar() {  
    console.log("Hola");  
}
```

Así es como definimos una función, pero si ejecutamos este programa.... no pasara nada por pantalla, no veremos el "Hola" de nuestro console log por pantalla. Esto es debido a que las funciones deben de ser llamadas, para que su código se ejecute, esto nos permite un mejor control del flujo.


Para llamar nuestra función simplemente debemos escribir el nombre de la función seguido de unos paréntesis abiertos y cerrados.



```
function Saludar( ){  
  
    console.log( "Hola" );  
  
}  
  
Saludar( );
```

Ahora si, estamos llamando a nuestra función. Por el momento son un poco inútiles estas funciones, dado que ejecutan un código muy sencillo.

Quizás nos gustaría hacer una suma de unos números, pero siempre tendremos los mismos números en nuestro programa, lo cual hacer que no importe mucho si el código está encapsulado en una función o no.



```
function Suma( ){  
  
  let nA = 2;  
  
  let nB = 4;  
  
  let res = nA + nB;  
  
}
```

Como vemos, nuestra variable res, siempre valdrá 6.... dado que no tenemos una forma efectiva de cambiar las variables. Ahí es donde entra la siguiente parte de nuestras variables

Parámetros

Los parámetros de un función están especificados después del paréntesis y nos permiten dar funcionalidad extra a nuestra función permitiendo una mayor reusabilidad del código. Veamos un ejemplo, de una suma en base a los valores que les pasamos como parámetros.



```
function Suma(nA, nB){  
  let res = nA + nB;  
}
```

```
Suma(3,6);
```

En este punto podemos observar que en nuestra función sólo recibimos variables(**nA,nB**) y no hay ningún número, lo que permite que el valor de res, sea en cada ocasión distinto, si los parámetros de entrada cambian. También nos puede interesar guardar el resultado de nuestra suma en una variable.



```
function Suma(nA, nB){  
  let res = nA + nB;  
}  
  
let miSuma = Suma(3,6);  
console.log(miSuma)
```

Si lanzamos este código nos va a dar un error **“Undefined”** esto lo resolvemos a continuación

Retorno de las funciones

De una forma acelerada el retorno de las funciones es el valor que devolverá la función, este podemos usar para pasarlo a otra función, en definitiva tratarlo como un dato específico: un string, un número, un booleano. Una ejemplificación rápida sería el ejemplo que tenemos justo arriba...

Como podemos hacer para que una variable sea el valor de una función determinada.



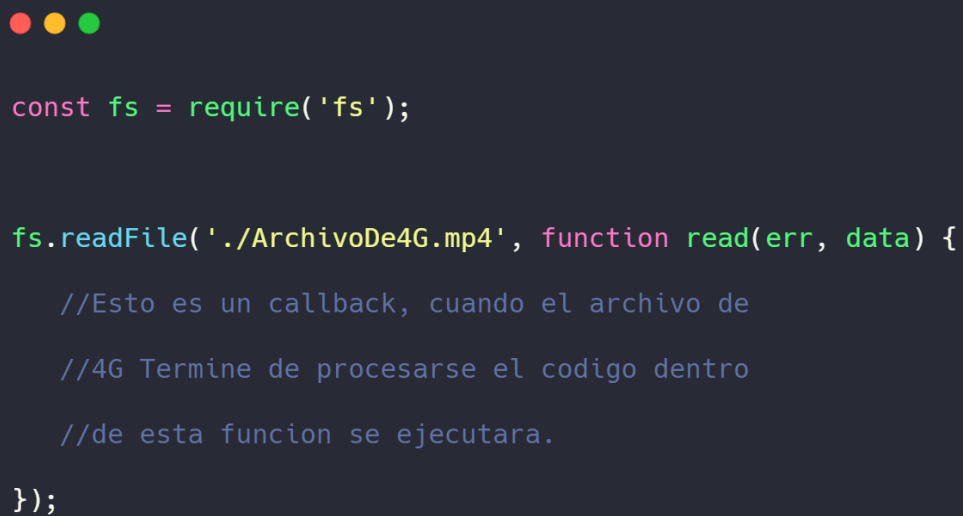
```
function Suma(nA, nB){  
  let res = nA + nB;  
  return res;  
}
```

```
let miSuma = Suma(3,6);  
console.log(miSuma)
```

De esta forma nuestra variable miSuma, en este caso no lanzará un valor **"Undefined"** , nos lanza un valor numérico **9**.

Callbacks

Los callbacks son funciones dentro de funciones. Un poco extraño, pero existe, imaginemos una función que tiene que procesar 4GB de archivos.... Esta función tarda un tiempo en procesarse y queremos ejecutar una porción de código cuando esos 4GBs terminen de ejecutarse, no antes, ni mucho después. Aquí es donde entran los callback, veamos cómo usar uno. Ya descubriremos como escribirlos en otra ocasión.



```
const fs = require('fs');

fs.readFile('./ArchivoDe4G.mp4', function read(err, data) {
  //Esto es un callback, cuando el archivo de
  //4G Termine de procesarse el codigo dentro
  //de esta funcion se ejecutara.
});
```

De este modo cuando nuestro archivo se procese podemos ejecutar cualquier pieza de código que escribamos