JS | Funciones

Después de esta lección podrás:

1. Entender qué es una función
2. Aprender a separar las necesidades en funciones
3. Comprender cómo funcionan las funciones

Qué es una función?

Una función se define para a resolver una **tarea concreta.**

```
console.log('mi primera funcion');
```

En el ejemplo anterior, vemos una función para **imprimir** un mensaje. Sencillo, ¿no? 😲

Pensad cuando tratamos de resolver un ejercicio de programación, escribiendo código en varias líneas... ¿y si existiera alguna manera de **dividir** en "trocitos" todo ese código? (para

que no esté todo en un bloque inmenso). Pues en esta sección aprenderemos a **separar** nuestro código por funcionalidades.

Volviendo al ejemplo inicial, tenemos la función `console.log` para imprimir por pantalla, y la podemos usar todas las veces que queramos. Imaginad que ahora quisiéramos una función para sumar dos números:

```
// Vamos a crear nuestra primera función!
function sum(num1, num2) {
  return num1 + num2;
}
// Ya tenemos declarada nuestra función para sumar dos números

var result = sum(2, 2); // Aqui estamos invocando a la función (ejecutándola)
console.log(result); // Y aquí vamos a imprimir el resultado
```

Analicemos este código:

- `function` es la palabra mágica (sentencia reservada) para crear **funciones**
- `sum` es el **nombre** que le doy a la función (hay que procurar darle un nombre descriptivo)
- `(num1, num2)` a esto se le llama **parámetros**, y es lo que **entra** en nuestra función
- `return` es la palabra mágica (reservada) para **devolver** el resultado, la **salida** de la función
- `{ }` entre los corchetes irá el cuerpo de la función, donde operamos

Tras la declaración de la función, vemos la línea en la que hacemos uso de ella `sum(2, 2)`, es fácil, basta con escribir su **nombre** y ponerle los **paréntesis** con los valores de **entrada**.

Ahora copia y pega el bloque de código en tu fichero `funciones.js` y ejecútalo con **Node**. El resultado es 4 (hemos conseguido sumar 2 + 2 🤪).

Conceptos importantes

Es importante que entendamos los siguientes conceptos:

- Una función me ayuda a **encapsular** una operación concreta
- Una función se crea con la sentencia `function`
- Una función siempre **debería** de llevar un `return` (aunque no es obligatorio)
- Una función se **ejecuta** con los **paréntesis** después de su nombre

Pero ¿qué ocurre si quiero crear una función tan simple que no necesite parámetros de entrada? Veamos un ejemplo:

```
// Declaremos una función de lo más simple
function myNameIs() {
  // Vamos a devolver una cadena de texto (string) con nuestro nombre
  return "I am Iron Man";
}

var result = myNameIs(); // Aqui invocamos la función
console.log(result); // Y aquí vamos a imprimir el resultado
```

Esta función NO ha necesitado parámetros en los paréntesis, y funciona igual que las demás:

Asignación de funciones

Existe otra forma de crear y nombrar funciones, que es **asignando** funciones a **variables**. Aquí tenemos un ejemplo:

```
var sayHello = function() {
  return "Hello!";
};

sayHello(); // Devuelve "Hello!"
```

Ejercicio guiado

Veamos ahora un pequeño ejercicio, en él debemos transformar un solo bloque de código con varios cálculos, en varias funciones.

Hemos ido con nuestra pareja a cenar a nuestro restaurante favorito, y hemos pedido una ensalada para compartir (9€) y dos hamburguesas (11€ cada una). Vamos a calcular el coste total la cena 🤑

```
var numSalads = 1; // Creamos una variable con el numero de ensaladas
var saladPrice = 9; // Creamos una variable con el precio de la ensalada
var numBurgers = 2; // Creamos una variable con el numero de burgers
var burgerPrice = 11; // Creamos una variable con el precio de la burger
var tax = 0.21; // Creamos una variable con IVA (21%) [Hacienda somos todos]
var total = (numSalads * saladPrice) + (numBurgers * burgerPrice);
var billAmount = total * tax + total;
console.log(billAmount) // Devuelve 37.51
```

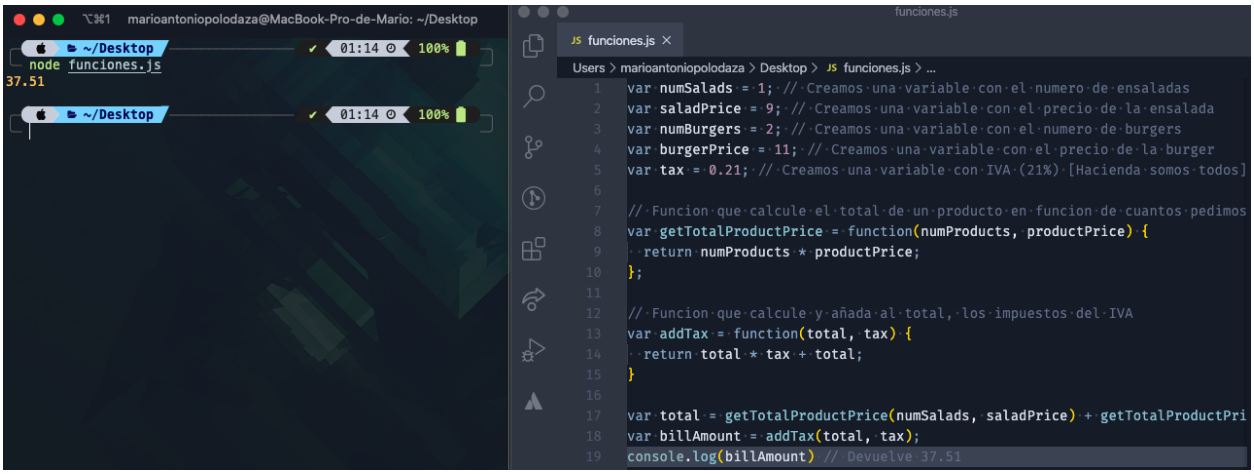
Y ahora, veremos el mismo ejercicio, pero con funciones. Podremos encapsular dichos cálculos, creando un código más limpio, que a futuro nos hará más fácil la vida cuando el código empiece a "crecer":

```
var numSalads = 1; // Creamos una variable con el numero de ensaladas
var saladPrice = 9; // Creamos una variable con el precio de la ensalada
var numBurgers = 2; // Creamos una variable con el numero de burgers
var burgerPrice = 11; // Creamos una variable con el precio de la burger
```

```
var tax = 0.21; // Creamos una variable con IVA (21%) [Hacienda somos todos]

// Funcion que calcule el total de un producto en funcion de cuantos pedimos
var getTotalProductPrice = function(numProducts, productPrice) {
  return numProducts * productPrice;
};

// Funcion que calcule y añada al total, los impuestos del IVA
var addTax = function(total, tax) {
  return total * tax + total;
}

var total = getTotalProductPrice(numSalads, saladPrice) + getTotalProductPrice(numBurgers, burgerPrice);
var billAmount = addTax(total, tax);
console.log(billAmount) // Devuelve 37.51
```

Como podemos comprobar, el resultado es el mismo, pero aprovechamos las funciones para:

- Crear la funcionalidad de calcular precio (en función de cuantos productos pedimos)
- Crear pequeños bloques de código, que hacen más fácil comprender el código



Funciones definidas en Javascript

En anteriores secciones ya hemos usado algunas funciones, lo que pasaba es que NO éramos conscientes de ello, pero ¿cómo es esto posible, si yo no recuerdo haber creado ninguna? 😱

Es fácil, existe un conjunto de funciones que ya vienen definidas en Javascript, existen en el estándar. Al principio de esta sección usábamos la función `console.log()`, y si nos paramos a pensar, es la función con nombre **log** creada dentro de la consola de Javascript, que sirve para imprimir por pantalla.

Os invitamos a repasar anteriores bloques, en búsqueda de funciones que hemos ido usando, aunque para hacéroslo más fácil, os dejamos aquí descritas algunas de las usadas:

```
var test = "Test";
console.log(test); // Imprime por pantalla "Test"
var res = test.indexOf("T"); // Devuelve la posición de la letra "T" en el string "Test" -> 0
console.log(res);
var res2 = test.includes("st"); // Indica si existe la cadena "st" en el string "Test" -> true
console.log(res2);
```

Resumen

En esta sección hemos entendido lo que son las funciones, y cual es su finalidad.

Ahora sabemos diferenciar lo que compone una función, su nombre semántico, sus parámetros de entrada y su valor de retorno en la salida.

De ahora en adelante seremos capaces de sintetizar nuestro código para separarlo en bloques funcionales.

Seguimos aprendiendo en:

JS | Arrays