UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Diplomatura Professional Front End Developer

Módulo 2:
Desarrollo Web con JavaScript

JavaScript

Universidad Tecnológica Nacional - Facultad Regional Buenos Aires
Medrano 951 (C1179AAQ) C.A.B.A. Tel: (54 11) 4867-7511

Funciones

Cuando se desarrolla una aplicación compleja, es muy habitual utilizar una y otra vez las mismas instrucciones. Un script para una tienda de comercio electrónico, por ejemplo, tiene que calcular el precio total de los productos varias veces, para añadir los impuestos y los gastos de envío.

Cuando una serie de instrucciones se repiten una y otra vez, se complica demasiado el código fuente de la aplicación, ya que:

- El código de la aplicación es mucho más largo porque muchas instrucciones están repetidas.
- Si se quiere modificar alguna de las instrucciones repetidas, se deben hacer tantas modificaciones como veces se haya escrito esa instrucción, lo que se convierte en un trabajo muy pesado y muy propenso a cometer errores.

Las funciones son la solución a todos estos problemas, tanto en JavaScript como en el resto de lenguajes de programación. **Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.**

En el siguiente ejemplo, las instrucciones que suman los dos números y muestran un mensaje con el resultado se repiten una y otra vez:

```
let resultado

let numero1 = 3;
let numero2 = 5;

//se suman los números y se muestra el resultado
resultado = numero1 + numero2
alert(`El resultado es ${resultado}`)

numero1 = 10
numero2 = 7
resultado = numero1 + numero2

alert(`El resultado es ${resultado}`)

numero1 = 5
numero2 = 8
resultado = numero1 + numero2

alert(`El resultado es ${resultado}`)
```

Aunque es un ejemplo muy sencillo, parece evidente que repetir las mismas instrucciones a lo largo de todo el código no es algo recomendable. La solución que proponen las funciones consiste en extraer las instrucciones que se repiten y sustituirlas por una instrucción del tipo "en este punto, se ejecutan las instrucciones que se han extraído".

Por lo tanto, en primer lugar se debe crear la función básica con las instrucciones comunes. Las funciones en JavaScript se definen mediante la palabra reservada function, seguida del nombre de la función. Su definición formal es la siguiente:

```
function nombre_funcion() {
   ……
}
```

El nombre de la función se utiliza para llamar a esa función cuando sea necesario. El concepto es el mismo que con las variables, a las que se les asigna un nombre único para poder utilizarlas dentro del código.

```
function suma_y_muestra() {
    resultado = numero1 + numero2;
    alert("El resultado es " + resultado);
 }
```

De esta forma, el ejemplo anterior queria asi:

```
function suma_y_muestra() {
     resultado = numero1 + numero2
     alert(`El resultado es ${resulado}`)
}

let resultado
let numero1 = 3;
let numero2 = 5;

suma_y_muestra()

numero1 = 10
numero2 = 7

suma_y_muestra()

numero1 = 5
numero2 = 8
suma_y_muestra()
```

Otra posibilidad de las funciones es la de recibir parámetros. Estos son valores que pueden ser usados dentro del código que la función ejecuta. De esta manera las funciones se vuelven sumamente reutilizables. Cuando definimos la función, damos

nombre a los parámetros que queremos recibir y estos estarán disponibles como variables dentro de la función.

Cuando llamemos a la función solo debemos pasar la variable o valor que deseemos usar como parámetro de la función en la posición correcta.

```
function saludar(nombre) {
  alert("Hola " + nombre);
}

saludar("Gabriela")
saludar("Antonio")
```

Por último, las funciones también pueden devolver un valor o resultado mediante el uso de la palabra clave return. Dicho valor puede ser almacenado en una variable o utilizado en alguna de las estructuras de control o bucles que vimos previamente.

```
function numero_al_cubo(numero) {
  return numero * numero * numero;
}

var tres_al_cubo =  numero_al_cubo(3);
alert(tres_al_cubo);
```

DOM

El DOM (Document Object Model en español Modelo de Objetos del Documento) es una API definida para representar e interactuar con cualquier documento HTML o XML. El DOM es un modelo de documento que se carga en el navegador web y que representa el documento como un árbol de nodos, en donde cada nodo representa

una parte del documento (puede tratarse de un elemento, una cadena de texto o un comentario).

El DOM es una de las APIs más usadas en la Web, pues permite ejecutar código en el navegador para acceder e interactuar con cualquier nodo del documento. Estos nodos pueden crearse, moverse o modificarse. Pueden añadirse a estos nodos manejadores de eventos (event listeners en inglés) que se ejecutarán/activarán cuando ocurra el evento indicado en este manejador.

El DOM surgió a partir de la implementación de JavaScript en los navegadores. A esta primera versión también se la conoce como DOM 0 o "Legacy DOM". Hoy en día el grupo WHATWG es el encargado de actualizar el estándar de DOM.

Selectores de elementos

Los selectores api proveen métodos que hacen más fácil y rápido devolver elementos del nodo Element del DOM mediante emparejamiento de un conjunto de selectores. Esto es mucho más rápido que las técnicas anteriores, donde fuera necesario, por ejemplo usar un loop en un código JavaScript para localizar el ítem específico que quisieras encontrar.

document.querySelector()

Devuelve la primera coincidencia del (elemento) Element nodo dentro de las subramas del nodo. Si no se encuentra un nodo coincidente, se devuelve null .

document.querySelectorAll()

Devuelve un listado de nodos NodeList conteniendo todos los elementos del nodo coincidentes(Element) dentro de las subramas del nodo, o Devuelve un Listado de Nodos vacío NodeList si no se encuentran coincidencias.

Estos métodos aceptan uno o más selectores separados por comas entre cada selector para determinar qué elemento o elementos deben ser devueltos. por ejemplo para seleccionar todos los elementos (p) del párrafo en un documento donde la clase CSS sea tanto warning or note, podes hacer lo siguiente:

```
const especial = document.querySelectorAll('p.warning, p.note')
```

También por usar query para etiquetas id. Por ejemplo:

```
const destacados = document.querySelector('#principal, #bienvenidos,
#notas')
```

Luego de ejecutar el código de arriba, la variable contiene el primer elemento del documento, su ID puede ser uno de los siguientes: principal, bienvenidos, o notas.

Podes usar cualquier selector CSS con los métodos **querySelector()** y **querySelectorAll().**

document.getElementById()

Este método permite seleccionar una elemento del DOM usando el atributo id del mismo.

```
const mensaje = document.getElementById('mensaje')
```

document.getElementsByClassName()
Este método devuelve un array o lista de elementos que contengan la clase que se especifique como parámetro.

```
const items = document.getElementsByClassName('item')
```

`document.getElementsByTagName()`
Este método devuelve un array o lista de elementos cuya etiqueta html sea la que se especifique como parámetro.

```
const parrafos = document.getElementsByTagName('p')
```

innerHTML e innerText

Las propiedades de los elementos del DOM innerHTML e innerText hacen referencia al contenido de los nodos. Mientras que innerText permite insertar y manipular contenido sencillo dentro de los nodos, innerHTML nos permite incluir código HTML más complejo sin la necesidad de crear los elementos a mano.

```
<body>
  <p id="demo">Este elemnto tiene espacio extra y contiene
  <span>una etiqueta span</span></p>
  <script>

      function getHTML() {
          alert(document.getElementById('demo').innerHTML);
      }

      function getInnerText() {
          alert(document.getElementById('demo').innerText);
      }
  </script>
</body>
```