Capítulo 4: Fundamentos de JavaScript

MSc. Juan Antonio Castro Silva

May 19, 2020

Versión: 0.1 (20200508_1156)

1 Introducción

JavaScript es el lenguaje de programación de HTML y la Web. Se debe utilizar en la capa de comportamiento (behavior), permite la iteración del usuario y el manejo de datos de forma dinámica.

JavaScript es uno de los 3 lenguajes que todos los desarrolladores web debe aprender:

1. HTML para definir el contenido y la estructura de las páginas web.
2. CSS para especificar el diseño (layout) de las páginas web.
3. JavaScript para programar el comportamiento (behavior) de las páginas web.

JavaScript no se utiliza solamente en las páginas web. Muchos programas de escritorio y servidor usan JavaScript. Node.js es el más conocido. Algunas bases de datos, tales como MongoDB, tambien usan JavaScript como su lenguaje de programación.

JavaScript y Java son lenguajes completamente diferentes, tanto en concepto como en diseño. JavaScript fue inventado por Brendan Eich en 1995, y se volvio un estandar ECMA en 1997. ECMA-262 es el nombre oficial del estandard. ECMAScript es el nombre oficial del lenguaje.

JavaScript no se instala, no se descarga, ya esta corriendo en el navegador (browser) del computador, tablet o smart-phone. Es software libre (Free to use).

En este capítulo, se discutirán las características básicas de JavaScript y como utilizarlo en la práctica. El codigo fuente de este capítulo se encuentra en el repositorio de github del curso `https://github.com/juancasi/programacion_web`.

2 Formas de insertar JavaScript

2.1 La etiqueta <script>

En HTML, el código JavaScript se inserta entre la etiquetas <script> y </script>.

```
<script>
....
</script>
```

Ejemplos viejos de JavaScript pueden usar un atributo type: <script type=”text/javascript”>. El atributo type no es requerido, JavaScript es el lenguaje de script por defecto en HTML.

2.2 JavaScript en <head>(No Recomendado)

En este caso, se pone una función JavaScript en la sección <head> de una página HTML.

La función se invoca (llama) cuando se hace click en un botón. ¡h2¿Something ¡b¿here¡/b¿¡

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
```

```
}
</script>
</head>
<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

2.3 JavaScript en <body>(No Recomendado)

En este ejemplo, se ubica una función en la sección <body> de una página HTML.

La función se invoca (llama) cuando se hace click en un botón.

```
<!DOCTYPE html>
<html>
<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
 document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

Colocar scripts (guiones) en la parte inferior del elemento <body> mejora la velocidad de visualización, porque la interpretación de los scripts ralentiza la visualización.

2.4 JavaScript Externo (Buena práctica - Recomendado)

Los scripts tambien se pueden ubicar en archivos externos:

External file: myScript.js

```
function myFunction() {
 document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

Los scripts externos son prácticos cuando el mismo código se utiliza en muchas páginas web diferentes.

Los archivos JavaScript tiene la extensión .js. Para usar un script externo, ponga el nombre del archivo de script en el atributo src (source) de una etiqueta <script>.

Ejemplo:

```
<script src="myScript.js"></script>
```

Los scripts externos no pueden contener la etiqueta (tag) <script>.

2.4.1 Ventajas de los JavaScripts externos

Ubicar scripts en archivos externos tiene algunas ventajas:

- Separa HTML y código Hace mas facil leer y mantener HTML y JavaScript
- Los archivos JavaScript en caché pueden acelerar la carga de las páginas

Para adicionar varios archivos de script a una página - use varias etiquetas script:

```
<script src="myScript1.js"></script>
<script src="myScript2.js"></script>
```

2.5 Referencias externas

Los scripts externos se pueden referenciar con una URL completa o con una ruta (path) relativa a la página web actual.
Este ejemplo utiliza una URL completa para enlazar a un script:
Ejemplo

```
1 <script src="https://www.w3schools.com/js/myScript1.js"></script>
```

Este ejemplo utiliza un script localizado en un folder especificado en el sitio web actual:

Ejemplo

```
1 <script src="/js/myScript1.js"></script>
```

¡h2¿Something ¡b¿here¡/b¿¡ Este ejemplo enlaza a un script ubicado en la misma carpeta que la página actual:
Ejemplo

Ejemplo

```
1 <script src="myScript1.js"></script>
```

3 Usos de JavaScript

3.1 JavaScript para cambiar el contenido HTML

El método getElementById() de JavaScript permite encontrar un elemento HTML.

```
1 document.getElementById("demo").innerHTML = "Hello JavaScript";
```

El método getElementById() de JavaScript permite encontrar el elemento HTML con id=”demo” dentro de la página (document) y al modificar la propiedad innerHTML le cambia el contenido al elemento a ”Hello JavaScript”.

3.1.1 JavaScript para cambiar el valor de una propiedad

3.1.2 JavaScript para cambiar el valor de un atributo

¡h2¿Something ¡b¿here¡/b¿¡

```
1 document.getElementById('myImage').src = 'pic_bulbon.gif'"
```

3.1.3 JavaScript para cambiar estilos HTML (CSS)

Cambiar el estilo de un elemento HTML, es una variante de cambiar un atributo HTML:

```
1 document.getElementById("demo").style.fontSize = "35px";
```

4 Fundamentos de JavaScript

4.1 Salida de JavaScript - Output

JavaScript puede ”mostrar” datos de diferentes maneras:

- Escribir dentro de un elemento HTML, usando innerHTML.
- Escribir dentro de un cajon de alerta, usando window.alert().
- Escribir dentro de la consola del navegador, usando console.log().

4.1.1 Usar innerHTML

Para accesar un elemento HTML, JavaScript puede usar el método document.getElementById(id). El atributo id define el elemento HTML. La propiedad innerHTML define el contenido HTML:

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Cambiar la propiedad innerHTML de un elemento HTML es una forma común para mostrar datos en HTML.

4.1.2 Usar alert()

Se puede usar un cajon de alerta para mostrar datos:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

4.1.3 Usar console.log()

Para fines de depuración, se puede llamar al método console.log() en el navegador para mostrar datos.

```
<!DOCTYPE html>
<html>
<body>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

4.2 Tipos de datos JavaScript

Las variables de JavaScript pueden contener muchos tipos de datos: números, cadenas, objetos y más:

```
var length = 16;                               // Number
var lastName = "Johnson";                      // String
var x = {firstName:"John", lastName:"Doe"};    // Object
```

JavaScript Types are Dynamic JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

```
var x;           // Now x is undefined
x = 5;           // Now x is a Number
x = "John";      // Now x is a String
```

4.2.1 Cadenas de texto (Strings)

Un string (o una cadena de texto) es una serie de caracteres como "John Doe". Las cadenas de texto se escriben con comillas. Se pueden usar comillas simples o dobles:

```
var carName1 = "Volvo XC60";   // Using double quotes
var carName2 = 'Volvo XC60';   // Using single quotes
```

Se pueden usar comillas dentro de una cadena de texto, siempre que no coincidan con las comillas que rodean la cadena:

```
var answer1 = "It's alright";             // Single quote inside double quotes
var answer2 = "He is called 'Johnny'";    // Single quotes inside double quotes
var answer3 = 'He is called "Johnny"';    // Double quotes inside single quotes
```

4.2.2 Números

JavaScript tiene solamente un tipo de números.

Los números se pueden escribir con, o sin decimales:

```
var x1 = 34.00;     // Written with decimals
var x2 = 34;        // Written without decimals
```

Los números extra largos o extra pequeños se pueden escribir con notación científica (exponencial).

```
var y = 123e5;      // 12300000
var z = 123e-5;     // 0.00123
```

4.2.3 Booleanos

Los booleanos solamente pueden tener dos valores: true o false (verdadero o falso).

```
var x = 5;
var y = 5;
var z = 6;
(x == y)       // Returns true
(x == z)       // Returns false
```

Los booleanos se usan a menudo en pruebas condicionales.

4.2.4 Arreglos (Arrays)

Los arreglos en JavaScript se escriben con corchetes.

Los elementos de un arreglo se separan por comas.

El siguiente código declara (crea) un arreglo llamado cars, que contiene tres elementos (nombres de carros):

```
var cars = ["Saab", "Volvo", "BMW"];
```

Los índices de los arreglos están basados en cero (zero-based), lo que significa que el primer elemento es [0], el segundo es [1], y asi sucesivamente.

4.2.5 Objetos

Los objetos en JavaScript se escriben con llaves .

Las propiedades de los objetos se escriben como parejas (pairs) nombre:valor, separados por comas.

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

El objeto person en el ejemplo de arriba tiene cuatro propiedades: firstName, lastName, age, y eyeColor.

4.2.6 Operador typeof

Se puede usar el operador typeof para encontrar el tipo de una variable de JavaScript.

```
typeof ""              // Returns "string"
typeof "John"          // Returns "string"
typeof "John Doe"      // Returns "string"
```

4.2.7 Undefined

En JavaScript, una variable sin un valor, tiene el valor de undefined (indefinido). El tipo también es undefined.

```
var car;    // Value is undefined, type is undefined
```

Cualquier variable se puede vaciar, estableciendo el valor en undefined. El tipo también será undefined.

```
var car = "Volvo";
car = undefined;    // Value is undefined, type is undefined
```

4.2.8 Funciones

Una función en JavaScript es un bloque de código diseñado paar ejecutar una tarea particular.

Un función se ejecuta cuando ”algo” la invoca (la llama).

```
function myFunction(p1, p2) {
  return p1 * p2;   // The function returns the product of p1 and p2
}
```

Sintaxis de una función

Una función se define con la palabra clave function, seguido por un nombre, seguido por paréntesis ().

Los nombres de funciones pueden contener letras, digitos, raya al piso (underscores), y signo dolar (las mismas reglas que las variables).

Los paréntesis pueden incluir nombres de parametros separados por comas: (parametro1, parametros2, ...)

El código a ser ejecutado, por la función, se ubica dentro de llaves: .

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Los parametros de la función se listan dentro de los paréntesis () en la definición de la función.

Los argumentos de la función son los valores recibidos por la función cuendo se invoca.

Dentro de la función, los argumentos (los parametros) se comportan como variables locales.

4.2.9 Retorno de una función (return)

Cuando JavaScript alcanza una declaración (statement) de retorno, la función dejará de ejecutarse.

Si la función fue invocada desde una declaración, JavaScript ”regresará” (return) para ejecutar el código después de la declaración de invocación.

Las funciones con frecuencia computan un valor de retorno. El valor de retorno se devuelve al llamador (caller).

El siguiente ejemplo calcula el producto de dos números, y retorna el resultado:

```
var x = myFunction(4, 3);   // Function is called, return value will end up in x

function myFunction(a, b) {
  return a * b;             // Function returns the product of a and b
}
```

El resultado en x será: 12

5 Eventos

Los eventos HTML son ”cosas” que suceden a los elementos HTML.

Cuando JavaScript se utiliza en las páginas HTML, JavaScript puede ”reaccionar” ante estos eventos.

5.1 Eventos HTML

Un evento HTML puede ser algo que hace el navegador, o algo que hace un usuario.
Ejemplos de eventos HTML:

- Una página web HTML ha finalizado de cargar.
- Un campo de entrada HTML ha cambiado.
- Se ha hecho click en un botón HTML.

A menudo, cuando los eventos ocurren, se desea hacer algo.
JavaScript permite ejecutar código cuando se detectan los eventos.
HTML permite agregar atributos de controlador de eventos, con código JavaScript, a los elementos HTML.
Con comillas sencillas o dobles:

```
<element event='some JavaScript'>

<element event="some JavaScript">
```

En el siguiente ejemplo, se adiciona un atributo onclick a un elemento <button>. Es común ver a los atributos evento llamar funciones:

```
<button onclick="displayDate()">The time is?</button>
```

5.2 Eventos HTML

La tabla muestra una lista de algunos de los eventos HTML más comunes.

Evento	Descripción
onchange	Un elemento HTML ha sido cambiado
onclick	El usuario hace click en un elemento HTML
onmouseover	El usuario mueve el mouse sobre un elemento HTML
onmouseout	El usuario aleja el mouse de un elemento HTML
onkeydown	El usuario presiona una tecla del teclado
onload	El navegador ha finalizado de cargar la página

```
1 <!DOCTYPE html>
```

My box.

One-line code formatting also works with minted. For instance, a simple html sample like this:

`<h2>Something <b>here</b></h2>`

can be properly formatted.