Capítulo 1

Introducción a JavaScript

1.1. Historia de JavaScript

JavaScript, comunmente abreviado como JS, es un lenguaje de programación interpretado. En los comienzos, el lenguaje se utilizaba para agregar dinamismo del lado del cliente a las páginas web. Sin embargo, hoy en día se pueden crear aplicaciones de escritorio o del lado del servidor.

El lenguaje fue creado por **Brendan Eich** en 1995, quien en ese entonces trabajaba para Netscape. Eich denominó a su lenguaje `LiveScript`, y el objetivo inicial del lenguaje era solucionar problemas de validación de formularios complejos en el lado del cliente para el navegador Netscape Navigator, tratando de adaptarlo a tecnologías ya existentes.

La empresa Netscape junto con Sun Microsystems desarrollaron en conjunto este lenguaje de programación. Pero por cuestiones de mercado antes del lanzamiento, Netscape decidió cambiar el nombre del lenguaje a JavaScript (ya que en ese entonces Java estaba de moda en el mundo informático).

Al poco tiempo, la empresa Microsoft lanzó `JScript` para Internet Explorer. Para no entrar en una guerra informática, Netscape decidió que lo mejor sería estandarizar el lenguaje. Para ello, enviaron la especificación de JavaScript 1.1 al organismo ECMA (European Computer Manufacturers Association).

ECMA creó el comité TC39 con el objetivo de "*estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa*". El primer estándar que creó el comité TC39 se denominó ECMA-262, en el que se definió por primera vez el lenguaje ECMAScript (abreviado comunmente como `ES`).

Es así entonces, que cuando hablamos de JavaScript, estamos haciendo referencia a una implementación de lo que se conoce como `ECMAScript`. El estándar ha ido evolucionando con el paso del tiempo. En la actualidad, la mayoría de los navegadores corren algún intérprete que soporta la mayoría de las características de las versiones 5.1 y 6.

Aunque la última versión sea la de ECMAScript 8 (lanzada en Junio de 2017), la versión 6 es más popular, ya que en ésta se han agregado muchos cambios significativos para el lenguaje. En este documento se hará énfasis en las versiones 5.1 y 6.

1.2. JavaScript en la actualidad

Después de más de 20 años de existencia, los usos del lenguaje han cambiado. JavaScript ya no es más un lenguaje para hacer validaciones de formularios complejos en páginas de Internet, ni tampoco para agregar dinamismo o animaciones a las páginas.
En la actualidad, se puede afirmar que JavaScript está en «la cresta de la ola». ¿Qué se puede hacer hoy en día con JavaScript?

- Páginas Web – Pareciera la respuesta obvia, sin embargo la forma de crear sitios web ha cambiado con el paso del tiempo. Hoy en día existe una gran cantidad de frameworks basados en JavaScript, tales como **React**, **AngularJS** o **Vue.JS**, entre otros.
- Aplicaciones móviles – Se pueden crear aplicaciones para celulares o dispositivos móviles programando en JavaScript, usando **Apache Cordova**, **Sencha**, **Ionic**, **NativeScript** o **Tabris.JS**.
- Aplicaciones de escritorio – Así como recién se hizo mención de las aplicaciones móviles, las de escritorio no se quedan atrás. Algunos frameworks como **Electron** ó **NW.JS** permiten crear aplicaciones multiplataforma.
- Robots – Mediante frameworks como **Cylon.JS** se pueden manejar dispositivos de hardware o robots. También existen kits basados en Arduino para programar en JS, tales como **Johnny-Five** o **Nodebots**.

1.3. Características del lenguaje

JavaScript es un lenguaje de alto nivel, interpretado y multiparadigma. Es dinámica y débilmente tipado.

Posee herencia basada en prototipos. Este tipo de herencia es muy particular, y muy pocos lenguajes lo tienen.

Se dice que es multiparadigma porque soporta los paradigmas imperativo, funcional, orientado a objetos (prototipado) y dirigido por eventos.

En el ecosistema de la Web, JavaScript es uno de los lenguajes más populares. Todos los navegadores en la actualidad tienen un intérprete del lenguaje.

Si bien tiene bastantes partes criticables, JavaScript tiene la fama de ser un lenguaje «*liviano*» y «*expresivo*».

1.3.1. Influencias

JavaScript tiene fuertes influencias de varios lenguajes. Sus características más sobresalientes surgen de los siguientes lenguajes:

Java y C – No solo tiene la influencia sobre el nombre, sino que además tiene influencia sobre la sintaxis del lenguaje. Tanto Java como JavaScript sintácticamente emergen del lenguaje C. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes.

Perl y Python – Tanto Perl como Python han influido en el manejo de strings, arreglos y expresiones regulares en JavaScript.

Scheme – De la familia del paradigma funcional. Adopta las funciones de primera clase y *closures*, los cuales se tratarán más adelante.

Self – Un lenguaje desarrollado por Sun Microsystems. Es de los pocos lenguajes que tienen herencia prototipada. Además de ésta característica, también se adopta la inusual notación de objetos.

1.3.2. Intérpretes

Ya se ha mencionado que JavaScript es un lenguaje interpretado. Sin embargo es necesario mencionar algunos «motores» que se encargan de interpretar el código en JavaScript.
Actualmente la gran mayoría de los navegadores (web browsers) viene con un intérprete de JS incorporado. A continuación se mencionan algunos de los más populares:

- **Rhino** – Gestionado por la fundación Mozilla, es de código abierto y está desarrollado completamente en Java.
- **SpiderMonkey** – También desarrollado por Mozilla para el navegador Firefox. Escrito en C++. Es utilizado en proyectos como MongoDB y GNOME.
- **Chakra** – Desarrollado por Microsoft, primero para Internet Explorer, y luego para Microsoft Edge.
- **V8** – El motor por defecto para Google Chrome, y también utilizado Node.JS, Opera y otros proyectos populares. Escrito en C++, maneja alocación en memoria y posee garbage collector.
- **JavaScriptCore** – Es utilizado por navegadores como Safari o PhantomJS. También es conocido como SquirrelFish o Nitro, bajo proyectos similares con otro nombre por cuestiones de mercado.

El objetivo de esta sección no es entrar en detalle ni hacer un análisis comparativo de los intérpretes. Basta con hacer una pequeña búsqueda para notar que varios de éstos intérpretes poseen garbage collection, compilación JIT (just in time), y estrategias para la optimización del código.
A lo largo de este documento se mostrarán ejemplos de código, cuya interpretación se realizará utilizando Node.JS (V8), y la consola de los navegadores Google Chrome (V8) y Mozilla Firefox (SpiderMonkey).
En caso de que el lector quiera ejecutar el código JavaScript, se deja a disposición los enlaces de descarga de las herramientas mencionadas:

- Node.JS – nodejs.org
- Google Chrome – google.com/chrome
- Mozilla Firefox – mozilla.org/firefox

Para abrir el intérprete desde Node.JS, basta con escribir `node` en la línea de comandos. Mientras que para el caso de los navegadores, hace falta apretar la tecla F12 para abrir la consola.

1.4. Nociones básicas

1.4.1. Tipos primitivos

Undefined

El tipo indefinido tiene un único valor, `undefined`. A toda variable que aún no se le haya asignado valor, tendrá el valor `undefined`.

Null

El tipo nulo tiene un único valor, `null`, que representa al valor nulo o «vacío».

Boolean

El tipo booleano representa una entidad lógica con dos posibles valores, `true` ó `false`.

String

Utilizado para representar datos de texto, el tipo String está definido como cero o más elementos, donde cada elemento es un entero no signado de 16 bits, de una longitud máxima de $2^{52} - 1$ elementos.

Number

Representa al conjunto de datos numérico. Se basa en la norma IEEE 754-2008, formato doble precisión de 64 bits en la aritmética de punto flotante. Toma algunos valores especiales de este conjunto para representar datos como NaN (Not a Number) y también +Infinity y -Infinity. La cantidad de valores reservados para NaN es dependiente de la implementación.

Symbol

Fue agregado en la versión de ES6. Abarca el conjunto de todos los valores no String que pueden ser usados como clave en la propiedad de un Object. Cada valor posible de Symbol es único e inmutable. Se los puede pensar como tokens que sirven como identificadores únicos.

Object

Es la forma básica de representar un objeto en JavaScript. Está compuesto por una colección de propiedades.

Las propiedades se identifican usando claves. El valor de una clave puede ser o bien un String, o bien un Symbol. Todos los valores String y Symbol son válidos como nombre clave para una propiedad, inclusive la cadena vacía.

1.4.2. Palabras reservadas

Las palabras reservadas del lenguaje se dividen en cuatro conjuntos:

- Palabras claves (*keywords*)
- Palabras reservadas a futuro
- Literal nulo (`null`)

- Literales booleanos (`true` y `false`)

Las siguientes son palabras claves, a excepción de `null`, `true` y `false`, que son literales.

CUADRO 1.1: Lista de palabras claves del lenguaje.

break	do	import	throw
case	else	in	true
catch	export	instanceof	try
class	extends	new	typeof
const	false	null	var
continue	finally	return	void
debugger	for	super	while
default	function	switch	with
delete	if	this	yield

Por otro lado, existe un conjunto de palabras reservadas a futuro. En un principio son solamente dos: `await` y `enum`. Pero si se especifica la directiva de *strict mode*, aparecen otras más: `implements`, `interface`, `package`, `private`, `protected` y `public`.

En resumen, las palabras reservadas a futuro (en modo estricto) son:

CUADRO 1.2: Lista de palabras reservadas a futuro.

await	implements	package	protected
enum	interface	private	public

1.4.3. Otras cuestiones a tener en cuenta

Identificadores

- Un identificador debe comenzar con una letra, signo pesos ($), ó guión bajo (_).
- Un identificador consiste en letras, números, signo pesos ($), ó guión bajo (_).
- Se permiten caracteres Unicode.
- No se permite el uso de palabras reservadas como identificadores.

Sensible a las mayúsculas

JavaScript es un lenguaje sensible a las mayúsculas, lo que significa que se entiende a `miVariable` y a `MIVARIABLE` como dos identificadores totalmente diferentes.

1.5. Sintaxis

A continuación se hará una introducción sintáctica al lenguaje de forma breve y mediante ejemplos. El objetivo de esta sección no es detallar la especificación del

lenguaje, sino dar un repaso general por los elementos básicos, las estructuras de control y de repetición. Para mayor detalle sobre la sintaxis, se recomienda leer el «Standard ECMA-262 (Language Specification)».

1.5.1. Comentarios

Los comentarios en JavaScript se realizan de forma similar a los lenguajes influenciados por C (como por ejemplo JavaScript o C++). Es posible hacer comentarios inline, asi como también multilínea.

```
1 // Esto es un comentario en una sola linea
```

Comentario inline

```
1 /*
2 Esto es un comentario
3 escrito en varias lineas
4 */
```

Comentario multilinea

1.5.2. Variables

Para la declaración de variables, el lenguaje posee la palabra reservada `var`. Una variable tendrá el valor inicial `undefined` a menos que se la inicialice en su declaración.

También se pueden hacer múltiples declaraciones en la misma línea, incluso con la asignación de un valor inicial.

```
1 var a;                    // Definiendo una variable con nombre a
2 var b = 1;                // Definiendo una variable con nombre b
3 var c, d, e;              // Definiendo varias variables
4 var f, g = true, h;       // Esto tambien es valido
5 var i = "Hola", j = 2;    // Definiendo y asignando multiples variables
```

Declarando variables

Vale la pena hacer mención también a dos nuevas formas de definir variables a partir de ES6. Se trata de `let` y `const`.

Sobre `let`, es una forma de declarar variables de alcance local. Se hará énfasis en este punto en futuros capítulos cuando se muestren los problemas de alcance que posee el lenguaje.

Por el lado de `const`, se tratan de variables de valor constante, cuyo valor no se puede cambiar y tampoco pueden ser redeclaradas.

1.5.3. Condicional `if`

Se posee uso del condicional `if` de manera similar a Java.

`if` («condición») «sentencia1» [`else` «sentencia2»]

1.5.4. Condicional `switch`

1.5.5. Repetición `for`

1.5.6. Repetición `while` y `do...while`

sdsdsdsd

Capítulo 2

Sistema de Tipos

JavaScript es un lenguaje de «scripting», con tipado dinámico y un sistema de tipos débil. Generalmente, en este tipo de lenguajes interpretados no es necesario definir el tipo de una variable al momento de declararla, por lo que es lógico preguntarse ¿Es JavaScript un lenguaje seguro?.

2.1. Tipos primitivos

Tal como se mencionó en el Capítulo 1, existen solamente 7 tipos primitivos del lenguaje:

- `undefined`
- `null`
- `number`
- `string`
- `boolean`
- `symbol`
- `object`

Algunos autores consideran que `Object` no es un tipo primitivo, sino que es un tipo que hereda de `Null`. Por otro lado, hay que mencionar que las funciones en JavaScript son consideradas objetos, por lo que `Function` es un subtipo de `Object`.