Javascript (Febrero 2014)

Raúl Andrés Ortiz Fuentes, 112508.

*Índice de términos*— *Javascript: es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,3 basado en prototipos, imperativo, débilmente tipado y dinámico..*

# INTRODUCCIÓN

L

a creación de sitios web sigue en aumento y gracias a la masificación de estas tecnologías hoy en día existen numerosos javascript que ayudan a acelerar el proceso de desarrollo de un sitio web.

# Javascript

1. *Definicion*

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Se llevó a cabo originalmente como parte de los navegadores web para que los scripts del lado del cliente podría interactuar con el usuario, controlar el navegador, comunicarse de forma asincrónica, y alterar el contenido del documento que se estaba visualizando.

# Sintaxis

La sintaxis de JavaScript es muy similar a la de otros lenguajes de programación como Java y C. Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

1. *No se tienen en cuenta los espacios en blanco y las nuevas líneas:*

Como sucede con XHTML, el intérprete de JavaScript ignora cualquier espacio en blanco sobrante, por lo que el código se puede ordenar de forma adecuada para entenderlo mejor (tabulando las líneas, añadiendo espacios, creando nuevas líneas, etc.)

Se distinguen las mayúsculas y minúsculas: al igual que sucede con la sintaxis de las etiquetas y elementos XHTML. Sin embargo, si en una página XHTML se utilizan indistintamente mayúsculas y minúsculas, la página se visualiza correctamente, siendo el único problema la no validación de la página. En cambio, si en JavaScript se intercambian mayúsculas y minúsculas el script no funciona.

1. *No se define el tipo de las variables:*

Al crear una variable, no es necesario indicar el tipo de dato que almacenará. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.

1. *No es necesario terminar cada sentencia con el carácter de punto y coma (;):*

En la mayoría de lenguajes de programación, es obligatorio terminar cada sentencia con el carácter ;. Aunque JavaScript no obliga a hacerlo, es conveniente seguir la tradición de terminar cada sentencia con el carácter del punto y coma (;).

1. *Se pueden incluir comentarios:*

Los comentarios se utilizan para añadir información en el código fuente del programa. Aunque el contenido de los comentarios no se visualiza por pantalla, si que se envía al navegador del usuario junto con el resto del script, por lo que es necesario extremar las precauciones sobre la información incluida en los comentarios.

Aparte de las cuatro normas básicas que defininen la sintaxis del lenguaje JavaScript, este se inserta en documentos HTML (*HyperText Markup Languaje*), de forma que su código queda reflejado en la propia página y no es llamado o cargado de ninguna fuente externa. Se trata de un lenguaje interpretado puro (ni compilación, ni generación de intermedios codificados de ningún tipo) y sensible a mayúsculas, aunque algunas implementaciones ignoran en parte este último extremo.

Todo el código de JavaScript debe explicitarse en la página HTML según el formato:

*<SCRIPT Language="JavaScript">*

*<!--// Aquí irá su código// -->*

*</SCRIPT>*

Se observa que el tag utilizado es *<SCRIPT LANGUAGE=...>.....</SCRIPT>*. En *LANGUAGE* se indica qué lenguaje script se utilizará.

Con lo aterior es necesario especificar los elementos que compenen el leguaje, como varibles, operadores, funciones y demás, los cualues se explicran a continuación:

1. *Las variables y sus tipos*

Una variable no es más que un contenedor para un valor. Las variables se declaran por medio de la palabra clave *var*, pero a diferencia de otros lenguajes de programación en JavaScript no es necesario declarar su tipo. En cuanto a su tipología se dividen en las siguientes:

* *Numéricas*

Se utilizan para almacenar valores numéricos enteros (llamados integer en inglés) o decimales (llamados float en inglés). En este caso, el valor se asigna indicando directamente el número entero o decimal. Los números decimales utilizan el carácter . (punto) en vez de , (coma) para separar la parte entera y la parte decimal:

* *Cadenas de texto*

Se utilizan para almacenar caracteres, palabras y/o frases de texto. Para asignar el valor a la variable, se encierra el valor entre comillas dobles o simples, para delimitar su comienzo y su final:

*var mensaje = "Bienvenido a nuestro sitio web";*

*var nombreProducto = 'Producto ABC';*

*var letraSeleccionada = 'c';*

* *Arrays*

En ocasiones, a los arrays se les llama vectores, matrices e incluso arreglos. No obstante, el término array es el más utilizado y es una palabra comúnmente aceptada en el entorno de la programación. Un array es una colección de variables, que pueden ser todas del mismo tipo o cada una de un tipo diferente.

Para definir un array, se utilizan los caracteres [ y ] para delimitar su comienzo y su final y se utiliza el carácter , (coma) para separar sus elementos:

*var nombre\_array = [valor1, valor2, ..., valorN];*

Cada elemento se accede indicando su posición dentro del array. Por último es válido a aclarar que las posiciones de los elementos empiezan a contarse en el 0 y no en el 1.

* *Booleanos*

Las variables de tipo boolean o booleano también se conocen con el nombre de variables de tipo lógico. Una variable de tipo boolean almacena un tipo especial de valor que solamente puede tomar dos valores: true (verdadero) o false (falso). No se puede utilizar para almacenar números y tampoco permite guardar cadenas de texto. Los únicos valores que pueden almacenar estas variables son true y false, por lo que no pueden utilizarse los valores verdadero y falso. A continuación se muestra un par de variables de tipo booleano:

*var clienteRegistrado = false;*

*var ivaIncluido = true;*

1. *Operadores*

Los operadores permiten manipular el valor de las variables, realizar operaciones matemáticas con sus valores y comparar diferentes variables. De esta forma, los operadores permiten a los programas realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones. Dichos operadores se clasifican en Adsignacion, Inrementeo, Decremento y Logicos, de los cuales se descripbiran a continuación:

* *Asignación*

El operador de asignación es el más utilizado y el más sencillo. Este operador se utiliza para guardar un valor específico en una variable. El símbolo utilizado es =.

* *Incremento y decremento*

Estos dos operadores solamente son válidos para las variables numéricas y se utilizan para incrementar o decrementar en una unidad el valor de una variable.

*var numero = 5; ++numero;*

El operador de incremento se indica mediante el prefijo ++ en el nombre de la variable. El resultado es que el valor de esa variable se incrementa en una unidad. De forma equivalente, el operador decremento (indicado como un prefijo -- en el nombre de la variable) se utiliza para decrementar el valor de la variable:

*var numero = 5; --numero;*

Los operadores de incremento y decremento no solamente se pueden indicar como prefijo del nombre de la variable, sino que también es posible utilizarlos como sufijo. En este caso, su comportamiento es similar pero muy diferente.

* *Lógicos*

Los operadores lógicos son imprescindibles para realizar aplicaciones complejas, ya que se utilizan para tomar decisiones sobre las instrucciones que debería ejecutar el programa en función de ciertas condiciones.

El resultado de cualquier operación que utilice operadores lógicos siempre es un valor lógico o booleano.

1. *Negación*

Uno de los operadores lógicos más utilizados es el de la negación. Se utiliza para obtener el valor contrario al valor de la variable. La negación lógica se obtiene prefijando el símbolo ! al identificador de la variable.

1. *AND*

La operación lógica AND obtiene su resultado combinando dos valores booleanos. El operador se indica mediante el símbolo && y su resultado solamente es true si los dos operandos son true.

1. *OR*

La operación lógica OR también combina dos valores booleanos. El operador se indica mediante el símbolo || y su resultado es true si alguno de los dos operandos es true.

* *Matemáticos*

JavaScript permite realizar manipulaciones matemáticas sobre el valor de las variables numéricas. Los operadores definidos son: suma (+), resta (-), multiplicación (\*) y división (/). Además de los cuatro operadores básicos, JavaScript define otro operador matemático, el cual es el operador "módulo", que calcula el resto de la división entera de dos números. El operador módulo en JavaScript se indica mediante el símbolo %.

* *Relacionales*

Los operadores relacionales definidos por JavaScript son idénticos a los que definen las matemáticas: mayor que (>), menor que (<), mayor o igual (>=), menor o igual (<=), igual que (==) y distinto de (!=). Los operadores relacionales también se pueden utilizar con variables de tipo cadena de texto.

# Estructuras de control de flujo

Las estructura de flujo de control pueden ser de selección o de recursión. Si se utilizan estructuras de control de flujo, los programas dejan de ser una sucesión lineal de instrucciones para convertirse en programas inteligentes que pueden tomar decisiones en función del valor de las variables.

* 1. *Estructura if*

La estructura más utilizada en JavaScript y en la mayoría de lenguajes de programación es la estructura if. Se emplea para tomar decisiones en función de una condición. Su definición formal es:

*if(condicion) {*

*...*

*}*

Si la condición se cumple se ejecutan todas las instrucciones que se encuentran dentro de {...}. Si la condición no se cumple no se ejecuta ninguna instrucción contenida en {...} y el programa continúa ejecutando el resto de instrucciones del script.

* 1. *Estructura if...else*

Contiene el mismo prinicpio del if, sin embargo a esta se le añade la instrucción else, que permite realizar la ejecución de instrucciones de código si llegado el caso la condición del if no se cunple. Su definición formal es la siguiente:

*if(condicion) {*

*...*

*}*

*else {*

*...*

*}*

Si la condición se cumple se ejecutan todas las instrucciones que se encuentran dentro del if(). Si la condición no se cumple se ejecutan todas las instrucciones contenidas en else { }.

* 1. *Estructura for*

Las estructuras if y if...else no son muy eficientes cuando se desea ejecutar de forma repetitiva una instrucción. La estructura for permite realizar repeticiones también llamadas bucles, de una forma muy sencilla. Su definición formal es:

*for(inicializacion; condicion; actualizacion) {*

*...*

*}*

La idea del funcionamiento de un bucle for es la siguiente: "mientras la condición indicada se siga cumpliendo, repite la ejecución de las instrucciones definidas dentro del for. Además, después de cada repetición, actualiza el valor de las variables que se utilizan en la condición".

La "inicialización" es la zona en la que se establece los valores iniciales de las variables que controlan la repetición.

La "condición" es el único elemento que decide si continua o se detiene la repetición.

La "actualización" es el nuevo valor que se asigna después de cada repetición a las variables que controlan la repetición.

*Ejemplo:*

*for(var i = 0; i < 5; i++) {*

*alert(mensaje);*

*}*

* 1. *Estructura for...in*

Una estructura de control derivada de for es la estructura for...in. este tipo de estrutura permite recorrer listas de objetos o en otros términos array de elementos, sin la necesidad del manejo de un índice que indica la posición del objeto como se hace en el for. En este caso la codicion esta relacionada directamente con la cantidad de objetos que existen en la lista, ya que el for..in se ejecutara hasta que llegue hasta el ulotimo de la lista. Para la inicialización no se declara un índice, si no que se declara un objeto del mismos tipo que se encuentra almacenado en la lista. Su definición es la siguiente:

for(indice in array) {

...

}

Si se quieren recorrer todos los elementos que forman un array, la estructura for...in es la forma más eficiente de hacerlo, como se muestra en el siguiente ejemplo:

var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];

for(i in dias) {

alert(dias[i]);

}

La variable que se indica como indice es la que se puede utilizar dentro del bucle for...in para acceder a los elementos del array. De esta forma, en la primera repetición del bucle la variable i vale 0 y en la última vale 6.

Esta estructura de control es la más adecuada para recorrer arrays (y objetos), ya que evita tener que indicar la inicialización y las condiciones del bucle for simple y funciona correctamente cualquiera que sea la longitud del array. De hecho, sigue funcionando igual aunque varíe el número de elementos del array.

Referencias

1. Luzcila, “LOS 11 MEJORES FRAMEWORKS GRATUITOS PARA APLICACIONES WEB”, 2013,Recuperado de: http://elbauldelprogramador.com/los-10-mejores-frameworks-gratis-de-aplicaciones-web/
2. Agustín Baraza, “20 frameworks responsivos para adaptar tu web a todos los dispositivos”, 2014, Recuperado de: http://www.nosolocss.com/blog/recursos/20-frameworks-responsivos-para-adaptar-tu-web-a-todos-los-dispositivos
3. GeRO Reimers, “bocetostipograficos”, 2011, Recuperado de: http://www.bocetostipograficos.com.ar/2011/02/lista-de-fuentes-instaladas-por-defecto.html

**Primer autor** Raul Andres Ortiz Fuentes, 112508, Aplicaciones Web Avanzadas.