Aproximación a JavaScript

Aproximation to JavaScript

Autor 1: Camilo Eduardo Muñoz Albornoz
Departamento o Escuela, Universidad, Ciudad, País
Correo-e: ejemplo@org.es

***Resumen*— Se realizará una aproximación al lenguaje de JavaScript**

***Palabras clave*— HTML, interfaz, programación,**

***Abstract*— An approximation to the JavaScript language will be made**

***Key Word* —HTML, interface, programming,.**

I. INTRODUCCIÓN

Cuando se creó JavaScript, inicialmente tenía otro nombre: "LiveScript". Pero Java era muy popular en ese momento, por lo que se decidió que posicionar un nuevo lenguaje como un "hermano menor" de Java ayudaría.

Pero a medida que evolucionaba, JavaScript se convirtió en un lenguaje totalmente independiente con su propia especificación llamada ECMAScript , y ahora no tiene ninguna relación con Java.

II. CONTENIDO

JavaScript es un robusto lenguaje de programación que puede ser aplicado a un documento HTML y usado para crear interactividad dinámica en los sitios web. Fue inventado por Brendan Eich, co-fundador del proyecto Mozilla, Mozilla Foundation y la Corporación Mozilla .

Puedes hacer casi cualquier cosa con JavaScript. Puedes empezar con pequeñas cosas como carruseles, galerías de imágenes, diseños fluctuantes, y respuestas a las pulsaciones de botones. Con más experiencia, serás capaz de crear juegos, animaciones 2D y gráficos 3D, aplicaciones integradas basadas en bases de datos, ¡y mucho más!.

JavaScript por si solo es bastante compacto, aunque muy flexible, y los desarrolladores han escrito gran cantidad de herramientas encima del núcleo del lenguaje JavaScript, desbloqueando una gran cantidad de funcionalidad adicional con un mínimo esfuerzo. Esto incluye:[1]

- Interfaces de Programación de Aplicaciones del Navegador (APIs) — APIs construidas dentro de los navegadores que ofrecen funcionalidades como crear dinámicamente contenido HTML y establecer estilos CSS, hasta capturar y manipular un vídeo desde la cámara web del usuario, o generar gráficos 3D y muestras de sonido.
- APIs de Terceros, que permiten a los desarrolladores incorporar funcionalidades en sus sitios de otros proveedores de contenidos como Twitter o Facebook.
- Marcos de trabajo y librerías de terceros que puedes aplicar a tu HTML para que puedas construir y publicar rápidamente sitios y aplicaciones.

Historia

El desarrollo de JavaScript comenzó en 1995 en Netscape Communications, los creadores del navegador Netscape. Se dieron cuenta de que añadir un "lenguaje pegajoso" para mejorar la experiencia del usuario de la web aumentaría la aceptación del usuario. Así que trajeron a Brendan Eich para que incrustara el lenguaje de programación de esquemas. Sin embargo, como Java era, en ese momento, el nuevo lenguaje de la web, decidieron acercar el lenguaje en sintaxis a Java. El resultado fue JavaScript, con características de Scheme, la orientación a objetos de SmallTalk y la sintaxis de Java. La primera versión de este lenguaje fue nombrada Mocha en mayo de 1995, renombrada a LiveScript en septiembre de 1995, y de nuevo a JavaScript en diciembre de 1995.

En 1996, JavaScript fue enviado a ECMA International para su finalización como especificación estándar. En junio de 1997, la primera especificación oficial de la lengua se publicó como ECMA-262. La última versión del lenguaje es ECMAScript 2017 que fue lanzado en junio de 2017.[2]

¿Para que sirve?

JavaScript es un intérprete de lenguaje de programación completo incrustado en el navegador web. Puede hacer cualquier cosa en JavaScript que un lenguaje normal como Java permita. Estos incluyen, por ejemplo:

- Declarar variables
- Almacenar y recuperar valores
- Definir e invocar funciones
- Defina sus propias clases

Fecha de Recepción: (Letra Times New Roman de 8 puntos)
Fecha de Aceptación: Dejar en blanco

- Cargar y utilizar módulos externos
- Escribir manejadores de eventos que respondan a eventos del usuario y otros eventos

El navegador carga una página web, analiza el HTML y crea lo que se conoce como un Modelo de Objeto de Documento (DOM) a partir de los contenidos. El DOM presenta una vista en vivo de la página web a su código JavaScript. El código puede entonces hacer actualizaciones al DOM y presentarlo instantáneamente al usuario. El navegador también le permite registrar el código para ser notificado de eventos de la interfaz de usuario como el movimiento del ratón, el clic de los botones, etc. Usando todas estas facilidades, usted puede construir aplicaciones pequeñas (y no tan pequeñas) para servir cualquier propósito que usted elija.

¿Como Funciona?

Cuando el navegador carga una página web, el analizador HTML comienza a analizar el código HTML y a crear el DOM. Cada vez que el analizador encuentra una directiva CSS o JavaScript (en línea o cargada externamente), se entrega al analizador CSS o al motor JavaScript según sea necesario. El motor JavaScript carga archivos JavaScript externos y código en línea, pero no ejecuta el código inmediatamente. Espera a que se complete el análisis HTML y CSS. Una vez hecho esto, el JavaScript se ejecuta en el orden en que se encuentran en la página web: se definen las variables y funciones, se ejecutan las invocaciones de funciones, se activan los manejadores de eventos, etc. Estas actividades hacen que el DOM sea actualizado por el JavaScript y que el navegador lo muestre instantáneamente.[2]

Sintaxis de JavaScript

La sintaxis de un lenguaje de programación se define como el conjunto de reglas que deben seguirse al escribir el código fuente de los programas para considerarse como correctos para ese lenguaje de programación.

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:

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.

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.

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 (;).

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.[3]

Comprensión de declaraciones, nombres de variables, espacios en blanco y otra sintaxis básica de JavaScript.[4]

Una declaración de variable simple

```
var foo = 'hello world';
```

El espacio en blanco no tiene ningún significado fuera de las comillas

```
var foo =        'hello world';
```

Los paréntesis indican precedencia.

```
2 * 3 + 5;   // retorna 11; multiplicacion pasa primero
2 * (3 + 5);  // retorna 16; suma pasa primer
```

Las pestañas mejoran la legibilidad, pero no tienen un significado especial

```
var foo = function() {
console.log('hello');
};
```

Operaciones en números y cuerdas

En JavaScript, los números y las cadenas de vez en cuando se comportarán de una manera que no podrías esperar.

Adición vs concatenación

```
var foo = 1;
var bar = '2';
console.log(foo + bar);  // 12. uh oh
```

Obligando a una cadena a actuar como un número

```
var foo = 1;
var bar = '2';
```

```
// coaccionar la cadena a un número
console.log(foo + Number(bar));
```

El constructor de números, cuando se le llama como una función (como arriba) tendrá el efecto de convertir su argumento en un número. También puede utilizar el operador Unary Plus, que hace lo mismo:

Forzar una cadena para que actúe como un número (utilizando el operador unary-plus)

```
console.log(foo + +bar);
```

Operadores lógicos

Los operadores lógicos le permiten evaluar una serie de operandos utilizando las operaciones AND y OR.

Operadores lógicos AND y OR

```
var foo = 1;
var bar = 0;
var baz = 2;

foo || bar;   // retorna 1, lo cual es true
bar || foo;   // returns 1, lo cual es true

foo && bar;   // retorna 0, lo cual es false
foo && baz;   // retorna 2, lo cual es true
baz && foo;   // retorna 1, lo cual es true
```

Aunque no quede claro en el ejemplo, el (||) operador devuelve el valor del primer operando de verdad o, en los casos en que ninguno de los operandos es verdadero, devolverá el último de los dos operandos. El (&&) operador devuelve el valor del primer operando falso, o el valor del último operando si ambos operandos son verdaderos.

Operadores de comparación

Los operadores de comparación le permiten probar si los valores son equivalentes o si los valores son idénticos.

Operadores de comparación

```
var foo = 1;
var bar = 0;
var baz = '1';
var bim = 2;

foo == bar;   // retorna false
foo != bar;   // retorna true
foo == baz;   // retorna true
foo === baz;          // retorna false
foo !== baz;          // retorna true
foo === parseInt(baz);   // retorna true

foo > bim;    // retorna false
bim > baz;    // retorna true
foo <= baz;   // retorna true
```

Codigo condicional

A veces solo desea ejecutar un bloque de código bajo ciertas condiciones. El control de flujo (vía if y else bloques) le permite ejecutar código solo bajo ciertas condiciones.

Control de flujo

```
var foo = true;
var bar = false;

if (bar) {
// this code will never run
console.log('hello!');
}

if (bar) {
// el codigo no funcionara
} else {
if (foo) {
// el codigo funcionara
} else {
// el codigo no funcionaria si foo y bar fueran falsas
}
}
```

Ejemplo[5]

Tabla de multiplicación simple que le pida al usuario el número de filas y columnas que quiere.

```
<html>
<head>
  <title>Multiplication Table</title>
  <script type="text/javascript">
    var rows = prompt("How many rows for your multiplication table?");
    var cols = prompt("How many columns for your multiplication table?");
    if(rows == "" || rows == null)
      rows = 10;
    if(cols== "" || cols== null)
      cols = 10;
    createTable(rows, cols);
    function createTable(rows, cols)
    {
```

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

```
    var j=1;
    var output = "<table border='1' width='500' cellspacing='0'cellpadding='5'>";
    for(i=1;i<=rows;i++)
    {
    output = output + "<tr>";
      while(j<=cols)
      {
      output = output + "<td>" + i*j + "</td>";
      j = j+1;
     }
      output = output + "</tr>";
      j = 1;
   }
   output = output + "</table>";
   document.write(output);
   }
  </script>
</head>
<body>
</body>
</html>
```

Ejemplo si el usuario digita 5 filas y 5 columnas

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

III. CONCLUSIONES

- JavaScript se creó inicialmente como un lenguaje solo para el navegador, pero ahora también se usa en muchos otros entornos.
- Hoy en día, JavaScript tiene una posición única como el lenguaje de navegador más ampliamente adoptado con integración completa con HTML / CSS.
- Hay muchos idiomas que se "transfieren" a JavaScript y proporcionan ciertas funciones. Se recomienda echarles un vistazo, al menos brevemente, después de dominar JavaScript.

REFERENCIAS

Fundamentos de JavaScript [Online]. Available: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics

J. Sridhar. (2017, Oct.). What Is JavaScript and How Does It Work? [Online]. Available: https://www.makeuseof.com/tag/what-is-javascript/

Uniwebsidad. Introducción a JavaScript [Online]. Available: https://uniwebsidad.com/libros/javascript/capitulo-1/sintaxis

JavaScript Basics [Online]. Available: https://autotelicum.github.io/Smooth-CoffeeScript/literate/js-intro.html#syntax-basics

Practical Code Examples using JavaScript [Online]. Available: https://www.guru99.com/practical-code-examples-using-javascript.html

(2019, May). Una introducción a SavaScripr [Online]. Available: https://javascript.info/intro