Java Script, funcionamiento y utilidades

K. Peña, E. Simbaña, MJ. Vizuete

Resumen

En el presente artículo se muestran los resultados y conclusiones de nuestra investigación acerca de JavaScript, lo que es, sus componentes, reglas básicas de uso, los compiladores, variables, entradas por teclado, estructuras secuenciales y estructuras condicionales pertenecientes a este lenguaje.

Además, para un mejor entendimiento se desarrollan y explican dos aplicaciones en donde se evidencia la teoría investigada, junto con diagramas de flujo.

JavaScript en la elaboración de una página web este lenguaje constituye solamente una capa de tres que constituyen el proceso (Html y css) Este lenguaje de programación permite al desarrollador crear contenido novedoso y eficiente, controlar archivos de multimedia, crear imágenes animadas y muchas otras cosas más

I. INTRODUCCIÓN

El objeto de investigación de este artículo es comprender el funcionamiento del lenguaje de programación Java-Script y sus principales estructuras, investigando los usos de este lenguaje por medio de la abstracción de información, para así poder conocer las funcionalidades que tiene.
A partir de la investigación se busca explicar más a fondo los componentes de este lenguaje y algunas reglas que lo caracterizan, de tal manera que la información sea comprensible y detallada. También se busca desarrollar ejemplos básicos donde se evidencie el lenguaje ya antes mencionado.

A. ¿Qué es JavaScript?

JavaScript es un lenguaje utilizado para dotar de efectos y procesos dinámicos e "inteligentes" a documentos HTML. Un documento HTML viene siendo coloquialmente "una página web". Así, podemos decir que el lenguaje JavaScript sirve para ejecutar acciones rápidas y efectos animados en páginas web.

Las acciones controladas por JavaScript pueden ser el despliegue de un menú, hacer aparecer, desaparecer o cambiar texto e imágenes, realizar cálculos y mostrar resultados, mostrar mensajes de aviso (por ejemplo, si faltan datos en un formulario) y "efectos animados" en general.

Este lenguaje es principalmente utilizado por parte de programadores web para dar respuestas rápidas a las acciones del usuario sin necesidad de enviar la información de lo que ha hecho el usuario al servidor y esperar respuesta de éste (lo que haría más lento los procesos). El código JavaScript se carga al mismo tiempo que el código HTML en el navegador, y reside en el cliente (computador en el que nos encontramos), por lo que JavaScript sigue funcionando incluso aunque se produzca un corte en la conexión a internet (en este caso no podremos seguir navegando hacia otras direcciones web, pero sí podremos ejecutar procesos "locales" en nuestro computador para la página web en que nos encontráramos).[1]

B. Compiladores

1) Google Closure Compiler

No es realmente un compilador de código en el sentido normal, pero sí que se puede considerar un compilador porque compila, junta diferentes fragmentos de código y transforma de JavaScript a JavaScript mejorado.

Google Closure Compiler es una herramienta que ayuda a optimizar y encontrar errores en el código JavaScript. Que realiza las siguientes acciones:
-Junta diferentes ficheros de JavaScript en uno solo.
-Elimina contenido innecesario para la ejecución, como los comentarios.
-Detecta código inútil que no se usa.
-Detecta código que contiene errores.[2]

2) Angular.JS

Se trata de un framework MVC (Modelo Vista Controlador) de JavaScript de código abierto, desarrollado por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.
Al usar un patrón MVVM (model view view-model) separamos la lógica de la de diseño, pero mantenemos ambas partes conectadas (data binding). De manera que la capa visual no sabe lo que está pasando en la capa lógica, pero manteniendo control sobre el DOM (el cuerpo de la web) y actualizar su contenido como queramos. [3]

3) Node.JS

Es un entorno de tiempo de ejecución de JavaScript en tiempo real, que se encuentra de lado del servidor y utiliza un modelo asíncrono y dirigido por eventos. Incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript.
Se trata de una Máquina Virtual rápida y de gran calidad. Además, las capacidades de Node.js para I/O (Entrada/Salida) son realmente ligeras y potentes, dando al desarrollador la posibilidad de utilizar toda la I/O del sistema. Uno de sus puntos fuertes, es su capacidad de mantener muchas conexiones abiertas y esperando. [4]

C. Variables

Una variable es un espacio de memoria donde se almacena un dato y se puede guardar cualquier tipo de información necesaria para realizar las acciones de nuestros programas. [6]

D. Estructuras secuenciales

Son aquellas en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso**.** También se las puede identificar cuando en un problema solo participan operaciones, entradas y salidas.
En JavaScript, como no se puede indicar el tipo de la variable, se requiere mucho más cuidado cuando operamos con sus contenidos.

E. Estructura iterativa

1) Entrada repetitiva do while

La sentencia do/while es una estructura repetitiva, la cual se utiliza cuando conocemos de antemano que por lo menos una vez se ejecutará el bloque repetitivo, a diferencia del while que puede no ejecutar el bloque.

La condición de la estructura está abajo del bloque a repetir y finaliza la ejecución del bloque repetitivo cuando la condición retorna falso.

2) Entrada repetitiva For

Un bucle for se repite hasta que la condición especificada se evalúa como falsa. Cuando un bucle for se ejecuta, ocurre lo siguiente:

- La expresión de inicialización, si existe, se ejecuta. Esta expresión habitualmente inicializa uno o más contadores del bucle, pero la sintaxis permite una expresión con cualquier grado de complejidad. Esta expresión puede también declarar variables.
- Se evalúa la expresión condición. Si el valor de condición es verdadero, se ejecuta la sentencia del bucle. Si el valor de condición es falso, el bucle for finaliza. Si la expresión condición es omitida, la condición es asumida como verdadera.
- Se ejecuta la sentencia. Para ejecutar múltiples sentencias, use un bloque de sentencias ({ ... }) para agruparlas.
- Se ejecuta la expresión expresionIncremento, si hay una, y el control vuelve al paso 2. [7]

F. Entradas por teclado

En Informática, la "entrada" de un programa son los datos que llegan al programa desde el exterior. Actualmente, el origen más habitual es el teclado (Sintes,2018).

1) Prompt()

El método prompt () se utiliza para mostrar un cuadro de diálogo con un mensaje opcional que nos permite solicitar información al visitante de la página (Barrena,2016). A menudo se usa si el usuario desea ingresar un valor antes de ir a una página. Entonces devuelve una cadena que contiene el texto ingresado por el usuario, o nulo.

Sintaxis:

mensaje (mensaje, predeterminado)

El mensaje es una cadena de texto para mostrar al usuario. Se puede omitir si no hay nada que mostrar en la ventana de solicitud, es decir, es opcional. **Default** es una cadena que contiene el valor predeterminado que se muestra en el campo de entrada de texto. También es opcional. Ejemplo:

prompt("Indique su edad");

Si se prueba en el editor online (*entre etiquetas <script></script>*), saldrá un pop-up con el texto que se haya incluido y una casilla para introducir la información solicitada (datos numéricos o texto).Se puede incluir una respuesta por defecto. Esta se mostrará en la casilla y puede ser modificada por el visitante(Barrena,2017).[8]

G. Estructuras condicionales

Un script consiste en una lista de enunciados que se van ejecutando a medida que se cargan. Sin embargo, en ocasiones es necesario controlar el flujo de la ejecución estableciendo alternativas, es decir, que una serie de enunciados se ejecuten en algunas ocasiones y en otras no(Fernández,2007). Para permitir esto existen las estructuras condicionales.

1) if … else

medio de if se puede indicar una condición que, de cumplirse, permite la ejecución de uno o más enunciados.Por medio de else se puede establecer una alternativa, aunque su uso es opcional. La sintaxis de esta estructura es la siguiente:

```
if(condición){
    …enunciados a ejecutar si se cumple la condición…
}
else {
    …enunciados a ejecutar si NO se cumple la primera condición…
}
```

Ejemplo:

En un programa se desea lanzar una advertencia si el valor de una variable es mayor que 100, y otra si es mayor:

```
if (variable _a_ comparar>100){
    alert("El valor de la variable es mayor que 100");
} else {
    alert("El valor de la variable es menor o igual que 100");
}
```

Además, las instrucciones se pueden vincular:

```
if (variable _a_ comparar>100){
    alert("El valor de la variable es mayor que 100");
} else if (variable _a_ comparar==100){
    alert("El valor de la variable es exactamente 100");
} else {
    alert("El valor de la variable es menor que 100");
}
```

2) Switch

Por medio de switch se puede listar una serie de bloques de enunciados que se ejecuten dependiendo del valor de una variable [9]

II. DIAGRAMAS

Diagrama de flujo aplicación: “Adivina el número”

El programa inicia solicitando datos, una vez estos sean ingresados, se dirigirá a dos condiciones una en la que compara el numeroUsuario y el numeroCPU si son iguales imprimirá Has acertado si no lo son llegara a la otra condición donde me indicara si el número es mayor o menor y de esta manera hasta que logremos acertar y de esta manera sea el númeroUsuario igual al númeroCPU.

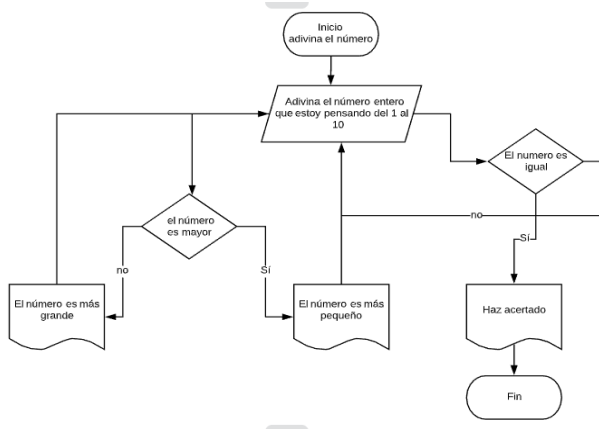
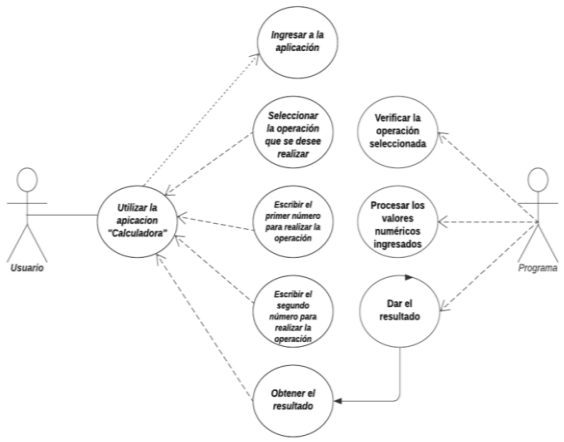


Diagrama casos de uso-clase de la aplicación "Calculadora"

En este diagrama se puede evidenciar de una manera más didáctica el funcionamiento de nuestra aplicación calculadora, mostrado desde el punto de vista de un usuario y del programa. Es así que el usuario desea usar esta aplicación y para ello debe cumplir con los pasos de ingresar, seleccionar la operación a realizar, escribir el primer y segundo número y posteriormente obtener un resultado; mientras tanto el programa debe verificar la operación seleccionada, procesar los valores y dar el resultado.



III. EXPLICACIÓN DE CÓDIGO FUENTE

A. Programa "Adivina número"

Primero el programa crea un número aleatorio del 1 al 10 mediante la función Math.floor((Math.random() *10) + 1); este número se compara con el valor ingresado por el usuario realizando una entrada por teclado mediante la función parseInt(prompt('Adivina el número que estoy pensando del 1 al 10'))
Si la comparación de estos dos números son distintos surgirá un alert mediante una condición if en donde si el número es mayor mostrará el mensaje 'el número es mayor' y si no lo es mostrara lo contrario.
El proceso está condicionado por un while que mantendrá un bucle hasta que el número generado aleatoriamente por el computador deje de ser distinto al del usuario y en este caso surgirá un alert con las palabras: 'has acertado'.

B. Programa "Calculadora"

Primero el programa solicita que el usuario elija la operación a ejecutar mediante una estructura HTML posicionando botones, del mismo modo, se condiciona al botón a ejecutar cierta función una vez se dé click en él por ejemplo :
<input type="button" class="boton rojo" value="suma" onclick="suma()" >
para la función suma una vez el usuario digite mediante la función prompt el primer y segundo número a operar esta crea una segunda función la cual realiza la operación denominada suma 2.

```
function suma(){
  numero1 = prompt('Introduce el primer numero entero');
  numero2 = prompt('Introduce el segundo numero entero');
  resultado = suma2(numero1,numero2);
  alert(resultado);
}
```

En la operación suma dos se lleva a cabo todo el proceso por el cual las dos variables van a ser sometidas para poder conseguir el resultado o valor que se mostrará mediante un alert.

```
function suma2(num1, num2){
  var valor;
  num1 = parseInt(num1);
  num2 = parseInt(num2);
  valor = num1 + num2;
  return(valor);
}
```

El proceso se repite con el resto de operaciones en donde al momento de llegar a operaciones más avanzadas como potencia o raíz cuadrada se hizo vital la utilización de dos funciones preestablecidas llamadas Math.pow y Math.sqrt respectivamente las cuales permiten el retorno del valor respuesta.

IV. METODOLOGÍA

La revisión bibliográfica fue el principal método empleado en la realización de este artículo, ya que nos centramos en el análisis de material bibliográfico y de consulta de información respecto al tema estudiado, lo que nos permitió afianzar los conceptos necesarios para el desarrollo del código de los programas. La investigación fue posible por

medio de herramientas utilizadas, así como el programa Atom, el cual nos sirvió para poder programar el código, el navegador en el cual ejecutamos los programas, la herramienta Github y del mismo modo la información recolectada nos permitió encontrar las múltiples estructuras a estudiar.

V. RESULTADOS

A partir del análisis y revisión de información se logró desarrollar y ejecutar dos programas, en los cuales se evidencian las principales estructuras del lenguaje de programación JavaScript, así como sus variables, entradas por teclado y sintaxis.

El editor de código fuente o id en múltiples ocasiones puede representar una ayuda para el programador puesto que cuentan con múltiples facilidades para hacer el desarrollo más ameno, por tal motivo se escogió Atom este es un editor de código fuente que cuenta con un autocompletado inteligente y en muchas ocasiones agiliza el trabajo.

JavaScript es un lenguaje de programación interpretado, por tal motivo, su estructura amerita de un lenguaje de etiquetado como HTML y CSS para poder ofrecer de una interfaz más amigable al usuario, se recomienda tener una idea base de un lenguaje HTML o CSS para poder realizar trabajos de mayor envergadura.

VI. CONCLUSIÓN

JavaScript es un lenguaje de programación que se emplea en la creación de páginas web, y que funciona integrándose con el HTML, además no necesita de compiladores ya que son los navegadores los que se encargan de leer el código.

Las estructuras condicionales de JavaScript como if-else y switch, permiten ejecutar ciertas partes del código dependiendo si se cumplen o no con las condiciones establecidas, estos conceptos se fueron comprendiendo a través del desarrollo de los programas.

Se cumplió nuestro objetivo acerca de desarrollar ejemplos básicos donde se evidencie el lenguaje JavaScript, esto debido a las investigaciones realizadas referentes a este lenguaje, su funcionamiento y características. De esta manera nuestras dos aplicaciones, tanto “calculadora” como “adivina número” se encuentran con un buen funcionamiento y listas para ser usadas.

REFERENCIA

[1] Pérez, J. E. (2019). introduccion a JavaScript.

[2] Compiler, C. (2017). Google Developers.

[3] Green, B., & Seshadri, S. (2013). AngularJS. " O'Reilly Media, Inc.".

[4] Gómez, D., Flores, Á., & Ureta, R. (2018). Programación Node. JS. Caribeña de Ciencias Sociales, (mayo).

[5] Pérez, J. E. (2019). introducción a JavaScript.

[6] Mohedano, J., Saiz, J. M., & Román, P. S. (2012). Iniciación a javascript. Ministerio de Educación

[7] Singh,P.(2015).Repensar los bucles de JavaScript como combinadores.

[8] Castillo, A. A. (2017). Curso de Programación Web: JavaScript, Ajax y jQuery. IT Campus Academy.

[9] Navarrete, T. (2006). El lenguaje JavaScript. *Argentina*.