## Qué es Javascript

Javascript es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web. Con Javascript podemos crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador.

Javascript es el siguiente paso, después del HTML, que puede dar un programador de la web que decida mejorar sus páginas y la potencia de sus proyectos. Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, a veces con ligereza. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica.

Entre las acciones típicas que se pueden realizar en Javascript tenemos dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, javascript nos permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

Javascript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Toda esta potencia de Javascript se pone a disposición del programador, que se convierte en el verdadero dueño y controlador de cada cosa que ocurre en la página.

**Introducción a algunas funciones en JavaScript**

**1. Función alert**Esta función es un método del objeto Window y es una de las más utilizadas al momento de iniciarnos en JavaScript, pues es la encargada de mostrar una pequeña ventana de aviso en la pantalla, así, si se requiere que aparezca un mensaje cuando ocurra determinada acción en el programa, podemos hacer uso de esta función. La función ***alert***recibe como parámetro el mensaje que se debe mostrar en la ventana.
Sintaxis:

**alert**(mensaje a mostrar)

A continuación se muestra un ejemplo donde aparece en la pantalla una pequeña ventana con un mensaje de saludo:

<html>

<head>
<title> Ejemplo alert </title>
<script language="javascript" type="text/javascript">

alert ( "Hola, como estás" )

</script>
</head>

<body>
</body>

</html>

 **2. Función write**Esta función es un método del objeto **document** y lo que hace es escribir en la página el texto que se ingresa como parámetro.
Sintaxis:

document.**write**(mensaje)

Ejemplo:

<html>

<head>
<title>Ejemplo write </title>
<script language="javascript" type="text/javascript">
document.write( "Hola, nos alegra que estés por aquí... ")
</script>
</head>
<body>
</body>
</html>

 **3. Función prompt**Al igual que la función **alert**, la función**prompt**es también un método del objeto Window. Esta función se utiliza cuando el usuario ingresa datos por medio del teclado. Con esta función aparece una ventana en la pantalla, con un espacio para el valor que se debe ingresar y un botón **aceptar** para que la información sea guardada. Esta función recibe dos parámetros, el primero es el mensaje que se muestra en la ventana y el segundo es el valor inicial del área de texto.

Sintaxis:

variable = **prompt**(mensaje, valor inicial)

Ejemplo:

<html>

<head>
<title>Ejemplo prompt </title>
<script language="javascript" type="text/javascript">

var nombre = prompt ("Ingrese su nombre", "")
document.write( "Hola "+ nombre)

</script>
</head>

<body>
</body>

</html>

<http://aprendeenlinea.udea.edu.co/lms/moodle/mod/resource/view.php?inpopup=true&id=47109>

http://librosweb.es/javascript/

# 3.1. Variables

Las variables en los lenguajes de programación siguen una lógica similar a las variables utilizadas en otros ámbitos como las matemáticas. Una variable es un elemento que se emplea para almacenar y hacer referencia a otro valor. Gracias a las variables es posible crear "programas genéricos", es decir, programas que funcionan siempre igual independientemente de los valores concretos utilizados.

De la misma forma que si en Matemáticas no existieran las variables no se podrían definir las ecuaciones y fórmulas, en programación no se podrían hacer programas realmente útiles sin las variables.

Si no existieran variables, un programa que suma dos números podría escribirse como:

resultado = 3 + 1

El programa anterior es tan poco útil que sólo sirve para el caso en el que el primer número de la suma sea el 3 y el segundo número sea el 1. En cualquier otro caso, el programa obtiene un resultado incorrecto.

Sin embargo, el programa se puede rehacer de la siguiente manera utilizando variables para almacenar y referirse a cada número:

numero\_1 = 3

numero\_2 = 1

resultado = numero\_1 + numero\_2

Los elementos numero\_1 y numero\_2 son **variables** que almacenan los valores que utiliza el programa. El resultado se calcula siempre en función del valor almacenado por las variables, por lo que este programa funciona correctamente para cualquier par de números indicado. Si se modifica el valor de las variables numero\_1 y numero\_2, el programa sigue funcionando correctamente.

Las variables en JavaScript se crean mediante la palabra reservada var. De esta forma, el ejemplo anterior se puede realizar en JavaScript de la siguiente manera:

var numero\_1 = 3;

var numero\_2 = 1;

var resultado = numero\_1 + numero\_2;

La palabra reservada var solamente se debe indicar al definir por primera vez la variable, lo que se denomina **declarar** una variable. Cuando se utilizan las variables en el resto de instrucciones del script, solamente es necesario indicar su nombre. En otras palabras, en el ejemplo anterior sería un error indicar lo siguiente:

var numero\_1 = 3;

var numero\_2 = 1;

var resultado = var numero\_1 + var numero\_2;

Si cuando se declara una variable se le asigna también un valor, se dice que la variable ha sido**inicializada**. En JavaScript no es obligatorio inicializar las variables, ya que se pueden declarar por una parte y asignarles un valor posteriormente. Por tanto, el ejemplo anterior se puede rehacer de la siguiente manera:

var numero\_1;

var numero\_2;

numero\_1 = 3;

numero\_2 = 1;

var resultado = numero\_1 + numero\_2;

Una de las características más sorprendentes de JavaSript para los programadores habituados a otros lenguajes de programación es que tampoco es necesario declarar las variables. En otras palabras, se pueden utilizar variables que no se han definido anteriormente mediante la palabra reservada var. El ejemplo anterior también es correcto en JavaScript de la siguiente forma:

var numero\_1 = 3;

var numero\_2 = 1;

resultado = numero\_1 + numero\_2;

La variable resultado no está declarada, por lo que JavaScript crea una variable global (más adelante se verán las diferencias entre variables locales y globales) y le asigna el valor correspondiente. De la misma forma, también sería correcto el siguiente código:

numero\_1 = 3;

numero\_2 = 1;

resultado = numero\_1 + numero\_2;

En cualquier caso, se recomienda declarar todas las variables que se vayan a utilizar.

El nombre de una variable también se conoce como **identificador** y debe cumplir las siguientes normas:

* Sólo puede estar formado por letras, números y los símbolos $ (dólar) y \_ (guión bajo).
* El primer carácter no puede ser un número.

Por tanto, las siguientes variables tienen nombres correctos:

var $numero1;

var \_$letra;

var $$$otroNumero;

var $\_a\_\_$4;

Sin embargo, las siguientes variables tienen identificadores incorrectos:

var 1numero; // Empieza por un número

var numero;1\_123; // Contiene un carácter ";"

#

# 3.4. Estructuras de control de flujo

Los programas que se pueden realizar utilizando solamente variables y operadores son una simple sucesión lineal de instrucciones básicas.

Sin embargo, no se pueden realizar programas que muestren un mensaje si el valor de una variable es igual a un valor determinado y no muestren el mensaje en el resto de casos. Tampoco se puede repetir de forma eficiente una misma instrucción, como por ejemplo sumar un determinado valor a todos los elementos de un array.

Para realizar este tipo de programas son necesarias las **estructuras de control de flujo**, que son instrucciones del tipo "si se cumple esta condición, hazlo; si no se cumple, haz esto otro". También existen instrucciones del tipo "repite esto mientras se cumpla esta condición".

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.

### 3.4.1. 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 (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro de {...}. Si la condición no se cumple (es decir, si su valor es false) no se ejecuta ninguna instrucción contenida en {...} y el programa continúa ejecutando el resto de instrucciones del script.

Ejemplo:

var mostrarMensaje = true;

if(mostrarMensaje) {

 alert("Hola Mundo");

}

En el ejemplo anterior, el mensaje sí que se muestra al usuario ya que la variable mostrarMensaje tiene un valor de true y por tanto, el programa entra dentro del bloque de instrucciones del if.

El ejemplo se podría reescribir también como:

var mostrarMensaje = true;

if(mostrarMensaje == true) {

 alert("Hola Mundo");

}

En este caso, la condición es una comparación entre el valor de la variable mostrarMensaje y el valor true. Como los dos valores coinciden, la igualdad se cumple y por tanto la condición es cierta, su valor es true y se ejecutan las instrucciones contenidas en ese bloque del if.

La comparación del ejemplo anterior suele ser el origen de muchos errores de programación, al confundir los operadores == y =. Las comparaciones siempre se realizan con el operador ==, ya que el operador = solamente asigna valores:

var mostrarMensaje = true;

// Se comparan los dos valores

if(mostrarMensaje == false) {

 ...

}

// Error - Se asigna el valor "false" a la variable

if(mostrarMensaje = false) {

 ...

}

La condición que controla el if() puede combinar los diferentes operadores lógicos y relacionales mostrados anteriormente:

var mostrado = false;

if(!mostrado) {

 alert("Es la primera vez que se muestra el mensaje");

}

Los operadores AND y OR permiten encadenar varias condiciones simples para construir condiciones complejas:

var mostrado = false;

var usuarioPermiteMensajes = true;

if(!mostrado && usuarioPermiteMensajes) {

 alert("Es la primera vez que se muestra el mensaje");

}

La condición anterior está formada por una operación AND sobre dos variables. A su vez, a la primera variable se le aplica el operador de negación antes de realizar la operación AND. De esta forma, como el valor de mostrado es false, el valor !mostrado sería true. Como la variable usuarioPermiteMensajes vale true, el resultado de !mostrado && usuarioPermiteMensajes sería igual a true && true, por lo que el resultado final de la condición del if() sería true y por tanto, se ejecutan las instrucciones que se encuentran dentro del bloque del if().

### Estructura if...else

En ocasiones, las decisiones que se deben realizar no son del tipo "si se cumple la condición, hazlo; si no se cumple, no hagas nada". Normalmente las condiciones suelen ser del tipo "si se cumple esta condición, hazlo; si no se cumple, haz esto otro".

Para este segundo tipo de decisiones, existe una variante de la estructura if llamada if...else. Su definición formal es la siguiente:

if(condicion) {

 ...

}

else {

 ...

}

Si la condición se cumple (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro del if(). Si la condición no se cumple (es decir, si su valor es false) se ejecutan todas las instrucciones contenidas en else { }. Ejemplo:

var edad = 18;

if(edad >= 18) {

 alert("Eres mayor de edad");

}

else {

 alert("Todavía eres menor de edad");

}

Si el valor de la variable edad es mayor o igual que el valor numérico 18, la condición del if() se cumple y por tanto, se ejecutan sus instrucciones y se muestra el mensaje "Eres mayor de edad". Sin embargo, cuando el valor de la variable edad no es igual o mayor que 18, la condición del if() no se cumple, por lo que automáticamente se ejecutan todas las instrucciones del bloque else { }. En este caso, se mostraría el mensaje "Todavía eres menor de edad".

El siguiente ejemplo compara variables de tipo cadena de texto:

var nombre = "";

if(nombre == "") {

 alert("Aún no nos has dicho tu nombre");

}

else {

 alert("Hemos guardado tu nombre");

}

La condición del if() anterior se construye mediante el operador ==, que es el que se emplea para comparar dos valores (no confundir con el operador = que se utiliza para asignar valores). En el ejemplo anterior, si la cadena de texto almacenada en la variable nombre es vacía (es decir, es igual a "") se muestra el mensaje definido en el if(). En otro caso, se muestra el mensaje definido en el bloque else { }.

La estructura if...else se puede encadenar para realizar varias comprobaciones seguidas:

if(edad < 12) {

 alert("Todavía eres muy pequeño");

}

else if(edad < 19) {

 alert("Eres un adolescente");

}

else if(edad < 35) {

 alert("Aun sigues siendo joven");

}

else {

 alert("Piensa en cuidarte un poco más");

}

No es obligatorio que la combinación de estructuras if...else acabe con la instrucción else, ya que puede terminar con una instrucción de tipo else if().

http://librosweb.es/javascript/capitulo\_3/estructuras\_de\_control\_de\_flujo.html

**Ciclos en JavaScript**

A menudo cuando se escribe código, se desea que el mismo bloque de código se ejecute una y otra, y otra vez en forma consecutiva. En vez de agregar muchas líneas casi iguales de código, en el script se pueden utilizar los ciclos para desempeñar una tarea como ésta.

En Javascript hay dos tipos distintos de ciclos (o loops):

* **for** – Ciclos a través de un bloque de código un número especificado de veces.
* **while** – Itera a través de un bloque de código mientras una condición específica es cierta.

**El ciclo for**

El ciclo for se utiliza cuando se conoce cuantas veces debe correr el script.

**Sintaxis**

|  |
| --- |
| For (var = varlorInicial; var <= valorFinal; var = var+incremento){ //código a ser ejecutado} |

**Ejemplo**

Explicación: El siguiente ejemplo define un ciclo que empieza con i = 0. El ciclo continuará corriendo siempre que i sea menor o igual a 10, i será incrementado en 1 cada iteración.

**Nota:** El parámetro de incremento podría ser también negativo, y el <= podría ser cualquier expresión de comparación.

|  |
| --- |
| <html><body><script type="text/javascript">var i=0for (i=0;i<=10;i++){ document.write("El número es " + i) document.write("<br />")}</script></body></html> |

 **El ciclo while**

El ciclo while se utiliza cuando se quiere que el ciclo se ejecute mientras que una condición especificada sea cierta.

|  |
| --- |
|   while (var <= valorFinal){  //código a ser ejecutado }  |

**Nota:** el <= puede ser cualquier expresión de comparación

  **Ejemplo**

Explicación: El siguiente ejemplo define un ciclo que empieza con i = 0. El ciclo continuará siempre y cuando i sea menor o igual a 10. i se incrementará en 1 cada iteración.

|  |
| --- |
|   <html> <body> <script type="text/javascript"> var i=0 while (i<=10){  document.write("El número es " + i)  document.write("<br />")  i=i+1 } </script> </body> </html>  |

 **El ciclo do…while**

  El ciclo do…while es una variante del ciclo while. Este ciclo siempre ejecuta el bloque de código al menos una vez, y entonces se repetirá siempre y cuando la condición especificada sea cierta.

Este ciclo siempre ejecutará al menos una vez, aún si la condición es falsa, porque el código es ejecutado antes de que la condición sea evaluada.

|  |
| --- |
|   do {  //código a ser ejecutado } while (var <= valorFinal)  |

**Ejemplo**

|  |
| --- |
| <html> <body> <script type="text/javascript"> var i=0 do{  document.write("El número es " + i)  document.write("<br />")  i=i+1 }while (i<0) </script> </body> </html>  |

**Declaraciones break y continue**

Hay dos declaraciones especiales que pueden ser usadas dentro de ciclos: break y continue.

**Break**

El comando break detendrá el ciclo y continuará ejecutando el código que sigue después del bloque del ciclo (si hay más código)

  **Ejemplo**

|  |
| --- |
| <html> <body> <script type="text/javascript"> var i = 0 for (i = 0; i <= 10; i++){  if (i==3){  break  }  document.write("El número es " + i)  document.write("<br />") } </script> </body> </html>  |

**Continue**

El comando continue obliga a que el ciclo continúe con el próximo valor.

  **Ejemplo**

|  |
| --- |
| <html> <body> <script type="text/javascript"> var i=0 for (i=0;i<=10;i++){  if (i==3){  continue  }  document.write("El número es " + i)  document.write("<br />") } </script> </body> </html>  |

**Ciclo for…in**

La declaración for…in se utiliza para iterar a través de los elementos de un array o a través de las propiedades de un objeto.

El código en el cuerpo (body) del ciclo for…in es ejecutado una vez por cada elemento o propiedad.

**Sintaxis**

|  |
| --- |
| for (variable in objeto){ //código a ser ejecutado} |

La variable argumento puede ser una variable, un arreglo o la propiedad de un elemento.

**Ejemplo**

Utilizando for…in para iterar sobre un arreglo:

|  |
| --- |
| <html><body><script type="text/javascript">var xvar misCarros = new Array()misCarros[0] = "Saab"misCarros[1] = "Volvo"misCarros[2] = "BMW" for (x in misCarros){ document.write(misCarros[x] + "<br />")}</script></body></html> |

**Sentencia Switch**

La sentencia switch permite que se seleccione un grupo de sentencias entre varias posibles.

La sintaxis de esta sentencia es:

|  |
| --- |
| **switch** ( Expresión ){ case Valor 1: Instrucción o bloque de instrucciones;[break;]case Valor 2: Instrucción o bloque de instrucciones;[break;]case Valor 3: Instrucción o bloque de instrucciones;[break;]...case Valor N: Instrucción o bloque de instrucciones;[break;][default:]Instrucción o bloque de instrucciones;}  |

Los prámetros utilizados en la sentencia switch son:

* Expresión: Es una variable que es comparada con cada uno de los case.
* Case **X** : Identificador usado para comparar la expresión.
* Instrucciones: Instrucción o bloque de instrucciones que son ejecutadas, una vez se identifica a qué case pertenece Expresión.

El funcionamiento general de la sentecia switch se describe a continuación: La Expresión entre paréntesis del **switch** debe ser entera. Su resultado se comparará con los distintos valores del case. Si coincide con uno de ellos se pasará a la instrucción siguiente al case con dicho valor y se seguirán ejecutando las instrucciones consecutivas hasta encontrar una instrucción break o alcanzar el cierra llaves del switch. En caso de que el resultado de la expresión no coincida con ningún valor se pasará la ejecución a la instrucción siguiente de la etiqueta default, si la hubiera, y se continuará como un case. Los valores en los case pueden ser una expresión constante. No puede haber dos case con el mismo valor.

Ejemplo:

|  |
| --- |
| <!-- Ejemplo Sentencia Switch--><html><head><title>Ejemplo Javascript</title></head><body><script language="JavaScript"> for(var Frutas=1; Frutas<=5; Frutas++){switch (Frutas) { case 1 : document.write("Las Naranjas cuestan $100."); break; case 2 : document.write("Las Manzanas cuestan $200."); break; case 3 : document.write("Las Bananas cuestan $50."); break; case 4 : document.write("Las Cerezas cuestan $500."); break; default : document.write("Lo siento, pero esa Fruta no la tenemos."); } document.write("<br>");}document.write("<br>");document.write("Desea comprar algo más?");</script></body></html>  |

La sentencia break dentro de un switche, está asociada con cada case y su función es asegurar que el programa salte fuera del switche una vez que la expresión haya coincidido con un case y se hayan ejecutado las instrucciones correspondientes. Así, el break hará que se ejecute la instrucción siguiente al switche. Si la sentencia break es omitida, el programa continua la ejecución en la siguiente instrucción dentro del switch.

Calcular la edad

<HTML>

<body>

<SCRIPT LANGUAGE='JavaScript'>

var nombre = parseInt(prompt('Ingrese nombre', ''));

var edad= parseInt(prompt('Ingrese la edad', ''));

if(edad>=18)

{

document.write("es mayor de edad");

}

else

{

document.write("es enor de edad");

}

</SCRIPT>

</body>

</HTML>

Casos

<!-- Ejemplo Sentencia Switch-->

<html>

<head>

<title>Ejemplo Javascript</title>

</head>

<body>

<script>

for(var Frutas=1; Frutas<=5; Frutas++)

{

switch (Frutas)

{

case 1 : document.write("Las Naranjas cuestan $100.");

break;

case 2 : document.write("Las Manzanas cuestan $200.");

break;

case 3 : document.write("Las Bananas cuestan $50.");

break;

case 4 : document.write("Las Cerezas cuestan $500.");

break;

default : document.write("Lo siento, pero esa Fruta no la tenemos.");

}

document.write("<br>");

}

document.write("<br>");

document.write("Desea comprar algo más?");

</script>

</body>

</html>

Ciclo for…

<html>

<body>

<script type="text/javascript">

var num=prompt('Cuantas veces quieres que lo repita','');

for(i=0;i<=num;i++){

 document.write("El número es " + i)

 document.write("<br />")

}

</script>

</body>

</html>

Otro for

<html>

<body>

<script type="text/javascript">

var i=0

for (i=0;i<=10;i++){

 document.write("El número es " + i)

 document.write("<br />")

}

</script>

</body>

</html>

Mas for

<html>

<body>

<script type="text/javascript">

var num=prompt('Cuantas veces quieres que lo repita','');

for(i=0;i<=num;i++){

 document.write("El número es " + i)

 document.write("<br />")

}

</script>

</body>

</html>

Do while

<html>

<body>

<script type="text/javascript">

var i=0

do{

 document.write("El número es " + i)

 document.write("<br />")

 i=i+1

}while (i<3)

</script>

</body>

</html>

Mientras

<html>

<body>

<script type="text/javascript">

var i=0

while (i<=10){

 document.write("El número es " + i)

 document.write("<br />")

 i=i+1

}

</script>

http://aprendeenlinea.udea.edu.co/lms/ova/mod/resource/view.php?id=1615

</body>

</html>

Sumar dos números

<HTML>

<body>

<SCRIPT LANGUAGE='JavaScript'>

var num1 = parseInt(prompt('Ingrese el número 1', ''));

var num2= parseInt(prompt('Ingrese el numero 2', ''));

var suma=num1+num2;

document.write("La suma es: ", suma);

</SCRIPT>

</body>

</HTML>