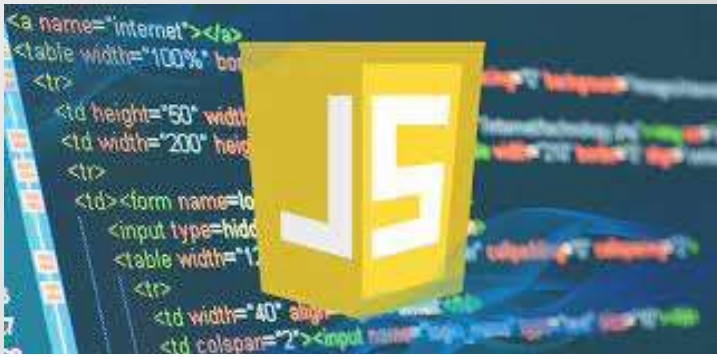
JavaScript

Parte I

Diseño de Sitios Web



¿Para qué necesitamos JavaScript?

- HTML y CSS le dan instrucciones al navegador de como mostrar la información, pero la interacción posible con el usuario es limitada.
- Si queremos alterar elementos de manera dinámica, según la interacción con el usuario necesitamos de lenguajes de script o de programación para la web
- Javascript es un lenguaje de programación que permite procesar y manipular documentos.

Javascript

- Cuando el navegador encuentra código javascript lo ejecuta y los cambios se muestran de manera inmediata en el navegador.
- Posee una sintaxis propia y un conjunto de instrucciones que se ejecutan de manera secuencial.
- Posee además un conjunto de funciones predefinidas y de API que facilitan la programación de ciertos componentes.
- Es el lenguaje del lado del cliente mas utilizado en las paginas web, en el cual se basan la mayoría de los frameworks de desarrollo.

¿Cómo introducir código Javascript?

- Hay tres técnicas para esto:
  - En íinea, es decir dentro de la etiqueta HMTL. Como por ejemplo en un botón para detectar el click del usuario (Ejemplo 2)
  - En el documento utilizando la etiqueta <Script>. Permitiendo crear bloques de código. (ejemplo 3)
  - Incrustar el código javascript vinculando un archivo externo con extensión js al documento HTML. Permitiendo cerar bloques extensos de código. (ejemplo 4)

Primer Javascript

- Colocar en **línea** a un elemento html el manejador que permite detectar el click
- Esto se puede colocar en cualquier elemento visible al cliente, dentro de la etiqueta HMTL

……

<section>

Clic aquí

 Aca no puede hacer click

</section>

Listado de Eventos 1 – Mouse

Aplicable a toda área que pueda ser clickeada por el usuario

- **onclick**—Este atributo responde al evento click. El evento se ejecuta cuando el usuario hace clic con el botón izquierdo del ratón. HTML ofrece otros dos atributos similares llamados ondblclick (el usuario hace doble clic con el botón izquierdo del ratón) y oncontextmenu (el usuario hace clic con el botón derecho del ratón).
- **onmousedown**—Este atributo responde al evento mousedown. Este evento se desencadena cuando el usuario pulsa el botón izquierdo o el botón derecho del ratón.
- **onmouseup**—Este atributo responde al evento mouseup. El evento se desencadena cuando el usuario libera el botón izquierdo del ratón.
- **onmouseenter**—Este atributo responde al evento mouseenter. Este evento se desencadena cuando el ratón se introduce en el área ocupada por el elemento.
- **onmouseleave**—Este atributo responde al evento mouseleave. Este evento se desencadena cuando el ratón abandona el área ocupada por el elemento.
- .

Listado de Eventos 2 – Mouse

- **onmouseover**—Este atributo responde al evento mouseover. Este evento se desencadena cuando el ratón se mueve sobre el elemento o cualquiera de sus elementos hijos.
- **onmouseout**—Este atributo responde al evento mouseout. El evento se desencadena cuando el ratón abandona el área ocupada por el elemento o cualquiera de sus elementos hijos.
- **onmousemove**—Este atributo responde al evento mousemove. Este evento se desencadena cada vez que el ratón se encuentra sobre el elemento y se mueve.
- **onwheel**—Este atributo responde al evento wheel. Este evento se desencadena cada vez que se hace girar la rueda del ratón

Lista de Eventos 3 -- Teclado

Aplicable a elementos <input> y <textarea>

- **onkeypress**—Este atributo responde al evento keypress. Este evento se desencadena cuando se activa el elemento y se pulsa una tecla.
- **onkeydown**—Este atributo responde al evento keydown. Este evento se desencadena cuando se activa el elemento y se pulsa una tecla.
- **onkeyup**—Este atributo responde al evento keyup. Este evento se desencadena cuando se activa el elemento y se libera una tecla.

Lista de Eventos 4 -- Documento

También contamos con otros dos atributos importantes asociados al documento:

- **onload**—Este atributo responde al evento load. El evento se desencadena cuando un recurso termina de cargarse.
- **onunload**—Este atributo responde al evento unload. Este evento se desencadena cuando un recurso termina de cargarse.

Ejemplo 2 En línea

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>JavaScript</title>
</head>
<body onload="alert (‘Bienvenido!!’)">
  <section>
    <p onclick="alert('Hizo clic!')" onmouseout="alert('No me abandone!')">Clic aquí</p>
  </section>
</body>
</html>
```

Insertar código en el Head

- Es apropiado solo para acciones inmediatas
- No es aconsejable para códigos extensos o lógica complicada

```
<head>
<script>
  // codigo javascript que sera ejecuta en el momento
</script>
</head>
```

Ejemplo 3 En el documento

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>JavaScript</title>
  <script>
    alert('Bienvenido!');
  </script>
</head>
<body>
  <section>
    <p>Hola</p>
  </section>
</body>
</html>
```

Incrustar código desde archivo externo

- La etiqueta script posee el atributo src que permite vincular un archivo externo, el código de ese archivo es incrustado en el documento.
- Necesario para bloque de código, reutilización de código
- Si varios documentos utilizan el mismo, el navegador lo descarga solo una vez

Ejemplo 4 Incrustar código JS

```
<head>
  <meta charset="utf-8">
  <title>JavaScript</title>
  <script src="micodigo.js"></script>
</head>
```

// donde el archivo micodigo contiene el codigo javascript y esta almacenado en el mismo directorio que el documento html

Elementos Básicos

- **Comentarios:**
  - Para una línea “// Esto es un Comentario”
  - Para un bloque “/* .... */”
- **Literales:** Valores que puede tomar una variable o una constante.
  - Ejemplos: **"Soy una cadena“ - 3434 - 3.43 - true, false**
- **Sentencias y bloques:** En JavaScript las sentencias se separan con un punto y coma, y se agrupan mediante llaves
- Variables: **Una variable es un espacio en memoria donde se almacena un dato.** Los nombres de las variables se construyen con caracteres alfanuméricos y el carácter guión bajo (_).

JavaScript Básico

- Tipos de datos: no es un lenguaje fuertemente tipado como Java o C. Es decir que el tipo del dato es inferido por el valor que almacena la variable. Luego surge typedscript para establecer el tipo de datos estático
- JavaScript permite trabajar con tres tipos de datos primitivos básicos:
  - Numéricos (54, 2.3698, etc.)
  - Cadenas o Strings (“Diseño Web”)
  - Booleanos (True o False)

JavaScript Básico

- **Tipado "dinámico":**
  - Los tipos de datos se asignan dinámicamente
  - La declaración de variables es diferente a los que conocen, ya no es necesario hacerlo de forma explicita.
  - La variable se puede convertir automáticamente de un tipo a otro durante la ejecución
- **Ejemplo:**

dato=48

.....

dato="Diseño Web"

JavaScript Básico

Tipos de datos Numéricos:

- Sólo existe un tipo de datos numérico.
- Todos los números son por tanto del tipo numérico.
- Los números reales también se pueden escribir en notación científica (2.482e12).
- Los números pueden escribirse en las bases decimal, hexadecimal (0x3EF) y octal (0234)

JavaScript Básico

Tipos de datos Cadena o String

- Sirve para guardar texto (números, letras, signos).
- Se pueden introducir cualquier número de caracteres.
- Los textos se escriben entre comillas, dobles o simples.
- Todo lo que se coloca entre comillas es tratado como una cadena de caracteres

JavaScript Básico

Tipos de datos Booleanos

- Verdadero o falso.
- Se utiliza para realizar operaciones lógicas
- Los dos valores que pueden tener las variables boleanas son true o false.

Variables

- Declaración: NO es obligación declarar las variables.
- Declaración: Se utiliza la palabra var.
- Ejemplos de declaraciones:

```
var nombre;
Edad=10;
```

- Se puede inicializar una variable en la declaración

```
var nombre = “Juan”;
var edad = 20;
```

- Se pueden declarar varias variables en la misma línea,

```
var nombre, edad;
```

Ejemplo 5 Variables

```
//codigo javascript a ser colocado en un documento
var minumero = 2;
minumero = 3;
alert(minumero);
```

Operadores

- Los operadores se clasifican según el tipo de acciones que realizan.
- **Se clasifican en:**
  - **Operadores Aritméticos:** Utilizados para operaciones matemáticas simples
  - **Operadores de asignación:** Utilizados para asignar valores a las variables
  - **Operadores de cadenas:** Utilizados para realizar acciones típicas sobre.
  - **Operadores lógicos** Utilizados para realizar operaciones lógicas
  - **Operadores condicionales:** utilizados para definir expresiones condicionales

Operadores Aritméticos

- + Suma de dos valores
- - Resta de dos valores,
- - cambio de signo de un número (un solo operando: –23)
- * Multiplicación de dos valores
- / División de dos valores
- % El resto de la división de dos números
- ++ Incremento en una unidad (un solo operando)
- -- Decremento en una unidad (un solo operando)

Operadores de asignación

- = Asignación. Asigna la parte de la derecha del igual a la parte de la izquierda.
- += Asignación con suma. Realiza la suma de la parte de la derecha con la de la izquierda y guarda el resultado en la parte de la izquierda.
- -= Asignación con resta
- *= Asignación de la multiplicación
- /= Asignación de la división
- %= Se obtiene el resto y se asigna

Operadores de cadenas

- **Javascript** sólo tiene un operador para cadenas
  - + Concatena dos cadenas.

- Se pueden realizar otras acciones con una serie de funciones predefinidas en el lenguaje.

Operadores lógicos

- ! Operador NO o negación.
- && Operador Y
- || Operador O

Operadores condicionales

- == Comprueba si los dos operandos son iguales
- != Comprueba si los dos operandos son distintos
- > Mayor que, devuelve true si el primer operando es mayor que el segundo
- < Menor que, es true si el primer operando es menor que el segundo
- >= Mayor igual.
- <= Menor igual

Precedencia de los operadores

- () [] . Paréntesis, corchetes y el operador punto que sirve para los objetos
- ! - ++ -- negación, negativo e incrementos
- * / % Multiplicación división y módulo
- +- Suma y resta
- << >> >>> Cambios a nivel de bit
- < <= > >= Operadores condicionales
- == != Operadores condicionales de igualdad y desigualdad
- & ^ | Lógicos a nivel de bit
- && || Lógicos boleanos
- = += -= *= /= %= <<= >>= >>>= &= ^= != Asignación

Ejemplos Operadores y variables

```
var mitexto = "Juan";
mitexto = "Mi nombre es " + mitexto;
alert(mitexto);

var otrotexto = "20" + 3;
alert(otrotexto);  // "203"

var valido = true;
alert(valido);
```

Arreglos

- Un arreglo es un conjunto de valores almacenados en espacios consecutivos de memoria
- Es javascript se los denomina Arrays
- Definición por extensión

```
var miarray = ["rojo", "verde", "azul"];
```

- Accedo por posición, con índice entero, primer posición 0

```
alert(miarray[0]);  // "rojo"
alert(miarray); // " rojo,azul,verde "
```

Ejemplo 6 Arrays

```
var miarray = ["rojo", 32, "HTML5 es genial!"];
alert(miarray[1]);
var mia = [64, 32];
mia[1] = mia[1] + 10;
Alert("El valor actual es " + mia[1]); // "El valor actual es 42«
Arreglo de arreglo:
var miarray = [[2, 45, 31], [5, 10], [81, 12]];
alert(miarray[1][0]);  // 5
Vaciar el arreglo
miarray=[];
```

Estructuras de Control

- Toma de decisiones: Para realizar unas acciones u otras en función de una condición.
  - IF
  - SWITCH
- Bucles o iteraciones: Para realizar ciertas acciones repetidamente.
  - FOR
  - WHILE
  - DO WHILE

Estructuras de Control: IF

- La sintaxis de la estructura IF es la siguiente.

```
if (expresión) {
  //   acciones a realizar en caso positivo
     }


if (expresión) {
//   acciones a realizar en caso positivo
   } else {
   //acciones a realizar en caso negativo
            }
```

Estructuras de Control: IF

- Estructuras if anidadas

```
if (x == y){   x+=y;
                alert("x + y = "+x) ;
  } else {
     if (x > y) { x-=y;   alert("x - y = "+x);
    }else{ x*=y;
             alert("x * y = "+x);   }
          }
```

Operador IF

- Es una forma más esquemática de realizar algunos IF sencillos.
- Sintaxis:

Variable = (condición) ? valor1 : valor2 ;

Estructuras de Control: Switch

- Se utiliza cuando tenemos múltiples posibilidades como resultado de la evaluación de una sentencia.

Estructuras de Control: Switch

- Sintaxis:

```
switch (expersión) {
  case valor1:
    Sentencias a ejecutar si la expresión tiene como valor a valor1
    break;
  case valor2:
    Sentencias a ejecutar si la expresión tiene como valor a valor2
    break;
  case valor3:
    Sentencias a ejecutar si la expresión tiene como valor a valor3
    break;
  default:
    Sentencias a ejecutar si el valor no es ninguno de los anteriores
}
```

Estructuras de Control: For

- Se utiliza para repetir una o más instrucciones un determinado número de veces.
- Se suele utilizar cuando sabemos seguro el número de veces que queremos que se ejecute las sentencias.
- Sintaxis

```
for (inicialización;condición;actualización) {
   //sentencias a ejecutar en cada iteración
}
```

Estructuras de Control: While

- Se utilizan cuando se quiere repetir la ejecución de una o más sentencias un número indefinido de veces, según se cumpla una condición.
- Sintaxis:

```
while (condición){
    //sentencias a ejecutar
  }
```

Estructuras de Control: Do ... While

- Se utiliza generalmente cuando se sabe que la iteración se ejecutará al menos una vez
- Sintaxis:

```
do {
    sentencias del bucle
} while (condición)
```

- Se ejecuta siempre una vez y al final se evalúa la condición.

Funciones

- Una función es una serie de instrucciones englobadas dentro de un mismo proceso que se podrá ejecutar luego desde cualquier otro sitio con solo llamarlo.

- Tipos de Funciones en JavaScript
  - Funciones definidas por el usuario
  - Funciones Predefinidas en el Lenguaje

Funciones: Definidas por el Usuario

- La sintaxis es:

```
function nombre (p1,…,pn){ //no tiene el tipo del parámetro
  /* instrucciones de la función
    … */
}
```

- Invocación

```
nombre (p1,p2,…,pn)
```

Funciones: Donde definirlas

- En cualquier lugar dentro de un bloque <SCRIPT>
- Orden definición – invocación
    - En el mismo bloque es indistinto
- En bloques distintos: Primero debe estar declarado el bloque de definición y luego el de invocación

Funciones: Valores de Retorno

- Toda función puede retornar un valor
- Sintaxis:

```
function nombre_funcion(){
        <sentencias>
        return <valor de retorno>
}
```

Funciones: Pasaje de Parámetros

Tipo de pasaje de parámetro es por Valor. Lo que implica que el valor del parámetro real no se modifica.

¿Cómo logramos modificar los valores dentro de la función de manera que queden cambiados al terminar la misma?

Ámbito de Variables

- Se le llama ámbito de las variables al lugar donde están disponibles.
- En JavaScript las variables son accesibles dentro de la página en la que se declaran.
- Existen dos ámbitos
  - Variables globales.
  - Variables locales

Ámbito de Variables: Variables globales

- Son las declaradas en el ámbito más amplio posible, que en Javascript es una página web.
- Una variable global a la página se declara en la cabecera del documento ya la etiqueta script o en el archivo correspondiente js.

```
<SCRIPT>
        var variableGlobal
</SCRIPT>
```

- Las variables globales son accesibles desde cualquier lugar de la página

Ámbito de Variables: Variables locales

- Son las declaradas en ámbitos más acotados (Ej, funciones, iteraciones)
- Sólo pueden accederse dentro del lugar donde se han declarado.
- **Ejemplos:**

```
<SCRIPT>
  function miFuncion (){
     var variableLocal
  }
</SCRIPT>
```

Ámbito de Variables

- Pueden existir variables con el mismo nombre en ámbitos distintos.
- **Ejemplo:**

```
<SCRIPT>
var numero = 2
function miFuncion (){
    var numero = 19
    alert(numero) //imprime ?
}
alert(numero) //imprime ?
</SCRIPT>
```

Ámbito de Variables

- En Javascript tenemos libertad para declarar o no las variables con la palabra var

- **Diferencias entre utilizar var o no**
  - Con **var** la variable es de ámbito local o global dependiendo de dónde se declaro.
  - Sin **var** tomará el valor de la variable global en caso de existir una con el mismo nombre.

Ejemplo 7: Ámbito de Variables

```
<SCRIPT>
  var numero = 2
  function miFuncion (){
     numero = 19
     alert(numero) //imprime ¿?
  }
  alert(numero) //imprime ??

miFuncion() //llamamos a la función
alert(numero) //imprime ??
</SCRIPT>
```