Unidad 1

JAVASCRIPT

Interacción con JavaScript

Acceso al contenido: Se puede usar Javascript para acceder a cualquier elemento, atributo o texto desde una página HTML

Modificar contenido: Se puede usar Javascript para agregar y o eliminar elementos, atributos, texto de una página HTML

Incluir reglas al programa: Se puede usar Javascript para regular la forma en que el usuario/navegador debe seguir para acceder o modificar el contenido de una página

Reaccionar a los eventos: Se puede usar Javascript para indicar un script que debe ejecutarse cuando ocurre un evento específico

Objetos y eventos

Cada concepto físico puede representarse como un objeto del mundo real-

Un objeto puede tener:

- Propiedad: valor para las características que lo componen
- Método: que hace el objeto
- Evento: la forma en la que el usuario interactúa con el evento.

Java VS JavaScript

JavaScript no es Java, son dos lenguajes de programación distintos, en realidad el nombre de JavaScript proviene de una asociacion en Netscape y los creadores de Java.

JavaScript tiene una sintaxis heredada de C.

Diferencias

JAVA

- Necesita compilador
- Orientado a objetos
- Propósito general
- Lenguaje propietario (pertenece a oracle)

JAVASCRIPT

- No necesita compilador
- Multiparadigma
- Programación web
- Lenguaje abierto

Formas de introducir JavaScript

Existen dos formas de utilizarlo:

1. Inclusión directa
2. Respuesta a evento

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Página con Javascript</h1>
    <p>Esta página tiene un cuadro de diálogo, que se mostrará nada más el navegador la procese.</p>

    <script>
      var personas = 4;
      var importeEntradas = 9.50;
      alert('Necesitas ' + personas * importeEntradas + ' euros para que entren todos al cine');
    </script>

    <p>Cuando el usuario pulse aceptar en el cuadro de diálogo, el navegador mostrará la página completa.</p>
</body>
</html>
```

```html
<body>
    <h1>Ejemplo Javascript</h1>
    <span id="mielemento">Pasa el ratón por aquí</span>
    <script>
    var pasadas = 0;
    function anunciarPasadas() {
      pasadas = pasadas + 1;
      alert('Has pasado el ratón encima ' + pasadas + ' veces');
    }
    document.getElementById('mielemento').addEventListener('mouseenter', anunciarPasadas);
    </script>
</body>
```

Recomendaciones

A la hora de utilizar JavaScript es recomendable utilizar un fichero externo por los siguientes motivos:

- Separación de código.
- Permite que el contenido del .js se almacene en la cache del navegador, lo que permite que si se vuelve a visitar la pagina esta cargue mucho mas rápido.

Sintaxis

JavaScript es un lenguaje sensible, lo que quiere decir que detecta entre mayúsculas y minúsculas.

Al igual que en C y C++ se utiliza ; para separar las diferentes sentencias

Si queremos introducir comentarios en JavaScript utilizamos // o /* */

Declaración de variables y tipos

Para declarar una variable en JavaScript utilizamos var nombre = expresión;

```
var nombre = "Pedro Garcia";
var num1 = 198;
var num2 = 3.14159265;
var VeroFal = true;
```

Las variables se convierten automáticamente en un tipo compatible en cuanto le pasamos la información (conversión automática)

Los identificadores tienen que empezar por una letra, pero pueden contener dígitos, $, _

Aunque no definamos un tipo de variable, JavaScript tiene sus propios tipos (number, boolean, String…)

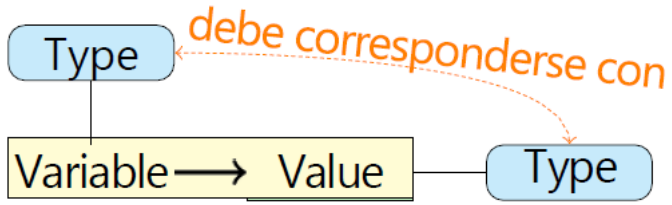
Se puede saber el tipo de variable utilizando typeof.

Sistema de tipos

Variables estáticamente tipado

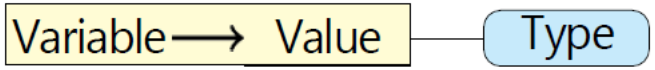
Cada variable esta vinculada a un tipo en particular.

Cada variable solo puede almacenar un valor de tipo coincidente.



Variables escritas dinamicamente

Cada variable puede almacenar un valor de tipo arbitrario.



Cada variable puede almacenar valores de diferentes tipos en diferentes momentos.

Sistema de tipos

Cuando invocamos una operación, cada valor de argumento debe ser de tipo coincidente.

En JavaScript, cada valor de argumento se convertirá implícitamente al tipo coincidente siempre que sea compatible.

X = 500; -> tipo number

X = ‘500’; -> tipo string

Problemas del sistema de tipos

Cuando declaramos funciones no declaramos los tipos de sus parámetros de entrada, cuando llamamos a una función sus parámetros se ajustan de forma automática (si es posible)

```
function sumar(x, y){
    var resultado = x + y;
    document.getElementById('demo').innerHTML="El resultado es: " + resultado + " del tipo: " + typeof(resultado)
}
```

Ventaja: programación flexible.

Desventaja: Posibilidad de generar errores potencialmente.

NaN e Infinity

Es posible que al realizar alguna operación nos de alguno de estos dos valores, NaN e Infinite, el primero indica que la operación no se ha podido realizar y a devuelto fallo, como por ejemplo 0/0, y el segundo un valor infinito.

Ambos son valores reservados, por lo cual podemos utilizarlos para comparar variables con los operadores lógicos.

¿Por qué son importantes estos dos valores?

Cuando intentados determinar si una variable contiene un tipo de dato, si este no es compatible devuelve NaN.

Por ejemplo si intentamos pasar una variable de tipo string a int esta puede devolver NaN

Ejemplos

N1 = parseInt (“una cadena”) -> devuelve NaN

N1 = parseInt (“una cadena 123”) -> devuelve 123

Dar valor desde HTML

Por medio de un elemento input de HTML podemos dar valores a las variables, por medio de document.getElementByID('ID del input').value

Salida de datos

JavaScript permite la salida de datos de varias formas:

document.getElementById(‘id del elemento').innerHTML=“elementos ”

Document.write(“”) -> introduce el elemento directamente dentro del body

Función alert -> utiliza un evento de alerta para mostrar la información alert(2+3);

Introducir elementos por JavaScript

Podemos introducir elementos pertenecientes a HTML por medio de funciones en JavaScript.

```
<body>
    <h1>Introducir elementos en html</h1>
    <br>
    <button type="button"
        onclick=introducirElem()>
        Introducir elemento.
    </button>
    <div id="elejs"></div>


</body>
```

```
function introducirElem(){
    document.getElementById("elejs").innerHTML="<p>Elemento introducido por JavaScript</p>";
}
```

Variables

Tenemos 3 tipos de variables en JavaScript, var, let y const:

- var: declara una variable, utiliza los métodos de la antigua versión de JavaScript (No es recomendable su uso)
- let: declara una variable con el nuevo método, permite que se modifique su valor
- const: declara una variable estatica, no se puede modificar su valor

“Tipos” de variables dentro de JS

Los tipos mas comunes que encontramos dentro de JavaScript son:

- String: cadena de caracteres
- Number: vale tanto como un int, doublé o float
- Bigint: representa un numero entero muy grande
- Boolean: representa verdadero o falso, podemos representar falso de varias formas (false, 0, NaN, “” y null), el resto significa true
- Object:
- Undefined: variable **SIN** contenido
- Null: indica que la ausencia
- Symbol: identificador único, suele utilizarse para definir conts

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="variable1.js"></script>
</head>
<body>

    <div>
        <h2>Variables en JavaScript</h2>
        <label for="entrada1">caja de la variable:</label><br>
        <input type="text" id="entrada1">

        <button type="button" onclick="cambTexto()">Cambiar</button>

        <p id="demo">Este texto cambiara con el boton</p>

    </div>

</body>
</html>
```

```javascript
function cambTexto(){
    let cadena1;
    cadena1=document.getElementById("entrada1").value;

    document.getElementById('demo').innerHTML=cadena1;

}
```

Modificar elementos HTML

Podemos modificar atributos de las etiquetas de los elementos del documento

```
<body>

    <p>Podemos cambiar los atributos de un elemento.</p>

    <button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Encender</button>

    <img id="myImage" src="pic_bulboff.gif" style="width:100px">

    <button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Apagar</button>

</body>
```

Introducir elementos en HTML

También podemos introducir elementos dentro de los div o del cuerpo.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
        let cont=1;
        function elemHTML() {

            document.getElementById("container").innerHTML="<p>Parrafo " + cont + "</p><br>";
            cont++;
        }

    </script>
</head>
<body>

    <button type="button" onclick="elemHTML()">Introduce un elemento HTML</button>
    <div id="container"></div>

</body>
</html>
```

Arrays

```
let array1 = ["AaD", "DIN", "PSE"];
let array2 = [];

array2[0]="Hola";
array2[1]="Mundo";
```

Al igual que en otros leguajes de programación, podemos definir arrays de datos.

Tenemos una serie de funciones útiles para ayudarnos a controlar los arrays:

- number array.push(valor1, valor2,…): agrega al final del array uno o mas elementos, devuelve el numero total de elementos del array.
- mixed array.pop(): extrae y devuelve el último elemento.
- mixed array.shift(): extrae y devuelve el primer elemento.
- number array.unshift(value 1, value 2,…): inserta uno o más elementos al comienzo del array devuelve el número de elementos del array resultante

Operadores

Los operadores que podemos encontrar en JavaScript son muy parecidos a otros vistos en distintos lenguajes de programación:

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
**	Exponente
/	División
%	Modulo (Resto)
++	Incremento
--	Decremento

Operador	Ejemplo	Significado
=	x = y	x = y
+=	x+=y	x = x + y
-=	X-=y	x = x – y
*=	x*=y	x = x * y
**=	x**=y	x = x ** y
/=	x/=y	x = x / y
%=	x%=y	x = x % y

Operador	Descripción
&&	Y lógico (AND)
||	O lógico (OR)
??	Multiplicación

Operador	Descripción
==	Igual valor
===	Igual valor y tipo
!=	Distinto valor
!==	Distinto valor y tipo
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

Funciones

Podemos definir funciones para ser invocadas desde cualquier parte y tantas veces como las necesitemos

function nombreFuncion(parametro1, parametro2,….)(){}

Eventos

Los eventos sirven para controlar la ocurrencia de algún suceso, podemos asociar un evento a una función para que ocurra una función en especifico. Los eventos mas comunes son:

Evento	Descripción
onclick	Ocurre al hacer click sobre un elemento
onmouseover	Ocurre al pasar con el ratón sobre un elemento
onmouseout	Ocurre al abandonar el ratón el área de un elemento
onfocus	Ocurre cuando un elemento recibe el enfoque
onchange	Ocurre cuando un elemento cambia su valor
onkeydown	Ocurre cuando un usuario ha presionado una tecla
onload	Ocurre cuando se está cargando la web
ondblclick	Ocurre cuando se hace doble click con el ratón

```
<input type="text" onchange="alert('evento change')">

<input type="text" onfocus="alert('evento focus')">
<input type="text" onkeypress="alert('evento keypress')">
<input type="text" onkeydown="alert('evento keydown')">
<div ondblclick="alert('evento doble click')">
esto es un div
</div>
```

Estructuras de control: if

```
if (condición){
Sentencias;
}
else if (condición){
Sentencias;
}
Else{
Sentencias;
}
```

Estructuras de control: switch

```
let x;
x = 3;

switch(x){
    case 1:
        alert("X vale 1");
        break;
    case 1:
        alert("X vale 2");
        break;
    case 1:
        alert("X vale 3");
        break;
    default:
        alert("X vale mas de 3");
        break;
}
```

Bucles: for

```
let i,cont;
cont = 1;
for(i=0;i<5;i++){
    cont*=2;
}

document.getElementById("salida").innerHTML = cont;
```

Bucles: while

```
let i,cont;
cont = 1;
while(i<5){
    i++;
    cont*=2;
}

document.getElementById("salida").innerHTML = cont;
```

Bucles: do ... while

```
let i,cont;
cont = 1;
do{
    i++;
    cont*=2;
}while(i<5)
```

Bucles

Podemos introducir un break en una sentencia condicional si queremos salir del bucle antes de tiempo.

Recoger agrupaciones de elementos

Podemos utilizar 2 funciones para recoger una serie de elementos del documento:

- getElementByTagName(etiqueta): recoge todos los elementos de un tipo de etiqueta en particular.
- getElementByClass(Clase): recoge todos los elementos pertenecientes a una clase en concreto.

```
function funcionclases(){
    let vector = document.getElementsByClassName("tparrafo");
    let i;

    for (i=0 ; i<vector.length ; i++){
        vector[i].innerHTML = i + "";
    }

}

function funcionetiqueta(){
    let vector = document.getElementsByTagName("p");
    let i;

    for (i=0 ; i<vector.length ; i++){
        vector[i].innerHTML = i + "";
    }

}
```