DESARROLLO DE APLICACIONES WEB

Tema 4.- Javascript

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Introducción

- JavaScript
  - JavaScript es un lenguaje de Script cuyo código es interpretado (no compilado)
  - Funcionalidad a la Web del lado del cliente

  - Hoy día también se utiliza como tecnología del servidor (NodeJs, Jaxer)

Introducción

- JavaScript del lado del cliente
  - Compuesto por:
    - Lenguaje EMACScript (ECMA-262)
      ECMA - European Computer Manufacturers Association
      Más información: https://en.wikipedia.org/wiki/ECMAScript
    - Modelo de Objetos del Navegador (BOM – Browser Object Model)
    - Modelo de Objetos del Documento (DOM – Document Object Model)
- JavaScript vs TypeScript

Introducción

- ¿Qué podemos hacer con JavaScript en la Web del lado del cliente?
  - Incluir contenido dinámico
  - Manejar, modificar o incluir tanto el contenido HTML como CSS de forma dinámica
  - Actuar ante eventos que se produzcan
  - Validación de datos
  - Identificación del navegador
  - Manejo de Cookies
  - etc, etc, etc…

- Herramienta online: CodePen

  https://codepen.io/pen/

Introducción

- JavaScript en un documento HMTL

```
<script … >  …  </script>
```

  - (inline script) en el propio documento

```
<script>
     // Código JavaScript
</script>
```

  - (external script) en un fichero externo

```
<script src="ruta/fichero.js"></script>
```

  - Para navegadores que no soporten JavaScript

```
<noscript>
      <p>Esta página usa JavaScript y tu navegador no lo soporta.</p>
</noscript>
```

Introducción

- ¿Dónde incluir el código JavaScript?

  La etiqueta <script> puede ir tanto en el <head> como en el <body>

  - Tradicionalmente se incluía en la cabecera

```
<!DOCTYPE html>
<html>
  <head>
     <title>Página HTML con JavaScript</title>
     <script type="text/javascript" src="funciones1.js"></script>
     <script type="text/javascript" src="funciones2.js"></script>
  </head>
  <body>
     <!– Aquí el contenido -->
  </body>
</html>
```

Introducción

- ¿Dónde incluir el código JavaScript?
  - Una mejor solución es insertarlo al final.

    El documento se renderiza antes de ejecutar el código Javascript por lo que no hay retardos innecesarios y permite usar Javascript no intrusivo

```
<!DOCTYPE html>
<html>
  <head>
     <title>Página HTML con JavaScript</title>
  </head>
  <body>
     <!– Aquí el contenido -->
     <script type=”text/javascript” src=”funciones1.js”></script>
     <script type=”text/javascript” src=”funciones2.js”></script>
  </body>
</html>
```

  - El atributo *async*: Javascript se ejecuta de forma asíncrona

```
<script async src=”example1.js”></script>
```

Introducción – Ejemplo 1

ejemplo1_js.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo 1 JS</title>
  <meta charset="utf-8" />

  <script type="text/javascript">

    function procesaDato() {
      let v, elto;
       v=document.getElementById("dato").value;
      if ( v=="" ) {
       alert("Debes insertar algún valor");
       return false;
      }
      elto=document.getElementById("valor");
      elto.innerText=v;
     }

  </script>
</head>
```

```
<body>

<script>
  document.write("<h1>Ejemplo 1</h1>");
</script>

<input type="text" name="dato" id="dato" />
<input type="button"
       value="validar“ id=“bProcesar”
       onclick="procesaDato();"/>

<p>Valor Insertado:
   <span id="valor”></span>
</p>


</body>
</html>
```

Introducción – Ejemplo 2 (no intrusivo)

ejemplo2_js.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo 2 JS</title>
    <meta charset="utf-8" />
</head>
<body>

<h1>Ejemplo 2</h1>

<input type="text" name="dato" id="dato" />
<input type="button" value="Procesar"
    id="bProcesar" />

<p>Valor Insertado:
<span id="valor"></span>
</p>

<script src="file.js"></script>

</body>

</html>
```

```
// fichero file.js


 let b, elto;
 b=document.getElementById('bProcesar');

 b.onclick = function() {
  let v;
  v=document.getElementById("dato").value;
  if ( v=="" ) {
       alert("Debes insertar algún valor");
       return false;
  }
  elto=document.getElementById("valor");
  elto.innerText=v;
 }
```

Introducción

- Depurando JavaScript
  - Usar consola JavaScript de las herramientas de desarrollador del navegador
  - Console.log, Console.error, Console.warn, Console.info, Console.group

    Artículo: How you can improve your workflow using the JavaScript console
  - Ejemplos

```
console.log(”Mensaje a la consola");

let msg = “Variable mostrada por consola";
console.log(msg);
```

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Fundamentos de JavaScript

- Sintaxis
  - Case sensitive
  - Es de típado débil y dinámico
  - Sentencias
    - En nueva línea o separadas por punto y coma (;)
    - (Recomendado) Finalizar con punto y coma (;)
    - (Recomendado) Uso siempre de paréntesis para bloques, aunque afecte solo a una sentencia
  - Comentarios
    - Línea: `// Comentario de una línea`
    - Varias líneas:
      ```
      /* comentario en
         varias líneas */
      ```

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Fundamentos de JavaScript

- Tipos de datos y variables
  - Declaración de variable. No requiere especificar tipo, se establece en cada asignación. Se recomiendo usar estilo "camel case" para identificadores

```
var miVariable ;      // declaración de variables globales o locales a función
let miVariabble;      // declaración de variable de bloque
const miConstante = 10; // declaración de una constante
```

Tipos de datos

number: Enteros y Reales	let num1=45, num2=3.14;
boolean: true y false	let ok=true;
string: cadenas caracteres.	let s1="Hola Mundo!" //Dobles comillas
	let s2= 'Hola Mundo!' // Comillas simples
object: Objetos (arrays, funciones, etc)	

```
typeof miVariable ;    /* undefined, boolean, string, number, obtect, function */
miVariable intanceof tipo /* true si la variable es del tipo indicado */
```

Fundamentos de JavaScript

- JavaScript realiza conversión implícita (automática) cuando lo necesita

- Se puede forzar de forma explícita:

  Number(x), String(x), etc

- Funciones de conversión de tipos

  int parseInt(*cad***):** Devuelve el entero incluido en *cad* o NAN si no es numérica

  float parseFloat(*cad***):** Devuelve el real incluido en *cad* o NAN si no es numérica

  String v.toString(): Devuelve una cadena con el valor de v

- Concatenación de cadenas y variables numéricas: +

  Ejemplo: "El valor de X es " + x ;

  NOTA: Plantillas literales se delimitan con el carácter tildes invertidas (` `) (acento grave), permite evaluar el contenido. Ejemplo. `La suma es ${suma}.`

Fundamentos de JavaScript: Boolean

- Valores

```
var found = true;
var lost = false;
```

- Conversión a Boolean

Tipo	Valor True	Valor False
String	No vacía	Vacía
Number	Cualquier número distinto de cero (incluso Infinity)	Cero y NaN
Object	Cualquier objeto	Null
Undefined	-	undefined

Fundamentos de JavaScript: Number

- Valores

```
var myInteger = 16;
var myFloat = 1.08;
var reallyBig = 8.67e10;  // 86700000000
var reallySmall = 15e-4;  // 0.0015
```

- Constantes and Funciones
  - Number.MAX_VALUE . Número mayor que se puede representar
  - Number.MIN_VALUE . Número menos
  - Infinity y –Infinity . Valor infinito
  - NaN . Valor que indica que no es un número

  - isNaN(x) . True si x no es un número o un valor de conversión directa

Fundamentos de JavaScript: String

- Valores

  var miNombre= “Soy ‘Pepe’ ”; // Comillas dobles

  var saludo = ‘Hola “Pepe”!’; // Comillas simples

  NOTA: Plantillas literales se delimitan con el carácter tildes invertidas (` `) (acento grave), permite evaluar el contenido. Por ejemplo. `La suma es ${suma}.`

- Caracteres especiales
  - \n - Salto de línea
  - \t - Tabulador
  - \b - Espacio
  - \r - Retorno de caro
  - \\ - Caracter \
  - \’ - Caracter ‘
  - \” - Caracter “

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Fundamentos de JavaScript

- Operadores
  - Asignación

Operador	Descripción
=	Asigna el valor del operando de la izquierda al operando de la derecha
+=	Suma el operando derecho al izquierdo y le asigna el resultado
-=	Resta el operando derecho al izquierdo y le asigna el resultado
*=	Multiplica el operando derecho por el izquierdo y le asigna el resultado
/=	Divide el operando izquierdo entre el derecho, asignando el resultado al operando izquierdo
%=	Divide el operando de la izquierda por el de la derecha y asigna el valor del resto de la división al operando de la izquierda

  - Ejemplos:

```
a = 5;
b += a;
b %= 7;
```

Fundamentos de JavaScript

- Operadores
  - Aritméticos

Operador	Descripción
binarios	
+	Sumar
-	Restar
*	Multiplicar
/	Dividir
%	Resto división
unarios	
++	Incremento
--	decremento

Ejemplos

```
c = a+b;
d = (a + b) / 2;
resto =  a % 2;
area = lado * lado;
valor = 12 / ( 4 * 3 );
a++;
++b;
--c;
d--;
```

Fundamentos de JavaScript

- Operadores
  - Relacionales y Lógicos

Operador	Descripción
==	Igual que
===	**Idénticamente igual**
!=	Distinto de
!==	**Idénticamente distinto**
> (>=)	Mayor (igual) que
< (<=)	Menor (igual) que
&&	Y (AND)
||	O (OR)
!	No

x	y	x&&y	x||y	!x
F	F	F	F	T
F	T	F	T	T
T	F	F	T	F
T	T	T	T	F

Ejemplos

a == “ok”
a=42; if (a==’42’) { … // true
a=42; if (a=== ‘42’ } … // false
if (salir != 100) ..
while (x1 <= 10) …
for (i=1; i<=10; i++) ...
if (!condicion) …
while (x<10 && y<10) …
if (!a || (b>0 && c!=“si”)) …

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Fundamentos de JavaScript

- Estructuras de control: Condicionales
  - **if (***cond***) { *sentSi* }**
  - **if (***cond***) { *sentSi* }**
    else {*sentSiNo* }
  - **if – else if –**
  - **switch (** exp **) {**
    case *val1* **:** *sent1*; **break;**
    case *val2* **:** *sent2*; **break;**
    …
    case *valN* **:** *sentN*; **break;**
    default : *sent*;
    }

Fundamentos de JavaScript

ejemplo3_js.html

- Estructuras de control: Condicionales

```
...
<form id="f" action="" method="">
    <label for="v1">Valor 1:</label>
    <input type="number" id="v1" name="v1"/>
    <label for="v2">Valor 2:</label>
    <input type="number" id="v2" name="v2" />
    <label for="op">Operación</label>
    <select id="op" name="op">
          <option>Elige opcion</option>
          <option value="+" selected>Sumar</option>
          <option value="-">Restar</option>
          <option value="*" >Multiplicar</option>
          <option value="/" >Dividir</option>
    </select>
    <h2><label for="result">Resultado:</label>
    <output id="result" name="result" size="3">
     </output>
    </h2>
    <br />
    <input id="b" type="button" value="Calcular" />
</form>
...
```

```
<script>
  function operar() {
    let f=document.getElementById("f");
    let resultado=0;
    let operacion=f.op.value;
    let op1=parseFloat(f.v1.value);
    let op2=parseFloat(f.v2.value);
    if (isNaN(op1) || isNaN(op2)) {
       alert("Introduce números");
    }
    else {
      switch (operacion) {
        case ("+") :
             resultado=op1+op2;break;
        case ("-") :
             resultado=op1-op2;break;
        case ("*") :
             resultado=op1*op2;break;
       }
       f.result.value=resultado;
      }
      return false;
}

  let b = document.getElementById("b");
  b.addEventListener("click", operar, false);
</script>
```

Fundamentos de JavaScript

- Estructuras de control: Bucles
  - **for** (*exp_inic ; cond ; exp_incr*) **{** *sent* **}**

    Las sentencias se ejecutan desde exp_inic mientras se cumple la condición incrementandose en cada iteración segú exp_incr

    Ejemplo:
    ```
    for( i=0; i<100; i++) {
        document.write( "Línea "+i+ "<br /> ");
    }
    ```
  - **for** (ind **in** objeto) **{** *sent* **}**

    Se repiten las sentencias para cada elemento en el objeto; por ejemplo, un array, tomando ind el índice (0, 1, …).
  - **for** (elto **of** objeto) **{** *sent* **}**

    Se repiten las sentencias para cada elemento en el objeto; por ejemplo, un array, tomando elto cada uno de los elementos (elto1, elto2, …).
  - **while** (*cond*) **{** *sent* **}**

    Las sentencias *sent* se ejecutará mientras que se cumpla la condición.

    Ejemplo:
    ```
    while ( dato!="Fin") {
        dato = prompt( "Inserta dato);
    }
    ```

Fundamentos de JavaScript

- Estructuras de control: Bucles
  - **do {***sent* **} while (***cond***);**
    Las sentancias *sent* se ejecutaran mientras la condición *cond* sea verdadera.
    Ejemplo:
    ```
    do {
        dato = prompt( "Inserta dato);
        …
    } while ( dato!= "Fin " );
    ```
  - **break** y **continue**
    break : finaliza la ejecución de un bucle.
    continue : Finaliza un ciclo de un bucle, comenzando uno nuevo.

Fundamentos de JavaScript

ejemplo4_js.html

- Estructuras de control: Bucles

```
<script>
function entrar() {
  let r, intentos=0, salir;
  do { intentos++;
       r=prompt("Introduce Palabra mágica ?");
       if(r==null) continue;
       if(r=="abracadabra") break;
       let i = document.getElementById("intentos");
       i.innerHTML=intentos;
       if(intentos==3) {
           salir=confirm("Quieres salir?");
           if (salir) {
                     break;
           }
           intentos=0; i.innerHTML=intentos;
       }
  } while(true);
  if (r=="abracadabra") {
    document.write("<h1>Has encontrado la palabra mágica</h1>");
  }
  else {
    document.write("<h1>No encontrate la palabra mágica</h1>");
  }
}  </script>
```

```
…
<body onload="entrar()">

<h1>Ejemplo JS: Bucles</h1>
<p>Acierta la palabra mágica</p>
<p>Intentos: <span
id="intentos">0</span></p>

</body>
…
```

Fundamentos de JavaScript

- Estructuras de control: Gestión de errores
  - **try {***sent1* **} catch(***ex***) {** sent2 **};**
    Si durante la ejecución de *sent1* se produce una excepción se ejecutará sent2.
    Ejemplo:

```
try{
    undefinedFunction();
    alert( 'Aparece si la función se ejecuta bien' );
}
catch(ex){
    alert( 'Se ha producido el error: ' + ex.message)
};
```

  - **throw** "*excepcion* "**;**
    Lanza una excepción.

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Funciones en JavaScript

- Declaración (tradicional)

  function *nombFuncion* **(** *listaParametros* **) {**

  sentencias;

  }

  O bien (como función anónima)

  var nombFuncion **= function (** *listaParametros* **) {**

  sentencias;

  };

- Llamada: *nombFuncion* **(** *valoresParametros* **);**

- Para devolver algún valor: **return**

- Expresiones Lambda o Funciones flecha

```
const suma= (x,y) => x+y;
console.log(suma(4,4))

// Lambda
[5, 8, 9].map(i => i + 1);
// -> [6, 9, 10]

// función anónima
[5, 8, 9].map(function(i) {
  return i + 1;
});
// -> [6, 9, 10]
```

Funciones en JavaScript

ejemplo5_js.html

- Funciones – Ejemplo

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo 5 JS. Funciones</title>
<meta charset="utf-8" />
</head>
<body>

<h1>Ejemplo JS: Funciones</h1>
<form>
<label for="v">Valor:</label>
<input type="type" id="v" name="v"
      onchange="salida.value=raizCuadrada(v.value)"/>
<label for="output">Resultado:</label>
<output id="output" name="salida"></output>
</form>
```

```
<script>
  var aviso = function (error, funcion) {
       alert("Error: "+error+" en función"+funcion);
  };

  function raizCuadrada(x) {
    let dato;
    if (x<0) aviso("Valor Negativo", "raizCuadrada");
    else {
      dato = Math.sqrt(x);
      return dato;
     }
   }
</script>

</body>

</html>
```

Funciones en JavaScript

- Ejemplo Callbacks: fichero **func.js**

```
function sumar(x,y) {
    return x+y;
}

function restar(x,y) {
    return x-y;
}

function calcular(func, x, y) {
    return func(x,y);
}
```

```
let sumar = (x,y) => x+y;
let restar = (x,y) => x-y;

let calcular = (funcion, x, y) => funcion(x,y);
```

```
/*
console.log("Calcular 4+2 = " + calcular(sumar, 4, 2));
console.log("Restar 4-2 = " + calcular(restar, 4, 2));
*/
```

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Objetos en JavaScript

- Objetos

```
var  miClase = new Object();  // crea un objeto vacío sin atributos ni métodos

miClase.nombre="Pepe"; // añade un atributo nombre con valor Pepe
miClase.edad = 40;        // añade un atributo edad con valor 40

miClase.dimeHola = function() {     // añade un método dimeHola
    alert("Hola " + this.nombre); // que tiene este código
};
```

Todo de una vez

```
var  miClase = {
    nombre :"Pepe",   // asignación con dos puntos (:) y atributos separados por coma (,)
    edad : 40,
    dimeHola : function() {
                alert("Hola " + this.nombre);
    }
};
```

Objetos en JavaScript

- Objetos
  - Métodos como funciones anónimas en la función constructora

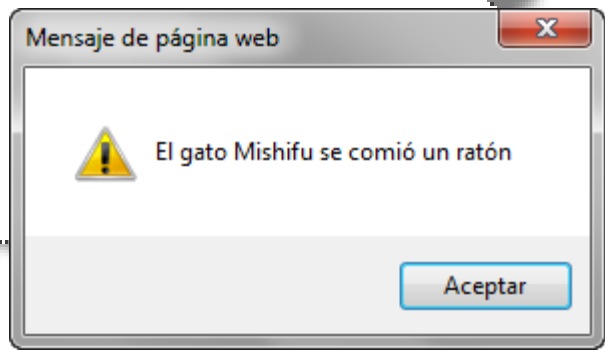
```
var Gato = function (nombre, edad) {
        this.nombre = new String(nombre);
        this.edad = new Number(edad);

        this.comer = function() {
            alert("El gato "+this.nombre+" se comió un ratón");
            };

    }

...

var miGato = new Gato("Mishifu", 2);
miGato.comer();
```



Objetos en JavaScript

- Objetos
  - Métodos como funciones externas en la función constructora

```
var Gato = function (nombre, edad) {
        this.nombre = new String(nombre);
        this.edad = new Number(edad);

        this.comer = comerExterna;

        }
    }

function comerExterna() {
    alert("El gato "+this.nombre+" se comió un ratón");
}


...

var miGato = new Gato("Mishifu", 2);

miGato.comer();
```

Objetos en JavaScript

- Objetos
  - Añadir métodos al prototipo, que no estaban en la función constructora

```
var Gato = function (nombre, edad) {
        this.nombre = new String(nombre);
        this.edad = new Number(edad);
    }

Gato.prototype.comer = function() {
            alert("El gato "+this.nombre+" se comió un ratón");
            };


...

var miGato = new Gato("Mishifu", 2);
miGato.comer();
```

Objetos en JavaScript

- **Objetos:** Herencia

```
var Gato = function () {
    this.ojos = 2;
    this.piernas = 4;
}
var Siames = function () {
    this.color = "blanco";
    this.color_ojos = "azul";
}
//Como vemos, ambos tienen propiedades distintas.
//Ahora, heredemos:
Siames.prototype = new Gato();
//Eso hace que se copie el prototipo de Gato y se añada al de Siames.
//Probemos a ver si es cierto
var Catboy = new Siames();
alert(Catboy.ojos);
//Retorna 2! ^_^
alert(Catboy.color);
//Retorna "blanco", asi que conserva sus propiedades
```

Objetos en JavaScript

ejemplo6_js.html

- Clases: Ejemplo clases usando elemento canvas HTML5

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo   JS. Clases</title>
<meta charset="utf-8" />
</head>
<body>

<h1>Ejemplo JS: Clases</h1>


<canvas id="miCanvas" width="200" height="100"
        style="border:1px solid red;">
Tu navegador no soporta Canvas.
</canvas>


<script type="text/javascript">
    function punto(x, y)  {
        this.x = x;
        this.y = y;
    };

    punto.prototype.damePunto = function() {
        return "("+ this.x + ", " + this.y + ")";
    };
```

```
    let linea1 = new Object();
    linea1.nombre="Mi Polígono";
    linea1.p1=new punto(10, 40); linea1.p2=new punto(100, 95);
    linea1.p3=new punto(150, 90); linea1.color="#00FF00";

    let linea2 = {
        p1: new punto(100, 50), p2: new punto(180, 70),
        p3: new punto(160, 20), p4: new punto(10, 20),
        ancho:2
    }

    let c=document.getElementById("miCanvas");
    let ctx=c.getContext("2d");
    // Línea 1
    ctx.beginPath();
    ctx.strokeStyle = linea1.color;
    ctx.moveTo(linea1.p1.x,linea1.p1.y);
    ctx.lineTo(linea1.p2.x,linea1.p2.y);
    ctx.lineTo(linea1.p3.x,linea1.p3.y);
    ctx.font="8px Arial";
    ctx.fillText(linea1.p1.damePunto(),linea1.p1.x,linea1.p1.y);
    ctx.fillText(linea1.p2.damePunto(),linea1.p2.x,linea1.p2.y);
    ctx.fillText(linea1.p3.damePunto(),linea1.p3.x,linea1.p3.y);
    ctx.stroke();
    // Línea 2
    ctx.beginPath();
    ctx.moveTo(linea2.p1.x, linea2.p1.y);
    ctx.lineTo(linea2.p2.x,linea2.p2.y);
    ctx.lineTo(linea2.p3.x,linea2.p3.y);
    ctx.lineTo(linea2.p4.x,linea2.p4.y);
    ctx.lineWidth = linea2.ancho;
    ctx.stroke();
```

```
</script>
</body>
</html>
```

Objetos en JavaScript -> JSON

- JSON – JavaScript Object Notation

  Notación de objeto de JavaScript

- Formato de texto ligero para el intercambio de datos, alternativa a XML
- JSON es un subconjunto de la notación de objetos de JavaScript

- Recibir datos

```
var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

- Enviar datos

```
var myJSON = '{"name":"John", "age":31, "city":"New York"}';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Clases en JavaScript

- Clases
  - ECMAScript 2015 (ES6)
    https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Classes
  - Sintaxis

```
class nombreClase {
    attributes
    constructor() { ... }
    method_1() { ... }
    method_2() { ... }
    method_3() { ... }
    }
  }
```

				
Chrome 49	Edge 12	Firefox 45	Safari 9	Opera 36
Mar, 2016	Jul, 2015	Mar, 2015	Oct, 2015	Mar, 2016

Clases en JavaScript

- Clases
  - Ejemplo

```
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }

  age(x) {
    return x - this.year;
  }

}

let myCar = new Car("Ford", 2014);


let date = new Date();
let year = date.getFullYear();


alert("My "+ myCar.name + " is " + myCar.age(year) + " years old.");
```

Clases en JavaScript

- Clases – Getters, Setters y Encapsulamiento

```
class Car {
  #carname
  constructor(name) {
    this.#carname = name;
  }

 show() {
    return `Marca: ${this.#carname}`
  }

  get getCarname() {
    return this.#carname;
  }

  set setCarname(x) {
    this.#carname = x;
  }
}
```

```
let c = new Car('Volvo');
console.log(c.show());
c.setCarname='Kia';
console.log(c.show());
```

Clases en JavaScript

- Clases – Herencia: extends

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return 'I have a ' + this.carname;
  }
}

class Model extends Car {
  constructor(brand, mod) {
    super(brand);
    this.model = mod;
  }
  show() {
    return this.present() + ', it is a ' + this.model;
  }
}
```

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Cadenas de caracteres

```
var cadena1 = “Mi Cadena”;                  // typeof -> "string"
var cadena2 = new String(“Mi Cadena”); // typeof -> “objetc”
```

- Propiedades:
  - length : longitud de la cadena
- Métodos
  - charAt(indice): Devuelve el carácter situado en la posición especificada
  - indexOf(cad, indice): Devuelve la posición de cad desde el índice indicado
  - lastIndexOf(cad, indice): Devuelve la posición de la última ocurrencia de cad
  - split(separador): Devuelve un array a partir de la cadena
  - substring(Indice1,Indice2): Devuelve la subcadena entre indice1 e indice2
  - substr(indice, long): Devuelve la subcadena de long que comienza en índice
  - toLowerCase(): Devuelve la cadena en minúsculas
  - toUpperCase() : Devuelve la cadena en mayúsculas
  - trim(), trimLeft(), trimRight() : Elimina espacios en blanco

Cadenas de caracteres

ejemplo7_js.html

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo 7 JS. Clases</title>
<meta charset="utf-8" />
</head>
<body>

<h1>Ejemplo JS: Clases</h1>
<p>Analizando la cadena "Texto de Prueba"</p>

<ul>
<script type="text/javascript">
let cadena = "Texto de Prueba";
document.write("<li>length: "+cadena.length); //  15
document.write("<li>indexOf('e'): "+cadena.indexOf('e')+"</li>");  //  1
document.write("<li>lastIndexOf('e'): "+cadena.lastIndexOf('e')+"</li>"); //  12
document.write("<li>charAt(5): "+cadena.charAt(9)+"</li>");        //  P
document.write("<li>substring(6,7): "+cadena.substring(6, 8)+"</li>");  //  de
document.write("<li>substr(6, 2): "+cadena.substr(6, 2)+"</li>");     //  de
document.write("<li>toUpperCase(): "+cadena.toUpperCase()+"</li>");    //  TEXTO DE PRUEBA
document.write("<li>toLowerCase('d'): "+cadena.toLowerCase()+"</li>");    //  texto de prueba
</script>
</ul>

</body>
</html>
```

Arrays

- Colección, de tamaño **dinámico**, cuyos elementos **no tienen que ser del mismo tipo.**

  Los arrays pueden ser Indexados o asociativos

  - Indexados
    - let m = new Array( );
      m[0]= 10;
    - let n = new Array(10);
    - let vocales =new Array('a','e','i','o','u');
    - let pizas = [] ;
    - let pizas = ['Peperoni', 'Margarita', 'Cuatro quesos']

  - Asociativos
    - let horasTrabajadas = new Array();
    - horasTrabajadas['lunes']=20;

  Usos: pizas[3]="Barbacoa";
  let a= m[0]+4;
  horasFinSemana = horasTrabajadas['sabado'] + horasTrabajadas['domingo'];

Arrays

- Propiedades y métodos (*objArray.método*) [I]
  https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array

  length: devuelve el número de elementos del array

  indexOf(elto): Devuelve el índice donde está almacenado el elemento
  includes(elto): Devuelve true o false si el elemento está (o no) en el array
  push(elto) : Inserte el elemento al final del array
  pop(): extrae el último elemento
  unshit(elto): Inserta un elemento al principio
  shift(): Extraer primer elemento

  join(*separador*): Devuelve una cadena separada por el parámetro indicado
  concat(array) : Concatena el array al pasado por parámetror
  reverse() : Devuelve el array con los elementos en orden invertido.
  sort() : Devuelve el array ordenado siguiendo el orden lexicográfico

Arrays

- Propiedades y métodos (*objArray.método*) [II]

  https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array

  filter(predicado): Devuelve un array que contiene todos los elementos que cumplan el predicado pasado como parámetro.

  find(predicado): Devuelve el primer elemento que cumpla el predicado que se pasa como parámetro, o undefined si ninguno lo cumple.

  findIndex(predicado): Devuelve el índice del primer elemento del array que cumpla el predicado que se pasa como parámetro, o -1 si nunguno lo cumple.

  forEach(funcion): Llama a la función pasada como parámetro para todos los elementos del array.

  map(funcion): Devuelve un nuevo array que contiene el resultado de llamar a la función pasada como parámetro a todos los elementos del array.

  values(): Devuelve un Array Iterator que contiene los valores para cada índice del array.

  keys(): Devuelve un Array Iterator que contiene las claves de cada índice del array.

  Predicado y función suelen ser expresión lambda

  Iterador:

Arrays

ejemplo8_js.html

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo 10 JS. Clases</title>
<meta charset="utf-8" />
</head>
<body>

<h1>Ejemplo JS: Clases</h1>


<script type="text/javascript">

let numeros=new Array(5,2,1,4,3); // Array con 5 elementos
numeros[5]=0;  // Array [5,2,1,4,3,0]
numeros.push(6); // Array [5,2,1,4,3,0,6]
numeros.unshift(7); // Array [7,5,2,1,4,3,0,6]
for (x in numeros) {
    document.write(numeros[x]+" ");
}

let ultimo = numeros.pop(); // Extraido: 6 - Array [7,5,2,1,4,3,0]
document.write("<p>"+numeros.toString()+"</p>");

let primero = numeros.shift(); // Extraido: 7 - Array [5,2,1,4,3,0]
document.write("<p>"+numeros.toString()+"</p>");


</script>

</body>
</html>
```

Fechas

- Objeto **Date**: Manipulación de fechas

  - Declaración

    new Date(); // fecha actual

    new Date(EnteroMs_1-1-1970); // fecha que ha transcurrido desde 1ene70

    new Date(*año, mes, día[, hora, minutos, segundos, mlseg*]**);** // fecha establecida

    donde año: xxxx, mes: 0-11, día: 1-(según mes), hora: 0-23, min y seg: 0-59 y mlseg 0-999

Ejemplo:

```
var fechaActual = new Date();
```

Fechas

- Principales métodos

getDate() : Devuelve el día del mes actual como un entero entre 1 y 31.

getMonth() : Devuelve el mes del año actual como un entero entre 0 (ene) y 11 (dic).

getYear() : Devuelve el año actual como un entero.

getDay() : Devuelve el día de la semana actual como un entero entre 0 (D) y 6 (S).

getHours() : Devuelve la hora del día actual como un entero entre 0 y 23.

getMinutes() : Devuelve los minutos de la hora actual como un entero entre 0 y 59.

getSeconds() : Devuelve los segundos del minuto actual, un entero entre 0 y 59.

setDate(día_mes), setDay(día_semana), setHours(horas), setMinutes(minutos),

setMonth(mes), setSeconds(segundos), setTime(milisegundos), setYear(año): Modifican la parte de la fecha indicada en el parámetro.

Objetos Predefinidos en JavaScript

ejemplo9_js.html

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo JS. Objeto Date</title>
<meta charset="utf-8" />
</head>
<body>

<h1>Ejemplo JS: Date</h1>

<script type="text/javascript">

let diasSemana = ["Domingo", "Lunes", "Martes", "Miércoles",
                  "Jueves", "Viernes", "Sábado"];
let meses = new Array("Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
          "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre");

let fechaActual = new Date();

let stringFecha = "Hoy es: ";

stringFecha += diasSemana[fechaActual.getDay()] + ", ";
stringFecha += fechaActual.getDate() + " de ";
stringFecha += meses[fechaActual.getMonth()] + " de ";
stringFecha += fechaActual.getFullYear();

document.write(stringFecha);

</script>

</body>
</html>
```

Objetos Predefinidos en JavaScript

ejemplo10_js.html

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo JS. Objeto Date</title>
<meta charset="utf-8" />
</head>
<body>

<h1>Ejemplo JS: Date</h1>

<form>
Inserta dia(1-31)/mes(1-12)/año(XXXX):<br />
<label for="dia"></label><input id="dia" type="text" name="dia"/> /
<label for="mes"></label><input id="mes" type="text" name="mes"/> /
<label for="anio"></label><input id="anio" type="text" name="anio"/>
<input type="date" />
</form>

<script>
let diasSemana = ["Domingo", "Lunes", "Martes", "Miércoles",
                  "Jueves", "Viernes", "Sábado"];

let elto = document.getElementById("anio");
elto.onchange = function() {
    let fecha = new Date(anio.value, mes.value-1, dia.value);
    let d = diasSemana[fecha.getDay()];
    document.write("<h1>Naciste un " + d + "</h1>");
}
</script>

</body>
</html>
```

Funciones Matemáticas

- Objeto **Math**: Funciones matemáticas
  - Utilización

    Math. *Metodo(…)_o_propiedad*;

  - Propiedades
    - E: Número 'e', base de los logaritmos naturales (neperianos).
    - LN2 : Logaritmo neperiano de 2.
    - LN10 : Logaritmo neperiano de 10.
    - LOG2E : Logaritmo en base 2 de e.
    - LOG10E : Logaritmo en base 10 de e.
    - PI : Número PI.
    - SQRT1_2 : Raíz cuadrada de 1/2.
    - SQRT2 : Raíz cuadrada de 2.

    Ejemplo: Math.E; Math.PI

Funciones Matemáticas

- Métodos:
  - abs(num): Valor absoluto.
  - acos(num): Arcocoseno. 'num' debe estar en el rango [-1,1], si no devuelve NaN.
  - asin(num): Arcoseno. 'num' debe estar en el rango [-1,1], si no devuelve NaN.
  - atan(num): Arcotangente. Si 'num' no es numérico devuelve NaN.
  - atan2(x,y): Devuelve el ángulo formado por el vector (x,y) respecto al eje X.
  - ceil(num): Devuelve el entero obtenido de redondear en exceso 'num'.
  - cos(num): Coseno de 'num ' o NaN si éste no es numérico.
  - exp(num): Devuelve e^{num}.
  - floor(num): Devuelve el entero obtenido de redondear por defecto 'num'.
  - log(num): Devuelve el logaritmo neperiano de 'num'.
  - max(x,y): Devuelve el máximo de 'x' e 'y'.
  - min(x,y): Devuelve el mínimo de 'x' e 'y'.
  - pow(base,exp: Devuelve $base^{exp}$.
  - random(): Devuelve un número pseudoaleatorio entre 0 y 1.
  - round(num): Redondea 'num' al entero más cercano.
  - sin(num): Devuelve el seno de 'num' o NaN.
  - sqrt(num): Devuelve la raíz cuadrada de número.
  - tan(num): Devuelve la tangente de 'num' o NaN.

Contenido

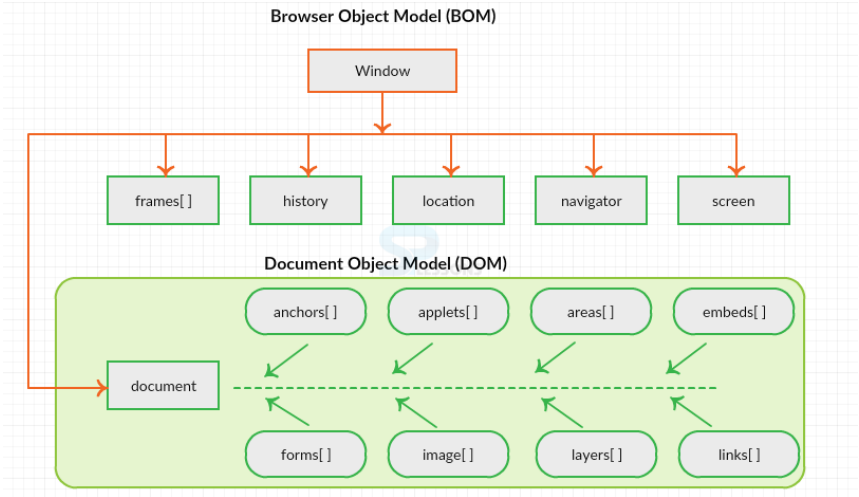
- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Objetos en JavaScript

- Objetos básicos

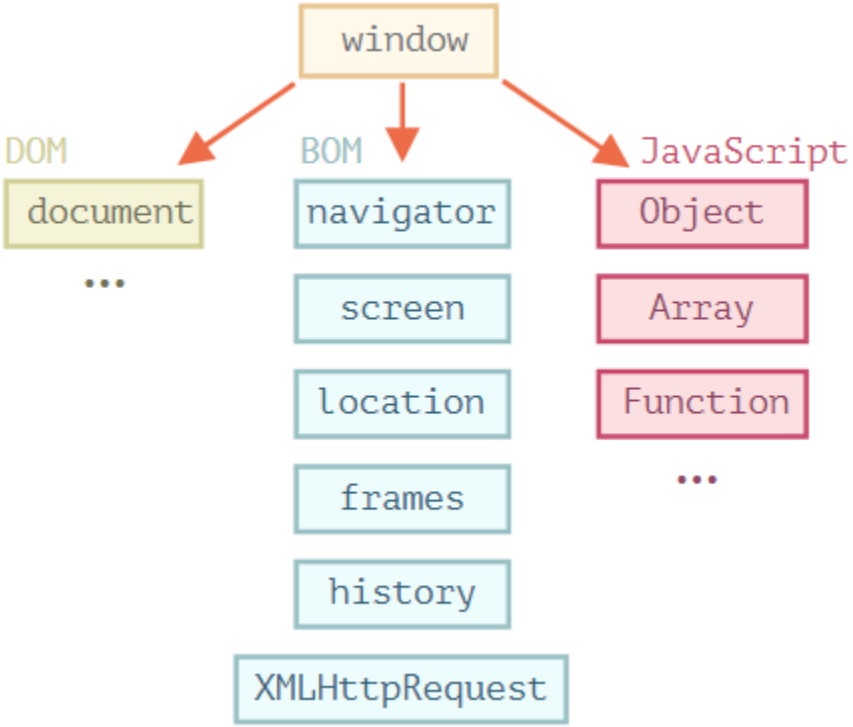
  Number
  Boolean
  String
  Array
  Date
  Math

- Modelo de Objetos del Navegador



Modelo de Objetos del Navegador

- BOM – Browser Object Model



Modelo de Objetos del Navegador

- **Objeto window**: Objeto principal de la jerarquía.

Propiedades principales
closed: Valor booleano que indica si la ventana está cerrada.
defaultStatus: Valor por defecto para la barra de estado del navegador.
document: Objeto document (DOM)
history: Array con las direcciones (URL's) visitadas en la ventana (historial).
innerHeight: Establece o devuelve el alto de la ventana actual
innerWidth: Establece o devuelve el ancho de la ventana actual
location : Dirección (URL) actual mostrada en ventana (objeto location).
name : Nombre de la ventana.
navigator: Objeto navigator
opener : Referencia al objeto window que abrió está ventana.
screen: Objeto Screen
self y window : Nombres alternativo para la ventana.
status : Información de la barra de estado.
top : Nombre de la ventana del nivel superior.

Modelo de Objetos del Navegador

- Objeto Windows

Métodos principales (I)

alert(msj) : Muestra el mensaje 'msj' en un cuadro de diálogo.

confirm(msj) : Muestra un cuadro de diálogo con el mensaje 'msj' y dos botones: Aceptar y Cancelar. Devuelve true si se pulsa aceptar y false si se pulsa cancelar.

prompt(msj, defecto) : Muestra un cuadro de diálogo con el mensaje "msj" y una caja de texto. El parámetro 'defecto' es opcional, y muestra la respuesta por defecto. El método devuelve la respuesta introducida en el cuadro de texto.

setTimeout(func, retardo) : Establece un retardo, en milisegundos, antes de evaluar func. Devuelve un identificador.

setInterval(func, retardo) : Establece un intervalo, en milisegundos, para evaluar "func", periódicamente. Devuelve un identificador.

clearTimeout(id) : Cancela el retardo "id".

clearInterval(id) : Cancela el intervalo "id".

Modelo de Objetos del Navegador

- Objeto Windows

Métodos principales (y II)

open(*url*, *nombre*, *opciones*) : Abre una ventana nueva con la *url* indicada. Donde *opciones* establecen las características de la nueva ventana. Estas son:

directories / location / menubar / resizable / scrollbars / status / toolbar (yes/no/ 1/0)

width / heigth / top / left / outerWidth / outerHeight (pixels)

Ejemplo: ventNueva=open(“”,”Prueba”, “directories=no, toolbar=yes”);

close() : Cierra la ventana.

focus() : Asigna el foco sobre la ventana.

blur() : Elimina el foco de la ventana.

moveBy(x, y) : Mueve la ventana hor. y vert. el número de pixels x e y, respect.

moveTo(x, y) : Mueve la ventana a las coordenadas (x, y).

scrollTo(x, y) : Desplaza el contenido de la ventana a las coordenadas (x, y).

scrollBy(x, y) : Desplaza el contenido x e y pixels, hor. y vert., respectivamente.

Modelo de Objetos del Navegador

ejemplo13_js.html

```
<!DOCTYPE html>
 <html>
 <head>
 <title> Ejemplo Objeto Windows </title>
 <meta charset="UTF-8" />
 </head>
 <body>
 <h1>Ejemplo con Objeto Window</h1>
 <script type="text/javascript">
 var v;
 function a()  {
   let opciones = "left=100,top=100,width=250,height=150";
   v = window.open("", "", opciones);
   v.document.write("<h1>Nueva Ventana</h1>");
   setTimeout("b()",2000); setTimeout("c()",4000); setTimeout("v.close()",6000);
  }

function b() {
    v.moveBy(200, 200);
    v.document.write("<p>Movimiento desde función b().</p>");
}
function c() {
    v.moveTo(500, 100);
    v.document.write("<p>Movimiento desde función c().</p>");
}
 a();
</script>
</body>
</html>
```

Modelo de Objetos del Navegador

- **Objeto Navigator**: Información del navegador

Propiedades

appName: Nombre del Navegador.

appVersion: Versión del Navegador.

platform: Nombre de la plataforma sobre la que se ejecuta el Navegador.

userAgent: Información del navegador enviada en la cabecera en una petición HTTP.

plugins: Array que contiene (name, description, filename, length) de todos los plugins soportados por el Navegador.

onLine: true si el navegador tiene acceso a Internet

geolocation: Geolocalización

javaEnabled(): Devuelve true si el navegador permite ejecutar Java applets

cookiesEnabled(): Devuelve true si el navegador tiene habilitadas las cookies.

Modelo de Objetos del Navegador

ejemplo11_js.html

```
<script>
function isNavegador(tipo) {
    let ok=false;
    let uA = navigator.userAgent.toLowerCase();
    if(uA.indexOf(tipo)!=-1) ok=true;
    return ok;
}
let browser = function() {
    let cadena;
    let navs = ["Firefox", "Chrome", "MSIE", "Opera"];
    let i=-1;
    for (n in navs) {
        let c = navs[n].toLowerCase();
        if (isNavegador(c)) {
            i=n;
            break;
        }
    }
    if (i!=-1) cadena = navs[i];
    else cadena="Desconocido";

    return cadena;
}

let txt = new String();

txt = "<p>CodeName: " + navigator.appCodeName + "</p>";
txt+= "<p>appName: " + navigator.appName + "</p>";
txt+= "<p>appVersion: " + navigator.appVersion + "</p>";
txt+= "<p>Cookies: " + navigator.cookieEnabled + "</p>";
txt+= "<p>Platform: " + navigator.platform + "</p>";
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";

txt += "<p>Navegador: " + browser() + "</p>";
document.getElementById("texto").innerHTML=txt;

</script>
```

Modelo de Objetos del Navegador

ejemplo12_js.html

```
<body>

<p id="msg">Click para conocer tu geolocalización</p>
<button onclick="getLocation()">Encuéntrame</button>

<div id="mapa"></div>

<script src="http://maps.google.com/maps/api/js?sensor=false"></script>

<script>
let m=document.getElementById("msg");
function getLocation()
  {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition,showError);
  }
  else{
    m.innerHTML="Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
  let lat=position.coords.latitude;
  let lon=position.coords.longitude;
  let latlon=new google.maps.LatLng(lat, lon)
  let mapa=document.getElementById('mapa')
  mapa.style.height='250px';
  mapa.style.width='500px';

  let myOptions={
  center:latlon,zoom:14,
  mapTypeId:google.maps.MapTypeId.ROADMAP,
  mapTypeControl:false,
  navigationControlOptions:{style:google.maps.NavigationControlStyle.SMALL}
  };
  let map=new google.maps.Map(mapa,myOptions);
  let marker=new google.maps.Marker({position:latlon,map:map,title:"Estás aquí!"});
  }
```

```
function showError(error)
  {
  switch(error.code)
    {
    case error.PERMISSION_DENIED:
      m.innerHTML="El usuario ha denegado la Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      m.innerHTML="Información de localización no disponible."
      break;
    case error.TIMEOUT:
      m.innerHTML="Petición fuera de tiempo."
      break;
    case error.UNKNOWN_ERROR:
      m.innerHTML="Error desconocido."
      break;
    }
  }
</script>
```

Modelo de Objetos del Navegador

- Objeto **history**: Histórico de páginas visitadas.

Propiedades:
length : Número de entradas en el historial.

Métodos:
back() : cargar la URL anterior en el historial.
forward() : cargar la URL siguiente dentro del historial.
go(pos) : cargar la URL especificado por 'pos' dentro del historial, valores negativos para back y positivos para forward.

Ejemplos:
```
history.back();  // Página anterior
history.go(-2);  // Dos páginas antes
```

Modelo de Objetos del Navegador

- Objeto **Location**: Información sobre la URL visualizada.

Propiedades:
hash : Nombre del enlace, dentro de la URL.
host : Nombre de dominio y número del puerto en la URL.
hostname : Nombre de dominio (o dirección IP) en la URL.
href : Cadena URL completa.
pathname : Ruta al recurso solicitado por la URL.
port : Puerto especificado en la URL
protocol : Protocolo utilizado (incluyendo los dos puntos), dentro de la URL.
search : Información pasada en una llamada a un script CGI, dentro de la URL.

Métodos:
reload() : vuelve a cargar la URL especificada en la propiedad href del objeto location.
assign(cadenaURL) : Carga un nuevo documento.
replace(cadenaURL) : Reemplaza el documento actual sin alterar el historial.

Ejemplo: location.replace("http://www.uhu.es");

Modelo de Objetos del Navegador

- Objeto **Screen**: Información sobre la pantalla.

Propiedades:
availHeight : Alto de la pantalla disponible (excluyendo barra de menús)
availWidth : Ancho de la pantalla disponible (excluyendo barra de menús)
colorDepth : Profundidad de color
height : Alto total de la pantalla
pixelDepth : Resolucion de la pantalla (en bits por pixel)
width : Ancho total de la pantalla

```
<body>
 <h1>Ejemplo con Objeto Screen</h1>
 <script type="text/javascript">
document.write("<p>Ancho Total: " + screen.width);
document.write("<p>Ancho Disponible: " + screen.availWidth);
document.write("<p>Alto Total: " + screen.height);
document.write("<p>Alto Disponible: " + screen.availHeight);
</script>
```

Modelo de Objetos del Navegador

- Modelo de Objeto del Documento
  - Objeto **document**: Contenido de la página visualizada.

Propiedades:

applets : Array con los applets existentes en el documento
body: Elemento <body>
cookie : Valores de las cookies del documento actual
doctype : el tipo de documento
documentElement : elemento <html>
documentURI : URL cargada (location)
domain : Nombre del servidor que ha servido el documento
forms : Array con los formularios del documento.
images : Array con todas las imágenes del documento.
lastModified : Fecha de la última modificación del documento.
links : Array con los enlaces externos
title : Título del documento actual
URL: Cadena completa con la URL

Modelo de Objetos del Navegador

- Objeto **document**: Contenido de la página visualizada.

Métodos:

close() : Cierra la escritura sobre el documento.
open(mime, "replace") : Abre la escritura sobre el documento, donde mime es el tipo de documento (text/html) y "replace" (opcional) si aparece indica que se utilice el historial

createAttribute("att") : Crea un nodo atributo. Usado con elto.setAttributeNode(att), que asigna el nodo atributo al elemento

createComment("texto") : Crea un nodo comentario con el texto.
Este nodo debe ser asignado a un padre con elto.appendChild(nodo).
createElement("tag") : Crea un nodo elemento con la tag. (appendChild() o insertBefore())
createTextNode("texto") : Crea un nodo texto. (appendChild())

getElementById("id") : Devuelve el elemento con identificador id
getElementsByTagName("tag"): Devuelve un array con los elementos tag
getElementsByName("name"): Devuelve un array con los elementos con nombre name

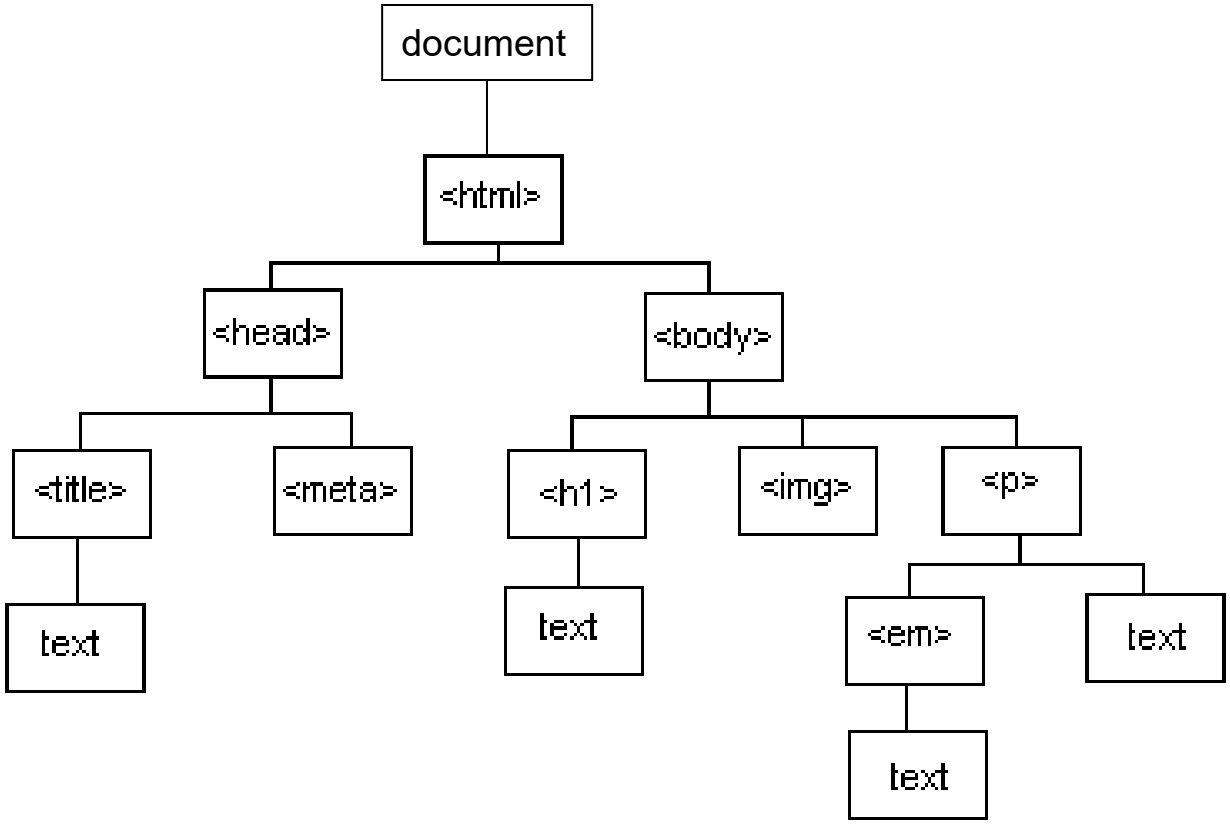
write() : Escribe texto en el documento.
writeln() : Escribe texto en el documento, y además lo finaliza con un salto de línea.

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Modelo de Objetos del documento

- DOM – Document Object Model



Modelo de Objetos del documento

- Modelo de Objetos del documento (objeto document)
  - Crear elementos y nodos de texto

```
var elto = document.createElement("span");
var texto = document.createTextNode("Hola Mundo!");
```

  - Añadir elementos: appendChild o append, o bien, insertBefore

```
elto.appendChild(texto);
document.body.appendChild(elto);

var miDiv = document.getElementById("miDiv");
miDiv.appendChild(elto);
```

  - Suprimir elementos: removeChild o remove

```
list.removeChild(list.childNodes[0]);

miDiv.parentNode.removeChild(elto);
```

Modelo de Objetos del documento

ejemplo14_js.html

- Ejemplo: Creando Elemento, nodos texto y añadiéndolo a otros elementos

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo JS - DOM</title>
<meta charset="UTF-8" />
</head>

<body>

<div id="area">
<p>Primer Párrafo</p>
</div>

<script type="text/javascript">

let elto = document.createElement("p");
let texto = document.createTextNode("Segundo Párrafo");
elto.appendChild(texto);

let area= document.getElementById("area");
area.appendChild(elto);

</script>

</body>
</html>
```

Modelo de Objetos del documento

- Modelo de Objetos del documento (objeto document)
  - Localizar elementos

document.getElementById(“id”) : Devuelve el elemento con identificador id
document.getElementsByTagName(“tag”): Devuelve un array con los elementos tag
document.getElementsByName(“name”): Devuelve un array con los elementos con nombre name
document.getElementsByClassName(“class”) : Array con los elementos con clase class
document.querySelector“selector”) : Primer elemento que cumplen el selector
document.querySelectorAll(“selector”) : NodeList con los elementos que cumplen el selector

  - Contenido de un elemento

elto.innerHTML= valor;
elto.innerText = valor;
elto.innerContentText = valor

  - Etiqueta y atributos de un elemento

elto.tagName
elto.id
elto.name
elto.className
element.attribute = valor
element.style.property = valor
elto.getAttribute(“atributo”);
elto.setAttribute(“atributo”, “valor”);
elto.removeAttribute(“atributo”);
elto.hasAttribute(“atributo”);

Modelo de Objetos del documento

ejemplo15_js.html

- Ejemplo: Comprobando Atributos, añadiendo y suprimiendo

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo JS - DOM</title>
<meta charset="UTF-8" />
</head>

<body>
<p id="p1" onclick="procesa('p1');">
<p id="p2" onclick="procesa('p2');">

<script type="text/javascript">

function procesa(id) {
  let elto = document.getElementById(id);
  if (elto.hasAttribute("title")) {
    elto.removeAttribute("title");
  } else {
    elto.setAttribute("title", "Parrafo "+id);
  }
}

</script>

</body>
</html>
```

Modelo de Objetos del documento

ejemplo16_js.html

- Ejemplo: Obteniendo elementos por etiqueta y cambiando su atributo class.

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo JS - DOM</title>
    <meta charset="UTF-8" />
    <style type="text/css">
    .blackWhite {
        background-color: black;
        color:white;
    }
    </style>

</head>
<body>

<p>
<p>
<p>

<button onclick="cambiarParrafos()">Click para cambiar éstilo de los párrafos</button>

<script type="text/javascript">
function cambiarParrafos() {
    let parrafos = document.getElementsByTagName("p");
    for (p in parrafos) {
        parrafos[p].className = "blackWhite";
    }
}
</script>


</body>
</html>
```

Contenido

- Introducción
- Fundamentos de JavaScript
  - Tipos básicos y variables
  - Operadores
  - Estructuras de Control
  - Funciones
  - Clases
- Objetos básicos
- Modelo de Objetos del Navegador (BOM)
- Modelo de Objetos del Documento (DOM)
- Eventos en JavaScript

Eventos en JavaScript

Eventos

Un suceso que se produce en un documento HTML y requiere tratamiento mediante un script

Posibles eventos

- cargar la página (onLoad , unLoad)
- clic en un enlace o botón (onClick)
- Movimiento del ratón por un elemento (onMouseOver, onMouseOut)
- Acciones en un formulario (onSubmit , onReset)
- Cambio del valor de un campo (onChange)
- Activar (foco) de un elemento (onFoco, onBlur)
- Selección de texto (onSelect)
- Pulsar teclas (onKeyDown, onKeyUp, onKeyPress)
- Pulsar boton del ratón (onMouseDown, onMouseUp)

Eventos en JavaScript

- Manejo de eventos

  Un objeto de tipo event es pasado a la función como parámetro

  - En el código HTML como atributos de las etiquetas

    <etiqueta onEvento="función()" ...>

    Ejemplo: UHU

  - Controladores de eventos (Event Handlers)

    var miElemento = document.getElementById("idElemento");

    miElemento.onEvento = funcion ; // no es posible varias funciones para un evento

  - Detectores de eventos (Event Listener)

    var miElemento = document.getElementById("idElemento");

    miElemento.addEventListener("Evento", funcion, false); // es posible registrar varias

    miElemento.removeEventListener("Evento", funcion, false);

Eventos en JavaScript

Evento	Causa	Principales Etiquetas
onLoad	El documento se carga	<body>
onUnload	El documento se descarga	<body>
onClick	Se pulsa el boton izq del ratón	button, submit, reset, radio, checkbox, <a href> y un área
onMouseOver	El ratón pasa sobre una zona	<a href> y cualquier área
onMouseOut	El ratón sale de una zona	<a href> y cualquier área
onSubmit	Se envía un formulario	<form>
onReset	Se resetea el formulario	<form>
onChange	Cambia el contenido	text y TextArea
onFocus	Se recibe el foco	text y TextArea
onBlur	Se pierde el foco	text y TextArea
onSelect	Se selecciona texto	text y TextArea
onAbord	Se interrumpe la carga	<body> <img>
onKeyDown onKeyUp onKeyPress	Pulsar una tecla Se deja de pulsar una tecla Se deja pulsada una tecla	Text y TextArea
onMouseDown onMouseUp	Pulsar un botón del ratón Dejar de pulsar un botón del ratón	<a href>, button, submit, reset
onResize	Redimensionar una ventana	<body>

Eventos en JavaScript

Evento	Causa	Principales Etiquetas
touchstart	Se toca la patalla táctil	<body>
touchend	Finaliza el tocar la pantalle táctil	<body>
touchmove	Mover el dedo por la pantalla	<body>
rientationchange	Se cambia de orientación	<body>

Para más información sobre los posibles eventos:

https://developer.mozilla.org/es/docs/Web/Events

https://www.w3schools.com/jsref/dom_obj_event.asp

Eventos en JavaScript

ejemplo17_js.html

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo JS - Eventos</title>
<meta charset="UTF-8" />
</head>

<body onload="entrar();">

<h1>Ejemplos de onLoad y onClick</h1>

<p id="par" onclick='tratarClick("link")'>Párrafo</p>

<table id="tab" border="1" onclick='tratarClick("tab")'>
<tr><td>Celda1</td><td>Celda2</td></tr></table>

<form id="form" name="Formulario" method="" action="">
<input type="checkbox" id="fcb" onclick='tratarClick("fcb")' /> CheckBox<br />
<input id="fbt" type="button" value="Enviar" onclick='tratarClick("fbt")' />
</form>

<script type="text/javascript">

function entrar(){
    alert("¡¡¡Bienvenido a mi Web!!!");
}

function tratarClick(id){
 switch(id) {
   case "link" : alert("Click en Enlace");break;
   case "tab" : alert("Click en Tabla");break;
   case "fbt" : alert("Click en boton de formulario");break;
   case "fcb" : alert("Click en checkbox de formulario");break;
  }
}

</script>
</body>
</html>
```

Eventos en JavaScript

ejemplo18_js.html

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo JS - Eventos</title>
<meta charset="UTF-8" />
<style type="text/css">
.sinRaton {
    color: #FFFFFF;
    background-color: #999999;
    border: thin ridge #006666;
}
.conRaton {
    color: #999999;
    background-color: #993300;
    border: thin inset #00CC66;
}
</style>
</head>

<body>
<h1>Ejemplo onMouserOver / onMouseOut </h1>
<p id="par" class="sinRaton"
    onMouseOver="colorFondo(true)"
    onMouseout="colorFondo(false)">
Pasa el ratón por aquí
</p>

<script type="text/javascript">
function colorFondo(bool) {
    var p=document.getElementById("par");
    if (bool) p.className = "conRaton";
    else p.className = "sinRaton";
}
</script>
</body>
</html>
```

Eventos en JavaScript

ejemplo19_js.html

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo JS - Eventos</title>
<meta charset="UTF-8" />
</head>

<body>

<h1>Ejemplo de onSubmit</h1>

<form id="form" method="post" action="procesar.jsp" onsubmit="return validar()">
Nombre: <input type="text" name="nombre" /><br />
Apellido: <input type="text" name="apellido" /><br />
Correo : <input type="text" name="email" /><br />
<input type="submit" value="Enviar" />
</form>


<script type="text/javascript">

function validar(){
  var f = document.getElementById("form");
  if(f.nombre.value=="") {alert("Introduce Nombre"); return false;}
  if(f.apellido.value=="") {alert("Introduce Apellido"); return false;}
  if(f.email.value=="") {alert("Introduce Correo"); return false;}
  if(f.email.value.indexOf("@", 0)==-1) {
    alert("Introduce Correo correcto"); return false;
  }
  return true;
}

</script>
</body>
</html>
```

Eventos en JavaScript

ejemplo20_js.html

```
<html>
<head>
<title>Ejemplo eventos JavaScript</title>
<meta charset="UTF8" />
</head>
<body>

<h1>Ejemplo eventos JavaScript</h1>

<table id="tabla" border="1">
  <tr id="fila">
    <td id="celda">ClickMe</td>
  </td>
</table>

<script>
  var t = document.getElementById("tabla");
  var f = document.getElementById("fila");
  var c = document.getElementById("celda");

  t.addEventListener("click", function() { alert("Tabla"); }, false);
  f.addEventListener("click", function() { alert("Fila"); }, false);
  c.addEventListener("click", function() { alert("Celda"); }, false);

  f.addEventListener("click", function() { alert("Fila Nueva"); }, true);
</script>

</body>
</html>
```

Drag and Drop

ejemplo21_js.html

```
<script>
    // Requiere que el elemento receptor capture el evento ondragover para permitir el Drop

    // Función que permite el Drop en el objeto receptor
    function allowDrop(ev) {
      ev.preventDefault();
    }

    // Requiere que el elemento desplazable capture el evento ondragstart para iniciar el desplazamiento

    // Función que inicia el desplazamiento del objeto
    function drag(ev) {
      ev.dataTransfer.setData("text", ev.target.id);
    }

    // Requiere que el elemento receptor capture el evento ondrop para finalizar el proceso

    // Función que finaliza el proceso Drag and Drop
    function drop(ev) {
      ev.preventDefault();
      let data = ev.dataTransfer.getData("text");
      ev.target.appendChild(document.getElementById(data));
    }
</script>
```

```
<body>


<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<div id="div2" draggable="true" ondragstart="drag(event)">

</body>
```