

Instituto de Enseñanza Secundaria
Santiago Hernández

DESARROLLO WEB EN ENTORNO CLIENTE

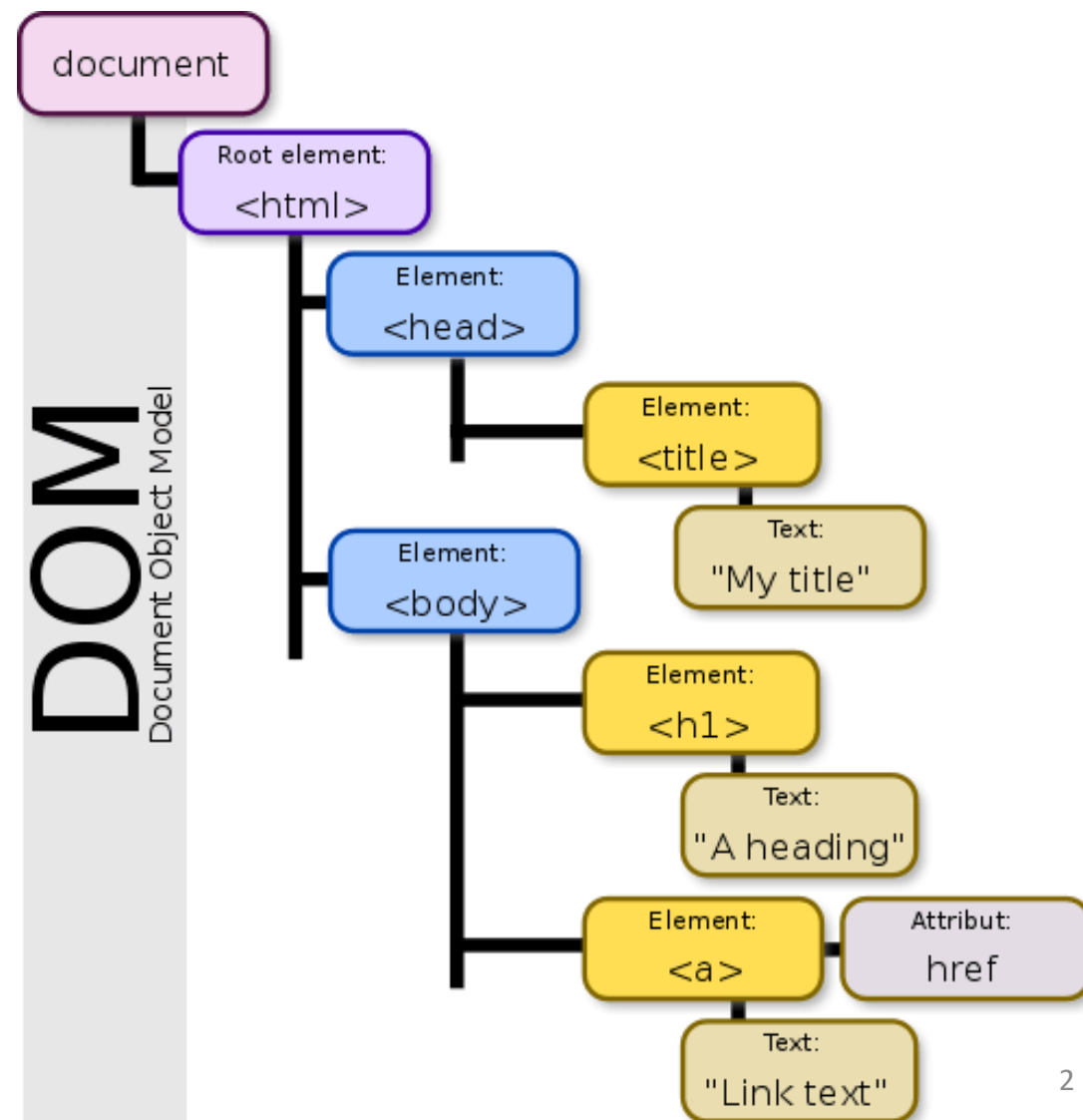
UD2. INTRODUCCIÓN A JAVASCRIPT. PARTE 2

DOM, EVENTO Y FORMULARIOS.

DOM

Document Object Model:

- Representación estructurada de un documento XML/HTML generada automáticamente por los navegadores.
- Permite acceder y modificar elementos de un documento web con programación (no solo JavaScript).
- Estándares W3C DOM y WHATWG DOM. Muchos navegadores extienden los estándares → cuidado.
- Formada por nodos y objetos con propiedades y métodos.
- ~~Etiqueta → Nodo.~~



DOM

Tipos de nodos generados:

➤ Habituales

- Document: Nodo raíz (<html>)
- Element: etiquetas HTML. Puede contener atributos y ser un nodo padre con nodos hijos y/o hermanos (siblings).
- Attr: se genera un nodo de este tipo para cada pareja atributo=valor.
- Text: nodo que contiene el texto de una etiqueta HTML.
- Comment: nodo que contiene los comentarios de la página.

➤ No tan habituales:

- DocumentType, CDataSection, DocumentFragment, Entity, EntityReference, ProcessingInstruction y Notation.

Ejemplo:

https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=802:tipos-de-nodos-dom-document-element-text-attribute-comment-arbol-de-nodos-para-javascript-cu01124e&catid=78&Itemid=206

DOM

Acceso a los nodos

- Solamente es posible cuando el árbol DOM ha sido construido completamente → página HTML cargada por completo → evento onload.
- Acceso jerárquico descendiendo desde el nodo raíz.
- Acceso directo: se utilizan APIs de los navegadores.
 - El objeto que devuelven es una “live collection” de HTML, que se adapta a los cambios del DOM.
 - `getElementById(id)`: devuelve el **elemento** con un id concreto.
 - `getElementsByClassName(clase)`: devuelve todos los elementos con una clase concreta.
 - `getElementsByName(nombre)`: devuelve todos los elementos con ese nombre.
 - `getElementsByTagName(tag)`: devuelve todos los elementos con ese tag.
 - Los selectores pueden combinarse para seleccionar los elementos.

DOM

Navegación entre nodos

- Una vez accedido un nodo con las APIs anteriores o desde la raíz podemos usarlo como referencia para utilizar las siguientes propiedades:
 - `childNodes[nodenumbr]`: devuelve una *live collection* con los nodos hijo del nodo. El acceso a cada nodo hijo se hace como en un *array*.
 - `parentNode`: devuelve el nodo padre del elemento.
 - `firstChild`: devuelve el primer nodo hijo del elemento o *null* si no tiene hijos.
 - `lastChild`: devuelve el último nodo hijo del elemento o *null* si no tiene hijos.
 - `nextSibling`: devuelve el siguiente nodo hermano del elemento o *null* si es el último.
 - `previousSibling`: devuelve el anterior nodo hermano del elemento o *null* si es el primero.

DOM

Creación de nodos con herramientas de DOM:

- Supone generar partes de código HTML mediante JavaScript.
- Creación de un nodo de tipo Element que represente al elemento.
 - `var parrafo = document.createElement("p");`
- Creación de un nodo de tipo Text que represente el contenido del elemento.
 - `var contenido = document.createTextNode("Hola Mundo!");`
- Añadir el nodo Text como nodo hijo del nodo Element.
 - `parrafo.appendChild(contenido);`
- Añadir el nodo Element a la página, en forma de nodo hijo del nodo correspondiente al lugar en el que se quiere insertar el elemento.
 - `document.body.appendChild(parrafo);`

DOM

Eliminación de nodos

- Localizar el nodo que se pretende eliminar.
 - `var parrafo = document.getElementById("provisional");`
- Acceder a su nodo padre para poder eliminarlo como nodo hijo:
 - `parrafo.parentNode.removeChild(parrafo);`

DOM

Acceso a los atributos de los nodos

- Valor de un atributo: nombre del nodo seguido del nombre del atributo HTML.
 - *`var enlace = document.getElementById("enlace");`*
 - *`alert(enlace.href);`*
- Propiedades CSS: haciendo uso del atributo style seguido de la propiedad:
 - *`var imagen = document.getElementById("imagen");`*
 - *`alert(imagen.style.margin);`*
- Si el nombre de una propiedad CSS es compuesto los guiones se eliminan y se capitaliza la letra siguiente:
 - font-weight se transforma en fontWeight.
- Atributo class:
 - class es palabra reservada en JavaScript → se modifica al generar el DOM.
 - class se transforma en className.

DOM

Propiedad innerHTML:

- Es otra forma de acceder al contenido del documento HTML y modificarlo. [Ej.](#)
- Permite pasar todo el contenido como una cadena de texto.
- Para acceder a las propiedades de un elemento HTML:
 - *`var x = document.getElementById("myList").innerHTML;`*
- Para modificar las propiedades:
 - *`document.getElementById("myP").innerHTML = "Hello Dolly.";`*
- Para añadir nodos:
 - *`var d = document.getElementById("d1");`*
 - *`d.innerHTML += "<p class='parrafos'>párrafo de prueba con innerHTML</p>";`*

EXPRESIONES REGULARES

Definición:

- Son patrones que usamos para encontrar una determinada combinación de caracteres en una cadena de texto.
- En JavaScript se representan por el objeto RegExp.
 - `var patron = /pato/;`
 - `var patron = new RegExp("pato");`
- Pueden ser un carácter o un patrón elaborado.
 - `var re = /abc/;`
 - `var re = /ab*c/;` //utiliza caracteres especiales
- Propiedades:
 - `ignoreCase` **Ignora mayusculas y minusculas (devuelve true si lo encuentra, si no false) /abc/i**
 - `lastMatch...` **te dice la ultima coincidencia.
Para buscar en toda la cadena: /abc/g**

EXPRESIONES REGULARES

Definición:

➤ Métodos de RegExp:

▪ *Exec*:

- `var patt1=new RegExp("e");`
- `document.write(patt1.exec("The best things in life are free"));`
- Devuelve e porque hay una e en la cadena.

var patt1=/e/. Devuelve lo que hay entre las barras. Lo muestra en HTML. Devuelve una e (el patron marcado) si existe

▪ *Test*:

- `var patt1=new RegExp("e");`
- `document.write(patt1.test("The best things in life are free"));`
- Devuelve true porque hay una "e" en la cadena.

test devuelve true o false, si lo encuentra true, si no false

➤ Métodos *match*, *search* y *split* de *String*

EXPRESIONES REGULARES

Caracteres especiales:

- Cuando la búsqueda requiere algo más que una coincidencia exacta, se usan patrones elaborados con caracteres especiales.

- **^** Indica coincidencia al principio de la cadena

/^abc/ busca dentro de la cadena a buscar, lo que empiece por abc

- **\$** Indica coincidencia al final de la cadena.

/abc\$/ buscará la coincidencia que esté al final, lo que acabe por abc

/alumn[ao] busca alumno o alumna

- `var cadena = /alumn[ao]/;` //buscaría tanto alumno como alumna

- `var cadena = /alumno[0-9]/;` //bucaría alumno0, alumno1, ... hasta alumno9

- `var cadena = /alumno[0-9a-zA]/;` //Formaría cadenas como alumno5, alumnor o alumnoA

Busca alumno + un numero del 0 al 9

Busca cadenas que tuviesen alumno + un num0-9 o a-z o A. Si no, seria cada posibilidad con un []

EXPRESIONES REGULARES

Caracteres especiales:

Buscar como patron del simbolo de dividir \ (contrabarra + barra)
Buscar asterisco *

➤ Repeticiones:

- **+** indica que lo que tiene a su izquierda puede estar 1 o mas veces.
- ***** indica que puede estar 0 o mas veces
- **?** indica opcionalidad, lo que tenemos a la izquierda puede aparecer 0 o 1 vez.
- **{3}** Indicar exactamente el número de veces que puede aparecer (3 en este caso).

➤ Comodines:

- **\d** un dígito. Equivale a **[0-9]**
- **\D** cualquier caracter que no sea un dígito.
- **\w** Cualquier carácter alfanumérico. Equivalente a **[a-zA-Z0-9_]**.
- **\W** cualquier carácter no alfanumérico **-_*\$...**
- **\s** espacio
- **\t** tabulador

EVENTOS

Programación basada en eventos

- Permite que los usuarios transmitan información a los programas.
- Los scripts quedan a la espera de que el usuario genere un evento.
- El navegador también puede generar eventos.
- El script procesa la información asociada al evento y genera un resultado.
- JavaScript define numerosos eventos que permiten la interacción del usuario.
- A cada evento puede asociársele una función (“event handlers”).
- Cada ejecución es lanzada por un “event listener”.
- Lista de eventos: https://www.w3schools.com/jsref/dom_obj_event.asp
- Relación evento-tipo de elemento: <https://uniwebsidad.com/libros/ajax/capitulo-6/modelo-basico-de-eventos?from=librosweb>

EVENTOS EN JAVASCRIPT

- El nombre de cada evento se construye mediante el prefijo `on`, seguido del nombre en inglés de la acción asociada al evento.
- Los eventos más utilizados en las aplicaciones web tradicionales:
 - `onload`: para esperar a que se cargue la página por completo.
 - `onclick`, `onmouseover`, `onmouseout`: permiten controlar el ratón.
 - `onsubmit`: para controlar el envío de los formularios.
- Las acciones típicas que realiza un usuario en una página web pueden dar lugar a una sucesión de eventos. Ejemplo:
 - Pulsar un botón de tipo `<input type="submit">` desencadena consecutivamente:
 - `onmousedown`
 - `onclick`
 - `onmouseup`
 - `onsubmit`

EVENTOS

Definición

```
document.getElementById("id").addEventListener("click",funcion,false);  
document.getElementById("id").addEventListener("click",funcion2,false);
```

- Integrados en el código HTML: atributo con el evento especificado:
 - `<p onclick="alert('me has pulsado');">Púlsame</p>`
 - `<p onmouseover="this.style.color='red'">Pasa sobre mi</p>`
- Integrados con funciones externas: atributo con el evento llamando a una función:
 - `<p onclick="alerta('roja');">Púlsame para alerta roja</p>;`
 - No permiten el uso de this.
- Manejadores semánticos: requiere ubicarse primero en el nodo del DOM (id) y luego añadir el evento:
 - `document.getElementById('id_elemento').onclick = funcionClick;`
 - Importante no terminar la función con () para que no se asigne el resultado sino el evento. JavaScript ejecuta automáticamente todo función con ().
- Mediante el uso de métodos: addEventListener / removeEventListener

EVENTOS

Uso de la variable this:

➤ Ejemplo de elemento HTML:

- `<div id="contenidos" style="width:150px; height:60px; border:thin solid silver">`
- Sección de contenidos...
- `</div>`

➤ Modificación de bordes al pasar el ratón por encima:

■ Sin this:

- `<div id="contenidos" style="width:150px; height:60px; border:thin solid silver" onmouseover="document.getElementById('contenidos').style.borderColor='black';" onmouseout="document.getElementById('contenidos').style.borderColor='silver';">`

■ Con this:

- `<div id="contenidos" style="width:150px; height:60px; border:thin solid silver" onmouseover="this.style.borderColor='black';" onmouseout="this.style.borderColor='silver';">`

EVENTOS

Manejadores de eventos del DOM

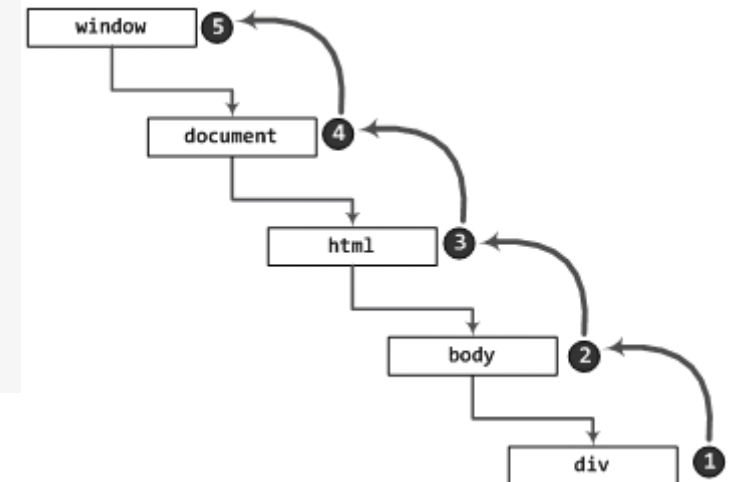
- Permiten asociar varios handler a un mismo evento.
- Sintaxis:
 - `document.getElementById('id_elemento').addEventListener("click", funcionClick, false);`
 - el evento se indica sin el prefijo 'on' y el tercer parámetro indica si el evento se emplea en la fase capture (true) o en la fase bubling (false)
 - Al igual que con los manejadores semánticos no terminar la función con ()

FLUJO DE EVENTOS

Definición y tipos

- Orden en que se ejecutan los eventos asignados a cada elemento de la página.
- Existen muchas diferencias en el flujo de eventos de cada navegador.
- Event Bubbling: desde el elemento más específico hasta el menos específico:

```
<html onclick="procesaEvento()">
  <head><title>Ejemplo de flujo de eventos</title></head>
  <body onclick="procesaEvento()">
    <div onclick="procesaEvento()">Pincha aqui</div>
  </body>
</html>
```



- Event Capturing: a la inversa.
- Aunque **window** no es un objeto del DOM los navegadores recorren los ascendentes del elemento hasta el objeto window.

FORMULARIOS

NO RECOMENDABLE!!!

Definición:

- Mediante el array `forms` del objeto del DOM `document` y sus elementos:
 - `document.forms[0];` //primer formulario de la página web
 - `document.forms[0].elements[0];` //elemento 0 del formulario
 - Inconvenientes: alteración del orden por ser la Web un entorno cambiante.
- Mediante `id` y accediendo con los métodos del DOM:
 - `var formularioPrincipal = document.getElementById("formulario");`
 - `var primerElemento = document.getElementById("elemento");`
 - `<form name="formulario" id="formulario" >`
 - `<input type="text" name="elemento" id="elemento" />`
 - `</form>`

MÁS RECOMENDABLE...

FORMULARIOS

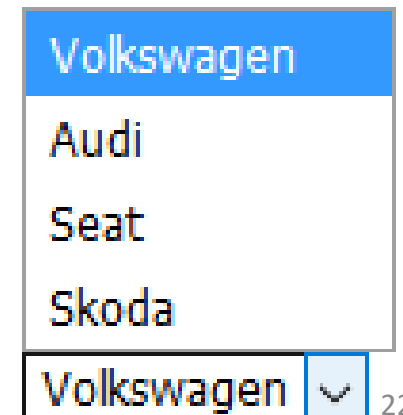
Definición:

- Mediante el atributo name del formulario y navegando por el DOM:
 - `var formularioPrincipal = document.formulario;`
 - `var primerElemento = document.formulario.elemento;`
 - `<form name="formulario">`
 - `<input type="text" name="elemento" />`
 - `</form>`

FORMULARIOS

Propiedades

- **type**: Indica el tipo de elemento del que se trata:
 - **text, textarea**: cuadros de texto. `<textarea name="area" ...></textarea>`
 - **radiobutton**: elementos excluyentes que **comparten name**. ☐ SI ☐ NO ☐ A VECES
 - **checkbox**: elementos no excluyentes. ☐ coche ☐ moto ☐ camion
 - **select**. `<select name=...>
<option value="1"...>
<option value="2"...>
</select>`
- **value**: obtiene el texto que se muestra en un botón o en
- **checked**: comprobar si un elemento ha sido marcado.
- **name**: se obtiene el valor del atributo name.
- **form**: para referirse al formulario de un elemento:
 - `document.getElementById("id_del_elemento").form`



Volkswagen

Audi

Seat

Skoda

Volkswagen ▼

FORMULARIOS

Eventos

- **onclick:** por ejemplo se usa al enviar un formulario.
 - `<form id="formulario" action="#">`
 - ...
 - `<input type="button" value="Enviar" onclick="this.disabled=true; this.value='Enviando...'; this.form.submit()" />`
 - `</form>`
- **onchange:** al cambiar el valor de un elemento y perder el foco
 - `var s= document.getElementById("sel1");`
 - `s.addEventListener("change",mostrarSelect,false);`
- **onfocus – onblur:** cuando el elemento tiene o pierde el foco.

este # lo envia
al propio html
o propia pagina

boton que envia
los datos

FORMULARIOS

Obtención de datos

➤ Cuadros de texto:

- `<input type="text" id="texto" />`
- `var valor = document.getElementById("texto").value;`
 - `<textarea id="parrafo"></textarea>`
- `var valor = document.getElementById("parrafo").value;`

FORMULARIOS

Obtención de datos

➤ Limitar número de caracteres de textarea:

■ function limita(maximoCaracteres) {

- var elemento = document.getElementById("texto");
- if(elemento.value.length >= maximoCaracteres) {
- return false;
- }
- else {
- return true;
- }

■ }

■ <textarea id="texto" onkeypress="return limita(100);"></textarea>

Limita las veces que nos
permite pulsar

FORMULARIOS

Obtención de datos

➤ Radiobutton:

- `<input type="radio" value="si" name="pregunta" id="pregunta_si"/> SI`
- `<input type="radio" value="no" name="pregunta" id="pregunta_no"/> NO`
- `<input type="radio" value="nsnc" name="pregunta" id="pregunta_nsnc"/> NS/NC`
- `var elementos = document.getElementsByName("pregunta");`
- `for(var i=0; i<elementos.length; i++) {`
 - `alert(" Elemento: " + elementos[i].value + "\n Seleccionado: " + elementos[i].checked);`
- `}`

FORMULARIOS

Obtención de datos

➤ Checkbox

- ☐ He leído y acepto las **condiciones de contratación**.
- ☐ He leído y acepto el **Aviso Legal** y la **Política de Privacidad**

Enviar

- `<input type="checkbox" value="condiciones" name="condiciones" id="condiciones"/>` He leído y acepto las condiciones
- `<input type="checkbox" value="privacidad" name="privacidad" id="privacidad"/>` He leído la política de privacidad
- `var elemento = document.getElementById("condiciones");`
- `alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);`
- `elemento = document.getElementById("privacidad");`
- `alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);`

FORMULARIOS

Obtención de datos

➤ Listas select:

- `<select id="opciones" name="opciones">`
 - `<option value="1">Primer valor</option>`
 - `<option value="2">Segundo valor</option>`
- `</select>`
- `var lista = document.getElementById("opciones"); //acceso a la lista.`
- `var indiceSeleccionado = lista.selectedIndex; //acceso al índice de la opción.`
- `var opcionSeleccionada = lista.options[indiceSeleccionado]; //acceso a la opción.`
- `var textoSeleccionado = opcionSeleccionada.text; //acceso al texto.`
- `var valorSeleccionado = opcionSeleccionada.value; //acceso al valor.`