6.- Utilización del modelo de objetos del documento (DOM-DocumentObjectModel):

a) El modelo de objetos del documento (DOM).

b) Objetos del modelo. Propiedades y métodos de los objetos.

c) Acceso al documento desde código.

d) Programación de eventos.

e) Diferencias en las implementaciones del modelo.

f) Uso de librerías de terceros.

a) El modelo de objetos del documento (DOM).

DOM (DocumentObjectModel) Modelo de Objeto de Documento, es la representación que hace el navegador de las etiquetas HTML, sus atributos y el orden en que aparecen en el archivo.

El DOM proporciona la información necesaria para que JavaScript se comunique con los elementos de la página Web, proporciona las herramientas necesarias para navegar a través de la página y añadir cambios al HTML de la página. El DOM es un protocolo del World Wide Web Consortium (W3C).

El Modelo de Objetos del Documento (DOM) es una interfaz de programación de aplicaciones (API) para documentos HTML y XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento.

En el DOM los documentos tiene una estructura lógica que es muy parecida a un árbol, aunque más bien se parece más a un bosque.

Podemos afirmar que el navegador cuando recibe el código HTML se encargar de generar un árbol con la estructura del documento HTML.

El DOM presenta los documentos como una jerarquía de objetos Node (nodos).

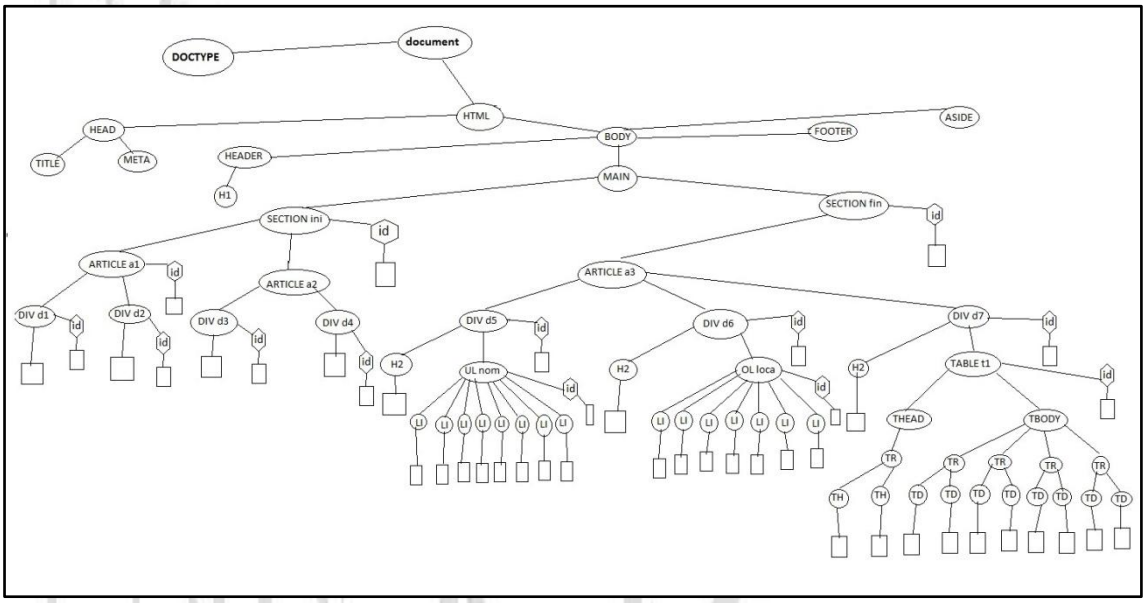
burgos-03.html

```
001 <!doctype html>
002 <html>
003 <head>
004 <title>Burgos</title>
005 <meta charset="utf-8"/>
006 </head>
007 <body>
008 <header>
009 <h1>Burgos</h1>
010 </header>
011 <main>
012 <section id="ini">
013 <article id="a1">
014 <div id="d1">
015                     Burgos es una ciudad española situada en el norte de la península ibérica, capital de la provincia homónima integrada en la comunidad autónoma de Castilla y León. Cuenta con unapoblación de 179.097 habitantes según los datos demográficos de 2013, repartidos en unasuperficie de 107,06 km², convirtiéndose en la 36.ª ciudad más poblada de España, y la segunda de la comunidad.
016 </div>
017 <div id="d2">
018                     Existen pruebas de asentamientos durante el Neolítico y la primera Edad del Hierro en el cerro del Castillo, el cual domina la ciudad. Sin embargo, se considera que la ciudad fue fundada como tal por el conde Diego Rodríguez "Porcelos" en el año 884. Hacia el año 930, se convirtió en capital del condado de Castilla primero dependiente del reino de León e independiente después por actuación del conde Fernán González. Fue la capital del reinode Castilla, y de manera intermitente de la Corona de Castilla, desde 1230 hasta el reinado de los Reyes Católicos. Estos dictaron en la ciudad en 1512 las Leyes de Burgos, las primeras leyes que la Monarquía Hispánica aplicó en América para organizar suconquista, germen de los actuales Derechos humanos. Posteriormente fue la capital de laantigua región histórica de Castilla la Vieja, y primera capital provisional de lacomunidad autónoma de Castilla y León.
019 </div>
020 </article>
021 <article id="a2">
022 <div id="d3">
023                     Cuenta con un anillo verde y un amplio conjunto monumental, entre los cuales destacan lacatedral de Santa María, exponente de la arquitectura gótica declarada Patrimonio de la Humanidad por la UNESCO,el monasterio de Santa María la Real de Las Huelgas y la cartujade Miraflores. Además, la ciudad es atravesada por el Camino de Santiago, otro Patrimonio de la Humanidad y se encuentra a menos de 15 km del yacimiento arqueológico de Atapuerca,también bajo la protección de la UNESCO desde el año 2000.7 El 13 de julio de 2010, abriósus puertas en la ciudad el Museo de la Evolución Humana, que expone los fósiles más importantes hallados en el yacimiento.
024 </div>
025 <div id="d4">
```

```
026                          La ciudad experimentó una fuerte industrialización durante el siglo XX, principalmente en torno al sector automovilístico y de alimentación, propiciada por su privilegiada localización geográfica, lo que le ha permitido convertirse en un importante nudo de comunicaciones en el norte de España, tanto de rutas nacionales como internacionales. Estos factores, unidos a un sector terciario desarrollado, con presencia destacada del turismo, la convierten en la 18ª ciudad en actividad económica de la nación.
027  </div>
028  </article>
029  <article id="a3">
030  <div id="d5">
031  <h2>Monumentos</h2>
032  <ul id="mon">
033  <li>Catedral</li>
034  <li>Monasteriod e San Juan</li>
035  <li>Iglesia de San Esteban</li>
036  <li>Murallas</li>
037  <li>Plaza del Cid</li>
038  <li>Iglesia de Santa María</li>
039  <li>Iglesia de San Gil</li>
040  <li>Iglesia de San Cosme y San Damián</li>
041  </ul>
042  </div>
043  <div id="d6">
044  <h2>Algunos Pueblos</h2>
045  <ol id="loca">
046  <li>Aranda de Duero</li>
047  <li>Covarrubias</li>
048  <li>Lerma</li>
049  <li>Medina de Pomar</li>
050  <li>Miranda de Ebro</li>
051  <li>Santa Gadea</li>
052  <li>Villarcayo</li>
053  </ol>
054  </div>
055  <div id="d7">
056  <h2>Producción agrária en 2009</h2>
057  <table id="t1">
058  <thead>
059  <tr>
060  <th>Cereal </th>
061  <th>Producción (T) </th>
062  </tr>
063  </thead>
064  <tbody>
065  <tr>
066  <td>Cebada </td>
067  <td>628.373 </td>
068  </tr>
069  <tr>
070  <td>Trigo </td>
071  <td>509.860 </td>
072  </tr>
073  <tr>
074  <td>Avena </td>
075  <td> 20.193</td>
076  </tr>
077  <tr>
078  <td>Centeno </td>
079  <td>7.898 </td>
080  </tr>
081  </tbody>
082  </table>
083  </div>
084  </article>
085  </section>
086  </main>
087  <footer>
088  </footer>
089  <aside>
090  </aside>
```

```
091 </body>
092 </html>
```

su árbol del DOM será



b) Objetos del modelo. Propiedades y métodos de los objetos.

Estas estructuras pueden considerarse como tipos de datos específicos del DOM y algunas de las más interesantes son:

- **Document**: nodo raíz del que derivan todos los demás nodos del árbol. Puede tener nodos hijos de tipo Element, ProcessingInstruction, Comment, DocumentType.
- **Node**: hace referencia a un nodo sea cual sea su tipo.
- **Element**: Es un tipo de datos que hace referencia a un nodo de tipo elemento (los nodos de tipo elemento son los que correspondiente a un elemento HTML, como p o form). Existen otros dos tipos de datos íntimamente relacionados con éste; por un lado tenemos uno más genérico llamado node y por otro lado tenemos uno más concreto llamado HTMLElement que hace referencia a nodos de tipo elemento HTML (no XML). Element hereda de node y HTMLElement hereda de element. Pude tener nodos hijos de tipo Element, Text, Comment, ProcesessingInstruction, CDATASection, EntityReference.
- **Attribute**: Es un tipo de datos que hace referencia a un nodo de tipo atributo. No debemos confundirlo con el tipo attr, que es un objeto asociado a un nodo de tipo atributo (¿y cuál es la diferencia? attribute hereda de Node y consecuentemente utiliza las propiedades nodeName y nodeValue, mientras que attr es un objeto que usa las propiedades name y value).
- **Attr**: Es un objeto asociado a un nodo de tipo atributo, pero en DOM 4 ya no hereda del objeto Node, de modo que no podemos utilizar nodeName ni nodeValue, pero en su lugar nos ofrece las propiedades name y value. Puede tener nodos hijos de tipo Text, EntityReference.
- **NodeList**: Es una colección **ordenada** de objetos de tipo nodo. Una colección nos permite acceder a sus elementos como si fuera un array, es decir, indicando el índice entre corchetes, pero no admite ciertos métodos que sí están disponibles en los arrays. Los nodeList pueden ser vivos o muertos (no vivos o estáticos, si lo prefiere); si son vivos, contendrán una referencia bidireccional con los nodos, de modo que

cualquier modificación que realicemos en ellos se reflejará en la representación del nodo en el documento, y cualquier modificación que pudieran sufrir esos nodos por acción de otra instrucción se reflejaría en el nodeList (**incluso si se añaden o eliminan nodos**); si son muertos la referencia también es bidireccional, pero sólo con los nodos que contenía el documento en el momento en el que fue creado el nodeList (al añadir o eliminar nodos posteriormente, el nodeList estático no será informado de este hecho). En la sección "Modificar el DOM" de este mismo tema encontrará un ejemplo sobre esta sutil diferencia entre nodeList vivo y muerto.

- **HTMLCollection**: Es una colección viva y ordenada (como un nodeList) de nodos de tipo elemento en la que se puede acceder a sus elementos por el índice o por el id de ese elemento usando en ambos casos la sintaxis de corchetes.
- **NamedNodeMap**: Es una colección viva de nodos, similar a nodeList, pero en la que además de acceder a los nodos por su índice (que en este caso y a diferencia de nodeList no corresponde a ningún orden convenido), podemos hacerlo por su nombre a través del método getNamedItem(*nombreDelNodo*).
- **Text**: nodo que contiene el texto encerrado por una etiqueta XHTML.
- **Comment**: representa los comentarios incluidos en la página XHTML.

El tipo **DOMString** sirve para almacenar secuencia de caracteres en formato unicode de 16 bits.

El tipo **DOMTimeStamp** se utiliza para almacenar una fecha absoluta o relativa. Es un número que representa milisegundos.

El tipo **DOMUserData** se usa para almacenar datos de la aplicación.

El tipo **DOMObject** se usa para recuperar un objeto.

El tipo interfaz Document

Representa el documento HTML completo. Hereda de Node

Propiedades

- **doctype**→ devuelve el tipo de documento.
- **documentElement**→ devuelve un valor que nos permite acceder directamente al nodo hijo, que es el elemento raíz del documento.
- **children**→ son todos los elementos hijos, es un objeto HTMLCollection.
- **firstElementChild**→ el primer hijo de tipo element.
- **lastElementChild**→ es el último hijo de tipo element.
- **childElementCount**→ número de elementos hijos de tipo element.
- **documentUri**→ devuelve la localización del documento.
- **URL**→devuelve la localización del documento.
- **implementation**→el objeto DOMImplementation.
- **compactMode**→ devuelve la cadena "BackCompat" si el modo del documento es "quirksmode" y devuelve "CSS1Compat" en otros casos.
- **characterSet**→devuelve la codificación usada para ese documento en el momento del análisis.
- **charset**→devuelve la codificación usada para ese documento en el momento del análisis.alias de characterSet.
- **inputEncoding**→ devuelve la codificación usada para ese documento en el momento del análisis.alias de characterSet.

Métodos

- **createElement**(*tipo*)→ crea un objeto de tipo Element del tipo indicado.

- **createTextNode**(*cadena*)→ crea un nodo Text con la cadena indicada
- **createAttribute**(*nombre-atributo*)→ devuelve un Attr con el nombre dado.
- **getElementById**(*identificador*)→ devuelve el elemento correspondiente al nodo con el identificados indicado (atributo id). El primero que encuentra, normalmente solamente debe existir uno con ese identificador.
- **getElementsByTagName**(*nombre-etiqueta*)→ devuelve un HTMLCollection de todos los elementos con el nombre de la etiqueta dado.
- **getElementsByClassName**(*nombre-clase*)→ devuelve los elementos que tiene esa clase, ese valor en el atributo **class**.HTMLCollection.
- **querySelector**(*selector*)→ devuelve el primer elemento correspondiente al selector indicado.
- **querySelectorAll**(*selector*)→ devuelve todos los elementos correspondientes al selector indicado. Es un NodeList.
- **importNode**(*nodo*, *subárbol*)→importa el nodo indicado, con subárbol se indica si se quiere importar el subárbol del nodo indicado a través de un valor lógico. devuelve una copia del nodo.
- **adopNode(***nodo***)**→Mueve el nodo indicadoperteneciente a otro documento. devuelve el nodo.
- **append**(nodo …) →añade el nodo indicado al final o los nodos indicados.
- **prepend**(nodo …) →añade el nodo indicado antes del primer hijo.
- **getElementsByName**(nombre) → devuelve los elementos que tienen ese nombre, ese valor en el atributo **name**. No se tiene en DOM 4.1
- **createEvent(***nombre-evento***)**→ crea el evento para ese evento.

El tipo interfaz Node

Es el tipo de dato primario para la integridad del DOM.

Propiedades

- **childNodes**→devuelve un nodoList con todos los nodos hijos de ese nodo.
- **parentNode**→ devuelve el nodo padre de ese nodo.
- **parentElement**→devuelve el elemet padre de ese nodo.
- **firstChild**→devuelve el primer nodo hijo.
- **lastChild**→devuelve el último nodo hijo.
- **nextSibling**→ devuelve el nodo siguiente a este nodo.
- **previousSibling**→ devuelve el nodo anterior a este nodo.
- **nodeName**→ devuelve el nombre de ese nodo.
  - **Element**
  - **Attr**
  - **Text**
  - **CDATASection**
  - **ProcessignInstruction**
  - **Comment**
  - **Document**
  - **DocumentType**
  - **DocumentFragment**
- **nodeType**→ devuelve el código del tipo de nodo u objeto.

NodeTypes - NamedConstants

NodeType	NamedConstant
1	ELEMENT_NODE
2	ATTRIBUTE_NODE

3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE

- **nodeValue**→contiene el valor de ese nodo. Utilizar en su lugar value.
- **ownerDocument**→ devuelve el objeto document asociado con ese nodo.
- **textContent**→ contiene el contenido de texto de ese nodo.
- **innerHTML**→ implementación HTML de un elemento. no en DOM 4
- **baseURI**→devuelve la URI absoluta de ese nodo.

Métodos

- **hasChildNodes()**→devuelve un valor lógico que nos indica si el nodo tiene algún nodo hijo.
- **appendChild(*nodo*)**→añade el nodo indicado al final de la lista de nodo hijos del ese nodo y devuelve el nodo añadido.
- **insertBefore(*nuevo-nodo*, *nodo-posición*)**→inserta el nuevo nodo antes del nodo posición, si este es null le añade al final. Devuelve el nodo insertado.
- **removeChild(*nodo*)**→quita el nodo hijo indicado. Devuelve el nodo quitado.
- **replaceChild(*nuevo-nodo*, *nodo-anterior*)**→ sustituye el nodo anterior por el nodo nuevo, siendo ambos nodos hijos del nodo actual. Devuelve el nodo anterior.
- **cloneNode(*subárbol*)**→devuelve una copia del nodo actual. Con subárbol indicamos con un valor lógico si queremos clonar su subárbol también.
- **isEqualNode(*nodo*)**→ indica si el nodo actual y el indicado son iguales Tienen las mismas propiedades.
- **isSameNode(*nodo*)**→ indica si el nodo actual y el indicado son los mismos.
- **contains(nodo)**→indica si ese nodo contiene el nodo indicado, es un descendiente.
- **compareDocumentPosition(nodo)** →devuelve un código (máscara) que nos indica la posición del nodo respecto a otro.

Constante	Valos	Significado
DOCUMENT_POSITION_DISCONNECTED	1	están en distinto árbol
DOCUMENT_POSITION_PRECEDING	2	cuando el nodo indicado es anterior.
DOCUMENT_POSITION_FOLLOWING	4	cuando el nodo indicado está más alla o sigue al nodo.
DOCUMENT_POSITION_CONTAINS	8	cuando el nodo indicado es un antecesor del nodo.
DOCUMENT_POSITION_CONTAINED_BY	10	cuando el nodo indicado es un descendiente del nodo.

Vamos a añadir un elemento a una lista no ordenada

```
001  var padre=document.getElementById("lista");
002  var elemento=document.createElement("li");
```

```
003  var valor=document.getElementById("provincia").value;
004  var dato=document.createTextNode(valor);
005  elemento.appendChild(dato);
006  padre.appendChild(elemento);
```

De esa manera se añaden elementos que pueden estar repetidos, si queremos que no estén repetidos deberemos poner:

```
001  var padre=document.getElementById("lista");
002  var valor=document.getElementById("provincia").value;
003  var todos=padre.getElementsByTagName("li");
004  var noexiste=true;
005  for(var i=0;i <todos.length;i++){
006      if(todos[i].textContent==valor)
007          noexiste=false;
008  }
009  if(noexiste){
010      var elemento=document.createElement("li");
011      var dato=document.createTextNode(valor);
012      elemento.appendChild(dato);
013      padre.appendChild(elemento);
014  }
```

El tipo interfaz Element

Representa un elemento en un documento HTML. Hereda de Node.

Propiedades

- **tagName**→ devuelve el nombre del elemento.
- **children**→ son todos los elemento hijos, es un objeto HTMLCollection.
- **firstElementChild**→ el primer hijo de tipo element.
- **lastElementChild**→ es el último hijo de tipo element.
- **childElementCount**→ número de elementos hijos de tipo element.
- **previousElementSibling**→ el hermano anterior element del elemento.
- **nextElementSibling**→ el hermano siguiente element del elemento.
- **attributes**→ es un NamedNodeMap con los atributos
- **namespaceURI**→ devuelve el namespaces.
- **prefix**→ devuelve el prefijo del namespaces
- **localName**→ nombre local
- **id**→identificador
- **className**→ clase

Métodos

- **querySelector**(***selector***)→ devuelve el primer elemento correspondiente al selector indicado.
- **querySelectorAll**(***selector***)→ devuelve todos los elementos correspondientes al selector indicado. Es un NodeList.
- **getElementsByTagName(*nombre-etiqueta*)**→ devuelve un HTMLCollection de todos los elementos descendentes que tiene la etiqueta cuyo nombre se indica.
- **getElementsByClassName(*nombre-clase*)**→ devuelve un HTMLCollection de todos los elementos descendentes que tiene la clase cuyo nombre se indica.
- **append**(nodo …)→ añade el nodo indicado al final o los nodos indicados.
- **prepend**(nodo …)→ añade el nodo indicado antes del primer hijo.
- **after**(nodo …)→ añade después del element.
- **before**(nodo …)→ añade delante del element.

- **remove**()→ elimina el element.
- **replaceWith**(nodo …)→ sustituye el elemento por el nodo.
- **getAttribute(*nombre*)**→ obtiene el valor del atributo cuyo nombre se indica.
- **getAttributeNode(*nombre*)**→ devuelve el nodo Attr del atributo cuyo nombre se indica.
- **hasAttribute(*nombre*)**→ nos indica si tiene el atributo cuyo nombre se indica.
- **removeAttribute(*nombre*)**→ elimina el atributo cuyo nombre se indica.
- **removeAttributeNode(*nodo-attr*)**→ elimina el nodo-attr indicado. Devuelve el elemento eliminado.
- **setAttribute(*nombre,valor*)**→ añade un nuevo atributo con el nombre y el valor indicado.
- **setAttributeNode(*nodo-attr*)**→ añade el nodo attr, si ya existe se reemplaza, devolviendo este elemento o null si le añade.
- **closest(*selector*)**→Devuelve el primer antecesor inclusivo (que comienza en el elemento) que coincide con los selectores y, de lo contrario, nulo.
- **matches(*selector*)**→Devuelve true si los selectores coinciden con la raíz del elemento, y false en caso contrario.
- **insertAdjacentElement(donde, elemento)**→ inserta el nodo como adyacente del elemento donde se indica, se va a insertar mediante uno de estos valores:
  - **beforebegin** delante del elemento padre
  - **afterbegin** antes del primer elemento secundario del elemento
  - **beforeend** antes de nulo
  - **afterend** antes que el siguiente hermano
- **insertAdjacentText(donde,texto)** →

El tipo interfaz NodeList

Proporciona la abstracción de un conjunto ordenado de nodos, son definir o restringir como se implementa este conjunto.

Propiedades

- **length**→ devuelve el número de nodos de la lista.
- **item(*posición*)**→ devuelve el elemento que ocupa la posición indicada.

El tipo interface HTMLCollection es un NodeList

Propiedades

- **length**→número de elementos en la lista.
- **item(posición)**→nodo que ocupa esa posición.
- **namedItem(nombre)**→devuelve el nodo que tiene ese nombre en el ido name.

El tipo interfaz NamedNodeMap

Los objetos que implementan la interfaz son utilizados para representar conjuntos de nodos a los que se pueden acceder por su nombre.

Propiedades

- **length**→ devuelve el número de nodos del mapa.

Métodos

- **getNamedItem(*nombre*)**→ devuelve el nodo cuyo nombre se indica.
- **removeNamedItem(*nombre*)**→ elimina el nodo cuyo nombre se indica. Devuelve el nodo quitado.

- **setNamedItem(*nodo*)**→ añade un nuevo nodo utilizando el atributo nodeName, si existía uno previo con ese nombre es reemplazo por el indicado. Devuelve null si añade uno nuevo o el nodo reemplazo.
- **item(*posicion*)**→ devuelve el nodo cuya posición se indica.

El tipo interfaz Attr

representa un atributo de un objeto Element. Hereda de Node.

Propiedades

- **name**→devuelve el nombre del atributo.
- **ownerElement**→ devuelve el Element correspondiente a este atributo.
- **specified**→ devuelve un valor lógico que indica si al atributo se le asigna un valor.
- **value**→ valor del atributo.
- **namespaceURI**→ devuelve el namespaces.
- **prefix**→ devuelve el prefijo del namespaces
- **localName**→ nombre local
- **nodeName**→ nombre del nodo

El tipo interfaz characterData

Extiende el Nodo con un conjunto de atributos y métodos para acceder a datos de caracteres DOM. Hereda de Node.

Propidades

- **data**→ cadena con los datos.
- **length**→ devuelve el número caracteres Unicode 16 bits.

Métodos

- **appendData(*cadena*)**→ añade la cadena al final de los datos.
- **deleteData(*posición*, *cantidad*)**→ borra de la cadena tantos caracteres como indica cantidad a partir de la posición indicada.
- **insertData(*posición*,*cadena*)**→ inserta la cadena indicada en la posición indicada de los datos.
- **replaceData(*posición*, *cantidad*, *cadena*)**→ sustituye los caracteres existes a partir de la posición indicada por posición en la cantidad indicada por la cadena indicada.
- **substringData(*posición*, *cantidad*)**→ devuelve una cadena con tantos caracteres como indica cantidad tomados de la cadena de datos a partir de la posición indicada.

El tipo interfaz Text

Es heredada de CharacterData y representa el contenido textual de un Element o Attr.

Propiedades

- **isElementContentWhitespace**→ devuelve un valor que indica si el texto contiene espacios en blanco.
- **wholeText**→ devuelve todo el texto de los nodos Text.

Métodos

- **replaceWholeText(*cadena*)**→ sustituye el texto por la cadena indicada en el nodo actual y los nodos textos adyacentes

- **splitText(*posición*)**→ divide ese nodo en dos nodos por la posición indica del texto. Devuelve el nuevo nodo.

El tipo interfaz Comment

Es heredada de CharacterData y representa el contenido de un comentario, es decir, todos los caracteres entre el '<!- -' y el '-- >'

El tipo interfaz TypeInfo

Representa el tipo referenciado de nodos Element o Attr, especificado en los esquemas asociados con el documento. Para XML. SE OMITE TODO.

c) Acceso al documento desde código.

Básicamente existen 4 formas de acceder a un elemento de un documento HTML:

- **A través de una propiedad directa**. Algunos elementos únicos, como head y body, son accesibles directamente a través de las propiedades head y body, respectivamente, del objeto document.
  Para acceder al body pondremos **document.body**.
  Para acceder al head pondremos **document.head**.
- **A través de una colección (HTMLCollection)**. Para otros elementos que no tienen por qué ser únicos, el objeto document nos ofrece propiedades que devuelven una colección ordenada de esos elementos, como forms, links, images, scripts, styleSheets, y frames (para iFrames).
- **Explorando el árbol de nodos**. Utilizar las propiedades **childNodes, children**, **firstChild**, **lastChild**,**firstElementChild**, **lastElementChild**, **previousSibling**, **nextSibling**, **previousElementSibling**, **nextElementSibling**, **parentElement, parentNode** y **ownerElement** procedentes de la interfaz nodo (al fin y al cabo el documento completo es también un nodo), para explorar los distintos elementos. Generalmente estas propiedades suelen utilizarse en combinación con nodeType y nodeName (ya tratadas en el tema anterior), o métodos como **hasChildNodes**(), que nos indica si el nodo contiene nodos anidados, para poder discriminar e identificar el nodo que nos interesa. Estas propiedades/métodos se describirán en la sección posterior de este mismo tema "Explorar el árbol DOM".
- **Buscándolo**. El objeto document nos ofrece los métodos **getElementById**() (obsérvese que es Element y no Elements, pues no puede haber más de un elemento con el mismo id en un documento HTML), **getElementsByTagName**() y **getElementsByClassName**(), con los que podemos buscar, respectivamente, los elementos HTML de un tipo concreto (similar a las propiedades forms, images, scripts, …, pero devolviendo un nodeList en lugar de una HTMLCollection), y los elementos que pertenecen a una clase determinada. En lugar de buscar por atributos del elemento, también podemos usar como criterio selectores CSS (recuerde del tema anterior que los selectores CSS pueden ser de etiqueta, de id y de clase, y que se pueden agrupar con una coma, o combinar mediante los operadores de descendiente (espacio), hijo (>) y hermano menor (+)) gracias a los métodos **querySelector**(*cadenaSelector*) y **querySelectorAll**(*cadenaSelector*); el primero devuelve el primer elemento que coincide con el selector, mientras que el segundo devuelve un nodeList de todos los elementos que coinciden con el selector.

Para acceder a un elemento del DOM pondremos

document.**getElementById**(*identificador*) → devuelve el elemento correspondiente al nodo con el identificados indicado (atributo id).

*Vamos a acceder al elemento cuyo nombre es **TablaCodigos**.*

```
001  var primero=document.getElementById("TablaCodigos");
```

document.**querySelector**(selector) → devuelve el primer elemento correspondiente al selector indicado. En el selector podemos cualquier selector que hemos visto con anterioridad.

*Vamos a acceder al elemento cuyo identificador es **TablaCodigos**.*

```
001  var primero=document.querySelector("#TablaCodigos");
```

*Vamos a acceder al primer elemento de la clase **Codigos**, que hay en el documento.*

```
001  var segundo=document.querySelector(".Codigos");
```

*Vamos a acceder al tbody que hay dentro de la tabla con el identificador **TablaCodigos**.*

```
001  var tercero=document.querySelector("#TablaCodigos>tbody");
```

nodo.**querySelector**(selector) → devuelve el primer elemento correspondiente al selector indicado a partir del nodo

element.**querySelector**(selector) → devuelve el primer elemento correspondiente al selector indicado a partir del nodo

*Vamos a acceder al tbody que hay dentro de la tabla con el identificador **TablaCodigos**.*

```
001  var primero=document.querySelector("#TablaCodigos");
002  var tercero=primero.querySelector("tbody");
```

*Vamos a acceder al primer elemento, li, de una lista ordenada llamada **ListaNombre**.*

```
001  var lista=document.querySelector("#ListaNombre");
002  var primerli=lista.querySelector("li");
```

Para obtener una serie de elementos tenemos

document.**getElementsByTagName**(*nombre-etiqueta*) → devuelve todos los elementos con el nombre de la etiqueta dado. HTMLCollection.

*Vamos a obtener todos las lista ordenadas (**ol**) que hay en el documento.*

```
001  var lista=document.getElementsByTagName("ol");
```

*Vamos a obtener todas las imágenes (**img**) que hay en el documento.*

```
001  var imagenes=document.getElementsByTagName("img");
```

document.**getElementsByClassName**(nombre-clase) → devuelve los elementos que tiene esa clase, ese valor en el atributo **class.** HTMLCollection.

*Vamos a obtener todos los elementos de la clase **Codigos***

```
001  var listacodigos = document.getElementsByClassName("Codigos");
```

document.**getElementsByName**(nombre) → devuelve los elementos que tienen ese nombre, ese valor en el atributo **name.** Ya no está disponible en DOM 4.1.

document.**querySelectorAll**(selector) → devuelve todos los elementos correspondientes al selector indicado.NodeList

*Vamos a obtener todas las lista ordenadas (**ol**) que hay en el documento.*

```
001  var lista=document.querySelectorAll("ol");
```

*Vamos a obtener todas las imágenes (**img**) que hay en el documento.*

```
001  var imagenes=document. querySelectorAll("img");
```

nodo.**querySelectorAll**(selector) → devuelve todos los elementos correspondientes al selector indicado. NodeList

element.**querySelectorAll**(selector) → devuelve todos los elementos correspondientes al selector indicado. NodeList

*Vamos a obtener todos los elementos, **li**, de la lista cuyo identificador es **listacodigos***

```
001 var lista = document.getElementById("listacodigos");
002 var datoslista=lista.querySelectorAll("li");
```

*Vamos a obtener todas las celdas, **td**, de una fila, **tr**, (la primera) en una tabla, cuyo identificador es **tablacc**.*

```
001 var tabla = document.querySelector("#tablacc");
002 var raiz=tabla.querySelector("tbody");
003 var lineas=raiz.querySelectorAll("tr");
004 var indice= 0;
005 var celdas=lineas.item(indice).querySelectorAll("td");
```

elemento.**getElementsByTagName(*nombre-etiqueta*)** → devuelve todos los elementos descendentes que tiene la etiqueta cuyo nombre se indica. HTMLCollection.

*Vamos a obtener todos los elementos, **li**, de la lista cuyo identificador es **listacodigos***

```
001 var lista = document.getElementById("listacodigos");
002 var datoslista=lista.getElementsByTagName("li");
```

*Vamos a obtener todas las celdas, **td**, de una fila, **tr**, (la primera) en una tabla, cuyo identificador es **tablacc***

```
001 var tabla = document.getElementById("tablacc");
002 var raiz=tabla.querySelector("tbody");
003 var lineas=raiz.getElementsByTagName("tr");
004 var indice= 0;
005 var celdas=lineas.item(indice).getElementsByTagName("td");
```

elemento.**getElementsByClassName**(nombre-clase) → devuelve los elementos que tiene esa clase, ese valor en el atributo **class.** HTMLCollection

*Vamos a obtener todos los elementos que hay en el elemento con identificador **sevilla**, correspondientes a la clase **monumentos**.*

```
001 var parrafo = document.getElementById("sevilla");
002 var todos=parrafo.getElementsByClassName("monumentos");
```

Para crear nodos tenemos

document.createElement(*tipo*) → crea un objeto de tipo Element del tipo (etiqueta de html) indicado.

*Vamos a crear un elemento de una lista, **li**, una fila para una tabla, **tr**, y un celda de una tabla, **td**.*

```
001 var elemento=document.createElement("li");
002 var linea=document.createElement("tr");
003 var celda=document.createElement("td"):
```

document.createTextNode(*cadena*) → crea un nodo de texto con la cadena indicada.

*Vamos a crea un node de tipo texto con el valor **0** y con el contenido de la variable **VstDato***

```
001 var datocelda=document.createTextNode(0);
002 var valor=document.createTextNode(VstDato);
```

nodo.**appendChild(*nodo*)** → añade el nodo indicado al final de la lista de nodo hijos del ese nodo y devuelve el nodo añadido.

Vamos a añadir una fila de una celda a una tabla, a su tbody.

```
001 celda.appendChild(datocelda);
002 linea.appendChild(celda);
003 raiz.appendChild(linea);
```

Vamos a añadir un elemento, con su valor, a una lista

```
001 elemento.appendChild(valor);
002 lista.appendChild(elemento);
```

nodo.**insertBefore(***nuevo-nodo***,** *nodo-posición***)** → inserta el nuevo nodo antes del nodo posición, si este es null le añade al final. Devuelve el nodo insertado.

Insertamos un elemento, li, en una lista delante de otro existente.

```
001 lista.insertBefore(elemento,datoslista.item(i));
```

Insertamos una fila en una tabla

```
001 raiz.insertBefore(linea,lineas.item(i));
```

nodo.**removeChild(***nodo***)** → quita el nodo hijo indicado. Devuelve el nodo quitado.

Borramos un elemento, li, de una lista.

```
001 lista.removeChild(datoslista.item(i));
```

Borramos una fila de una tabla

```
001 raiz.removeChild(lineas.item(i));
```

nodo.**replaceChild(***nuevo-nodo***,** *nodo-anterior***)** → sustituye el nodo anterior por el nodo nuevo, siendo ambos nodos hijos del nodo actual. Devuelve el nodo anterior.

Sustituimos un elemento, li, en una lista por otro existente.

```
001 lista.replaceChild(elemento,datoslista.item(i));
```

Sustituimos una fila en una tabla

```
001 raiz.replaceChild(linea,lineas.item(i));
```

document.prepend(*nodo*|*cadena* …**)** → añade el nodo indicado antes del primer hijo
*elemento***.prepend(***nodo*|*cadena* …**)** → añade el nodo indicado antes del primer hijo del elemento

Añadimos un elemento al inicio de una lista

```
001 lista.prepend(elemento);
```

document.append(*nodo*|*cadena* … **)** → añade el nodo indicado al final o los nodos indicados
*elemento***.append(***nodo*|*cadena* ,…**)** → añade el nodo indicado o los nodos indicados al final del elemento

Añadimos un elemento al final de una lista

```
001 lista.append(elemento);
```

*elemento***.before(***nodo*|*cadena* …**)** → añade el nodo delante del elemento

Insertamos una fila en una tabla

```
001 lineas.item(i).before(linea);
```

*elemento***.after(***nodo*|*cadena* …**)** → añade el nodo después del elemento

Insertamos una fila en una tabla

```
001 lineas.item(i).after(linea);
```

*elemento***.replaceWith(***nodo*|*cadena* …**)** → sustituye el elemento por el nodo

Sustituimos una fila en una tabla

```
001 lineas.item(i).replaceWith(linea);
```

*elemento***.remove()** → Borrar el elemento indicado

Borramos un elemento, li, de una lista.

```
001 datoslista.item(i).remove();
```

Valores

nodo.**nodeValue** → contiene el valor de ese nodo. Utilizar en su lugar **value**.
nodo.**value** → contiene el valor de ese nodo.
nodo.**textContent** → contiene el contenido de texto de ese nodo.
nodo.**innerHTML** → contiene el contenido de HTML de ese nodo no en dom 4

```
001  var VstNombre=document.getElementById("nombre").value;
002  var VstContenido=document.getElementById("texto").textContent;
```

Padres

nodo.**parentNode** → devuelve el nodo padre de ese nodo
nodo.**parentElement** → devuelve el elemento padre de ese nodo

Hijos

nodo.**hasChildNodes**() → devuelve un valor lógico que nos indica si el nodo tiene algún nodo hijo.
nodo.**childNodes** → devuelve un nodoList con todos los nodos hijos de ese nodo
nodo.**firstChild** → devuelve el primer nodo hijo.
nodo.**lastChild** → devuelve el último nodo hijo.

element.**children** → devuelve todos los hijos de ese elemento. HTMLCollection
element.**firstElementChild** → devuelve el primer hijo de tipo elemento de ese elemento.
element.**lastElementChild** → devuelve el último hijo de tipo elemento de ese elemento.
element.**childElementCount** → devuelve el número de elementos hijo de tipo elemento de ese

document.children → devuelve todos los hijos de document. HTMLCollection
document.firstElementChild → devuelve el primer hijo de tipo elemento de document
document.lastElementChild → devuelve el último hijo de tipo elemento de document
document.childElementCount → devuelve el número de elementos hijo de tipo elemento de

Hermanos

nodo.**nextSibling** → devuelve el nodo siguiente a este nodo.
nodo.**previousSibling** → devuelve el nodo anterior a este nodo
element.**nextElementSibling** → siguiente hermano de tipo elemento del elemento
element.**previousElementSibling** → hermano anterior de tipo elemento del elemento

Valores

nodo.**nodeValue** → contiene el valor de ese nodo. Utilizar en su lugar value.
nodo.**value** → contiene el valor de ese nodo
nodo.**textContent** → contiene el contenido de texto de ese nodo.
nodo.**innerHTML** → contiene el contenido de HTML de ese nodo no en dom 4

Atributos

document.createAttribute(*cadena***)** → crea el atributo indicado
element.**attributes** → devuelve los atributos del elemento
element.**hasAttribuites()** → indica si el elemento tiene atributos
element.**hasAttribute(***nombre***)** → nos indica si tiene el atributo cuyo nombre se indica.
element.**getAttribute(***nombre***)** → obtiene el valor del atributo cuyo nombre se indica

element.**setAttribute(***nombre, valor***)** → añade un nuevo atributo con el nombre y el valor indicado
element.**removeAttribute(***nombre***)** → elimina el atributo cuyo nombre se indica

Conjunto de elementosNodelist
nodelist.**item(***posicion***)** → elemento que ocupa la posición indicada en el conjunto
nodelist.**length** → número de elementos del conjunto

Conjunto de elementosHTMLCollection
htmlcollection.**length** → número de elementos del conjunto
htmlcollection.**item(***posicion***)** → elemento que ocupa la posición indicada en el conjunto
htmlcollection.**namedItem(***nombre*) → elemento con el nombre indicado en el conjunto

Más
document.**importNode**(*nodo*, *subárbol*) → importa el nodo indicado, con subárbol se indica si se quiere importar el subárbol del nodo indicado a través de un valor lógico
document.**adoptNode**(*nodo*) →
nodo.**cloneNode(***subárbol***)**→ devuelve una copia del nodo actual. Con subárbol indicamos con un valor lógico si queremos clonar su subárbol también

d) Programación de eventos.

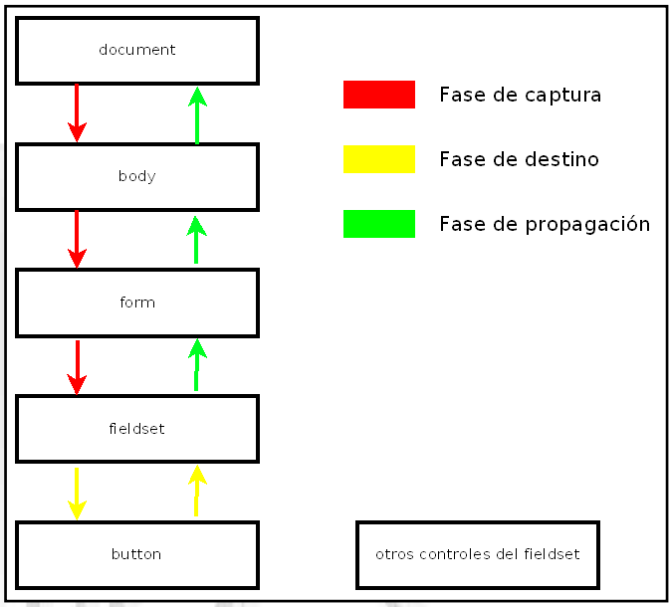
Los eventos son avisos que genera el navegador en respuesta a actuaciones del usuario (como hacer clic sobre un elemento) o del propio sistema (como terminar de cargar un documento en el navegador).

Los elementos pueden tener asociados oyentes de eventos, que son funciones que se ejecutan cuando el elemento es notificado del evento.

Las notificaciones de los eventos viajan por el árbol DOM, partiendo desde el elemento document y buscando el elemento que ha sido el disparador del aviso (por ejemplo, un botón sobre el que se ha hecho clic), y luego de vuelta al elemento document. Este modelo de propagación permite a cada elemento del DOM relacionado con un evento reaccionar ante él tanto antes como después de que reaccione el elemento disparador. El elemento disparador o blanco del evento se denomina técnicamente target. La comunicación del evento se divide en tres fases:

- **Fase de captura (*capture*):** Es el recorrido desde el elemento document hasta el padre del elemento target.
- **Fase de destino (*target*):** Es la comunicación del evento al elemento target.
- **Fase de propagación (*bubbling*)**: Es el recorrido de regreso desde el target hasta el document.

En la siguiente figura se reflejan estas 3 fases usando como ejemplo un elemento de tipo botón. Obsérvese que la comunicación es de padres a hijos en la fase de captura y de hijos a padres en la de destino, pero que en ningún caso se comunica a los hermanos.



Existen 3 formas de asociar un oyente de evento a un elemento:

- Usando el método **addEventListener(*nombreDeEvento*,*función*,*soloFaseCaptura*)**: Es la forma más recomendable. Este método está disponible en todos los objetos de nodo de elemento. El argumento *nombreDeEvento* es una cadena que indica qué evento queremos "escuchar" (debemos tener mucha precaución con esta cadena porque debe coincidir en mayúsculas y minúsculas con el nombre usado por la implementación del DOM en el navegador, no lleva on), y *función* es el bloque de código que vamos a ejecutar en respuesta a ese evento. Esta función recibe de la notificación un argumento que es un objeto con información sobre el evento. Si utilizamos true como *soloFaseCaptura*, el oyente sólo estará "pendiente" en las fases de captura y destino; si usamos false estará atento sólo en las de destino y propagación. En la siguiente tabla se describen las principales propiedades y métodos del objeto de evento que se envía como argumento a la función. Tenga en cuenta que esta colección de propiedades/métodos puede ser ampliada en función del tipo de evento (por ejemplo, los eventos relacionados con el teclado poseen además una propiedad llamada char que contiene el carácter correspondiente a la tecla).

```
001  document.body.addEventListener('click',mostrarInformacion,true);
002  var tabla =document.getElementById('tabla');
003  tabla.addEventListener('click',mostrarInformacion,true);
004  document.addEventListener('readystatechange',asignarEventos,false);
```

En Internet Explorer 8 y versiones anteriores, así como en Opera 6.0 y versiones anteriores el método **addEventListener** no funciona y en su lugar hay que utilizar el método **attchEvent**, cuyo formato es el siguiente.
attachEvent(*nombre-evento*, *función-ejecutar*)
en donde el nombre-evento lleva prefijo la palabra on.
con lo cual para poner un código independiente del navegador deberemos poner

```
001  var codigo=document.getElementById("cp");
002  if(document.addEventListener){
003  // soportaaddEventListener
004      codigo.addEventListener("click",pulsar);
005  }elseif(document.attachEvent){
006  // NO soporta addEventListener pero soporta attachEvent
007      codigo.attachEvent("onclick",pulsar);
008  }
```

*En la función **inicio** declaramos:*

- *Cuando se hace **click** sobre el elemento de identificador **aceptar** se ejecuta la función **pulsar**.*
- *Cuando **se va a enviar un formulario** de identificador **formulario** se ejecutará la función **validar**.*
- *Cuando **se pulsa una tecla** sobre el elemento de identificador **codigopostal** se va a realizar la función **digitos**.*
- *Cuando **se modifica el valor** del elemento con identificador **comunidades** se realizará la función **mostrar**.*

```
001 function inicio(){
002     var boton=document.getElementById("aceptar");
003     var formula=document.getelementById("formulario");
004     var codigo=document.getElementById("codigopostal");
005     var opciones=document.getElementById("comunidades");
006     if(document.addEventListener){
007         boton.addEventListener("click",pulsar);
008         formula.addEventListener("submit",validar);
009         codigo.addEventListener("keypress", digitos);
010         opciones.addEventListener("change",mostrar);
011     } else if (document.attachEvent){
012         boton.attachEvent("onclick",pulsar);
013         formula.attachEvent("onsubmit",validar);
014         codigo.attachEvent("onkeypress", digitos);
015         opciones.attachEvent("onchange",mostrar);
016     }
017 }
```

```
001 if(document.addEventListener)
002     window.addEventListener("load",iniciar)
003 else if(document.attachEvent)
004     window.attachEvent("onload",iniciar);
005 function iniciar(){
006     var primero=document.getElementById("primero");
007     var segundo=document.getElementById("segunda");
008     var tercero=document.getElementById("tercero");
009     var cuarto=document.getElementById("cuarto");
010     if(document.addEventListener){
011         primero.addEventListener("click",crearElemento);
012         segundo.addEventListener("click",nuevoElemento);
013         tercero.addEventListener("click",insertarElemento);
014         cuarto.addEventListener("change",borrarElemento);
015     }else if(document.attachEvent){
016         primero.attachEvent("onclick",crearElemento);
017         segundo.attachEvent("onclick",nuevoElemento);
018         tercero.attachEvent("onclick",insertarElemento);
019         cuarto.attachEvent("onchange",borrarElemento);
020     }
021 }
```

Vamos a ver los elementos del objeto **Event** objeto evento.
En las funciones manejadoras de evento el primer parámetro va a hacer referencia al evento, pero en internet explorer en la versión 8 y anteriores no es así, sino que el evento lo toma de la variable global **window.event**. Con lo cual deberemos poner:

```
001 function funcion(even){
002     var evento=even || window.event;
003 }
```

Propiedad/método	Descripción
currentTarget	Asociada al elemento al que se está notificando actualmente el evento. Recuerde que el evento es disparado por un elemento, el target o blanco, pero que es comunicado a todos sus ancestros en las fases de captura y propagación.
eventPhase	Devuelve un número que nos indica en qué fase de la

	notificación del evento nos encontramos (1 corresponde a la fase de captura, 2 a la de destino, y 3 a la de propagación).
target	Asociada al elemento que disparó el evento.
timeStamp	Contiene la hora a la que se disparó el evento expresada en número de milisegundos desde la época UNIX.
type	Indica de qué tipo de evento se trata.
cancelBubble	El establecedor del atributo cancelBubble debe establecer el indicador de propagación de detención del objeto de contexto si el valor dado es verdadero, y no hacer nada de lo contrario.
bubbles	Devuelve verdadero o falso según cómo se inicializó el evento. Verdadero si el evento pasa por los antepasados de su valor de atributo de destino en orden de árbol inverso, y falso de lo contrario.
cancelable	Devuelve verdadero o falso según cómo se inicializó el evento. Su valor de retorno no siempre tiene significado, pero verdadero puede indicar que parte de la operación durante la cual se envió el evento, puede cancelarse invocando el método preventDefault ().
defaultPrevented	Devuelve true si preventDefault () se invocó con éxito para indicar la cancelación, y false en caso contrario.
composed	Devuelve verdadero o falso según cómo se inicializó el evento. Verdadero si el evento invoca a oyentes más allá de un nodo ShadowRoot que es la raíz de su valor de atributo de destino, y en caso contrario, es falso.
isTrusted	Devuelve true si el agente de usuario ha enviado el evento, y false en caso contrario.
preventDefault()	Algunos elementos tienen asociados oyentes implícitos. Por ejemplo, al hacer clic sobre un enlace actúa implícitamente un oyente que redirige al navegador al href especificado en ese enlace, o al pulsar una tecla cuando el foco está en un campo de texto se inserta en él el carácter correspondiente a la tecla. Al ejecutar este método se anula la ejecución de estos oyentes implícitos. Si queremos que no se realice la opción por defecto del evento deberemos mandar ejecutar este método. Rechazar carácter pulsado en el evento keypess o no mandar el contenido del formulario en el evento submit.
stopPropagation()	Impide que el evento continúe su comunicación por el árbol DOM una vez atendidos los oyentes del currentTarget actual. Tenga en cuenta que un mismo elemento puede contar con varios oyentes para un mismo tipo de evento.
stopInmediatePropagation()	Impide que el evento continúe su comunicación una vez atendidos el oyente actual. Ni siquiera permite que se atiendan los demás oyentes que pudiera tener para ese evento el currentTarget.

composedPath()	

```
001 if(document.addEventListener)
002     window.addEventListener("load",comienzo)
003 else if(document.attachEvent)
004     window.attachEvent("onload",comienzo);
005 function comienzo(){
006     var cajanombre=document.getElementById("nombre");
007     var cajaedad=document.getElementById("edad");
008     if(document.addEventListener){
009         cajanombre.addEventListener("keypress",sololetras);
010         cajaedad.addEventListener("keypress",solodigitos);
011     }else if(document.attachEvent){
012         cajanombre.attachEvent("onkeypress",sololetras);
013         cajaedad.attachEvent("onkeypress",solodigitos);
014     }
015 }
016 function solodigitos(evnt){
017     var evento=evnt || window.event;
018     var dato=String.fromCharCode(evento.charCode);
019     if(dato <"0"|| dato >"9")
020         evento.preventDefault();
021 }
022 function sololetras(evnt){
023     var evento=evnt || window.event;
024     var dato=String.fromCharCode(evento.charCode).toLowerCase();
025     var adicionales="áéíóúüñ ";
026     if(dato <"a"|| dato >"z")
027        if(!adicionales.includes(dato))
028            evento.preventDefault();
029 }
```

- Utilizar un atributo de tipo **onevento** del elemento HTML. Los elementos HTML ofrecen atributos de evento, como **onclick** u **onfocus**, que están pendientes de que se produzca ese evento y responden a él con el código JavaScript que tengan asignado como valor. Dentro de ese código puede usarse la variable event, que contiene una referencia al objeto de evento, cuyas propiedades/métodos se describieron en la tabla anterior. Los oyentes definidos de este modo sólo están pendientes durante la fase de propagación.

```
001 <td onclick='mostrarInformacion(event);'>Celda</td>
```

- Utilizar una propiedad de tipo **onevento** del objeto de tipo elemento. Los atributos HTML del párrafo anterior son interpretados por el DOM como nodos de atributo, como cualquier otro atributo, pero también ofrece para ellos propiedades de acceso directo dentro del elemento, como **onclick** u **onfocus**. Estas propiedades admiten como valor el nombre de una función (a la que se enviará implícitamente el mismo argumento con información sobre el evento que a las funciones registradas con **addEventListener**). Los oyentes definidos de este modo sólo están pendientes durante la fase de destino y propagación.

  ***elemento*.onclick=*función*;**

```
001 var tabla =document.getElementById('tabla');
002 tabla.onclick=mostrarInformacion;
```

El resto de la sección lo dedicaremos a describir algunos de los eventos más importantes del DOM.

El objeto **document** posee el evento **readystatechange**, que se dispara cada vez que se produce un cambio de valor en la propiedad **readyState** (recuerde que esta propiedad

puede contener los valores '**loading**', '**interactive**' y '**complete**'). Este evento es muy útil para retrasar la asignación de otros oyentes de eventos hasta tener la certeza de que se ha creado el DOM completo del documento, pues de otro modo podría ocurrir que intentásemos asignar un evento a un objeto que aún no ha sido insertado en el DOM (tenga en cuenta que no está correctamente implementado en Opera).

Si deseamos quitar la asociación de una función con un evento vamos a utilizar el método **removeEventListener** cuyo formato es el siguiente

removeEventListener(*nombre-evento*, *nombre-función*, *fase-captura*)

En Internet Explorer 8 y versiones anteriores, así como en Opera 6.0 y versiones anteriores este método no funciona y en su lugar hay que utilizar el método **detachEvent**.

detachEvent(*nombre-evento* , *nombre-función*)

Deberemos tener en cuenta que el nombre del evento en este caso va precedido de la palabra on.

```
001  var codigo=document.getElementById("cp");
002  if(document.removeEventListener){
003  // soporta addEventListener
004     codigo.removeEventListener("click",pulsar);
005  }elseif(document.detachEvent){
006  // NO soporta removeEventListener pero soporta detachEvent
007     codigo.detachEvent("onclick",pulsar);
008  }
```

*Vamos a hacer que **no se ejecute** la función **pulsar** cuando **se hace click** sobre el elemento con identificador **aceptar**.*

```
001  var boton=document.getElementById("aceptar");
002  if(document.removeEventListener){
003  // soporta removeEventListener
004      boton.removeEventListener("click",pulsar);
005  } else if (document.detachEvent){
006  // NO soporta removeEventListener pero soporta detachEvent
007      boton.detachEvent("onclick",pulsar);
008  }
```

e) Diferencias en las implementaciones del modelo.

Implementaciones directas del DOM1, DOM2, DOM3 y DOM4 a través de funciones de terceros como jQuery o Angularjs.

f) Uso de librerías de terceros.

Utilización de JQuery para realizar las mismas operaciones.

jQuery es una biblioteca JavaScript rápida, pequeña y rica en funciones. Hace las cosas del documento HTML como el recorrido y la manipulación, el control de eventos, animación y Ajax. Es una API fácil de usar que funciona a través de una multitud de navegadores. Con una combinación de versatilidad y capacidad de ampliación, jQuery ha cambiado la forma en que millones de personas escriben JavaScript. Existen diversas versiones de jQuery y siempre es conveniente revisar las distintas versiones que hay y la compatibilidad con las versiones anteriores.

jQuery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

La versión actual actual es la 3.6.0

Para poder utilizar jquery deberemos incluir en nuestro documento html:

Versión completa, desde la página de jQuery

<script src="http://code.jquery.com/jquery-3.6.0.js"></script>

Versión minimalista, desde la página de jQuery
<script src="http://code.jquery.com/jquery-3.6.0.min.js"></script>
Versión minimalista desde el CDN de Microsoft
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.6.0.min.js"></script>
Versión minimalista desde el CDN de google
<script src="http://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"> </script>
Desde nuestro servidor
Versión completa
<script src="jquery/jquery-3.6.0.js"></script>
Versión minimalista
<script src="jquery/jquery-3.6.0.min.js"></script>

Avanzados, podemos indicar que se cargue desde el CDN de google. Dicen que funciona pero yo lo he probado y falla.
<script src="http://ajax/googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"> </script>
Y por si no carga la librería podemos poner a continuación, para que se cargue desde el CDN de jquery.
<script>windows.JQuery || document.write(unespace('%3cscript src="http://code.jquery.com/jquery-3.6.0.min.js" %3E%3C/script%3E'))</script>

Para hacer referencia a un elemento mediante jQuery vamos a poder utilizar las siguientes formas.

Para inicializar, cuando se ha cargado el documento HTML usaremos

$(document).ready(function() { *cuerpo-función*})

Cuando se cargue el formulario va a salir un cuadro de mensaje con el mensaje "hecho" ejemplo-001.js.

```
001  $(document).ready(function(){
002      alert("hecho");
003  });
```

Otra forma que podemos utilizar

$(document).ready(*nombre-función*)

Cuando se cargue el formulario va a salir un cuadro de mensaje con el mensaje "hecho" ejemplo-004.js

```
001  $(document).ready(iniciado);
002  function iniciado(){
003      alert("hecho");
004  }
```

También podemos utilizar

$(function() {*cuerpo-función*})

Cuando se cargue el formulario va a salir un cuadro de mensaje con el mensaje "hecho" ejemplo-003.js

```
001  $(function(){
002      alert("hecho");
003  });
```

También se consigue lo mismo de la siguiente forma

$(*nombre-función*)

Cuando se cargue el formulario va a salir un cuadro de mensaje con el mensaje "hecho" ejemplo-006.js

```
001 $(iniciado);
002 function iniciado(){
003     alert("hecho");
004 }
```

El signo de $ se puede sustituir por la palabra **jQuery**.

Selectores

- **$("*")** → aplicado a todos los elementos.

El color del texto de todos los elemento va a ser azul. ejemplo-010.js

```
001 $(function (){
002     $("*").css("color","blue");
003 });
```

- **$("*etiqueta*")** → aplicado a todos los elementos cuya etiqueta es la indicada.

El color del texto de la cabeceras h2 va a ser rojo ejemplo-011.js

```
001 $(function (){
002     $("h2").css("color","red");
003 });
```

- **$("*etiqueta.clase*")** → aplicado a todos los elementos cuya etiqueta es la indica y que además se corresponden con la clase indicada.

El color del texto de los párrafos div de la clase leyenda va a ser verde ejemplo-012.js

```
001 $(function (){
002     $("div.leyenda").css("color","green");
003 });
```

- **$(".*clase*")** → aplicado a todos los elementos de la clase indicada.

El color del texto de los elementos de la clase leyenda va a ser verde ejemplo-013.js

```
001 $(function (){
002     $(".leyenda").css("color","green");
003 });
```

- **$("*etiqueta#identificador*")** → aplicado a todos los elementos de la etiqueta indicada con el atributo id igual a identificador.

El color del texto del párrafo div con identificador acueducto es naranja, el color del texto del párrafo div identificador salto es amarillo y el color del texto del párrafo div con identificador parral es marrón ejemplo-014.js

```
001 $(function (){
002     $("div#acueducto").css("color","orange");
003     $("div#salto").css("color","yellow");
004     $("div#parral").css("color","brown");
005 });
```

- **$("*#identificador*")** → aplicado a todos los elementos con el atributo id igual a identificador.

El color del texto del elemento con identificador vida es violeta, el color del texto del elemento con identificador muerte es oliva, el color del texto del elemento con identificador corpus es gris y el color del texto del elemento con identificador sala es cyan. ejemplo-015.js

```
001 $(function (){
002     $("#vida").css("color","violet");
003     $("#muerte").css("color","olive");
004     $("#corpus").css("color","gray");
005     $("#sala").css("color","cyan");
006 });
```

- **$("*selector-1*, selector-2, *selector-3*,..")** → aplicada a todos los elementos de los selectores indicados.

El color del texto de la cabcera h1, del párrafo de la clase madrid, del elemento de la clase cristo, del párrafo div con identificador muerte y del elemento con identificador sala es magenta. ejemplo-016.js

```
001  $(function(){
002      $("h1,div.madrid,.cristo,div#muerte,#sala").css("color","magenta");
003  });
```

- **$("*etiqueta*[*atributo*]")** → aplicada a todos los elementos con la etiqueta indicada que tienen el atributo indicado.

El color del texto de los elementos con identifiador va a ser lima y el color del texto de las cabeceras h2 con el atributo class va a ser turquesa. ejemplo017.js

```
001  $(function(){
002      $("[id]").css("color","lime");
003      $("h2[class]").css("color","turquoise");
004  });
```

- **$("*etiqueta*[*atributo='valor'*]")** → aplicada a todos los elementos con la etiqueta indicada y que tienen en el atributo el valor indicado.

El color del texto de los elementos cuyo atributo class tenga el leyen va a ser salmon y el color del texto de las cabeceras h2 con el el atributo class con el valor cabeza va a ser rosa. ejemplo-018.js

```
001  $(function(){
002      $("[class='leyen']").css("color","salmon");
003      $("h2[class='cabeceza']").css("color","pink");
004  });
```

- **$("*etiqueta*[*atributo*!='*valor*']")** → aplicada a todos los elementos con la etiqueta indicada y que tienen en el atributo un valor distinto del indicado o bien a los elementos con la etiqueta indicada y que no tienen el atributo indicado.

El color del texto de todas las cabeceras que no tengan el atributo class, así como todas las cabeceras que tengan el atributo class con un valor distinto de cabeceza se van a mostrar en color salmon. ejemplo-019.js.

```
001  $(function(){
002      $("h2[class!='cabeceza']").css("color","salmon");
003  });
```

- **$("*etiqueta*[*atributo*|='*valor*']")** → aplicada a todos los elementos con la etiqueta indicada y que tienen en el atributo entre su valores el indicado.

El color del texto de todos los elementos que tengan en el atributo class como uno de sus valores cabeceza va a ser naranja. ejemplo-020.js

```
001  $(function(){
002      $("[class|='cabeceza']").css("color","orange");
003  });
```

- **$("*etiqueta*[*atributo*^='*valor*']")** → aplicada a todos los elementos con la etiqueta indicada y que tienen el atributo un valor que empieza por el valor indicado.

Todos los elementos que tengan el atributo class y éste comience por la cadena cabe va a tener como color de texto marrón. ejemplo-021.js

```
001  $(function(){
002      $("[class^='cabe']").css("color","brown");
003  });
```

- **$("*etiqueta*[*atributo*$='*valor*']")** → aplicada a todos los elementos con la etiqueta indicada y que tienen el atributo un valor que termina por el valor indicado.

*Todos los elementos que tengan el atributo **class** y éste termine por la letra **a** va a tener como color de texto rojo. ejemplo-022.js.*

```
001  $(function(){
002      $("[class$='a']").css("color","red");
003  });
```

- **$("*etiqueta*[*atributo**='*valor*']")** → aplicada a todos los elementos con la etiqueta indicada y que tienen el atributo un valor que contiene el valor indicado.

Todos los elementos que tengan el atributo class y éste contenga la cadena en va a tener como color de texto azul. ejemplo-023.js.

```
001  $(function(){
002      $("[class*='en']").css("color","blue");
003  });
```

- **$("*selector-1 selector-2*")** → descendiente: selector-2 dentro del selector-1.

Todos los elementos de la clase solera, que están dentro del elemento con identificador acueducto va a tener de color del texto rosa. Todos los elementos de la clase origen, que están dentro del elemento con identificador acueducto van a tener de color de texto cyan. Todos los elementos de la clase castillo, que están dentro del elemento de identificador alcazar van a tener de color de texto naranja. Todos los elementos de la clase bellos, que están dentro del elemento con identificador catedral van a tener de color de texto marrón. ejemplo-027.js.

```
001  $(function(){
002      $("#acueducto .solera").css("color","pink");
003      $("#acueducto .origen").css("color","cyan");
004      $("#alcazar .castillo").css("color","orange");
005      $("#catedral .bellos").css("color","brown");
006  });
```

- **$("*selector-1* > *selector-2*")** → hijo: selector-2 hijo del selector-1.

Todos los elementos de la clase solera, que son hijos del elemento con identificador acueducto va a tener de color del texto rosa. Todos los elementos de la clase castillo, que son hijos del elemento de identificador alcazar van a tener de color de texto naranja. Todos los elementos de la clase bellos, que son hijos del elemento con identificador catedral van a tener de color de texto marrón. ejemplo-028.js.

```
001  $(function(){
002      $("#acueducto > .solera").css("color","pink");
003      $("#alcazar > .castillo").css("color","orange");
004      $("#catedral > .bellos").css("color","brown");
005  });
```

- **$("*selector-1* ~ *selector-2*")** → hermanos precedidos de otro elemento. El selector-2 que está detrás del selector-1.

Todos los párrafos p que están a continuación de un hermano de la clase Segovia van a tener de color del texto naranja; y todas las cabeceras h2 que están a continuación de un hermano de la clase parral van a tener de color del texto verde. ejemplo-026.js.

```
001  $(function(){
002      $(".segovia ~ p").css("color","orange");
003      $(".parral ~ h2").css("color","green");
004  });
```

- **$("*selector-1* + *selector-2*")** → hermano adyacente directo, seguido. El selector-2 está inmediatamente a continuación del selecto-1.

*Todos los elementos que tengan el atributo class con un valor que termine con la cadena **to** y que esté precedido, tenga juntamente antes, de un elemento de la clase **leyenda** va a tener como color de texto rojo. Y todas las cabeceras **h2**, que estén inmediatamente a continuación de un elemento de la clase **leyen** va a tener como color de texto azul. ejemplo-025.js.*

```
001  $(function(){
002      $(".leyenda +[class$='to']").css("color","red");
003      $(".leyen + h2").css("color","blue");
004  });
```

- **$("*selector-1 selector-2*:nth-child(*posición*)")** → el hijo que ocupa la posición indicada (que debe ser un selector-2) dentro del selector-1.

El segundo hijo, que es un párrafo div, de los elementos de la clase leyendas va a tener de color del texto rosa. El cuarto hojo de los elementos de la clase leyendas va a tener de color de texto naranja. El séptimo hijo, que

es una cabecera h2, de los elementos de la clase leyendas va a tener de color del texto marrón. Se empiezan a numerar por 1. ejemplo-29.js

```
001  $(function(){
002      $(".leyendas div:nth-child(2)").css("color","pink");
003      $(".leyendas :nth-child(4)").css("color","orange");
004      $(".leyendas h2:nth-child(7)").css("color","brown");
005  });
```

- **$("*selector-1 selector-2*:first-child")** → el primer hijo delselector-1 que debe ser un selector-1, si no es un selector-2 no hace nada.

El primer hijo, que es una cabecera h2, del elemento con identificador catedral va a tener de color del texto azul. El primer hijo, que es un párrafo div, del elemento con identificador alcazar va a tener de color del texto rojo. El primer hijo del elemento con identificador acueducto va a tener de color del texto verde. El primer hijo de los elementos de la clase solera va a tener de color del texto amarillo. ejemplo-030.js.

```
001  $(function(){
002      $("#catedral h2:first-child").css("color","blue");
003      $("#alcazar div:first-child").css("color","red");
004      $("#acueducto :first-child").css("color","green");
005      $(".solera :first-child").css("color","yellow");
006  });
```

- **$("selector-1 selector-2:last-chid")** → el último hijo del selector-1, que debe ser un selector-2.

El último hijo, que es un párrafo div, del elemento con identificador catedral va a tener de color del texto azul. El último hijo, que es un párrafo div, del elemento con identificador alcazar va a tener de color del texto rojo. El último hijo del elemento con identificador acueducto va a tener de color del texto verde. El último hijo de los elementos de la clase solera va a tener de color del texto amarillo. ejemplo-031.js. Cuidado no se puede aplicar a p.

```
001  $(function(){
002      $("#catedral div:last-child").css("color","blue");
003      $("#alcazar div:last-child").css("color","red");
004      $("#acueducto :last-child").css("color","green");
005      $(".solera :last-child").css("color","yellow");
006  });
```

- **$("*selector-1 selector-2*:only-child")** → hijos únicos del selector. El selector-2 que es el único hijo del selector-1.

El párrafo div, que es el único hijo de cada uno de los elementos de la clase bellos, va a tener de color del texto azul. La cabecera h2, que es la única hija de cada uno de los elementos de la clase castillo, va a tener de color del texto rojo. El elemento, que es el único de los elementos de las clase solera, va a tener de color del texto amarillo. ejemplo-32.js.

```
001  $(function(){
002      $(".bellos div:only-child").css("color","blue");
003      $(".castillo h2:only-child").css("color","red");
004      $(".solera :only-child").css("color","yellow");
005  });
```

- **$("*selector*:eq(*posición*)")** → elemento que ocupa la posición indicada por índice dentro de un grupo.

El tercer párrafo div hijo del elemento con identificador catedral va a tener de color del texto azul. El cuarto párrafo div hijo del elemento con identificador alcazar va a tener de color del texto rojo. El segundo párrafo div hijo del elemento con identificador acueducto va a tener de color del texto ver. El quinto párrafo div hijo de cada uno de los elementos de la clase solera va a tener de color de texto amarillo. Se empiezan a numerar por 0.ejemplo-033.js.

```
001  $(function(){
002      $("#catedral>div:eq(2)").css("color","blue");
003      $("#alcazar>div:eq(3)").css("color","red");
004      $("#acueducto>div:eq(1)").css("color","green");
005      $(".solera>div:eq(4)").css("color","yellow");
006  });
```

- **$("selector:first")** → primer elemento de un grupo. El selector que es el primer elemento de un grupo.

El primer párrafo div hijo del elemento con identificador catedral va a tener de color del texto azul. El primer párrafo div hijo del elemento con identificador alcazar va a tener de color del texto rojo. El primer párrafo div del elemento con identificador acueducto va a tener de color del texto verde. El primer párrafo div de cada uno de los elementos de la clase solera va a tener de color de texto amarillo. ejemplo-34.js.

```
001 $(function(){
002     $("#catedral>div:first").css("color","blue");
003     $("#alcazar>div:first").css("color","red");
004     $("#acueducto>div:first").css("color","green");
005     $(".solera>div:first").css("color","yellow");
006 });
```

- **$("selector:last")** → último elemento de un grupo.

El último párrafo div hijo del elemento con identificador catedral va a tener de color del texto azul. El último párrafo div hijo del elemento con identificador alcazar va a tener de color del texto rojo. El último párrafo div del elemento con identificador acueducto va a tener de color del texto verde. El último párrafo div de cada uno de los elementos de la clase solera va a tener de color de texto amarillo. ejemplo-35.js.

```
001 $(function(){
002     $("#catedral>div:last").css("color","blue");
003     $("#alcazar>div:last").css("color","red");
004     $("#acueducto>div:last").css("color","green");
005     $(".solera>div:last").css("color","yellow");
006 });
```

- **$("selector:gt(índice)")** → todos los elementos que ocupan una posición superior al índice.

Todos los párrafos div hijos del elemento con identificador catedral, que ocupen un posición superior a la 2 van a tener como color del texto azul. Todos los párrafos div hijos del elemento con identificador alcazar, que ocupan una posición superior a la 1, van a tener como color del texto rojo. Todos los párrafos div hijos del elemento con identificador acueducto, que ocupen una posición superior a la 0, van a tener como color del texto verde. Todos los párrafos div hijos de los elementos de la clase solera, que ocupen una posición superior a la 4 van a tener de color del texto amarillo. Se empiezan a numerar por 0. ejemplo-36.js.

```
001 $(function(){
002     $("#catedral>div:gt(2)").css("color","blue");
003     $("#alcazar>div:gt(1)").css("color","red");
004     $("#acueducto>div:gt(0)").css("color","green");
005     $(".solera>div:gt(4)").css("color","yellow");
006 });
```

- **$("selector:lt(índice)")** → todos los elementos que ocupan una posición inferior a índice.

Todos los párrafos div hijos del elemento con identificador catedral, que ocupen un posición onferior a la 5 van a tener como color del texto azul. Todos los párrafos div hijos del elemento con identificador alcazar, que ocupan una posición inferior a la 4, van a tener como color del texto rojo. Todos los párrafos div hijos del elemento con identificador acueducto, que ocupen una posición inferior a la 2, van a tener como color del texto verde. Todos los párrafos div hijos de los elementos de la clase solera, que ocupen una posición inferior a la 3 van a tener de color del texto amarillo. Se empiezan a numerar por 0. ejemplo-37.js.

```
001 $(function(){
002     $("#catedral>div:lt(5)").css("color","blue");
003     $("#alcazar>div:lt(4)").css("color","red");
004     $("#acueducto>div:lt(2)").css("color","green");
005     $(".solera>div:lt(3)").css("color","yellow");
006 });
```

- **$("selector:even")** → elementos pares de un grupo.

Todos los párrafos div hijos del elemento con identificador catedral, que ocupen posiciones pares, van a tener de color del texto oliva. Todos los párrafos div del elemento con identificador alcazar, que ocupen posiciones pares, van a tener como color del texto salmon. Todos los párrafos div hijos del elemento con identificador acueducto, que ocupen posiciones pares, van a tener de color del texto magenta. Teniendo en cuenta que se empiezan a numerar por cero y el cero es un número par. ejemplo-038.js.

```
001 $(function(){
002     $("#catedral>div:even").css("color","olive");
```

```
003     $("#alcazar>div:even").css("color","salmon");
004     $("#acueducto>div:even").css("color","magenta");
005 });
```

- **$("selector:odd")** → elementos impares de un grupo.

Todos los párrafos div hijos del elemento con identificador catedral, que ocupen posiciones impares, van a tener de color del texto teal. Todos los párrafos div del elemento con identificador alcazar, que ocupen posiciones impares, van a tener como color del texto marrón. Todos los párrafos div hijos del elemento con identificador acueducto, que ocupen posiciones impares, van a tener de color del texto gris. Teniendo en cuenta que se empiezan a numerar por cero y el cero es un número par. ejemplo-039.js.

```
001 $(function(){
002     $("#catedral>div:odd").css("color","teal");
003     $("#alcazar>div:odd").css("color","maroon");
004     $("#acueducto>div:odd").css("color","gray");
005 });
```

- **$("*selector*:contains(*texto*)")** → contiene el texto

Todos los párrafos div hijos del elemento con identificador catedral, que en su contenido contengan la palabra 'Mayor' van a tener como color de los caracteres rojo. Todos los párrafos div hijos del elemento con identificador alcazar, que en su contenido contengan la palabra 'Zamarramala' van a tener como color del texto verde. Todos los párrafos div hijos del elemento con identificador acueducto, que contengan la cadena 'Segovia' van a tener como color del texto navy. ejemplo-040.js

```
001 $(function(){
002     $("#catedral>div:contains('Mayor')").css("color","red");
003     $("#alcazar>div:contains('Zamarramala')").css("color","green");
004     $("#acueducto>div:contains('Segovia')").css("color","navy");
005 });
```

- **$("*selector-1*:has(selector-2)")** → selector-1 que contiene el elemento indicado por el selector-2.

*Todos los párrafos div hijos del elemento con identificador catedral, que en su interior contengan la etiqueta **span** van a tener como color de los caracteres rojo. Todos los párrafos div hijos del elemento con identificador alcazar, que en su interior contengan la etiqueta **b** van a tener como color del texto verde. Todos los párrafos div hijos del elemento con identificador acueducto, que en su interior contengan la etiqueta i van a tener como color del texto navy. ejemplo-041.js*

```
001 $(function(){
002     $("#catedral>div:has(span)").css("color","red");
003     $("#alcazar>div:has(b)").css("color","green");
004     $("#acueducto>div:has(i)").css("color","navy");
005 });
```

- **$("*selector-1*:not(*selector-2*)")** → selector-1 que no contiene el selector2.

*Todos los párrafos div hijos del elemento con identificador catedral, que no sean de la clase **exterior** van a tener como color de los caracteres rosa. Todos los párrafos div hijos del elemento con identificador alcazar, que no tengan el atributo **class** van a tener como color del texto naranja. Todos los párrafos div hijos del elemento con identificador acueducto, que no sean de la clase uno van a tener como color del texto marron. Todas las cabeceras h4 hijos del elemento con identificador acueducto, que no tengan el atributo id van a tener como color del texto cyan. ejemplo-042.js*

```
001 $(function(){
002     $("#catedral>div:not(.exterior)").css("color","pink");
003     $("#alcazar>div:not([class])").css("color","orange");
004     $("#acueducto>h4:not(.uno)").css("color","brown");
005     $("#acueducto>div:not([id])").css("color","cyan");
006 });
```

- **$(":text")** → pseudoclase cajas de texto.

Todas las cajas de texto van a tener como color de fondo azul y color de los caracteres blanco. ejemplo-043.js

```
001 $(function(){
002     $(":text").css({"background-color":"blue","color":"white"});
003 });
```

- **$(":button")** → pseudoclase botón. <input type="button">

Todos los botones del tipo <input type="button" van a tener de color de fondo amarillo y de color del texto rojo. ejemplo-044.js

```
001 $(function(){
002     $(":button").css({"background-color":"yellow","color":"red"});
003 });
```

- **$(":header")** → pseudoclase cabeceras.

Todas las cabeceras, h1, h2, h3, h4, h5 y h6 van a tener como color del texto rojo. ejemplo-047.js.

```
001 $(function(){
002     $(":header").css("color","red");
003 });
```

- **$(":image")** → pseudoclase image, input type=image.

Todos los input type="image" van a tener un ancho del 5% y van a tener un borde continuo de 1 pixel de color rojo. ejemplo-066.js.

```
001 $(function(){
002     $(":image").css({"width":"5%","border": "1px solid red"});
003 });
```

- **$(":input")** → pseudoclase input, todos los elementos de entrada.

Todos los elementos de entrada, cajas de texto, textareas, botones, .. van a tener como color de fondo el amarillo. jemplo-046.js

```
001 $(function(){
002     $(":input").css("background-color","yellow");
003 });
```

- **$(":password")** → pseudoclase contraseña.

Todas las cajas de texto de las contraseñas, input type="password2 van a tener como color de fondo gris y color del texto amarillo. ejemplo-048.js.

```
001 $(function(){
002     $(":password").css({"background-color":"gray","color":"yellow"});
003 });
```

- **$(":radio")** → pseudoclase radiobutton.
- **$(":checkbox")** → pseudoclase checkbox.
- **$(":checked")** → pseudoclase radiobutton y checkbox seleccionados.

Al hacer click sobre el botón vamos a asignar a las variables el número de checkbox seleccionados dentro del elemento con identificador viajes y el número de radio seleccionados dentro del elemento con identificador estado, luego mostramos los valores en la consola. ejemplo-045.js

```
001 $(function(){
002     $("#comprobar").on("click", function() {
003         var VnbNumSelecioandos=$("#viajes :checkbox:checked ").length;
004         var VnbNumSelecioana=$("#estado :radio:checked ").length;
005         console.log(VnbNumSelecioandos);
006         console.log(VnbNumSelecioana);
007     });
008 });
```

- **$(":selected")** → pseudoclase opciones seleccionadas de un select.

Asignamos a la variable todas las opciones seleccionadas de la lista (select) de identificador provincias.

```
001 var seleccioandos=$("#provincias>option:selected");
```

- **$("selector:parent")** → elemento padre. El selector indicado, que debe ser un elemento padre, debe tener al menos un hijo.

El elemento con identificador cate, que es padre de otro/s elemento/s, va a tener un borde continuo de 1 pixel de color verde. ejemplo-074.js

```
001 $(function(){
002     $("#cate:parent").css("border", "1px solid green");
```

```
003 });
```

- **$(":submit")** → pseudoclase submit.
- **$(":reset")** → pseudoclase reset.

El botón de submit va a tener color del texto amarillo y de fondo rojo. El botón de reset va a terner color del texto blanco y color de fondo verde. ejemplo-075.js

```
001 $(function(){
002     $(":submit").css({"color": "yellow","background-color":"red" });
003     $(":reset").css({"color": "white","background-color":"green" });
004 });
```

- **$(":file")** → pseudoclase archivo.

El nombre del fichero seleccionado o "no se ha seleccionado ningún archivo" va a tener color de fondo rojo y color del texto verde. ejemplo-076.js

```
001 $(function(){
002     $(":file").css({"color": "green","background-color":"red" });
003 });
```

Para saber si hay algún elemento seleccionado se pregunta por su longitud.
Se puede asignar un selector a una variable, la variable va a comenzar por $.

Para acceder a un elemento, a partir de un selector

- **.length** → número de elementos.

Asignamos a la primera variable el número de párrafos div, que hay en el elemento de identificador acueducto. A la segunda variable el número se elementos seleccionados en la lista select de identificador provincias. A la tercera variable el número de li que hay en la lista (ol o ul) de identificador lista.

```
001 var VnbNumeroParrafos=$("#acueducto>div").length;
002 var VnbNumeroSeleccioandos=$("#provincias:selected").length;
003 var VnbNumeroLista=$("#lista>li").length;
```

- **.add("selector")** → añade el elemento a la colección de selectores, para tener más elementos a los que aplicaremos una característica.

Al conjunto de cabeceras h2 y h4 le añadimos los párrafos p, para que todos estos elementos tengan de color de fondo azul y color del texto amarillo. ejemplo-049.js

```
001 $(function(){
002     $("h2,h4").add("p").css({"background-color":"blue","color":"yellow"});
003 });
```

- **.not("selector")** → borra el elemento de la colección de selectores.

Al conjunto de todos los párrafos div hijos del elemento con identificador catedral, le quitamos los elementos de la clase cate y a los que nos queda les asignamos de color de fondo azul y color de los caracteres amarillo. ejemplo-050.js.

```
001 $(function(){
002     $("#catedral>div").not(".cate").css({"background-color":"blue"
    ,"color":"yellow"});
003 });
```

- **.filter("selector|función")** → filtrar de la colección de selectores el selector indicado.

Del conjunto de todos los párrafos div hijos del elemento con identificador catedral, nos quedamos con los elementos de la clase cate y a los que nos queda les asignamos de color de fondo rojo y color de los caracteres verde. ejemplo-051.js.

```
001 $(function(){
002     $("#catedral>div").filter(".cate").css({"background-color":"red",
    "color":"green"});
003 });
```

- **.has(elemento)** → tiene ese elemento o selector.

Todos los párrafos div hijos de otro párrafo div que contenga, como descendiente, una cabecera h4, el texto va a estar en cursiva y en negrilla. ejemplo052.js.

```
001 $(function(){
```

```
002     $("div>div").has("h4").css({"font-style":"italic","font-weight":"bold"});
003 });
```

- **.eq(posición)** → elemento que ocupa la posición indicada dentro de un conjunto.

El tercer párrafo div hijo del elemento con identificador alcazar va a tener el texto en negrilla y subrayado. ejemplo-053.js.

```
001 $(function(){
002     $("#alcazar>div").eq("2").css({"text-decoration":"underline","font-weight":
        "bold"});
003 });
```

- **.first()** → primer elemento de un conjunto.

El primer párrafo div hijo del elemento con identificador acueducto va a tener el texto subrayado y en cursiva. ejemplo-054.js.

```
001 $(function(){
002     $("#acueducto>div").first().css({"text-decoration":"underline","font-
        style":"italic"});
003 });
```

- **.last()** → último elemento de un conjunto.

El último párrafo div hijo del elemento con identificador catedral va a tener el texto en color salmon y en cursiva. ejemplo-055.js

```
001 $(function(){
002     $("#catedral>div").last().css({"color":"salmon","font-style":"italic"});
003 });
```

- **.slice(inicio[,fin])** → subcolección desde inicio hasta fin (excluido) o hasta el final si se omite fin.

Los párafos div hijos del elemento con identificador acueducto, que ocupen una posición a partir de la cuarta, incluida, van a tener los caracteres de color naranja en negrilla. Los párrafos div hijos del elemento con identificador alcazar, que ocupen una posición entre la segunda y la cuarta, ambas incluidas, van a tener los caracteres de color rosa en negrilla. ejemplo-056.js.

```
001 $(function(){
002     $("#acueducto>div").slice(3).css({"color":"orange","font-weidth":"bold"});
003     $("#alcazar>div").slice(1,4).css({"color":"pink","font-weidth":"bold"});
004 });
```

- **.odd()** → elementos de un conjunto, que ocupan posiciones impares. a partir de versión 3.5.0.
- **even()**→ elementos de un conjunto, que ocupan posiciones pares. a partir de versión 3.5.0.

Todos los párrafos div hijos de otros párrafos div, los que ocupan posiciones pares van a tener de color de los caracteres rojo y los que ocupan posiciones inmpares color de los caracteres azul. ejmplo-jquery-051.js

```
001 $(function (){
002     $("div>div").odd().css("color","blue");
003     $("div>div".even().css("color","red");
004 })
```

- **.children([selector])** → descendientes directos, si se incluye el selector serán los descendientes directos que se correspondan con ese selector.

Todos los elementos hijos del elemento con identificador alcazar van a tener los caracteres de color fucsia subrayados. Todos los elementos hijos, que sean h4, del elemento con identificador acueducto van a tener los caracteres de color oliva subrayado. ejemplo-057.js.

```
001 $(function(){
002     $("#alcazar").children().css({"color":"fuchsia","text-decoration":
        "underline"});
003     $("#acueducto").children("h4").css({"color":"olive","text-decoration":
        "underline"});
004 });
```

- **.find(selector)** → descendientes que se corresponden con el selector.

Todas las cabeceras h4, que sean descendientes del elemento con identificador acueducto van a tener los caracteres de color aqua en cursiva y negrilla. ejemplo-058.js

```
001 $(function(){
002     $("#acueducto").find("h4").css({"color":"aqua","font-style":"italic","font-
    weight":"bold"});
003 });
```

- **.contents()** → descendientes directos incluyendo el texto plano.

Todos los descendientes directos, hijos, del elemento cuyo identificador es catedral van a tener de color del texto azul en negrilla y con un borde continuo de 1 pixel de color rojo. ejemplo-067.js.

```
001 $(function(){
002     $("#catedral").contents().css({"color":"blue","font-weight":"bold","border":
    "1px solid red"});
003 });
```

- **.next()** → siguiente hermano.

El elemento siguiente hermano del segundo párrafo div hijo del elemento con identificador catedral, va a tener los caracteres de color navy en cursiva y subrayados. ejemplo-059.js.

```
001 $(function(){
002     $("#catedral>div").eq(1).next().css({"color":"navy","text-decoration":
    "underline","font-style":"italic"});
003 });
```

- **.prev()** → hermano anterior.

El elemento hermano anterior del último párrafo div hijo del elemento con identificador acueducto, va a tener los caracteres de color plata en negrilla y subrayados. ejemplo-060.js.

```
001 $(function(){
002     $("#acueducto>div").last().prev().css({"color":"silver","text-decoration":
    "underline","font-weight":"bold"});
003 });
```

- **.nextAll()** → todos los hermanos siguientes.

Todos los elementos hermanos siguientes al tercer párrafo div hijo del elemento con identificador alcazar, van a tener los caracteres de color oro en negrilla y subrayados. ejemplo-061.js.

```
001 $(function(){
002     $("#alcazar>div").eq(2).nextAll().css({"color":"gold","text-decoration"
    :"underline","font-weight":"bold"});
003 });
```

- **.prevAll()** → todos los hermanos anteriores.

Todos los elementos hermanos anteriores al último párrafo div hijo del elemento con identificador catedral, van a tener los caracteres de color marrón en cursiva y en negrilla. ejemplo-062.css.

```
001 $(function(){
002     $("#catedral>div").last().prevAll().css({"color":"brown","font-style":"italic",
    "font-weight":"bold"});
003 });
```

- **.nextUntil(selector)** → los hermanos siguientes hasta el indicado por el selector, el cual se omite

Todos los elementos hermanos siguientes al tercer párrafo div hijo del elemento con identificador alcazar hasta el elemento de la clase quinto (excluido éste), van a tener los caracteres de color oro en negrilla y subrayados. ejemplo-061.js.

```
001 $(function(){
002     $("#alcazar>div").eq(2).nextUntil(".quinto").css({"color":"gold","text-
    decoration" :"underline","font-weight":"bold"});
003 });
```

- **.siblings()** → todos los hermanos.

Todos los elementos hermanos del cuarto párrafo div hijo del elemento con identificador acueducto, van a tener los caracteres de color lima en negrilla y subrayados. ejemplo-065.js.

```
001 $(function(){
002     $("#acueducto>div").eq(3).siblings().css({"color":"lime","text-decoration":
    "underline","font-weight":"bold"});
003 });
```

- **.parent()** → el elemento padre.

El elemento padre del elemento con identificador cate va a tener un borde continuo de 1 pixel de color verde. ejemplo-068.js.

```
001 $(function(){
002     $("#cate").parent().css("border", "1px solid green");
003 });
```

- **.parents()** → todos los ascendientes.

Todos los elementos ascendentes o ancestros del elemento con identificador cate va a tener un borde continuo de 1 pixel de color verde. ejemplo-069.js

```
001 $(function(){
002     $("#cate").parents().css("border", "1px solid green");
003 });
```

- **.clone()** → devuelve un nuevo elemento que es un clon del actual.

Se asigna a la variable nuevo una copia del tercer párrafo div hijo del elemento con identificador catedral y luego se añade al final de ese elemento. En resume, se está copiando el tercer div existente en el elemento con identificador catedral al final del mismo. ejemplo-070.js

```
001 $(function(){
002     var nuevo=$("#catedral>div").eq(2).clone();
003     $("#catedral").append(nuevo);
004 });
```

- **.index([elemento|selector])** → posición ocupa el elemento o selector indicado dentro de un conjunto, sobre el que se aplica el método index. Sin parámetros nos indica la posición del selector sobre el que se aplica en su conjunto.

Se va a visualizar en la caja de texto con identificador posiciones, la posición que ocupan el elemento con identificador capillas en el conjunto de párrafos div hijos del elemento con identificador catedral. La posición que ocupa el elemento con identificador foso dentro de los elementos hijos del elemento con identificador alcazar. La posición que ocupa el elemento con identificador solera dentro de su padre. La posición que ocupa la primera cabecera h4 dentro de su padre. La posición que ocupa la primera cabecera h2 dentro de los descendientes directos, hijos, del elemento con identificador catedral. La posición que ocupa el elemento con identificador caracol dentro de los hijos del elemento con identificador alcazar. ejemplo-071.js

```
001 $(function(){
002     var primero=$("#capillas");
003     var uno=$("#catedral>div").index($(primero));
004     var cinco=$("#catedral").contents().index($("h2"));
005     var segundo=$("#foso");
006     var dos=$("#alcazar").children().index($(segundo));
007     var seis=$("#alcazar").children().index($("#caracol"));
008     var tres=$("#solera").index();
009     var cuatro=$("h4").index();
010     $("#posiciones").val(uno + " - " + dos + " - " + tres + " - " + cuatro + " - "
    + cinco + " - " + seis);
011 });
```

Para modificar atributos a partir de un elemento.

- **.attr(atributo, valor)** → asigna valor al atributo

El hiperenclace con identificador primero va a tener en el atributo href el valor http://www.iesleonardo.com.html.

```
001 $('a#primero').attr("href","http://www.iesleonardo.com.html");
```

- **.attr({formato-estilos-css})** → asigna valor a varios atributos

Todos los hiperenlaces van a tener en el atributo title el valor 'all titles are the same too' y en el atributo href va a tener el valor 'somethingNew.html'.

```
001 $('a').attr({
```

```
    'title':'all titles are the same too',
    'href':'somethingNew.html'
});
```

- **.css("propiedad","valor")** → las propiedades suelen seguir las reglas de Javascript, aunque también se puede poner el nombre de css.

Todos los párrafos div van a tener como color del texto salmon.

```
$("div").css("color","salmon");
```

- **.css({estilos-css})** → añade los estilos indicados

Todos los párrafos div van a tener como color de fondo azul y como color del texto blanco.

```
$('div').css({
    'background-color':'blue',
    'color':'white'
});
```

- **.addClass("nueva-clase-elemento")** → añade la clase a los elementos.
- **.removeClass("clase")** → elimina la clase de los elementos

*Cuando entra el ratón en el elemento con identificador **alcazar** se ejecuta la función **entrar**, que añade o asigna la clase **colorear** a dicho elemento. Cuando sale el ratón del elemento con identificador **alcazar** se ejecutara función **salir**, que quita la clase **colorear** a dicho elemento. ejemplo-072.js.*

```
$(function(){
    $("#alcazar").on("mouseenter", entrar);
    $("#alcazar").on("mouseleave", salir);
});
function entrar(){
    $("#alcazar").addClass("colorear");
}
function salir(){
    $("#alcazar").removeClass("colorear");
}
```

- **.toggleClass("clase")** →añade/elimina clase de los elementos. Si no tiene la clase se la asigna y si la tiene se la elimina.´

*Cuando entra y sale el ratón en el elemento con identificador **alcazar** se ejecuta la función **asignar**, que añade o asigna la clase **colorear** a dicho elemento, si no la tiene, y que quita la clase **colorear** a dicho elemento, si la tiene. ejemplo-073.js.*

```
$(function(){
    $("#alcazar").on("mouseenter", asignar);
    $("#alcazar").on("mouseleave", asignar);
});
function asignar(){
    $("#alcazar").toggleClass("colorear");
}
```

Clase colorear, que se utiliza en los dos casos anteriores. Pone cono color de fondo verde, color del texto blanco, el tipo de letra Cambria, el tamaño del texto 13 puntos, la altura de línea un 25% mayor que el tipo de letra. ejemplo072.css.

```
.colorear{
    background-color:green;
    color:white;
    Font-family:Cambria;
    font-size:13pt;
    line-height:1.25rem;
}
```

- **.append(contenido)** → añade contenido al final.

Se añade un elemento a una lista (ordenada 'ol' o bien no ordenada 'ul') al final de la misma.

```
$("#lista").append("<li>"+dato+"</li>");
```

- **.prepend(contenido)** → añade contenido al inicio.

Se añade un elemento a una lista (ordenada 'ol' o bien no ordenada 'ul') al inicio de la misma.

```
001  $("#lista").prepend("<li>"+dato+"</li>");
```

- **.before(contenido)** → añade contenido antes.

Se añade un elemento a una lista (ordenada 'ol' o bien no ordenada 'ul') que se va a colocar delante del elemento indicado al principio, el que ocupa la posición indicada por índice.

```
001  $(todos).eq(indice).before("<li>"+dato+"</li>");
```

- **.after(contenido)** → añade contenido después.

Se añade un elemento a una lista (ordenada 'ol' o bien no ordenada 'ul') que se va a colocar a continuación del elemento indicado al principio, el que ocupa la posición indicada por índice.

```
001  $(todos).eq(indice).after("<li>"+dato+"</li>");
```

- **.remove()** → elimina elemento.

Se va a eliminar el elemento que ocupa la posición indicada por VnbIndice, dentro del conjunto conjuntoanimal

```
001  $(conjuntoanimal).eq(VnbIndice1).remove();
```

- **.html([contenido])** →obtiene/modifica contenido.

Obtenemos el contenido HTML del elemento con identificador párrafo, le añadimos un párrafo div, con su contenido, y le volvemos a asigna ese contenido HTML al elemento con identificador párrafo.

```
001  var VstContenido=$("#parrafo").html();
002  VstContenido+="<div>Prueba correcta</div>";
003  $("#parrafo").html(VstContenido);
```

- **.text([contenido])** → obtiene/modifica contenido solo texto del elemento.

Obtenemos el contenido del segundo elemento del conjunto casillas, le transformamos a valor numérico, le incrementamos en uno y al mismo elemento le asignamos el valor, en resumen incrementamos en uno el valor del segundo elemento del conjunto casillas

```
001  var VstNumero=$(casillas).eq(1).text();
002  var VnbNumero=parseInt(VstNumero,10);
003  VnbNumero+=1;
004  $(casillas).eq(1).text(VnbNumero);
```

- **.val([valor])** → valor de los elementos de un formulario.

Obtenemos el valor del elemento con identificador valor, le convertimos en número, le incrementamos en uno y volvemos a asignarle el valor, en resumen, incrementamos en uno el valor el elemento con identificador valor.

```
001  var VstNumero=$("#valor").val();
002  var VnbNumero=parseInt(VstNumero,10);
003  VnbNumero+=1;
004  $("#valor").val(VnbNumero);
```

- **.hide()** → oculta elemento.
- **.show([tiempo-milisegundo])** → muestra elemento.

Cuando se carga el formulario se ejecuta la función inicio. En la misma hacemos que el elemento con identificador mostrar, un botón, al hacer click sobre el mismo se ejecute la función muestra y que el elemento con identificador oculta, un botón, al hacer click sobre el mismo se ejecute la función oculta. En la función muestra se va a mostrar el elemento con identificador forma, un dialogo. En la función oculta se va a ocultar el elemento con identificador forma. ejemplo-077.js

```
001  $(window).on("load",inicio);
002  function inicio(){
003      $("#mostrar").on("click",muestra);
004      $("#ocultar").on("click",oculta);
005  }
006  function muestra(){
007      $("#forma").show();
008  }
009  function oculta(){
010      $("#forma").hide();
011  }
```

Cuando se carga el formulario hacemos que el elemento con identificador mostrar, un botón, al hacer click sobre el mismo se ejecute la función muestra y que el elemento con identificador oculta, un botón, al hacer click sobre el mismo se ejecute la función oculta. En la función muestra se va a aparecer poco a poco, en un periodo de 5 segundo, el elemento con identificador forma, un dialogo. En la función oculta se va a ocultar el elemento con identificador forma. ejemplo-078.js

```
001  $(function(){
002      $("#mostrar").on("click",muestra);
003      $("#ocultar").on("click",oculta);
004  });
005  function muestra(){
006      $("#forma").show(5000);
007  }
008  function oculta(){
009      $("#forma").hide();
010  }
```

- **.fadeIn([tiempo-milisegundos])** → cambia opacidad al 100% de forma animada.
- **.fadeOut([tiempo-milisegundos])** → cambia la opacidad al 0% de forma animada.

Cuando se carga el formulario hacemos que el elemento con identificador catedral al entrar con el ratón en él se ejecute la función muestra y al salir el ratón de dicho elemento se ejecute la función oculta. Así mismo vamos a ocultar el elemento con identificador imgcate. En la función muestra se va a mostrar el elemento con identificador imgcate. En la función oculta se va a ocultar el elemento con identificador imgcate. ejemplo-081.js

```
001  $(function(){
002      $("#catedral").on("mouseleave",oculta );
003      $("#catedral").on("mouseenter", muestra);
004      $("#imgcate").hide() ;
005  });
006  function muestra(){
007      $("#imgcate").fadeIn() ;
008  }
009  function oculta(){
010      $("#imgcate").fadeOut() ;
011  }
```

Cuando se carga el formulario hacemos que el elemento con identificador catedral al entrar con el ratón en él se ejecute la función muestra y al salir el ratón de dicho elemento se ejecute la función oculta. Así mismo vamos a ocultar el elemento con identificador imgcate. En la función muestra se va a mostrar el elemento con identificador imgcate en un periodo de 6 segundos. En la función oculta se va a ocultar el elemento con identificador imgcate en un periodo de 3 segundos. ejemplo-082.js

```
001  $(function(){
002      $("#catedral").on("mouseleave",oculta );
003      $("#catedral").on("mouseenter", muestra);
004      $("#imgcate").hide() ;
005  });
006  function muestra(){
007      $("#imgcate").fadeIn(6000);
008  }
009  function oculta(){
010      $("#imgcate").fadeOut(3000);
011  }
```

- **.slideDown()** → muestra el elemento con un movimiento de deslizamiento vertical.
- **.slideUp()** → oculta el elemento con un movimiento de deslizamiento vertical.

Cuando se carga el formulario hacemos que el elemento con identificador mostrar, un botón, al hacer click sobre el mismo se ejecute la función muestra y que el elemento con identificador oculta, un botón, al hacer click sobre el mismo se ejecute la función oculta. En la función muestra se va a mostrar el elemento con identificador forma, un dialogo. En la función oculta se va a ocultar el elemento con identificador forma. ejemplo-079.js

```
001  $(function(){
002      $("#mostrar").on("click",muestra);
```

```
003     $("#ocultar").on("click",oculta);
004 });
005 function muestra(){
006     $("#forma").slideDown() ;
007 }
008 function oculta(){
009     $("#forma").slideUp() ;
010 }
```

- **.slideToggle()** → muestra/oculta el elemento con un movimiento de deslizamiento vertical.

Cuando se carga el formulario hacemos que el elemento con identificador mostrar, un botón, al hacer click sobre el mismo se ejecute la función muestra y que el elemento con identificador oculta, un botón, al hacer click sobre el mismo se ejecute la función muestra. En la función muestra se va a mostrar el elemento con identificador, si está oculto, y se va a ocultar, si está visible. ejemplo-080.js

```
001 $(function(){
002     $("#mostrar").on("click",muestra);
003     $("#ocultar").on("click",muestra);
004 });
005 function muestra(){
006     $("#forma").slideToggle() ;
007 }
```

Eventos

.on(*evento* [, *selector*], *nombre-función*) → Se asigna la función indicada para el evento indicado sobre el selector. La función va a tener un parámetro que es el evento.
.on(*evento* [, *selector*], function(){*cuerpo-función*}) → Se asigna la función anónima indicada para el evento indicado sobre el selector.
.on(*evento* [, *selector*], *datos*, *nombre-función*) → Se asigna la función anónima indicada para el evento indicado sobre el selector.
.on(*evento* [, *selector*], *datos*, function(){*cuerpo-función*}) → Se asigna la función anónima indicada para el evento indicado sobre el selector. Los datos se pasan en el formato **{*nombre*:*valor*, …}**. En la función ese dato le obtenemos mediante **event.data.*nombre***.
.on({*evento-1*: *nombre-función* [, *evento-2*: *nombre-función*]..} [, *selector*]) → asigna sobre selector los eventos indicados con sus correspondientes funciones. Los datos se pasan en el formato **{*nombre*:*valor*, …}**. En la función ese dato le obtenemos mediante **event.data.*nombre***.
.on({*evento-1*:function(){*cuerpo-función-1*} [, *evento-2*:function(){*cuerpo-función-2*}]..} [, *selector*]) → asigna sobre el selector los eventos indicados con sus correspondientes funciones.
Con on, cuando dentro se pone un selector, se pone delante un selector padre del que se pone dentro

La caja de texto del nombre y la caja de texto de los apellidos solamente van a admitir letras y el espacio.
La caja de texto de la edad y la caja de texto del código postal solamente van a admitir dígitos.
Cuando nos situemos en las cajas de texto el color de fondo va a ser azul y el color de los caracteres blanco y además se va a borrar su contenido
Cuando avandonemos las cajas de texto el color de fondo y el colro del texto va a ser el anterior.
Cuando salgamos de la caja de texto del código postal, si el mismo está comprendido entra 1000 y 52999 vamos a mostrar en la caja de
texto de la provincia el nombre de la provinvia.
Cuando entre el ratón en el texto del guepardo el color de fondo va a ser rojo,el color del texto va a ser amarillo y el tamaño de las
letras va a ser de 16 puntos.
Cuando salga el ratón del texto del guepardo va a volver a su situación anterior.
Cuando estemos con el ratón sobre el texto del leopardo el color del texto va a ser rojo.
Cuando movamos el ratón sobre el texto del leopardo el tamaño del texto va a variar, así como el color de fondo.
Cuando se pulse la tecla F8 se va a cambiar la imagen de fondo del texto del ocelote.

ejemplo-jquery-521.js

```
001  $(window).on("load",comienzo);
002  function comienzo(){
003       $("#nombre").on("keypress",sololetras);;
004       $("#apellidos").on("keypress",sololetras);
005       $("#edad").on("keypress",solodigitos);
006       $("#codigopostal").on("keypress",solodigitos);
007       $("#codigopostal").on("blur",muestra);
008       $("#guepardo").on("mouseenter",entrarcaja);
009       $("#guepardo").on("mouseout",salircaja);
010       $("#leopardo").on("mouseover",dentro);
011       $("#leopardo").on("mouseout",salirdentro);
012       $("#leopardo").on("mousemove",mover);
013       $(":text").on("focus",entrar);
014       $(":text").on("blur",salir);
015       $("body").on("keyup",imagenes);
016  }
```

ejemplo-jquery-521a.js

```
001  $(window).on("load",comienzo);
002  function comienzo(){
003       $(document).on("keypress","#nombre",sololetras);;
004       $(document).on("keypress","#apellidos",sololetras);
005       $(document).on("keypress","#edad",solodigitos);
006       $(document).on("keypress","#codigopostal",solodigitos);
007       $(document).on("blur","#codigopostal",muestra);
008       $(document).on("mouseenter","#guepardo",entrarcaja);
009       $(document).on("mouseout","#guepardo",salircaja);
010       $(document).on("mouseover","#leopardo",dentro);
011       $(document).on("mouseout","#leopardo",salirdentro);
012       $(document).on("mousemove","#leopardo",mover);
013       $(document).on("focus",":text",entrar);
014       $(document).on("blur",":text",salir);
015       $(document).on("keyup","body",imagenes);
016  }
```

ejemplo-jquery-521b.js

```
001  $(window).on("load",comienzo);
002  function comienzo(){
003       $("#nombre").on("keypress",sololetras);;
004       $("#apellidos").on("keypress",sololetras);
005       $("#edad").on("keypress",solodigitos);
006       $("#codigopostal").on({"keypress":solodigitos , "blur":muestra});
007       $("#guepardo").on({"mouseenter":entrarcaja , "mouseout":salircaja});
008       $("#leopardo").on({"mouseover":dentro , "mouseout":salirdentro , "mousemove":
          mover});
009       $(":text").on({"focus":entrar , "blur":salir});
010       $("body").on("keyup",imagenes);
011  }
```

ejemplo-jquery-521c.js

```
001  $(window).on("load",comienzo);
002  function comienzo(){
003       $(document).on("keypress","#nombre",sololetras);;
004       $(document).on("keypress","#apellidos",sololetras);
005       $(document).on("keypress","#edad",solodigitos);
006       $(document).on({"keypress":solodigitos , "blur":muestra},"#codigopostal");
007       $(document).on({"mouseenter":entrarcaja , "mouseout":salircaja},"#guepardo");
008       $(document).on({"mouseover":dentro , "mouseout":salirdentro,
          "mousemove":mover},"#leopardo");
009       $(document).on({"focus":entrar , "blur":salir},":text");
010       $(document).on("keyup","body",imagenes);
011  }
```

ejemplo-jquery-521d.js

```
001  $(window).on("load",comienzo);
002  function comienzo(){
003       $("body").on("keypress","#nombre",sololetras);;
004       $("body").on("keypress","#apellidos",sololetras);
```

```
005     $("body").on("keypress","#edad",solodigitos);
006     $("body").on({"keypress":solodigitos , "blur":muestra},"#codigopostal");
007     $("body").on({"mouseenter":entrarcaja , "mouseout":salircaja},"#guepardo");
008     $("body").on({"mouseover":dentro , "mouseout":salirdentro,
        "mousemove":mover},"#leopardo");
009     $("body").on({"focus":entrar , "blur":salir},":text");
010     $(document).on("keyup","body",imagenes);
011 }
```

ejemplo-jquery-521e.js

```
001 $(window).on("load",comienzo);
002 function comienzo(){
003     $("body").on("keypress","#nombre",sololetras);;
004     $("body").on("keypress","#apellidos",sololetras);
005     $("body").on("keypress","#edad",solodigitos);
006     $("body").on("keypress","#codigopostal",solodigitos);
007     $("body").on("blur","#codigopostal",muestra);
008     $("body").on("mouseenter","#guepardo",entrarcaja);
009     $("body").on("mouseout","#guepardo",salircaja);
010     $("body").on("mouseover","#leopardo",dentro);
011     $("body").on("mouseout","#leopardo",salirdentro);
012     $("body").on("mousemove","#leopardo",mover);
013     $("body").on("focus",":text",entrar);
014     $("body").on("blur",":text",salir);
015     $(document).on("keyup","body",imagenes);
016 }
```

ejemplo-jquery-521f.js

```
001 $(window).on("load",comienzo);
002 function comienzo(){
003     $("main").on("keypress","#nombre",sololetras);;
004     $("main").on("keypress","#apellidos",sololetras);
005     $("main").on("keypress","#edad",solodigitos);
006     $("main").on("keypress","#codigopostal",solodigitos);
007     $("main").on("blur","#codigopostal",muestra);
008     $("main").on("mouseenter","#guepardo",entrarcaja);
009     $("main").on("mouseout","#guepardo",salircaja);
010     $("main").on("mouseover","#leopardo",dentro);
011     $("main").on("mouseout","#leopardo",salirdentro);
012     $("main").on("mousemove","#leopardo",mover);
013     $("main").on("focus",":text",entrar);
014     $("main").on("blur",":text",salir);
015     $(document).on("keyup","body",imagenes);
016 }
```

Cuando nos situemos en las cajas de texto el color de fondo va a ser azul y el color de los caracteres blanco y además se va a borrar su contenido. Cuando abandonemos las cajas de texto el color de fondo y el color del texto va a ser el anterior. Cuando entre el ratón en el texto del guepardo el color de fondo va a ser rojo, el color del texto va a ser amarillo y el tamaño de las letras va a ser de 16 puntos. Cuando salga el ratón del texto del guepardo va a volver a su situación anterior. Cuando estemos con el ratón sobre el texto del leopardo el color del texto va a ser rojo. Cuando salga el ratón del texto del leopardo va a volver a su situación anterior. ejemplo-jquery-526.js

```
001 const VstFondoInicial="white";
002 const VstColorInicial="black";
003 const VstFondoDentro ="blue";
004 const VstColorDentro="white";
005 const VstColorEntrar="red";
006 const VstColorSalir="white";
007 const VstSizeDentro="16pt";
008 const VstSizeSalir="1rem";
009 const VstBcolorDentro="red";
010 const VstFcolorDentro="green";
011 const VstBcolorSalir="green";
012 const VstFcolorSalir="white";
013 $(window).on("load",comienzo);
014 function comienzo(){
015     $("#guepardo").on("mouseenter",{tletras:VstSizeDentro,Bcolor:VstBcolorDentro,
    Fcolor:VstFcolorDentro},entrarcaja);
016     $("#guepardo").on("mouseout",{tletras:VstSizeSalir,Bcolor:VstBcolorSalir,
```

```
    Fcolor:VstFcolorSalir},salircaja);
017     $("#leopardo").on("mouseover",{colorTexto:VstColorEntrar},dentro);
018     $("#leopardo").on("mouseout",{colorTexto:VstColorSalir},salirdentro);
019     $(":text").on("focus",{colorFondo:VstFondoDentro,colorTexto:VstColorDentro},
        entrar);
020     $(":text").on("blur",{colorFondo:VstFondoInicial,colorTexto:VstColorInicial},
        salir);
021 }
022 function entrar(evento){
023     $(evento.target).css({"background-color":evento.data.colorFondo,
        "color":evento.data.colorTexto});
024     $(evento.target).val("");
025 }
026 function salir(evento){
027     $(evento.target).css({"background-
        color":evento.data.colorFondo,"color":evento.data.colorTexto });
028 }
029 function salircaja(evento){
030     $(evento.target).css({"font-size":evento.data.tletras,"background-
        color":evento.data.Bcolor,"color":evento.data.Fcolor});
031 }
032 function entrarcaja(evento){
033     $(evento.target).css({"font-size":evento.data.tletras,"background-
        color":evento.data.Bcolor, "color":evento.data.Fcolor});
034 }
035 function dentro(evento){
036     $(evento.target).css("color",evento.data.colorTexto);
037 }
038 function salirdentro(evento){
039     $(evento.target).css("color",evento.data.colorTexto);
040 }
```

La caja de texto del código postal solamente va a admitir dígitos. Los caracteres que no sean dígitos se van a mostrar en la caja de texto correspondiente. Cuando salgamos de la caja de texto del código postal, si el mismo está comprendido entra 1000 y 52999 vamos a mostrar en la caja de texto de la provincia el nombre de la provincia. ejemplo-jquery-525.js

```
001 $(window).on("load",comienzo);
002 function comienzo()
003     $("#codigopostal").on("keypress",{enviar:"pulsadas"}, solodigitos);
004     $("#codigopostal").on("blur",muestra);
005 }
006 function solodigitos(evento){
007     var dato=String.fromCharCode(evento.charCode);
008     var continuar=true;
009     if (dato < "0" || dato > "9") {
010         continuar=false;
011         var todo="#"+evento.data.enviar;
012         console.log(evento.data.enviar);
013         console.log(evento.data);
014         var tiene=$(todo).val()+dato;
015         $(todo).val(tiene);
016     }
017     if (!continuar)
018         evento.preventDefault();
019 }
020 function muestra(evento){
021     var provincias=new Array("Araba/Álava","Albacete","Alicante/Alacant",
        "Almería","Ávila","Badajoz","Balears, Illes","Barcelona","Burgos","Cáceres",
        "Cádiz", "Castellón/Castelló","Ciudad Real","Córdoba","Coruña, A","Cuenca",
        "Girona","Granada","Guadalajara","Guipuzcua/Gipuzkoa","Huelva","Huesca","Jaén",
        "León","Lleida","Rioja, La","Lugo","Madrid","Málaga","Murcia", "Navarra",
        "Ourense", "Asturias","Palencia","Palmas, Las","Pontevedra","Salamanca", "Santa
        Cruz de Tenerife","Cantabria", "Segovia","Sevilla","Soria","Tarragona", "Teruel",
        "Toledo","Valencia/València","Valladolid","Bizkaia/Vizcaya","Zamora","Zaragoza",
        "Ceuta","Melilla")
022     var dato=$(evento.target).val().trim();
023     var continuar=true;
024     if (dato!=""){
025         var digitos="0123456789";
026         var VnbIndice=0
027         while( continuar && VnbIndice< dato.length){
```

```
028             if (!digitos.includes(dato.charAt(VnbIndice)))
029                 continuar=false;
030             VnbIndice+=1;
031         }
032         if (continuar){
033             var numero=parseInt(dato,10);
034             if (numero <1000 || numero >= 53000)
035                 continuar=false
036             else {
037                 posicion= Math.floor(numero / 1000) -1;
038                 $("#provincia").val(provincias[posicion]);
039             }
040         }
041     }
042     if (!continuar)
043         evento.preventDefault();
044 }
```

Con delegate se pone delante un selector padre del que se pone dentro

.delegate(*selector*, *evento*, *nombre-función*) → Se asigna la función indicada para el evento indicado sobre el selector.

.delegate(*selector*, *evento*, function() {*cuerpo-función*}) → Se asigna la función anónima indicada para el evento indicado sobre el selector.

.delegate(*selector*, *evento*, *datos*, function() {*cuerpo-función*}) → Se asigna la función anónima indicada para el evento indicado sobre el selector. Los datos se pasan en el formato **{*nombre*:*valor*, …}**. En la función ese dato le obtenemos mediante **event.data.*nombre***.

.delegate(*selector*, { *evento-1*:function(){*cuerpo-función-1*} [, *evento-2*:function(){*cuerpo-función-2*}]..}) → asigna sobre selector los eventos indicados con sus correspondientes funciones.

La caja de texto del nombre y la caja de texto de los apellidos solamente van a admitir letras y el espacio.
La caja de texto de la edad y la caja de texto del código postal solamente van a admitir dígitos.
Cuando nos situemos en las cajas de texto el color de fondo va a ser azul y el color de los caracteres blanco y además se va a borrar su contenido
Cuando abandonemos las cajas de texto el color de fondo y el color del texto van a ser los anteriores.
Cuando salgamos de la caja de texto del código postal, si el mismo está comprendido entra 1000 y 52999 vamos a mostrar en la caja de texto de la provincia el nombre de la provincia.
Cuando entre el ratón en el texto del guepardo el color de fondo va a ser rojo, el color del texto va a ser amarillo y el tamaño de las letras va a ser de 16 puntos.
Cuando salga el ratón del texto del guepardo va a volver a su situación anterior.
Cuando estemos con el ratón sobre el texto del leopardo el color del texto va a ser rojo.
Cuando movamos el ratón sobre el texto del leopardo el tamaño del texto va a variar, así como el color de fondo.
Cuando se pulse la tecla F8 se va a cambiar la imagen de fondo del texto del ocelote.

ejemplo-jquery-522.js

```
001 $(window).delegate(document,"load",comienzo);
002 function comienzo(){
003     $(document).delegate("#nombre","keypress",sololetras);;
004     $(document).delegate("#apellidos","keypress",sololetras);
005     $(document).delegate("#edad","keypress",solodigitos);
006     $(document).delegate("#codigopostal","keypress",solodigitos);
007     $(document).delegate("#codigopostal","blur",muestra);
008     $(document).delegate("#guepardo","mouseenter",entrarcaja);
009     $(document).delegate("#guepardo","mouseout",salircaja);
010     $(document).delegate("#leopardo","mouseover",dentro);
011     $(document).delegate("#leopardo","mouseout",salirdentro);
012     $(document).delegate("#leopardo","mousemove",mover);
013     $(document).delegate(":text","focus",entrar);
014     $(document).delegate(":text","blur",salir);
015     $(document).delegate("body","keyup",imagenes);
016 }
```

ejemplo-jquery-522a.js

```
001 $(window).delegate(document,"load",comienzo);
```

```
002  function comienzo(){
003      $("body").delegate("#nombre","keypress",sololetras);;
004      $("body").delegate("#apellidos","keypress",sololetras);
005      $("body").delegate("#edad","keypress",solodigitos);
006      $("body").delegate("#codigopostal","keypress",solodigitos);
007      $("body").delegate("#codigopostal","blur",muestra);
008      $("body").delegate("#guepardo","mouseenter",entrarcaja);
009      $("body").delegate("#guepardo","mouseout",salircaja);
010      $("body").delegate("#leopardo","mouseover",dentro);
011      $("body").delegate("#leopardo","mouseout",salirdentro);
012      $("body").delegate("#leopardo","mousemove",mover);
013      $("body").delegate(":text","focus",entrar);
014      $("body").delegate(":text","blur",salir);
015      $(document).delegate("body","keyup",imagenes);
016  }
```

ejemplo-jquery-522b.js

```
001  $(window).delegate(document,"load",comienzo);
002  function comienzo(){
003      $("main").delegate("#nombre","keypress",sololetras);;
004      $("main").delegate("#apellidos","keypress",sololetras);
005      $("main").delegate("#edad","keypress",solodigitos);
006      $("main").delegate("#codigopostal","keypress",solodigitos);
007      $("main").delegate("#codigopostal","blur",muestra);
008      $("main").delegate("#guepardo","mouseenter",entrarcaja);
009      $("main").delegate("#guepardo","mouseout",salircaja);
010      $("main").delegate("#leopardo","mouseover",dentro);
011      $("main").delegate("#leopardo","mouseout",salirdentro);
012      $("main").delegate("#leopardo","mousemove",mover);
013      $("main").delegate(":text","focus",entrar);
014      $("main").delegate(":text","blur",salir);
015      $(document).delegate("body","keyup",imagenes);
016  }
```

ejemplo-jquery-522c.js

```
001  $(window).delegate(document,"load",comienzo);
002  function comienzo(){
003      $(document).delegate("#nombre","keypress",sololetras);;
004      $(document).delegate("#apellidos","keypress",sololetras);
005      $(document).delegate("#edad","keypress",solodigitos);
006      $(document).delegate("#codigopostal",{"keypress":solodigitos ,
"blur":muestra});
007      $(document).delegate("#guepardo",{"mouseenter":entrarcaja ,
"mouseout":salircaja});
008      $(document).delegate("#leopardo",{"mouseover":dentro , "mouseout":salirdentro ,
"mousemove":mover});
009      $(document).delegate(":text",{"focus":entrar,"blur":salir});
010      $(document).delegate("body","keyup",imagenes);
011  }
```

Cuando nos situemos en las cajas de texto el color de fondo va a ser azul y el color de los caracteres blanco y además se va a borrar su contenido. Cuando abandonemos las cajas de texto el color de fondo y el color del texto va a ser el anterior. Cuando entre el ratón en el texto del guepardo el color de fondo va a ser rojo, el color del texto va a ser amarillo y el tamaño de las letras va a ser de 16 puntos. Cuando salga el ratón del texto del guepardo va a volver a su situación anterior. Cuando estemos con el ratón sobre el texto del leopardo el color del texto va a ser rojo. Cuando salga el ratón del texto del leopardo va a volver a su situación anterior. ejemplo-jquery-526a.js

```
001  const VstFondoInicial="white";
002  const VstColorInicial="black";
003  const VstFondoDentro ="blue";
004  const VstColorDentro="white";
005  const VstColorEntrar="red";
006  const VstColorSalir="white";
007  const VstSizeDentro="16pt";
008  const VstSizeSalir="1rem";
009  const VstBcolorDentro="red";
010  const VstFcolorDentro="green";
011  const VstBcolorSalir="green";
012  const VstFcolorSalir="white";
013  $(window).delegate(document,"load",comienzo);
```

```
014 function comienzo(){
015     $(document).delegate("#guepardo","mouseenter",{tletras:VstSizeDentro,
Bcolor:VstBcolorDentro,Fcolor:VstFcolorDentro},entrarcaja);
016     $(document).delegate("#guepardo","mouseout",{tletras:VstSizeSalir,
Bcolor:VstBcolorSalir,Fcolor:VstFcolorSalir},salircaja);
017     $(document).delegate("#leopardo","mouseover",{colorTexto:VstColorEntrar},
dentro);
018     $(document).delegate("#leopardo","mouseout",{colorTexto:VstColorSalir},
salirdentro);
019     $(document).delegate(":text","focus",{colorFondo:VstFondoDentro,
colorTexto:VstColorDentro},entrar);
020     $(document).delegate(":text","blur",{colorFondo:VstFondoInicial,
colorTexto:VstColorInicial},salir);
021 }
022 function entrar(evento){
023     $(evento.target).css({"background-color":evento.data.colorFondo,
"color":evento.data.colorTexto});
024     $(evento.target).val("");
025 }
026 function salir(evento){
027     $(evento.target).css({"background-
color":evento.data.colorFondo,"color":evento.data.colorTexto });
028 }
029 function salircaja(evento){
030     $(evento.target).css({"font-size":evento.data.tletras,"background-
color":evento.data.Bcolor,"color":evento.data.Fcolor});
031 }
032 function entrarcaja(evento){
033     $(evento.target).css({"font-size":evento.data.tletras,"background-
color":evento.data.Bcolor, "color":evento.data.Fcolor});
034 }
035 function dentro(evento){
036     $(evento.target).css("color",evento.data.colorTexto);
037 }
038 function salirdentro(evento){
039     $(evento.target).css("color",evento.data.colorTexto);
040 }
```

La caja de texto del código postal solamente va a admitir dígitos. Los caracteres que no sean dígitos se van a mostrar en la caja de texto correspondiente. Cuando salgamos de la caja de texto del código postal, si el mismo está comprendido entra 1000 y 52999 vamos a mostrar en la caja de texto de la provincia el nombre de la provincia. ejemplo-jquery-525a.js

```
001 const VstFondoInicial="white";
002 const VstColorInicial="black";
003 $(window).delegate(document,"load",comienzo);
004 function comienzo(){
005     $(document).delegate("#codigopostal","keypress",{enviar:"pulsadas"},
solodigitos);
006     $(document).delegate("#codigopostal","blur",muestra);
007 }
008 function solodigitos(evento){
009     var dato=String.fromCharCode(evento.charCode);
010     var continuar=true;
011     if (dato < "0" || dato > "9") {
012         continuar=false;
013         var todo="#"+evento.data.enviar;
014         console.log(evento.data.enviar);
015         console.log(evento.data);
016         var tiene=$(todo).val()+dato;
017         $(todo).val(tiene);
018     }
019     if (!continuar)
020         evento.preventDefault();
021 }
022 function muestra(evento){
023     var provincias=new
Array("Araba/Álava","Albacete","Alicante/Alacant","Almería","Ávila",
"Badajoz","Balears, Illes","Barcelona","Burgos",
"Cáceres","Cádiz","Castellón/Castelló","Ciudad Real","Córdoba","Coruña,
A","Cuenca","Girona","Granada","Guadalajara", "Guipuzcua/Gipuzkoa",
"Huelva","Huesca","Jaén", "León","Lleida","Rioja, La","Lugo", "Madrid",
```

```
    "Málaga","Murcia","Navarra","Ourense","Asturias",          "Palencia","Palmas,
Las","Pontevedra","Salamanca","Santa      Cruz      de      Tenerife","Cantabria",
"Segovia","Sevilla",
"Soria","Tarragona","Teruel","Toledo","Valencia/València","Valladolid",
"Bizkaia/Vizcaya", "Zamora","Zaragoza","Ceuta","Melilla")
024     var dato=$(evento.target).val().trim();
025     var continuar=true;
026     if (dato!=""){
027         var digitos="0123456789";
028         var VnbIndice=0
029         while( continuar && VnbIndice< dato.length){
030             if (!digitos.includes(dato.charAt(VnbIndice)))
031                 continuar=false;
032             VnbIndice+=1;
033         }
034         if (continuar){
035             var numero=parseInt(dato,10);
036             if (numero <1000 || numero >= 53000)
037                 continuar=false
038             else {
039                 posicion= Math.floor(numero / 1000) -1;
040                 $("#provincia").val(provincias[posicion]);
041             }
042         }
043     }
044     if (!continuar)
045         evento.preventDefault();
046 }
```

.blind(*lista-eventos*, nombre-función) → Cuando se produzcan los eventos indicados en el selector se va a ejecutar la función cuyo nombre ponemos.

.blind(*lista-eventos*, function() {*cuerpo-función*}) → Cuando se produzcan los eventos indicados en el selector se va a ejecutar la función anónima indicada.

.blind(*lista-eventos*, *datos*, nombre-función) → Cuando se produzcan los eventos indicados en el selector se va a ejecutar la función cuyo nombre ponemos. Los datos se pasan en el formato **{*nombre*:*valor*, …}**. En la función ese dato le obtenemos mediante **event.data.*nombre***.

.blind(*lista-eventos*, *datos*, function() {*cuerpo-función*}) → Cuando se produzcan los eventos indicados en el selector se va a ejecutar la función anónima indicada. Los datos se pasan en el formato **{*nombre*:*valor*, …}**. En la función ese dato le obtenemos mediante **event.data.*nombre***.

.blind({*evento-1*: *nombre-función-1*} [, *evento-2*: *nombre-función-2*}]..}) → Asigna a los eventos indicados las funciones anónimas que ponemos.

.blind({*evento-1*:function(){*cuerpo-función-1*} [, *evento-2*:function(){*cuerpo-función-2*}]..}) → Asigna a los eventos indicados las funciones anónimas que ponemos.

La caja de texto del nombre y la caja de texto de los apellidos solamente van a admitir letras y el espacio.
La caja de texto de la edad y la caja de texto del código postal solamente van a admitir dígitos.
Cuando nos situemos en las cajas de texto el color de fondo va a ser azul y el color de los caracteres blanco y además se va a borrar su contenido
Cuando abandonemos las cajas de texto el color de fondo y el color del texto van a ser los anteriores.
Cuando salgamos de la caja de texto del código postal, si el mismo está comprendido entra 1000 y 52999 vamos a mostrar en la caja de texto de la provincia el nombre de la provincia.
Cuando entre el ratón en el texto del guepardo el color de fondo va a ser rojo, el color del texto va a ser amarillo y el tamaño de las letras va a ser de 16 puntos.
Cuando salga el ratón del texto del guepardo va a volver a su situación anterior.
Cuando estemos con el ratón sobre el texto del leopardo el color del texto va a ser rojo.
Cuando movamos el ratón sobre el texto del leopardo el tamaño del texto va a variar, así como el color de fondo.
Cuando se pulse la tecla F8 se va a cambiar la imagen de fondo del texto del ocelote.
ejemplo-jquery-523.js

```
001 $(window).bind("load",comienzo);
002 function comienzo(){
003     $("#nombre").bind("keypress",sololetras);;
```

```
004     $("#apellidos").bind("keypress",sololetras);
005     $("#edad").bind("keypress",solodigitos);
006     $("#codigopostal").bind("keypress",solodigitos);
007     $("#codigopostal").bind("blur",muestra);
008     $("#guepardo").bind("mouseenter",entrarcaja);
009     $("#guepardo").bind("mouseout",salircaja);
010     $("#leopardo").bind("mouseover",dentro);
011     $("#leopardo").bind("mouseout",salirdentro);
012     $("#leopardo").bind("mousemove",mover);
013     $(":text").bind("focus",entrar);
014     $(":text").bind("blur",salir);
015     $("body").bind("keyup",imagenes);
016 }
```

ejemplo-jquery-523a.js

```
001 $(window).bind("load",comienzo);
002 function comienzo(){
003     $("#nombre").bind("keypress",sololetras);;
004     $("#apellidos").bind("keypress",sololetras);
005     $("#edad").bind("keypress",solodigitos);
006     $("#codigopostal").bind({"keypress":solodigitos , "blur":muestra});
007     $("#guepardo").bind({"mouseenter":entrarcaja , "mouseout":salircaja});
008     $("#leopardo").bind({"mouseover":dentro , "mouseout":salirdentro ,
"mousemove":mover});
009     $(":text").bind({"focus":entrar,"blur":salir});
010     $("body").bind("keyup",imagenes);
011 }
```

Cuando nos situemos en las cajas de texto el color de fondo va a ser azul y el color de los caracteres blanco y además se va a borrar su contenido. Cuando abandonemos las cajas de texto el color de fondo y el color del texto va a ser el anterior. Cuando entre el ratón en el texto del guepardo el color de fondo va a ser rojo, el color del texto va a ser amarillo y el tamaño de las letras va a ser de 16 puntos. Cuando salga el ratón del texto del guepardo va a volver a su situación anterior. Cuando estemos con el ratón sobre el texto del leopardo el color del texto va a ser rojo. Cuando salga el ratón del texto del leopardo va a volver a su situación anterior. ejemplo-jquery-526b.js

```
001 const VstFondoInicial="white";
002 const VstColorInicial="black";
003 const VstFondoDentro ="blue";
004 const VstColorDentro="white";
005 const VstColorEntrar="red";
006 const VstColorSalir="white";
007 const VstSizeDentro="16pt";
008 const VstSizeSalir="1rem";
009 const VstBcolorDentro="red";
010 const VstFcolorDentro="green";
011 const VstBcolorSalir="green";
012 const VstFcolorSalir="white";
013 $(window).bind("load",comienzo);
014 function comienzo(){
015     $("#guepardo").bind("mouseenter",{tletras:VstSizeDentro,Bcolor:VstBcolorDentro,
Fcolor:VstFcolorDentro},entrarcaja);
016     $("#guepardo").bind("mouseout",{tletras:VstSizeSalir,Bcolor:VstBcolorSalir,
Fcolor:VstFcolorSalir},salircaja);
017     $("#leopardo").bind("mouseover",{colorTexto:VstColorEntrar},dentro);
018     $("#leopardo").bind("mouseout",{colorTexto:VstColorSalir},salirdentro);
019     $(":text").bind("focus",{colorFondo:VstFondoDentro,colorTexto:VstColorDentro},
entrar);
020     $(":text").bind("blur",{colorFondo:VstFondoInicial,colorTexto:VstColorInicial},
salir);
021 }
022 function entrar(evento){
023     $(evento.target).css({"background-color":evento.data.colorFondo,
"color":evento.data.colorTexto});
024     $(evento.target).val("");
025 }
026 function salir(evento){
027     $(evento.target).css({"background-
color":evento.data.colorFondo,"color":evento.data.colorTexto });
028 }
029 function salircaja(evento){
030     $(evento.target).css({"font-size":evento.data.tletras,"background-
```

```
    color":evento.data.Bcolor,"color":evento.data.Fcolor});
}
function entrarcaja(evento){
    $(evento.target).css({"font-size":evento.data.tletras,"background-color":evento.data.Bcolor, "color":evento.data.Fcolor});
}
function dentro(evento){
    $(evento.target).css("color",evento.data.colorTexto);
}
function salirdentro(evento){
    $(evento.target).css("color",evento.data.colorTexto);
}
```

La caja de texto del código postal solamente va a admitir dígitos. Los caracteres que no sean dígitos se van a mostrar en la caja de texto correspondiente. Cuando salgamos de la caja de texto del código postal, si el mismo está comprendido entra 1000 y 52999 vamos a mostrar en la caja de texto de la provincia el nombre de la provincia. ejemplo-jquery-525c.js

```
const VstFondoInicial="white";
const VstColorInicial="black";
$(window).bind("load",comienzo);
function comienzo() {
    $("#codigopostal").bind("keypress",{enviar:"pulsadas"}, solodigitos);
    $("#codigopostal").bind("blur",muestra);
}
function solodigitos(evento){
    var dato=String.fromCharCode(evento.charCode);
    var continuar=true;
    if (dato < "0" || dato > "9") {
        continuar=false;
        var todo="#"+evento.data.enviar;
        console.log(evento.data.enviar);
        console.log(evento.data);
        var tiene=$(todo).val()+dato;
        $(todo).val(tiene);
    }
    if (!continuar)
        evento.preventDefault();
}
function muestra(evento){
    var provincias=new Array("Araba/Álava","Albacete","Alicante/Alacant","Almería","Ávila","Badajoz","Balears, Illes","Barcelona","Burgos","Cáceres","Cádiz","Castellón/Castelló","Ciudad Real", "Córdoba", "Coruña, A","Cuenca","Girona","Granada", "Guadalajara", "Guipuzcua/Gipuzkoa","Huelva","Huesca","Jaén", "León","Lleida","Rioja, La","Lugo","Madrid","Málaga","Murcia", "Navarra","Ourense","Asturias", "Palencia","Palmas, Las","Pontevedra","Salamanca", "Santa Cruz de Tenerife","Cantabria","Segovia","Sevilla","Soria","Tarragona","Teruel","Toledo","Valencia/València","Valladolid", "Bizkaia/Vizcaya","Zamora", "Zaragoza","Ceuta","Melilla")
    var dato=$(evento.target).val().trim();
    var continuar=true;
    if (dato!=""){
        var digitos="0123456789";
        var VnbIndice=0
        while( continuar && VnbIndice< dato.length){
            if (!digitos.includes(dato.charAt(VnbIndice)))
                continuar=false;
            VnbIndice+=1;
        }
        if (continuar){
            var numero=parseInt(dato,10);
            if (numero <1000 || numero >= 53000)
                continuar=false
            else {
                posicion= Math.floor(numero / 1000) -1;
                $("#provincia").val(provincias[posicion]);
            }
        }
    }
    if (!continuar)
        evento.preventDefault();
```

```
046 }
```

.one(*evento* , *función*) → Se asigna la función indicada para el evento indicado sobre el selector. La función va a tener un parámetro que es el evento. Se ejecuta una única vez.
.one({*evento*:*función*, …}) → Se asignan las funciones indicadas para los eventos indicados sobre el selector. Se ejecuta una única vez.
.one(*evento*, *dato* , *función*) → Se asigna la función indicada para el evento indicado sobre el selector. La función va a tener un parámetro que es el evento y los datos los obtenemos a través de **event.data**. Se ejecuta una única vez. Los datos se pasan en el formato **{*nombre*:*valor*, …}**. En la función ese dato le obtenemos mediante **event.data.*nombre***.
.one(*evento*, *selector* [, *dato*], *función*) → Se asigna la función indicada para el evento indicado sobre el selector. La función va a tener un parámetro que es el evento y los datos los obtenemos a través de **event.data**. Se ejecuta una única vez. Los datos se pasan en el formato **{*nombre*:*valor*, …}**. En la función ese dato le obtenemos mediante **event.data.*nombre***.
.one({*evento*:*función*, …}, *selector* [, *dato*]) → Se asignan las funciones indicadas para los eventos indicados sobre el selector. Se ejecuta una única vez. Los datos se pasan en el formato **{*nombre*:*valor*, …}**. En la función ese dato le obtenemos mediante **event.data.*nombre***.
Con one, cuando dentro se pone un selector, se pone delante un selector padre del que se pone dentro.

La caja de texto del nombre y la caja de texto de los apellidos solamente van a admitir letras y el espacio. 1 vez.
La caja de texto de la edad y la caja de texto del código postal solamente van a admitir dígitos. 1 vez.
Cuando nos situemos en las cajas de texto el color de fondo va a ser azul y el color de los caracteres blanco y además se va a borrar su contenido. 1 vez.
Cuando abandonemos las cajas de texto el color de fondo y el color del texto va a ser el anterior. 1 vez.
Cuando salgamos de la caja de texto del código postal, si el mismo está comprendido entra 1000 y 52999 vamos a mostrar en la caja de texto de la provincia el nombre de la provincia. 1 vez
Cuando entre el ratón en el texto del guepardo el color de fondo va a ser rojo, el color del texto va a ser amarillo y el tamaño de las letras va a ser de 16 puntos. 1 vez.
Cuando salga el ratón del texto del guepardo va a volver a su situación anterior. 1 vez.
Cuando estemos con el ratón sobre el texto del leopardo el color del texto va a ser rojo. 1 vez.
Cuando movamos el ratón sobre el texto del leopardo el tamaño del texto va a variar, así como el color de fondo. 1 vez.
Cuando se pulse la tecla F8 se va a cambiar la imagen de fondo del texto del ocelote. 1 vez.
ejemplo-jquery-524.js

```
001 $(window).one("load",comienzo);
002 function comienzo(){
003     $("#nombre").one("keypress",sololetras);;
004     $("#apellidos").one("keypress",sololetras);
005     $("#edad").one("keypress",solodigitos);
006     $("#codigopostal").one("keypress",solodigitos);
007     $("#codigopostal").one("blur",muestra);
008     $("#guepardo").one("mouseenter",entrarcaja);
009     $("#guepardo").one("mouseout",salircaja);
010     $("#leopardo").one("mouseover",dentro);
011     $("#leopardo").one("mouseout",salirdentro);
012     $("#leopardo").one("mousemove",mover);
013     $(":text").one("focus",entrar);
014     $(":text").one("blur",salir);
015     $("body").one("keyup",imagenes);
016 }
```

ejemplo-jquery-524a.js

```
001 $(window).one("load",comienzo);
002 function comienzo(){
003     $("#nombre").one("keypress",sololetras);;
004     $("#apellidos").one("keypress",sololetras);
005     $("#edad").one("keypress",solodigitos);
006     $("#codigopostal").one({"keypress":solodigitos , "blur":muestra});
007     $("#guepardo").one({"mouseenter":entrarcaja , "mouseout":salircaja});
008     $("#leopardo").one({"mouseover":dentro      ,      "mouseout":salirdentro      ,
```

```
      "mousemove":mover});
009       $(":text").one({"focus":entrar,"blur":salir});
010       $("body").one("keyup",imagenes);
011   }
```

ejemplo-jquery-524b.js

```
001   $(window).one("load",comienzo);
002   function comienzo(){
003       $(document).one("keypress","#nombre",sololetras);;
004       $(document).one("keypress","#apellidos",sololetras);
005       $(document).one("keypress","#edad",solodigitos);
006       $(document).one("keypress","#codigopostal",solodigitos);
007       $(document).one("blur","#codigopostal",muestra);
008       $(document).one("mouseenter","#guepardo",entrarcaja);
009       $(document).one("mouseout","#guepardo",salircaja);
010       $(document).one("mouseover","#leopardo",dentro);
011       $(document).one("mouseout","#leopardo",salirdentro);
012       $(document).one("mousemove","#leopardo",mover);
013       $(document).one("focus",":text",entrar);
014       $(document).one("blur",":text",salir);
015       $(document).one("keyup","body",imagenes);
016   }
```

ejemplo-jquery-524c.js

```
001   $(window).one("load",comienzo);
002   function comienzo(){
003       $(document).one("keypress","#nombre",sololetras);;
004       $(document).one("keypress","#apellidos",sololetras);
005       $(document).one("keypress","#edad",solodigitos);
006       $(document).one({"keypress":solodigitos , "blur":muestra},"#codigopostal");
007       $(document).one({"mouseenter":entrarcaja , "mouseout":salircaja},"#guepardo");
008       $(document).one({"mouseover":dentro , "mouseout":salirdentro,
      "mousemove":mover},"#leopardo");
009       $(document).one({"focus":entrar , "blur":salir},":text");
010       $(document).one("keyup","body",imagenes);
011   }
```

.off(*evento*)→quita las funciones de eventos para ese evento.
.off(*evento, selector*)→ quita las funciones de eventos para ese evento y ese selector.
.off(*evento, selector, nombre-función*)→ quita la función de evento indicada para ese evento y ese selector.
.off()→ quita todas las funciones de eventos.

```
001   function quitar(){
002       $("#nombre").off("keypress",sololetras);;
003       $("#apellidos").off("keypress",sololetras);
004       $("#edad").off("keypress",solodigitos);
005       $("#codigopostal").off("keypress",solodigitos);
006       $("#codigopostal").off("blur",muestra);
007       $("#guepardo").off("mouseenter",entrarcaja);
008       $("#guepardo").off("mouseout",salircaja);
009       $("#leopardo").off("mouseover",dentro);
010       $("#leopardo").off("mouseout",salirdentro);
011       $("#leopardo").off("mousemove",mover);
012       $(":text").off("focus",entrar);
013       $(":text").off("blur",salir);
014       $("body").off("keyup",imagenes);
015   }
```

.undelegate(*selector*) → quita todos las funciones de eventos para ese selector.
.undelegate(*evento*) → quita todas las funciones de eventos para ese evento.
.undelegate(*selector, evento*) → quita todas las funciones de eventos para ese evento y ese selector.

.undelegate(*selector*, *evento*, *nombre-función*) → quita la función indicada para ese evento y ese selector.
.undelegate()→quita todas las funciones de eventos.

```
001  function quitar(){
002      $(document).undelegate("#nombre","keypress",sololetras);;
003      $(document).undelegate("#apellidos","keypress",sololetras);
004      $(document).undelegate("#edad","keypress",solodigitos);
005      $(document).undelegate("#codigopostal","keypress",solodigitos);
006      $(document).undelegate("#codigopostal","blur",muestra);
007      $(document).undelegate("#guepardo","mouseenter",entrarcaja);
008      $(document).undelegate("#guepardo","mouseout",salircaja);
009      $(document).undelegate("#leopardo","mouseover",dentro);
010      $(document).undelegate("#leopardo","mouseout",salirdentro);
011      $(document).undelegate("#leopardo","mousemove",mover);
012      $(document).undelegate(":text","focus",entrar);
013      $(document).undelegate(":text","blur",salir);
014      $(document).undelegate("body","keyup",imagenes);
015  }
```

Objeto event

Propiedades

- **target** → elemento sobre el cual se produce el evento.
- **type** → nos indica el evento que se ha producido.
- **currentTarget** → elemento que detecta el evento en la fase de propagación.
- **delegateTarget** → elemento sobre el que se ha declarado el evento.
- **relatedTarget** → el otro elemento involucrado en el evento, si existe o le hay.
- **data** → mediante esta propiedad vamos a conseguir accede a todos los valores que se pasan como datos a la función de manejo del evento.
- **which** → para los evento del ratón indica el botón pulsado (1 principal, 2 medio y 3 secundario) y para los eventos del teclado indica la tecla pulsada.
- **pageX** → coordenada horizontal de donde está el ratón.
- **pageY** → coordenada vertical de donde está el ratón.
- **clientX** → Coordenada horizontal del cursor dentro del área del navegador, sin tener en cuenta las barras de desplazamiento
- **clientY** → Coordenada horizontal del cursor dentro del área del navegador, sin tener en cuenta las barras de desplazamiento
- **screenX** → Coordenada horizontal del cursor dentro de la pantalla
- **screenY** → Coordenada vertical del cursor dentro de la pantalla
- **offsetX** → Coordenada horizontal del cursor dentro del elemento HTML que haya provocado el evento
- **offsetY** → Coordenada vertical del cursor dentro del elemento HTML que haya provocado el evento
- **button** → Botones del ratón pulsados al ocurrir el evento:0 ó 1=botónizquierdo (principal),2 = botón derecho, (secundario o alternativo), 4 = botón central.El valor de event.buttones la suma de los valores correspondientes a los botones pulsados. Por ejemplo si se pulsa simultáneamente los botones izquierdo y central se obtiene 5, 0 si no se pulsa ninguno.
- **buttons** → Botones del ratón pulsados al ocurrir el evento:0 ó 1=botónizquierdo (principal),2 = botón derecho, (secundario o alternativo), 4 = botón central.El valor de event.buttones la suma de los valores correspondientes a los botones pulsados. Por ejemplo si se pulsa simultáneamente los botones izquierdo y central se obtiene 5, 0 si no se pulsa ninguno.
- **detail** → número de veces que se pulsa el ratón, incrementado en uno.

- **result** →el ultimo valor devuelto por la function de manejo del evento.
- **altKey** → valor lógico que indica si se ha pulsado la tecla «Alt».
- **shiftKey** → valor lógico que indica si se ha pulsado la tecla «Shift» (mayúsculas).
- **ctrlKey** → valor lógico que indica si se ha pulsado la tecla «Control».
- **metaKey** → indica mediante un valor lógico, si se ha pulsado la tecla de windows.
- **key** → Carácter de la tecla pulsada
- **char** → Carácter de la tecla pulsada
- **charCode** → Valor Tecla
- **keyCode** → Valor ASCII de la tecla pulsada
- **bubbles** → Devuelve verdadero o falso según cómo se inicializó el evento. Verdadero si el evento pasa por los antepasados de su valor de atributo de destino en orden de árbol inverso, y falso de lo contrario.
- **cancelable** → Devuelve verdadero o falso según cómo se inicializó el evento. Su valor de retorno no siempre tiene significado, pero verdadero puede indicar que parte de la operación durante la cual se envió el evento, puede cancelarse invocando el método preventDefault ().
- **eventPhase** → Devuelve un número que nos indica en qué fase de la notificación del evento nos encontramos (1 corresponde a la fase de captura, 2 a la de destino, y 3 a la de propagación).
- **toElement** → Elemento HTML posterior a (mouseover o mouseout).
- **view** → ventana.
- **timeStamp** → los milisegundos transcurridos desde el 1 de enero de 1970 hasta el momento en que se produjo el evento.
- **namespace** → el espacio de nombre dende se produjo el evento.
- **originalEvent** → Para acceder a las propiedades de eventos que no figuran en la lista anterior.

Métodos

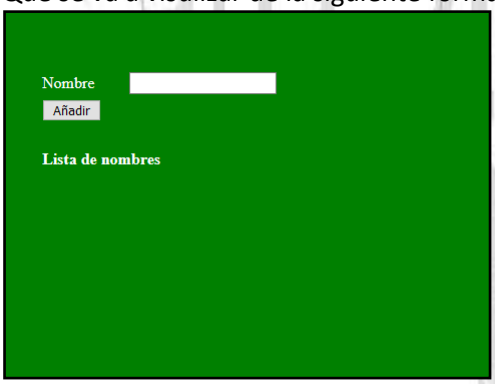
- **isDefaultPrevented()** → devuelve un valor lógico, que indica si el método precentDefault() fue ejecutado con anteriorida en este manejador de eventos (function).
- **isImmediatePropagationStopped()** → devuelve un valor lógico, que indica si el método stopImediatePropagation() fue ejecutado con anteriorida en este manejador de eventos (function).
- **isPropagationStopped()** → devuelve un valor lógico, que indica si el método stopPropagation() fue ejecutado con anteriorida en este manejador de eventos (function)
- **preventDefault()** → No se realiza la acción predeterminada del evento.
- **stopImmediatePropagation()** → Detiene la fase de propagación del evento, para que otros controladores de evento no le detecten.
- **stopPropagation()** → Impide que el evento se propage en el árbol DOM, evitando que los controladores principales sean notificados del evento.

Ejemplo de utilización de jQuery.
Disponemos del siguiente formulario.
ejemplo-jquery-501.html

```
001 <!doctype html>
002 <html lang="es">
003     <head>
004         <title>ejemplo501 jQuery inserta nodo li</title>
005         <meta charset="utf-8" />
```

```
006         <meta name="author" content="felix angel muñoz bayon" />
007         <link href="css/ejemplo-jquery-501.css" rel="stylesheet" type="text/css" />
008         <style type="text/css">
009         </style>
010         <script src="jquery/jquery-3.6.0.min.js" type="text/javascript"></script>
011         <script src="javascript/ejemplo-jquery-501a.js"
        type="text/javascript"></script>
012         <script type="text/javascript">
013         </script>
014     </head>
015     <body>
016         <header>
017         </header>
018         <nav>
019         </nav>
020         <main>
021             <form>
022                 <label for="nombre">Nombre </label>
023                 <input type="text" id="nombre" name="nombre"  /> <br />
024                 <input type="button" id="hacer" value="Añadir" />
025                 <h4>Lista de nombres</h4>
026                 <ul id="lista">
027                 </ul>
028             </form>
029         </main>
030         <footer>
031         </footer>
032     </body>
033 </html>
```

Que se va a visulizar de la siguiente forma



Sobre el mismo vamos a ejecutar diferentes códigos javascript para realizar tareas sobre él.

Cuando se pulse el botón Añadir si la caja de texto no está vacía se va a añadir su contenido al final de la lista no ordenada, que hay más abajo. ejemplo-jquery-501.js

```
001 $(function(){
002     $("#hacer").on("click",insertar);
003 });
004 function insertar() {
005     var dato=$("#nombre").val().trim();
006     if (dato!="") {
007         $("#lista").append("<li>"+dato+"</li>");
008     }
009 }
```

Cuando se pulse el botón Añadir si la caja de texto no está vacía se va a añadir su contenido al inicio de la lista no ordenada, que hay más abajo. ejemplo-jquery-501a.js

```
001 $(function(){
002     $("#hacer").on("click",insertar);
003 });
004 function insertar() {
005     var dato=$("#nombre").val().trim();
```

```
006     if (dato!="") {
007         $("#lista").prepend("<li>"+dato+"</li>");
008     }
009 }
```

Cuando se pulse el botón Añadir si la caja de texto no está vacía se va a añadir su contenido al final de la lista no ordenada, que hay más abajo, siempre y cuando no exista en la misma. ejemplo-jquery-502.js

```
001 $(function(){
002     $("#hacer").on("click",insertar);
003 });
004 function insertar() {
005     var dato=$("#nombre").val().trim();
006     if (dato!="") {
007         var todos=$("#lista>li");
008         var inexistente=true;
009         var indice=0;
010         while (inexistente && indice < $(todos).length) {
011             if (dato == $(todos).eq(indice).text())
012                 inexistente=false;
013             indice+=1;
014         }
015         if (inexistente) {
016             $("#lista").append("<li>"+dato+"</li>");
017         }
018     }
019 }
```

Cuando se pulse el botón Añadir si la caja de texto no está vacía se va a añadir su contenido al inicio de la lista no ordenada, que hay más abajo, siempre y cuando no exista en la misma. ejemplo-jquery-502a.js.

```
001 $(function(){
002     $("#hacer").on("click",insertar);
003 });
004 function insertar() {
005     var dato=$("#nombre").val().trim();
006     if (dato!="") {
007         var todos=$("#lista>li");
008         var inexistente=true;
009         var indice=0;
010         while (inexistente && indice < $(todos).length) {
011             if (dato == $(todos).eq(indice).text())
012                 inexistente=false;
013             indice+=1;
014         }
015         if (inexistente) {
016             $("#lista").prepend("<li>"+dato+"</li>");
017         }
018     }
019 }
```

Cuando se pulse el botón Añadir si la caja de texto no está vacía se va a añadir su contenido a la lista no ordenada, que hay más abajo, siempre y cuando no exista en la misma y ordenado alfabéticamente en orden descendete. ejemplo-jquery-503.js

```
001 $(function(){
002     $("#hacer").on("click",insertar);
003 });
004 function insertar() {
005     var dato=$("#nombre").val().trim();
006     if (dato!="") {
007         var todos=$("#lista>li");
008         var inexistente=true;
009         var indice=0;
010         while (inexistente && indice < $(todos).length) {
011             if (dato == $(todos).eq(indice).text())
012                 inexistente=false
013             else if (dato > $(todos).eq(indice).text()){
014                 inexistente=false;
015                 $(todos).eq(indice).before("<li>"+dato+"</li>");
016             }
017             indice+=1;
```

```
        }
        if (inexistente) {
            $("#lista").append("<li>"+dato+"</li>");
        }
    }
}
```

Cuando se pulse el botón Añadir si la caja de texto no está vacía se va a añadir su contenido a la lista no ordenada, que hay más abajo, siempre y cuando no exista en la misma y ordenado alfabéticamente en orden descendete. ejemplo-jquery-503a.js

```
$(function(){
    $("#hacer").on("click",insertar);
});
function insertar() {
    var dato=$("#nombre").val().trim();
    if (dato!="") {
        var todos=$("#lista>li");
        var inexistente=true;
        var indice=$(todos).length-1;
        while (inexistente && indice >=0 ) {
            if (dato == $(todos).eq(indice).text())
                inexistente=false
            else if (dato < $(todos).eq(indice).text()){
                inexistente=false;
                $(todos).eq(indice).after("<li>"+dato+"</li>");
            }
            indice-=1;
        }
        if (inexistente) {
            $("#lista").prepend("<li>"+dato+"</li>");
        }
    }
}
```

Cuando se pulse el botón Añadir si la caja de texto no está vacía se va a añadir su contenido a la lista no ordenada, que hay más abajo, siempre y cuando no exista en la misma y ordenado alfabéticamente en orden ascendete. ejemplo-jquery-504.js

```
$(function(){
    $("#hacer").on("click",insertar);
});
function insertar() {
    var dato=$("#nombre").val().trim();
    if (dato!="") {
        var todos=$("#lista>li");
        var inexistente=true;
        var indice=0;
        while (inexistente && indice < $(todos).length) {
            if (dato == $(todos).eq(indice).text())
                inexistente=false
            else if (dato < $(todos).eq(indice).text()){
                inexistente=false;
                $(todos).eq(indice).before("<li>"+dato+"</li>");
            }
            indice+=1;
        }
        if (inexistente) {
            $("#lista").append("<li>"+dato+"</li>");
        }
    }
}
```

Cuando se pulse el botón Añadir si la caja de texto no está vacía se va a añadir su contenido a la lista no ordenada, que hay más abajo, siempre y cuando no exista en la misma y ordenado alfabéticamente en orden ascendete. . ejemplo-jquery-504a.js.

```
$(function(){
    $("#hacer").on("click",insertar);
});
function insertar() {
    var dato=$("#nombre").val().trim();
    if (dato!="") {
```

```
007             var todos=$("#lista>li");
008             var inexistente=true;
009             var indice=$(todos).length - 1;
010             while (inexistente && indice >=0 ) {
011                 if (dato == $(todos).eq(indice).text())
012                     inexistente=false
013                 else if (dato > $(todos).eq(indice).text()){
014                     inexistente=false;
015                     $(todos).eq(indice).after("<li>"+dato+"</li>");
016                 }
017                 indice-=1;
018             }
019             if (inexistente) {
020                 $("#lista").prepend("<li>"+dato+"</li>");
021             }
022         }
023 }
```