Programación del Navegador

Andrés Pastorini

TRIA – Tecnólogo Informático

Programación del navegador

Objetivo

Seguir profundizando en los conceptos y las tecnologías involucradas en el desarrollo de aplicaciones web.

Programación del navegador

Repaso

¿Qué es una RIA?

Es una aplicación que corre en un navegador web(Chrome, ie, firefox, safari,etc) y proporciona una mejor experiencia al usuario que la utiliza, se asemeja más a una aplicación de escritorio, que a una aplicación web tradicional.

DOM

DOM

¿Qué es el DOM?

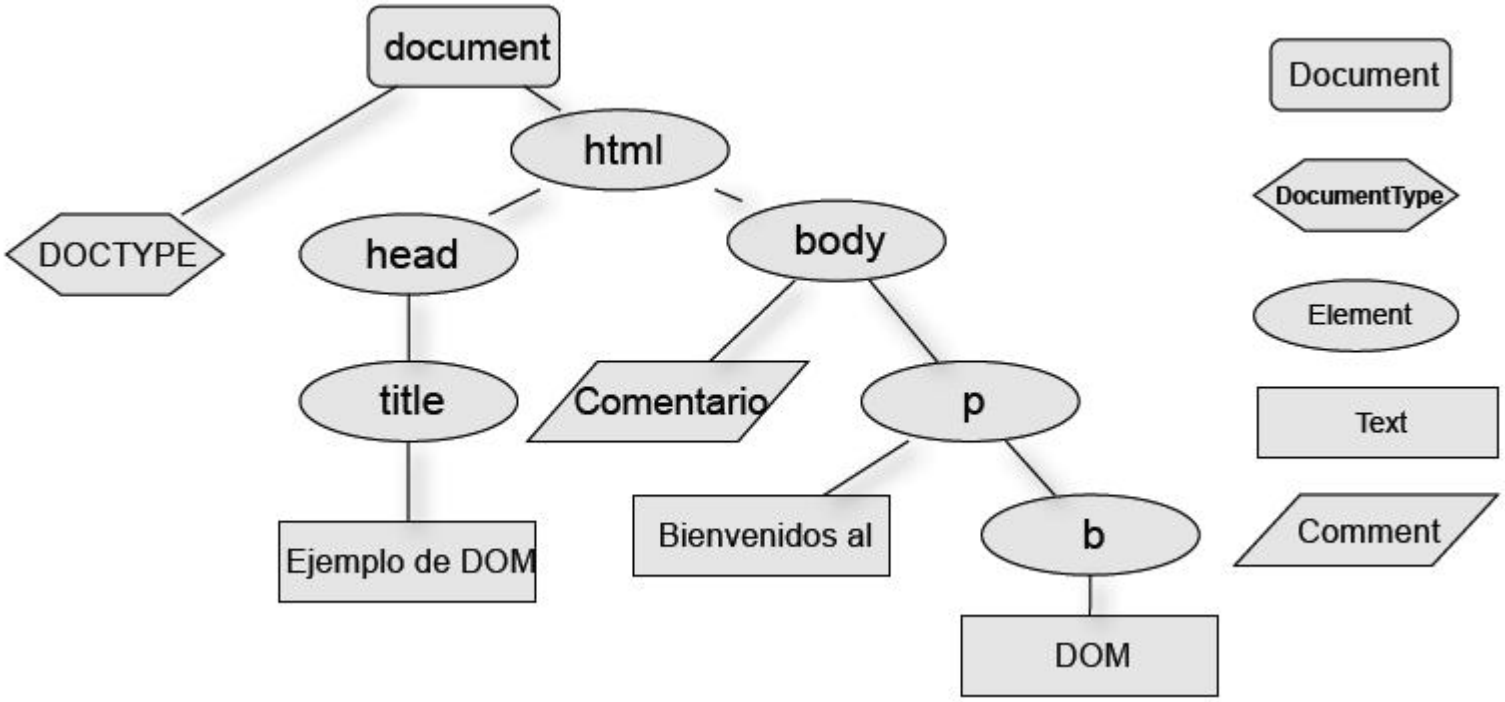
DOM es una interfaz para acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML

API orientado a objetos que permite interactuar con el documento HTML

- Cambiar/leer contenido y estructura
- Cambiar/leer estilos CSS
- Gestionar eventos con *listeners de una forma mucho más sofisticada que* con *handlers*

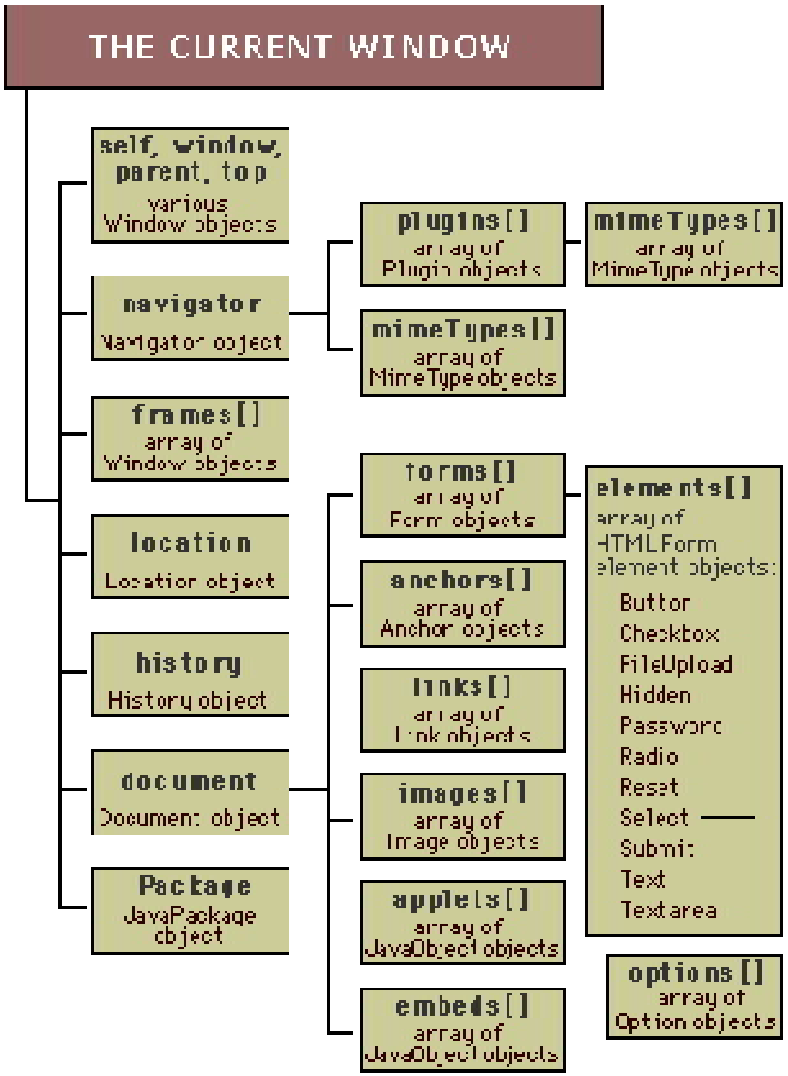
Programación del navegador

DOM



Programación del navegador

DOM



Programación del navegador

DOM

Para manipular el documento se manipulan los nodos.

Todos los nodos son del “tipo” Node, pero hay distintos “subtipos”:

Document, DocumentType, Element, Text, Comment, …

Aunque los atributos de las etiquetas son nodos de tipo Attr, no están en el árbol, hay que acceder a ellos con métodos del nodo que los posee.

DOM (Acceso a la información)

Una vez obtenida la referencia a un nodo (ej. con document.getElementById()) se pueden obtener sus propiedades.

Algunas props. de Node son:

- nodeType: cte. entera que representa el tipo de nodo
- nodeName: nombre, nodeValue: valor.

DOM (Acceso a la información)

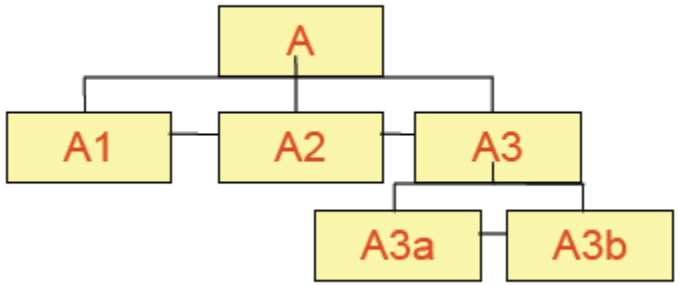
Tipo de nodo	nodeType	nodeName	nodeValue
Etiqueta	1 (Node.ELEMENT_NODE)	Nombre de la etiqueta sin los “<>” y en máyúsc.	null
Texto	3 (Node.TEXT_NODE)	#text	Texto del nodo
Comentario	8 (Node.COMMENT_NODE)	#comment	Texto del comentario
DOCTYPE	10(Node.DOCUMENT_TYPE_NODE)	Nombre de la etiq. raíz del DOCTYPE	null
Documento	9 (Node.DOCUMENT_NODE)	#document	null

DOM (Acceso a la información)

Cada nodo tiene una serie de propiedades que reflejan el "parentesco" con otros, algunas de las cuales son

- childNodes: array con los nodos hijos
- firstChild: primer nodo hijo, lastChild: último nodo hijo
- parentNode: nodo padre
- nextSibling: siguiente hermano (nodo al mismo nivel)
- prevSibling:hermano anterior.

DOM (Acceso a la información)



A.firstChild = A1
A.lastChild =A3
A.childNodes.length = 3
A.childNodes[0] = A1
A.childNodes[1] = A2
A.lastChild.firstChild = A3a
A3b.parentNode.parentNode = A

A1.nextSibling = A2
A3.prevSibling = A2
A3.nextSibling = null

En el estándar se interpretan los espacios en blanco entre etiquetas como nodos de texto.

DOM (Modificar la información)

Cambiar el valor: cambiar la propiedad nodeValue

Cambiar un atributo: setAttribute(nombre,nuevoValor)
Otras muchas propiedades son **solo de lectura (**nodeName**,** firstChild, parentNode,…)

Para crear un nodo

document.createElement(nombre): crea nodo etiqueta. Se le pasa el nombre de la etiqueta sin <>.
document.createTextNode(texto): crea nodo de texto, con el contenido especificado

DOM (Modificar la información)

Insertar nodos

appendChild(nuevoHijo): Añade el hijo al final de todos los hijos actuales.

insertBefore(nuevoHijo, hijoReferencia). Inserta el nuevo hijo justo antes del "hijo de referencia"

setAttribute(nuevoAtributo, nuevoValor). Si el atributo no existía, lo crea. Como ya se ha visto, si existía cambia su valor

Eliminar nodos

removeChild(hijoABorrar): un nodo deja de ser hijo

replaceChild(nuevoHijo, hijoAntiguo): reemplaza un hijo por otro nuevo

DOM (Especifico para html)

rows: propiedad de un nodo tabla que contiene todas sus filas
cells: propiedad de un nodo fila que contiene todas sus celdas
Insertar y borrar filas: los llama un nodo tabla
insertRow(pos): insertar nueva fila vacía (nodo tr) en la posición pos.
Comienzan por 0.
deleteRow(pos): borrar la fila nº pos
Insertar y borrar celdas: los llama un nodo fila
insertCell(pos): insertar nueva celda vacía (nodo td) en la posición pos.
Comienzan por 0
deleteCell(pos): borrar la celda nº pos.

DOM (Especifico para html)

Array predefinido **document.forms**

Por posición
Cada formulario tiene un array “elements” con los campos

Cada campo tiene un atributo “value” (campos de texto, textarea,...) o
bien uno booleano “checked” en casillas de verificación o botones de radio

Por nombre (name).
Automáticamente se define una propiedad con ese name. A los formularios también se les puede poner name

XML

Xml

Es un lenguaje de marcado, como HTML.

Un **lenguaje de marcado** o **lenguaje de marcas** es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación.

Xml

Es un meta lenguaje que permite definir lenguajes de marcados adecuados a usos determinados. Es un estandar internacional reconocido.

Cada documento XML posee una estructura lógica y una física . La estructura lógica del documento es una serie de declaraciones, elementos, comentarios, etc. que se indican en el documento mediante marcas explícitas. La estructura física del documento es una serie de unidades llamadas entidades, es decir, indica los datos que contendrá el documento.

Xml

Todos los documentos XML deben estar **bien formados**, lo que significa que se debe cumplir lo siguiente:

•si no se utiliza DTD, el documento debe comenzar con un declaración de Documento *Standalone*
•todas las etiquetas deben estar balanceadas
•todos los valores de los atributos deben ir entrecomillados
•no debe haber etiquetas aisladas en el texto
•los elementos deben anidar dentro de sí sus propiedades

Programación del navegador

Xml

```
<?xml version="1.0" encoding="UTF-7" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  "DTD/xhtml1-strict.dtd">
<html>
  <body bgcolor="yellow ">
    <hr/>
    <p>
      Entre lineas
    </p>
    <hr/>
  </body>
</html>
```

Xml

Los elementos XML pueden tener contenido (más elementos, caracteres o ambos a la vez), o bien ser elementos vacíos. Un elemento con contenido es, por ejemplo: <nombre>pepe</nombre>

Hay que tener en cuenta que el símbolo "<" siempre se interpreta como inicio de una etiqueta XML. Si no es el caso, el documento no estará bien-formado. Para usar ciertos símbolos se usarán las entidades predefinidas.

Xml

Permite elementos sin contenido, pero la etiqueta debe ser de la siguiente forma: <elemento-sin-contenido/>, que puede contener atributos o no, esto es debido que no hay una etiqueta de cierre que delimite el contenido .

<identificador CI="23123244"/>

Xml

Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento.

<alumno nota=“7"
asistencia="Nula">Gonzales</alumno>

Programación del navegador

Xml

En XML 1.0 se definen cinco entidades para representar caracteres especiales y que no se interpreten como marcas por el parser XML.

```
<?xml version="1.0" encoding="UTF-7" standalone='yes'?>
<ejemplos>
<descripcion>Lo siguiente es un ejemplo de HTML.</descripcion>
<ejemplo>
     &lt;HTML&gt;
       &lt;HEAD&gt;
        &lt;TITLE&gt;Rock &amp; Roll&lt;/TITLE&gt;
       &lt;/HEAD&gt;
</ejemplo>
</ejemplos>
```

ENTIDAD	CARACTER
&amp;	&
&lt;	<
&gt;	>
&apos;	'
&quot;	"

Xml

Las secciones CDATA permiten especificar datos, utilizando cualquier carácter, especial o no, sin que se interprete como una marca XML.

Las secciones CDATA empiezan por la cadena "<![CDATA[" y terminan con la cadena "]]>" y sólo ésta última se reconoce como marca. No se pueden anidar secciones CDATA

```
<?xml version="1.0" encoding="UTF-7" standalone='yes'?>
<ejemplos>
<descripcion>Lo siguiente es un ejemplo de HTML.</descripcion>
<ejemplo>
   <![CDATA[
     <HTML>
       <HEAD>
        <TITLE>Rock & Roll</TITLE>
       </HEAD>
   ]]>
</ejemplo>
</ejemplos>
```

Java Script

Programación del navegador

Java Script

Lenguaje de programación de lado cliente:

- Está integrado en cada navegador.

- Permite modificar el código generado por HTML, obteniendo un comportamiento específico a las necesidades de una página.

- Es un lenguaje “orientado a objetos”.

- Tiene una sintaxis similar a C.

Programación del navegador

Java Script

Incluir código JavaScript entre dos etiquetas <script></script> en el mismo
documento HTML.
<script> alert(“Hola Mundo”);</script>

• Incluir un link a un archivo .js que es el que contiene el código JavaScript.
<script src="js/file.js"></script>

• Todo el código (variables, atributos) dentro de “file.js” estará disponible desde el documento HTML.

Programación del navegador

Java Script

Script: cada uno de los programas, aplicaciones o trozos de código creados con el lenguaje de programación JavaScript.

Sentencia: cada una de las instrucciones que forman un script.

Palabras reservadas: son las palabras que se utilizan para construir las sentencias de JavaScript.
Las palabras reservadas son:
break, case, catch, continue, default, delete, do, else, finally , for,function, if, in, instanceof, new, return, switch, this, thro w, try, typeof, var, void,while, with.

Programación del navegador

Java Script

Sintaxis

•No se tienen en cuenta los espacios en blanco y las nuevas líneas.

•Se distinguen las mayúsculas y minúsculas

•No se define el tipo de las variables

•No es necesario terminar cada sentencia con el carácter de punto y coma (;)

•Se pueden incluir comentarios

Programación del navegador

Java Script

Sintaxis(Variables)

var num=…;

Sólo puede estar formado por letras, números y los símbolos $ y _

El primer carácter no puede ser un número.

Pueden ser de tipo cadena de texto, numéricas, arreglos, booleanos u objetos.

Programación del navegador

Java Script

Sintaxis(Operadores)

Matemáticos: +,-,/,*

Lógicos: !, &&, ||

Relacionales: <,>,=>,<=,!=

Programación del navegador

Java Script

Sintaxis (Estructuras de control)

if(condicion){}

if(condicion){}else{}

for(var i=valIni; i<valFin; i++) {}

for(indice in array) { ... }

while(condicion){}

do{}while(condicion);

switch(variable) { case valor_1: ... break; case…default: ... break; }

Programación del navegador

Java Script

Sintaxis(Funciones)

```
function nombre_funcion(parametros) { ... }
```

No es obligatorio que la función retorne un valor.

```
function nombre_funcion(parametros) { ... return valor;}
```

Java Script

Sintaxis (Eventos)

Cada elemento o etiqueta XHTML define su propia lista de posibles eventos que se le pueden asignar.
Un mismo tipo de evento puede estar definido para varios elementos XHTML diferentes y un mismo elemento XHTML puede tener asociados varios eventos diferentes.

El nombre de cada evento se construye mediante el prefijo on, seguido del nombre en inglés de la acción asociada al evento.

Programación del navegador

Java Script

Evento	Descripción	Elementos para los que está definido
onblur	Deseleccionar el elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onchange	Deseleccionar un elemento que se ha modificado	<input>, <select>, <textarea>
onclick	Pinchar y soltar el ratón	Todos los elementos
ondblclick	Pinchar dos veces seguidas con el ratón	Todos los elementos
onfocus	Seleccionar un elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onkeydown	Pulsar una tecla (sin soltar)	Elementos de formulario y <body>
onkeypress	Pulsar una tecla	Elementos de formulario y <body>
onkeyup	Soltar una tecla pulsada	Elementos de formulario y <body>
onload	La página se ha cargado completamente	<body>
onmousedown	Pulsar (sin soltar) un botón del ratón	Todos los elementos

Programación del navegador

Java Script

onmouseup	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
onreset	Inicializar el formulario (borrar todos sus datos)	<form>
onresize	Se ha modificado el tamaño de la ventana del navegador	<body>
onselect	Seleccionar un texto	<input>, <textarea>
onsubmit	Enviar el formulario	<form>
onunload	Se abandona la página (por ejemplo al cerrar el navegador)	<body>

Programación del navegador

Java Script

Sintaxis (Eventos)

Manejadores de eventos

Se deben asociar funciones o código JavaScript a cada evento. De esta forma, cuando se produce un evento se ejecuta el código indicado, por lo que la aplicación puede *responder* ante cualquier evento que se produzca durante su ejecución. Las funciones o código JavaScript que se definen para cada evento se denominan *"manejador de eventos“* existen varias formas diferentes de indicar los manejadores:

- Manejadores como atributos de los elementos XHTML.
- Manejadores como funciones JavaScript externas.
- Manejadores *"semánticos"*.

Java Script

Sintaxis (Eventos)

Manejadores de eventos como atributos XHTML

Se trata del método más sencillo y a la vez *menos profesional* de indicar el código JavaScript que se debe ejecutar cuando se produzca un evento. En este caso, el código se incluye en un atributo del propio elemento XHTML. En el siguiente ejemplo, se quiere mostrar un mensaje cuando el usuario pinche con el ratón sobre un botón:

```
<input type="button" value="Pinchame y verás"
onclick="alert('Gracias por pinchar');" />
```

Java Script

Sintaxis (Eventos)

Manejadores de eventos y variable this

JavaScript define una variable especial llamada this que se crea automáticamente. En los eventos, se puede utilizar la variable this para referirse al elemento XHTML que ha provocado el evento.

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

Programación del navegador

Java Script

Sintaxis (Eventos)

Manejadores de eventos como funciones externas

```
function muestraMensaje() { alert('Gracias por pinchar'); }
 <input type="button" value="Pinchame y verás"
onclick="muestraMensaje()" />
```

Programación del navegador

Java Script

Sintaxis (Eventos)

Manejadores de eventos semánticos

Una de las buenas prácticas básicas en el diseño de aplicaciones web es la separación del contenido (XHTML) y su presentación (CSS).

Java Script

Sintaxis (Eventos)

Manejadores de eventos semánticos

La técnica de los manejadores semánticos consiste en:

1)asignar un identificador único al elemento XHTML mediante el atributo id.

2)Crear una función de JavaScript encargada de manejar el evento.

3)Asignar la función externa al evento correspondiente en el elemento deseado.

```
var componente=document.getElementById("pinchable");
componente.onclick = funcionAsignada(…)
```

Java Script

Sintaxis (Eventos)

Manejadores de eventos semánticos

El único inconveniente de este método es que la página se debe cargar completamente antes de que se puedan utilizar las funciones DOM que asignan los manejadores a los elementos XHTML.

Para asegurar que cierto código se va a ejecutar después de que la página se cargue por completo es utilizar el evento onload:

```
window.onload = function() {
  document.getElementById("componenteId").onclick =
  muestraMensaje; }
```

La técnica anterior utiliza el concepto de *funciones anónimas.*

Java Script

Información del evento

En los navegadores tipo Internet Explorer, el objeto event se obtiene directamente mediante: var evento = window.event;

En el resto de navegadores, el objeto event se obtiene a partir del argumento que el navegador crea automáticamente:
function manejadorEventos(elEvento) { var evento = elEvento; }

forma correcta de obtener el objeto event en cualquier navegador:
function manejadorEventos(elEvento) { var evento = elEvento || window.event; }

En el objeto event se encuentra la información del evento

Java Script

Manejo de errores

```
try {
  throw “excepcion"; //lanza una excepción
  }
  catch (e) {
  //tratamos la excepción
  alert(e);
  }
```

Programación del navegador

Java Script

Orientación a objetos

JavaScript, no esta basado en clases, sino en prototipos

Un prototipo es un objeto abstracto, capaz de contener otros objetos dentro, los cuales pueden ser distintos tipos de datos: variables (numeros, cadenas de texto, valores lógicos), vectores, funciones e inclusive otros grupos de objetos.

Las variables dentro de este serán las propiedades, y las funciones serán los métodos:

[Objeto = Prototípo**]{** **[** Propiedad = Variable **]** **[** Método = Funcion **] }**

Programación del navegador

Java Script

Orientación a objetos

En JavaScript las funciones son objetos, para crear una clase e instanciarla:

```
var Gato = function (nombre) {
    this.nombre = nombre;
    //Creamos una variable privada sin asignarle "this" a esta
    var frecuencia = "moderada";
    // Creamos un metodo privado en muchos aspectos (XD) definiendo una
funcion normal
    function irAlBano(frecuencia) {
        alert("El gato va al baño con frecuencia "+frecuencia);
    }
    this.leerFrec = function() {
        irAlBano(frecuencia);
    }
}
var Michin = new Gato("Michifu");
Michin.leerFrec();
//Nos retorna "El gato va al baño con frecuencia moderada" ;)
```

Programación del navegador

Java Script

Orientación a objetos

Al ser las funciones objetos podemos tener variables que sean funciones

```
this.comer = comerExterna;

[...]

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

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

```
Michi.comer();
```

Programación del navegador

Java Script

Orientación a objetos

Para extender una clase:

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

Programación del navegador

Java Script

JSON

JSON es un formato de datos muy ligero basado en un subconjunto de la sintaxis de JavaScript: literales de matrices y objetos. Como usa la sintaxis JavaScript, las definiciones JSON pueden incluirse dentro de archivos JavaScript y acceder a ellas sin ningún análisis adicional como los necesarios con lenguajes basados en XML.

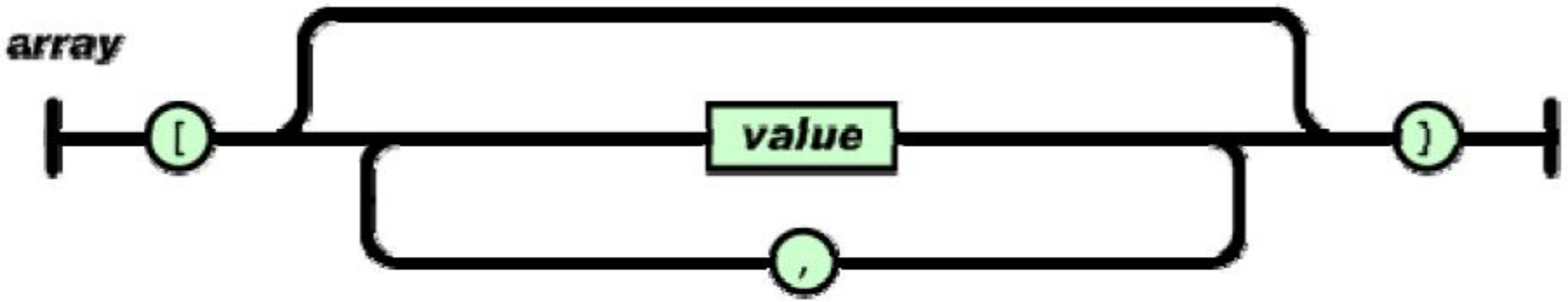
Programación del navegador

Java Script

JSON

LITERALES DE MATRIZ

Estos elementos se especifican utilizando corchetes ([y]) para encerrar listas de valores delimitados por comas de JavaScript



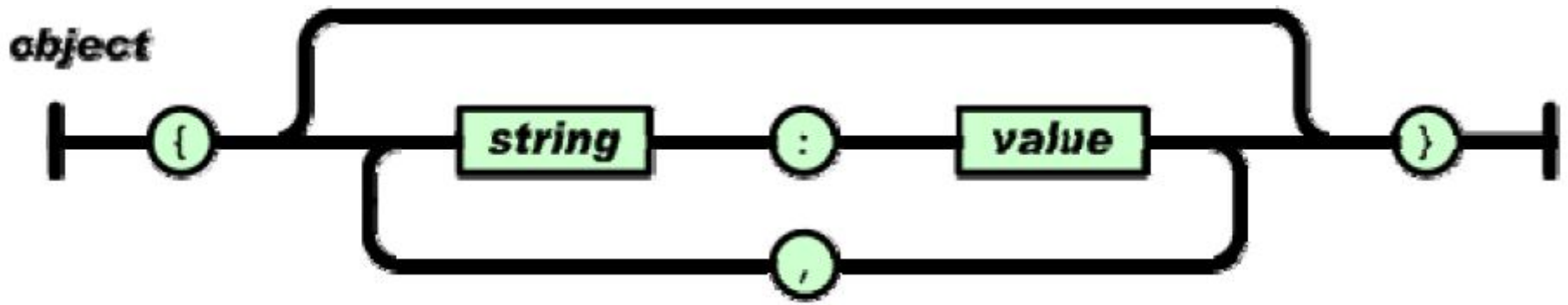
Programación del navegador

Java Script

JSON

LITERALES DE OBJETO

Los literales de objeto se utilizan para almacenar información en parejas nombre-valor para crear un objeto, Un literal de objeto se define mediante llaves ({ y }) Dentro de estas, se pueden colocar cualquier número de parejas nombre-valor, definida mediante una cadena, un símbolo de dos puntos y el valor. Cada pareja nombre-valor deben estar separadas por coma.



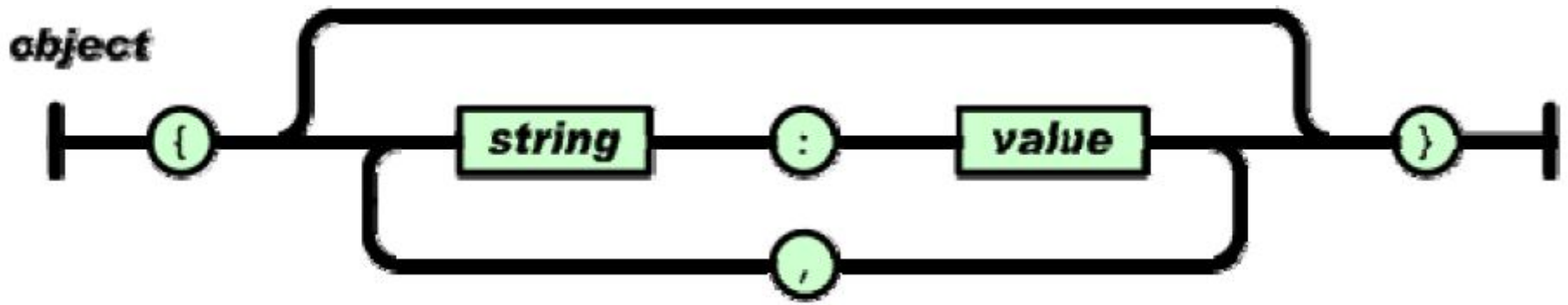
Programación del navegador

Java Script

JSON

LITERALES DE OBJETO

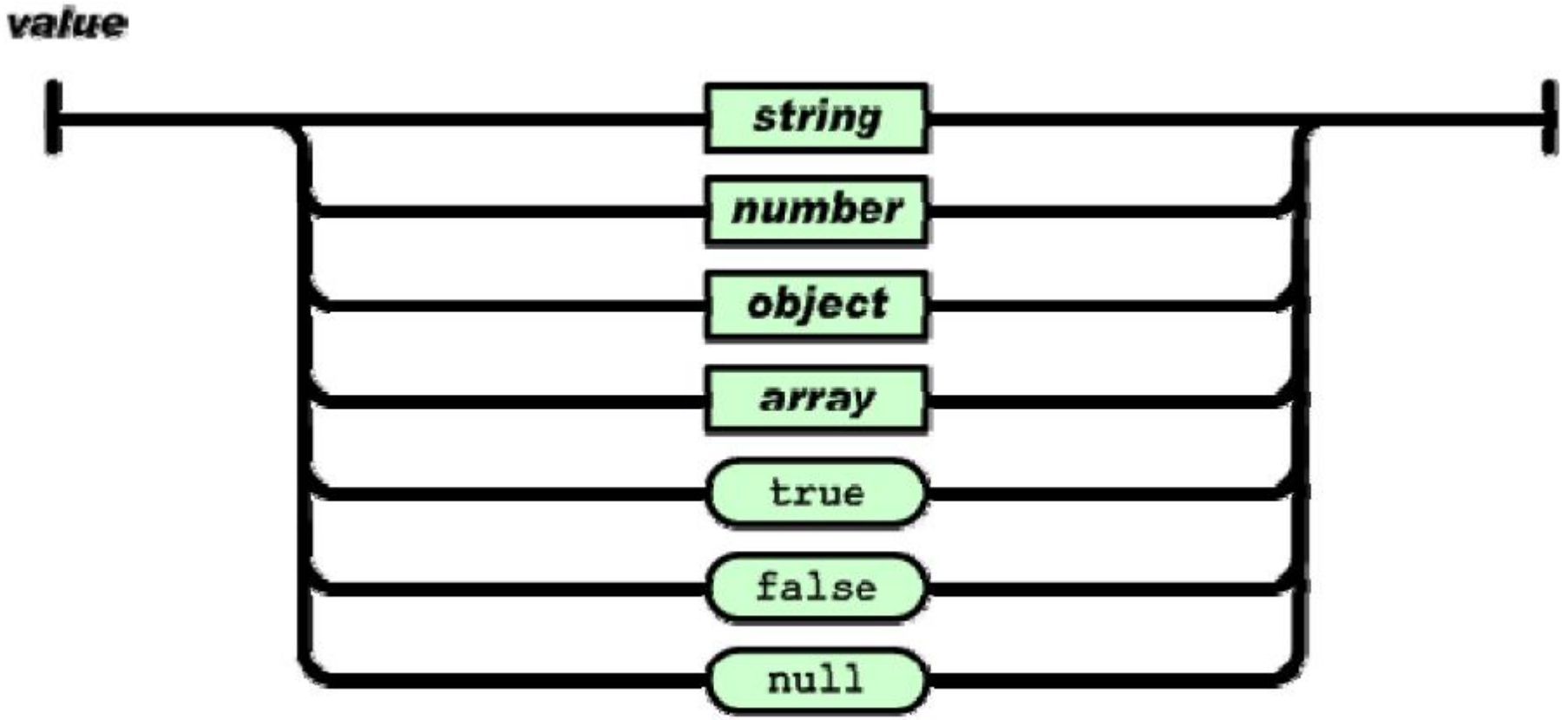
Los literales de objeto se utilizan para almacenar información en parejas nombre-valor para crear un objeto, Un literal de objeto se define mediante llaves ({ y }) Dentro de estas, se pueden colocar cualquier número de parejas nombre-valor, definida mediante una cadena, un símbolo de dos puntos y el valor. Cada pareja nombre-valor deben estar separadas por coma.



Programación del navegador

Java Script

JSON



Programación del navegador

Java Script

JSON

Codificar y decodificar JSON

Para decodificar se puede utilizar la función eval () pero esta evalúa cualquier código JavaScript que se pasa a la función y esto es un gran riesgo de seguridad. Existen librerías que se ocupan de esto Por ejemplo MOOTOOLS .

Programación del navegador

Ajax

(Asynchronous Java Script and xml)

Programación del navegador

Ajax

- Permite actualizar partes de la página sin la necesidad de cargar toda devuelta.
- El contenido se carga en forma dinámica
- Uso frecuente
  - Web-Chat
  - Validación de formularios
  - Sugerencias
  - Select multiples

Programación del navegador

Ajax

Objeto Clave:

XMLHttpRequest (compatible con todos los navegadores, menos versiones antiguas de IE)

•Tipos de Peticiones

- GET,POST
- Sincrónicas, Asincrónicas

XML es el formato nativo de ajax, pero puede procesar también texto plano

Programación del navegador

Ajax(Estructura básica)

```
<html>
   <head>
      <script type="text/javascript">
      function ejecutarajax(){
         var conexion;
         if (window.XMLHttpRequest)          {
           conexion=new XMLHttpRequest();
         }else
           {
           conexion=new ActiveXObject("Microsoft.XMLHTTP");
         }
         conexion.onreadystatechange=function(){
            if(conexion.readyState==4 && conexion.status==200){
               document.getElementById("midiv").innerHTML=conexion.responseText;
            }
         }
         conexion.open("GET","ejemplo.txt",true);
         conexion.send();
      }
      </script>
   </head>
   <body>
      <div id="midiv"></div>
      <button type="button" onclick="ejecutarajax()">Ejecutar</button>
   </body>
</html>
```

```
[0].nodeValue +"</td><td>"
```

Programación del navegador

Ajax

ReadyState:

0: la petición no se ha inicializado
1: conexión con el servidor establecida
2: petición recibida
3: procesando petición
4: petición finalizada y la respuesta esta lista

Status:

200: OK
404: Pagina no encontrada

Programación del navegador

Ajax(Procesando XML)

```
var txt,x,i;
if (window.XMLHttpRequest)
  {
  conexion=new XMLHttpRequest();
  }
else
  {
  conexion=new ActiveXObject("Microsoft.XMLHTTP");
  }
conexion.onreadystatechange=function()
  {
  if (conexion.readyState==4 && conexion.status==200)
    {
    xmlDoc=conexion.responseXML;
    txt="<table>";

    x=xmlDoc.getElementsByTagName("titulo");
    artista=xmlDoc.getElementsByTagName("artista");
    pais=xmlDoc.getElementsByTagName("pais");
    firma=xmlDoc.getElementsByTagName("firma");
    precio=xmlDoc.getElementsByTagName("precio");
    anio=xmlDoc.getElementsByTagName("anio");

    for (i=0;i<x.length;i++)
      {
      txt=txt + "<tr><td>" + x[i].childNodes[0].nodeValue +"</td><td>"+ artista[i].childNodes[0].nodeValue +"</td><td>
      }
    document.getElementById("myDiv").innerHTML=txt;
    }
  }
  conexion.open("GET","discoteca.xml",true);
  conexion.send();
}
```

JQuery

Programación del navegador

JQuery

Librería de Java Script

Se descarga desde http://jquery.com/download/

Para usar JQuery incluir en el cabezal:

```
<head>
        <script type="text/javascript" src="jquery.js"></script>
</head>
```

JQuery

Selectores

Por Id: $(“#ID”)
//Selecciona todos los elementos con id=ID
Por clase css: $(“.Clase”)
//Selecciona todos los elementos con class=Clase
Por etiqueta Html: ej $(“h1”)
//Selecciona todos los elementos cuya etiqueta sea h1
Por atributo: $(“[nomAtt]”)
//Selecciona todos los elementos que tengan un atributo
//nomAtt

JQuery

Selectores

Por atributo: $(“[nomAtt!=‘#’]”)
//Selecciona todos los elementos que tengan un atributo
//nomAtt y que su valor no contenga #

Por atributo: $(“[nomAtt$=‘.png’]”)
//Selecciona todos los elementos que tengan un atributo
//nomAtt y que su valor termine en .png

Programación del navegador

JQuery

Si x es una variable JS de tipo texto
x.text() devuelve el contenido
x.text("…") lo sobreescribe

Si x es una variable que contiene un tag html
x.attr("atributo") obtiene el valor del atributo

Si x es una variable que contiene un tag html
x.attr("atributo","valor") asigna el valor al atributo

Si x es una variable que contiene un tag html
x.removeAttr("atributo") elimina el atributo

Programación del navegador

JQuery

Si x es una variable que contiene elementos html
x.addClass(“clase”) agrega la clase css a los elementos en x.

Si x es una variable que contiene elementos html
x.removeClass(“clase”) desasigna la clase css a los elementos en x.

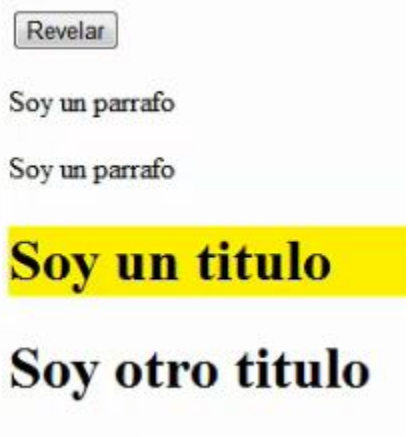
x.html(“cadena html”) incrustanuevo contenido html en x

Programación del navegador

JQuery (ejemplos)

```
<html>
   <head>
      <script type="text/javascript" src="../jquery.js"></script>
      <script type="text/javascript">
         var x;
         x=$(document);
         x.ready(inicio);

         function inicio(){
            var x = $("#revelar");
            x.click(mostrar);
         }
         function mostrar(){
            var x=$("h1#cambiame");
            x.css("background","yellow");
         }
      </script>
   </head>
   <body>
   <input type="button" id="revelar" value="Revelar"><br>
   <p>Soy un parrafo</p>
   <p id="cambiame">Soy un parrafo</p>
   <h1 id="cambiame">Soy un titulo</h1>
   <h1>Soy otro titulo</h1>
   </body>
</html>
```

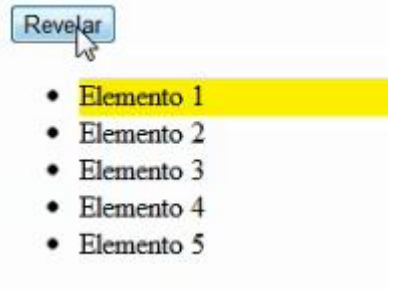


Programación del navegador

JQuery (ejemplos)

```
<html>
   <head>
      <script type="text/javascript" src="../jquery.js"></script>
      <script type="text/javascript">
         var x;
         x=$(document);
         x.ready(inicio);

         function inicio(){
            var x = $("#revelar");
            x.click(mostrar);
         }
         function mostrar(){
            var x=$("li:first");
            x.css("background","yellow");
         }
      </script>
   </head>
   <body>
   <input type="button" id="revelar" value="Revelar"><br>
   <ul>
      <li>Elemento 1</li>
      <li>Elemento 2</li>
      <li>Elemento 3</li>
      <li>Elemento 4</li>
      <li>Elemento 5</li>
   </ul>
   </body>
</html>
```



Programación del navegador

JQuery (métodos)

```
function escribir(){
   var x=$("p");
   x.prepend("XXXXXXXXXXXXXXXXXXX");
}
```

XXXXXXXXXXXXXXXXXXXEste es el el texto antiguo

```
function escribir(){
   var x=$("p");
   x.append("XXXXXXXXXXXXXXXXXXX");
}
```

Este es el el texto antiguoXXXXXXXXXXXXXXXXXXX

```
function escribir(){
   var x=$("p");
   x.after("XXXXXXXXXXXXXXXXXXX");
}
```

Este es el el texto antiguo

XXXXXXXXXXXXXXXXXXX

```
function escribir(){
   var x=$("p");
   x.before("XXXXXXXXXXXXXXXXXXX");
}
```

XXXXXXXXXXXXXXXXXXX

Este es el el texto antiguo

Programación del navegador

JQuery (métodos)

```
var x;
x=$(document);
x.ready(inicializar);

function inicializar()
{
var x;
x=$("#hola");
x.click(clickHecho);
}

function clickHecho()
{
var x;
x=$("#hola");
x.css("color","green");
}
```

Programación del navegador

JQuery (ejemplo)

```
<html>
   <head>
      <script type="text/javascript" src="../
jquery.js"></script>
      <script type="text/javascript" >
         var x;
         x=$(document);
         x.ready(inicializar);

         function inicializar(){
            var x;
            x=$("h1");
            x.click(presionar);
         }
         function presionar(){
            var x;
            x=$(this);
            x.css("font-size","12px");
         }
      </script>
   </head>
   <body>
      <h1>Yo soy un título</h1>
      <p>Yo soy un parágrafo</p>
      <h1>Yo soy un título</h1>
      <p>Yo soy un parágrafo</p>
      <h1>Yo soy un título</h1>
      <p>Yo soy un parágrafo</p>
      <h1>Yo soy un título</h1>
      <p>Yo soy un parágrafo</p>
   </body>
</html>
```

JQuery

Efectos

show(“fast”|”slow”), hace aparecer el elemento al cual se le aplica este método

hide(“fast”|”slow”) hace desaparecer el elemento al cual se le aplica este método, ejecuta una animación de dimensionamiento y transparencia

fadeOut(“fast”|”slow”) // efecto de ocultamiento con transparencia

fadeIn(“fast”|”slow”) // efecto de aparición con transparencia

JQuery

Efectos

fadeTo(“fast”|”slow”,0.0-1.0), se puede controlar el nivel preciso de transparencia de la animación. Cuanto más cerca a uno se encuentre el segundo parámetro más opaco se verá el objeto.

toggle(“fast”|”slow”), equivalente a show y hide, pero se da cuenta de forma automática cuando ejecutar cada uno de ellos.

JQuery

Efectos

slideUp((“fast”|”slow”), el objeto seleccionado se oculta con una animación de barrido hacia arriba

slideDown((“fast”|”slow”), , el objeto seleccionado aparece con una animación de barrido hacia abajo

slideToggle((“fast”|”slow”), equivalente a los anteriores, se da cuenta de forma automática cual ejecutar.

JQuery

Efectos

slideUp((“fast”|”slow”), , el objeto seleccionado se oculta con una animación de barrido hacia arriba

slideDown((“fast”|”slow”), , el objeto seleccionado aparece con una animación de barrido hacia abajo

slideToggle((“fast”|”slow”), equivalente a los anteriores, se da cuenta de forma automática cual ejecutar.

Animate, construcciones personalizadas de animación:

```
x.animate({height:300},"slow");
x.animate({width:300},3000);
x.animate({height:100},"normal");
x.animate({width:100},1000);
```

Programación del navegador

Ajax y JQuery

```
<html>
   <head>
      <script type="text/javascript" src="../jquery.js"></scrip
      <script type="text/javascript">
         var x;
         x=$(document);
         x.ready(inicio);

         function inicio(){
            var x;
            x=$("a");
            x.click(muestrame);
         }
         function muestrame(){
            var pagina=$(this).attr("href");
            var x=$("#hablame");
            x.load(pagina);
            return false;
         }
      </script>
      <style type="text/css">

      </style>
   </head>
   <body>
      <a href="colores.php?color=verde">Hablame del verde</a>
      <a href="colores.php?color=rojo">Hablame del rojo</a>
      <a href="colores.php?color=azul">Hablame del azul</a>
      <div id="hablame"></div>
   </body>
</html>
```

Programación del navegador

Ajax y JQuery
(Post)

```
<html>
    <head>
        <script type="text/javascript" src="../jquery.js"></script>
        <script type="text/javascript">
            var x;
            x=$(document);
            x.ready(inicio);

            function inicio(){
                var x;
                x=$("#enviar");
                x.click(calcular);
            }
            function calcular(){
                var v=$("#numero").attr("value");
                $.post("calculadora.php",{numero:v},recibir);
                return false;
            }
            function recibir(datos){
                alert(datos);
            }
        </script>
        <style type="text/css">

        </style>
    </head>
    <body>
        Multiplicar por 5 el siguiente numero

        <form action="calculadora.php" method="post">

        <input type="text" id="numero" name="numero">

        <input type="submit" value="Enviar" id="enviar">
                </form>
                <div id="resultado"></id>
    </body>
</html>
```

Servidor PHP

```
<?php

$porcinco = $_POST['numero']*5;
echo $porcinco;

?>
```

Programación del navegador

Ajax y Jquery

(Get)

```
<html>
   <head>
      <script type="text/javascript" src="../jquery.js"></script>
      <script type="text/javascript">
         var x;
         x=$(document);
         x.ready(inicio);

         function inicio(){
            var x;
            x=$("#enviar");
            x.click(calcular);
         }
         function calcular(){
            var v=$("#numero").attr("value");
            $.get("calculadora.php",{numero:v},recibir);
            return false;
         }
         function recibir(datos){
            alert(datos);
         }
      </script>
      <style type="text/css">

      </style>
   </head>
   <body>
      Multiplicar por 5 el siguiente numero

      <form action="calculadora.php" method="get">

      <input type="text" id="numero" name="numero">

      <input type="submit" value="Enviar" id="enviar">
            </form>
            <div id="resultado"></id>
   </body>
</html>
```

Servidor PHP

```
<?php

$porcinco = $_GET['numero']*5;
echo $porcinco;

?>
```

Programación del navegador

Ajax y JQuery(otra forma)

```
function enviar()
   {
     var v=$("#numero").attr("value");
     $.ajax({
              async:true,
              type: "POST",
              dataType: "html",
              contentType: "application/x-www-form-
urlencoded",
              url:"ajaxcompleto.php",
              data:"anio="+v,
              beforeSend:inicioEnvio,
              success:llegada,
              timeout:4000,
              error:problemas
            });
     return false;
   }
```

```
function inicioEnvio()
   {
     var x=$("#resultados");
     x.html('Cargando...');
   }
function llegada(datos)
   {
     $("#resultados").text(datos);
   }
function problemas()
   {
     $("#resultados").text('Problemas en el servidor.');
   }
```

Programación del navegador

JQuery

Eventos

ready(“f1”)//Invoca a la función f1 cuando se haya cargado todos los elementos de la página.

click(“f1”) //Invoca a la función f1 cuando se hace click en el elemento al cual se le asigno el evento click

mouseover(f1)//invoca a la función f1 cuando el mouse pasa por sobre el objeto

mouseout(f2)

hover(f1,f2)// equivalente a mouseover(f1), mouseout(f2)

Programación del navegador

JQuery

Eventos

focus(f1)// se dispara el evento focus sobre determinado elemento cuando el elemento adquiere el foco, y se invoca a f1

Blur(f1)// se dispara el evento focus sobre determinado elemento cuando el elemento pierde el foco, y se invoca a f1

JQuery

Eventos

mousemove(f1) //se dispara cada vez que el ratón se mueve (event.clientX y event.clientY contienen las coordenadas X, y del puntero desde f1)

mousedown

mouseup

dblclick

focus(f1)// se dispara cuando el foco aparece sobre el elemento, se invoca al f1

JQuery UI

Conjunto de plugins empaquetados para el diseño de interfaces de usuarios.

Se debe descargar desde Jqueryui.com

JQuery UI

Contiene una serie de métodos y elementos para diseñar interfaces gráficas.

Métodos:

- Draggable, permite desplazar al objeto seleccionado en la pantalla.
- Droppable, permite detectar cuando un objeto se suelta dentro de otro div.
- Resizable. Permite cambiar las dimensiones del elemento seleccionado.
- Selectable, permite seleccionar un elemento.

JQuery UI

Contiene una serie de métodos y elementos para diseñar interfaces gráficas.

Elementos(widgets):

- Accordion
- Autocomplete
- Calendar
- Dialog
- Progress bar
- Slider
- Tab

Programación del navegador

JQuery UI (Método Draggable)

```
<!DOCTYPE html>
<html>
   <head>
      <link type="text/css" href="..css/ui-lightness/jquery-
ui-1.8.16.custom.css" rel="stylesheet" />
      <script type="text/javascript" src="../jquery.js"></script>
      <script type="text/javascript" src="../jqueryui.js"></script>
      <script type="text/javascript">
         var x;
         x=$(document);
         x.ready(inicio);

         function inicio(){
            var x=$("#muevemueve");
            x.draggable();
         }
      </script>
      </script>
      <style type="text/css">
         #muevemueve{width:100px;height:100px;border:1px solid black;}
      </style>
   </head>
   <body>
      <div id="muevemueve">Mueveme</div>
   </body>
</html>
```

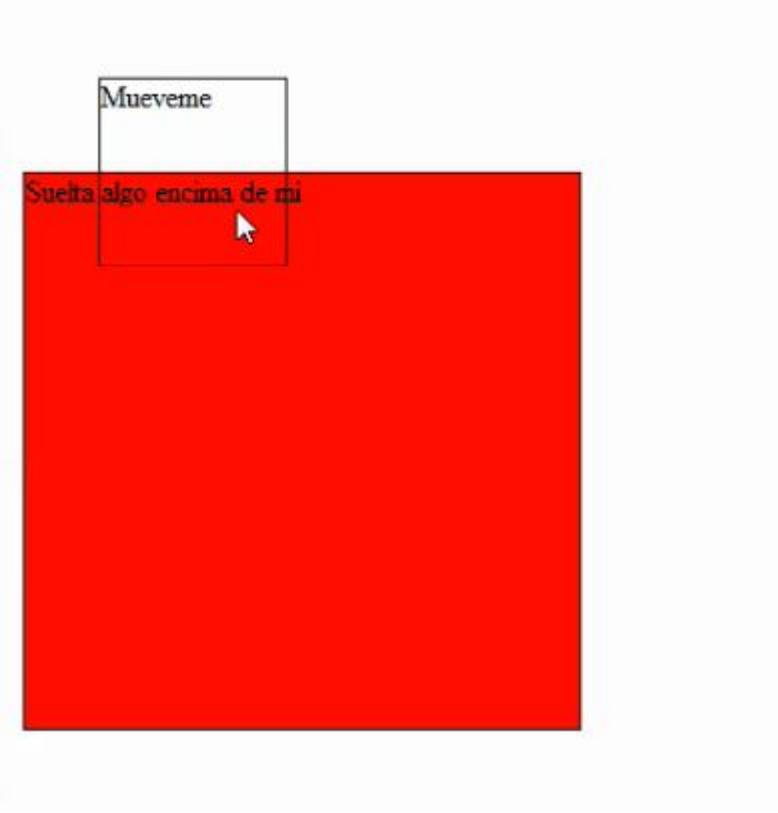
Programación del navegador

JQuery UI (Método Droppable)

```
    <link type="text/css" href="..css/ui-lightness/jquery-
i-1.8.16.custom.css" rel="stylesheet" />
    <script type="text/javascript" src="../jquery.js"></sc
    <script type="text/javascript" src="../jqueryui.js"></
    <script type="text/javascript">
       var x;
       x=$(document);
       x.ready(inicio);

       function inicio(){
          var x=$("#muevemueve");
          x.draggable();
          x=$("#sueltame");
          x.droppable({drop:soltado});
       }
       function soltado(){
          var x=$("#sueltame");
          x.css("background","red");
       }

    </script>
    </script>
```



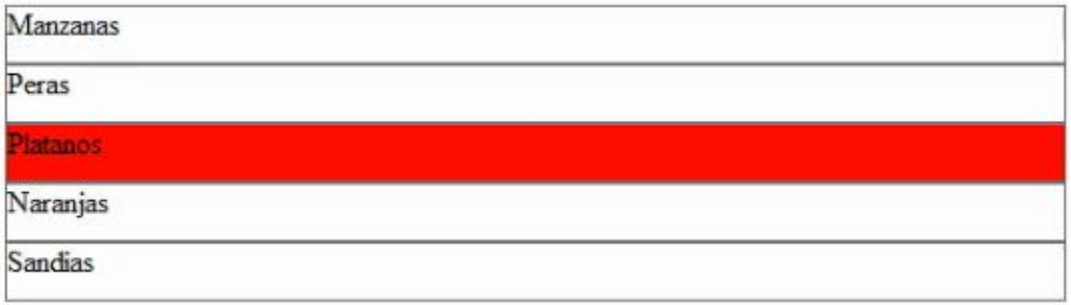
Programación del navegador

Jquery UI (Método Selectable)

```
        var x=$(document);
        x.ready(inicio);
        function inicio(){
            x=$("ul");
            x.selectable();
        }
    </script>

    <style type="text/css">
        ul{list-style-type:none;}
        li{border: 1px solid grey;with:500px;height:30px;}
        ul .ui-selecting{background:yellow;}
        ul .ui-selected{background:red;}
    </style>
</head>
<body>
    <ul id="seleccionable">
        <li>Manzanas</li>
        <li>Peras</li>
        <li>Platanos</li>
        <li>Naranjas</li>
        <li>Sandias</li>

    </ul>
```



JQuery UI (Widget Accordion)


```
$(function() {
$( "#accordion" ).accordion();
});
```

Programación del navegador

JQuery UI (Widget Autocompletar)

```
function inicio(){
   var posibilidades = [
      "Manzana",
      "Limon",
      "Naranja",
      "Platano",
      "Sandia",
      "Melon",
      "Pomelo",
      "Fresa",
      "Melocoton",
   ];
   x=$("#completar");
   x.autocomplete({source: posibilidades})
```

Programación del navegador

JQuery UI (Widget Calendario)

```
<!DOCTYPE html>
<html>
   <head>
      <link type="text/css" href="../css/ui-lightness/jquery-
ui-1.8.16.custom.css" rel="stylesheet" />
      <script type="text/javascript" src="../jquery.js"></script>
      <script type="text/javascript" src="../jqueryui.js"></script>
      <script type="text/javascript">
         var x;
         x=$(document);
         x.ready(inicio);
         function inicio(){
            var x=$("#calendario");
            x.datepicker();
         }
   </script>
      </script>
      </script>
      <style type="text/css">

      </style>
   </head>
   <body>
         <input type="text" id="calendario">
   </body>
</html>
```

Programación del navegador

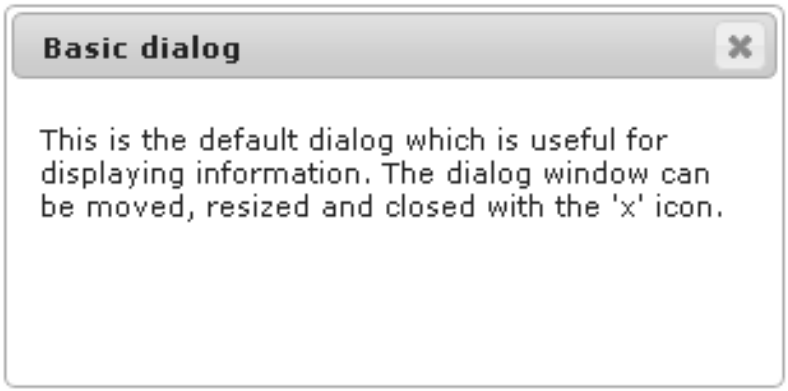
JQuery UI (Widget Dialog)

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>jQuery UI Dialog - Default functionality</title>
  <link rel="stylesheet" href="http://code.jquery.com/ui/1.10.2/themes/s
  <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
  <script src="http://code.jquery.com/ui/1.10.2/jquery-ui.js"></script>
  <link rel="stylesheet" href="/resources/demos/style.css" />
  <script>
  $(function() {
    $( "#dialog" ).dialog();
  });
  </script>
</head>
<body>

<div id="dialog" title="Basic dialog">
  <p>This is the default dialog which is useful
</div>


</body>
</html>
```



Programación del navegador

JQuery UI (Widget Barra de progreso)

```
        var x;
        x=$(document);
        x.ready(inicio);
        function inicio(){
            var x=$("#barradeprogreso");
            x.progressbar({value:50});
        }
</script>
    </script>
    </script>
    <style type="text/css">

    </style>
</head>
<body>
    <div id="barradeprogreso"></div>
```

Programación del navegador

JQuery UI (Widget Deslizador)

```
x=$("#deslizador");
x.slider();
```

```
<div id="deslizador"></div>
```

Programación del navegador

JQuery UI (Widget Pestañas)

```
var x;
x=$(document);
x.ready(inicio);
function inicio(){
    var x=$("#pestanas");
    |
}
```


```
<div id="pestanas">
   <ul>
      <li><a href="#pestana1">Enlace 1</a></li>
      <li><a href="#pestana2">Enlace 2</a></li>
      <li><a href="#pestana3">Enlace 3</a></li>
   </ul>
   <div id="pestana1">
```

Ejercicios

Entrar a http://jsfiddle.net/ y probar la ejecución de un script que muestre un mensaje de alerta y escriba en la consola del navegador (console.log()).

Programación del navegador

Ejercicios

Utilizar http://jsfiddle.net/ para probar los ejercicios de la clase 1.

Ejercicios

Realizar un formulario de alta de usuarios, donde el país y la ciudad se establecen a partir de un select multiple, este select debe ser cargado a través de ajax, intercambiando datos en formato xml.

Validar los datos con javascript accediendo a los mismos utilizando DOM.

Ejercicios

Realizar una interfaz gráfica utilizando Jquery UI.

Ejercicios

Repasar CSS en http://www.w3schools.com/

Programación del navegador

Referencias

http://librosweb.es/javascript
http://flanagan.ugr.es/xml/xml.htm
http://www.cristalab.com/tutoriales/programacion-orientada-a-objetos-oop-con-javascript-c232l/
http://si.ua.es/es/documentacion/mootools/documentos/pdf/json.pdf

Bibliografía

JavaScript: The Definitive Guide, Sixth Edition, *David Flanagan*
Referencia exhaustiva de Javascript, incluyendo DOM
Javascript Cookbook*, Shelley Powers*
Poca teoría, básicamente ejemplos de código
DOM en Caps 11 y 12

Fin