LaboratorioIII 2017

Resumen ppts

**HTLM**

Elementos semánticos:

* <header>, <footer>, <article> y <section>.
* <hr>: Define un cambio temático en el contenido (línea horizontal).

Etiquetas relevantes:

IMÁGENES:

* <img>: define una imagen.
* <picture>: define un contenedor de múltiples imágenes.
* <figure>: define contenidos para imágenes.
* <figcaption>: Define un titulo para un elemento <figure>.
* <picture>: define un contenedor de multiples imágenes.

LINKS:

* <a>
* <link>: Define la relación entre un documento y un recurso externo.
* <nav>: define un enlace de navegación.

AUDIO / VIDEO

* <audio>: define un contenido de auido.
* <video>: define video o película.
* <source>: Define recursos para elementos <video>, <audio> y <picture>.
* <track>: Define pistas de texto para elementos <video> y <audio>.

LISTAS

* <ul>: define una lista desordenada.
* <ol>: define una lista ordenada.
* <li>: define un ítem de una lista.
* <dl>: define una lista de descripción.
* <dd>: define una descripción de un término en una lista <dl>
* <menu>: define un ítem de un comando de lista/menú.
* <menuitem>: define un ítem de comando / menú que el usuario puede invocar desde el menú “popup”.

TABLAS

* <table>: define una tabla.
* <caption>: define el título de la tabla.
* <th>: define una celda cabecera en la fila de la tabla.
* <tr>: define una fila en la tabla.
* <td>: define una celda en la fila de la tabla.
* <thead>: agrupa las celdas **<th>** de la tabla.
* <tbody>: agrupa el contenido del cuerpo en la tabla.
* <tfoot>: agrupa el contenido del pie de la tabla.

ESTILOS y SEMANTICAS

* <styles>: define estilos para un documento.
* <div>: define una sección (contenedora) en un documento.
* <main>: especifica el contenido principal de un documento.
* **<header>: define el encabezado para un documento o sección.**
* <section>: define una sección de un documento.
* <article>: define un articulo.
* <aside>: define contenido “de lado” de una pagina contenedora.
* **<details>**: define detalles que el usuario puede ver u ocultar.
* **<dialog>**: define una ventana o caja de dialogo.

META DATOS

* <meta>: define metadatos para documentos HTML.
* <**head**>: define información acerca del documento.

PROGRAMACION

* <script>: define código del lado del cliente.
* <param>: define un parámetro para un objeto.

FORMULARIOS E INPUTS

* **<form>:** define un formulario HTML para entradas.
* **<input>:** define un control de entrada.
* <textarea>: Define un control de entrada multilínea.
* **<button>: Define un botón ‘clicleable’.**
* **<select>: Define una lista desplegable (combo box).**
* **<optgroup>: Define un grupo de opciones para una lista desplegable.**
* <option>: Define una opción para una lista desplegable.
* <label>: Define una etiqueta para los controles <input>.

**FORMULARIOS**

* Permiten solicitar información al visitante.
* Estan compuestos por tantos campos como informaciones deseamos obtener.

Sintaxis

<form method=“[GET|POST]” action=“URL” name=“frm” id=“frm” >

Contenido del formulario

</form>

**<input>**

* Tag multifunción, dependiendo del valor del atributo **type**.

Sintaxis

<input type=“[TEXT|PASSWORD|CHECKBOX|RADIO|SUBMIT|

RESET|FILE|HIDDEN|IMAGE|BUTTON]”

name=“nombre” id=“nombre” value=“valor” />

**Caja de texto multilínea**

Sintaxis

<textarea name=“mensaje” id=“mensaje” rows=“5” cols=“20” wrap >

</textarea >

**Listas Desplegables**

Sintaxis

<select name=“cboLista” size=“2” id=“cboLista” >

<option value=“1”> Opción 1 </option>

<option value=“2” selected > Opción 2 </option>

</select>

Opciones Agrupadas

Sintaxis

<select name=“cboLista” id=“cboLista” >

<optgroup label=“Grupo 1”>

<option value=“1”> Opción 1 </option>

<option value=“2”> Opción 2 </option>

</optgroup>

<optgroup label=“Grupo 2”>

<option value=“3”> Opción 3 </option>

<option value=“4”> Opción 4 </option>

</optgroup>

</select>

**Datalist**

Sintaxis

<input list=“dlLista” placeholder=“Elija un destino” />

<datalist id=“dlLista” >

<option value=“Cuba”>

<option value=“Mexico” >

</datalist>

**JAVASCRIPT**

Sintaxis como llamar JS en HTML

<!doctype html>

<html>

<head>

<script type=“text/javascript” >

/\* código javascript aquí \*/

</script>

</head>

<body>

</body>

</html>

Sintaxis para llamar un archivo externo JS en HTML

<!doctype html>

<html>

<head>

<script type=“text/javascript”

src=“Funciones/archivo.js” >

</script>

</head>

<body>

</body>

</html>

Normas básicas de 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.
* Cada sentencia termina con el carácter de punto y coma **(;)**
* Se pueden incluir comentarios (son los mismos que en C).
* Utiliza los mismos operadores y estructuras (selectivas e iterativas) que en C.

Variables, Tipos y Ambitos

* Variables se declaran con **var**.
* No se declara el tipo de variable.
* Define dos ambitos para las variables: global y local.

Funciones

Sintaxis

function NombreFuncion([param1, param2, paramN])

{

// Código

// [return valor;]

}

* alert: Son las ventanas de mensaje.
* Prompt: ventana que se crea, la cual el usuario puede introducir lo que se pide.
  + Sintaxis:

var nombre;

nombre = prompt(“Ingrese su nombre”,”Usuario”);

alert(“Su nombre es: ” + nombre);

* Confirm: es la ventana que presenta dos botones, aceptar y cancelar. DEVUELVE UN BOOL.
  + Sintaxis:

if(confirm(“Desea continuar?”))

// Se ejecuta una acción

else

// Se ejecuta otra acción

**Acceso Directo a Nodos**

Funciones de acceso directo

.getElementsByTagName(): Obtiene todos los elementos de la página XHTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

.getElementsByName():Es similar a la anterior, pero en este caso se buscan los elementos cuyo atributo name sea igual al parámetro proporcionado.

.getElementById():Es la más utilizada cuando se desarrollan aplicaciones Web dinámicas. Se trata de la función preferida para acceder directamente a un nodo y poder leer o modificar sus propiedades.

**Acceso Directo a Atributos**

Sintaxis

var enlace = document.getElementById(“miLink”);

alert(enlace.href); // Muestra el atributo href

<a id=“miLink” href=“miPagina.php” > link </a>

**EVENTOS**

* Hacen possible que los usuarios transmitan información a los programas.
* JS permite asignar una función a cada uno de los eventos.
* Este tipo de funciones se denominan “event handlers” (manejador de eventos).

Tipos

|  |  |  |
| --- | --- | --- |
| Evento | Descripción | Elementos |
| onblur | Deseleccionar el elemento. | button, input, label, select, textarea, body |
| onchange | Deseleccionar el elemento que se ha modificado. | input, select, textarea |
| onclick | Clickear y soltar el mouse. | Todos los elementos |
| ondblclick | Clickear dos veces seguidas. | Todos los elementos |
| onfocus | Seleccionar un elemento. | button, input, label, select, textarea, body |
| onkeydown | Pulsar una tecla (sin soltarla). | Eventos de formulario y body |
| onkeypress | Pulsar una tecla. | Eventos de formulario y body |
| onkeyup | Soltar una tecla pulsada. | Eventos de formulario y body |
| onload | La página se ha cargado completamente. | body |
| onmousedown | Pulsar, sin soltar, un botón del mouse. | Todos los elementos |
| onmousemove | Mover el mouse. | Todos los elementos |
| onmouseout | El mouse sale del elemento. | Todos los elementos |
| onmouseover | El mouse entra en el elemento. | Todos los elementos |
| onmouseup | Soltar el botón que estaba pulsado. | Todos los elementos |
| onreset | Borrar los datos de un formulario. | form |
| onresize | Se ha modificado el tamaño de la ventana del navegador. | body |
| onsubmit | Enviar el formulario. | form |
| onunload | Se abandona la página. | body |

Los elementos más utilizados:

* Onload: para esperar a que se cargue la pagina por completo.
* onclick, onmouseover, onmouseout: Controlar el raton.
* Onsubmint: controlar el envio de los formularios.

**MANEJADORES COMO ATRIBUTOS**

Sintaxis para indicar el código JS que se debe ejecutar cuando se produzca un evento determinado.

<input type=“button” value=“Click Aquí”

onclick=“alert(‘hola mundo!!’);” />

**MANEJADORES COMO FUNCIONES (esta es la que usamos)**

Consiste en extraer todas las instrucciones de JS y agruparlas en una función externa.

Sintaxis

< input type=“button” value=“Click Aquí”

onclick=“MostrarMensaje();” />

**CSS**

* Son hojas de estilo en cascada. Son usadas para definir la presentación estética de un documento HTML.

Sintaxis

* Se agrega en el atributo **style** de un elemento:

<p style=“color:red; font-size:10pt;” > </p>

* En el tag <style> dentro del encabezado:

<style type=“text/css” > body{ background:black; } </style>

* En un archivo con extensión .css
  + Utilizando el tag <link> para referenciar al archivo.

<link type=“text/css” rel=“stylesheet” href=“URL” />

* + Invocándolo con el comando de CSS “@import...”.

<style type=“text/css” > @import url(URL); </style>

**Agrupacion de reglas**

* Una misma declaracion a varios sectores:

H1,H2,H3,H4 { color : black; }

* Varias declaraciones a un solo selector:

H1 {color:black; font-size:6pt;}

* Varias declaraciones a varios selectores:

H1,H2 {color:black; font-size:6pt;}

**CLASES**

Utilizando el atributo class se puede crear un sub-sector:

P { color : red; }

P.negrita { font-weight : bold; }

<p> Párrafo en color rojo </p>

<p class=“negrita” > Párrafo en negrita </p>

Tambien se puede definir una clase genérica para varios sectores.

.cursiva { font-style : italic; }

<span class=“cursiva” > Texto en cursiva </span>

**Identificadores únicos**

Sintaxis: #xlr8 { letter-spacing : 0.5em; }

Atributo id de un elemento html permite referenciar en forma única como selector.

**Pseudo Clases**

Sintaxis: A:link { color : red; }

Nota: las pseudo clases utilizan **:** mientras que las clases reales usan el punto.

**Pseudo Clases**

Permiten referirse a porciones de elementos reales.

Sintaxis:

P:first-line{font-style:small-camps;}

P:first-letter

{font-size:220%;

float:left;

}

**AJAX**

* Conjunto de tecnologias que nos permiten hacer paginas de internet interactivas.
* Permite que las paginas sean capaces de actualizar partes en segundo plano.

**TECNOLOGIAS**

* XHTML y CSS, para crear una presentación basada en estandares.
* DOM
* XML, XSLT y JSON, para intercambio y manipulación de información
* XMLHttpRequest, para el intercambio asicrono de información
* JS para unir todas.

**XMLHttpRequest**

* Objeto que es utilizado para intercambiar datos con el servidor en segundo plano.
* Para enviar una petición al servidor, se utilizan los métodos open() y send().

Sintaxis: xhttp.open("Método", "Url", Async);

* + Metodo: Especifica el tipo de pedido (GET/POST).
  + Url: indica la ubicación del archivo en el servidor.
  + Async: true (asincornico; false (sincronico).

xhttp.send(); //Si es GET

xhttp.send("string"); //Si es POST

Envia la petición al servidor

var xhttp = new XMLHttpRequest();

xhttp.open("GET", "ajax\_test.php", true);

xhttp.send();

**onreadystatechange**

* El evento se dispara cada vez que cambia **readyState**.
* La propiedad readyState mantiene el estado de XMLHttpRequest.
* Almacena una función (o el nombre de una función) que se invoca de forma atomatica cada vez que cambia de propiedad readyState.

readyState: mantiene el estado del XMLHttpRequest:

* + 0 : Solicitud no inicializado
  + 1 : Conexión de servidor establecido
  + 2 : Solicitud recibida
  + 3 : Solicitud de procesamiento
  + 4 : Solicitud terminada y respuesta lista
* **status**: 200 –> OK; 404 –> No encontrado.

**JQUERY** es un framework que lo utilizamos (ej: node.js).

**JSON**

* Es un formato de intercambio de datos ligero.
* Esta basado en un subconjunto de la sintaxis de JS.
* Es completamente independiente del lenguaje.

Se basa en dos estructuras:

* Una colección de pares nombre/valor.
* Una lista ordenada de valores

Estas son estructuras de datos universales. Todos los lenguajes de programación modernos los soportan de una forma u otra.

Literales de Objetos

Sintaxis:

var persona = **{** "nombre" **:** "Juan"**,** "edad" **:** 35 **}**;

alert(persona.nombre); // Juan

alert(persona.edad); // 35

alert(persona["nombre"]); // Juan

Literales de Arrays

Sintaxis:

var nombres = **[** "Juan"**,** "Pablo"**,** "Ramiro" **]**;

alert(nombres[0]); // Juan

alert(nombres[2]); // Ramiro

var personas = **[{** "nombre" **:** "Juan"**,** "edad" **:** 35 **},**

**{** "nombre" **:** "Anibal"**,** "edad" **:** 26 **}]**;

alert(personas[0].nombre); // Juan

alert(personas[1].nombre); // Anibal

**JSON.parse**

* El método analiza una cadena de texto como JSON, transoformando opcionalmente el valor producido por el análisis.

Sintaxis:

var str = **'**{ "nombre" : "Juan", "edad" : 35 } **'**;

var obj = JSON.parse(str);

alert(obj.nombre); //Juan

**JSON.stringify**

* convierte un valor a notación JSON representándolo:
  + Las propiedades de objetos no-array no se garantiza que se realicen en un orden particular.
  + Los objetos *Boolean, Number*, y *String* se convierten a sus valores primitivos.
  + Si se encuentra *undefined* durante la conversión o se omite (cuando se encuentra en un objeto) o se sustituye por *null* (cuando se encuentra en un array).

Sintaxis:

var obj = { "nombre" : "Juan", "edad" : 35 };

var str = JSON.stringify(obj);

// {"nombre" **:** "Juan"**,** "edad" **:** 35 }