Javascript

Introduccion

JavaScript es un lenguaje de programación que nos permite controlar todo lo que sucede en un documento HTML.

Su sintaxis se basa en otro lenguaje conocido como C.

JavaScript es un lenguaje que corre del lado del cliente, esto quiere decir que se ejecuta en el browser o agente del cliente (Internet Explorer, Firefox, Safari, Chrome, etc).

Alcance del lenguaje

Por medio de javascript uno toma control de todo lo que sucede en un documento HTML. Veamos algunas de las cosas que se pueden realizar:

- Redireccionar a otro documento
- Controlar los distintos eventos que van sucediendo tanto por parte del documento como del usuario
- Abrir ventanas secundarias
- Mostrar mensajes emergentes
- Confirmar con el usuario una acción
- Tomar datos de parte del usuario
- Validar datos en un formulario
- Intercambiar imágenes
- Agregar, modificar o quitar elementos de un documento

Donde y como se escribe

Encontramos 3 formas de incluir javascript dentro de un documento.

Por medio de la etiqueta <script>

Dentro de un evento

En un archivo independiente .js

Por medio de la etiqueta <script>

Ya sea en el head o en el body podemos incluir las etiquetas <script></script> para poder agregar código en javascript.

Como atributo de estas etiquetas debemos agregar el tipo de codificación que vamos a utilizar por medio del atributo type = “text/javascript”

Ejemplo:

```
<script type=”text/javascript”>

<!--

      Código javascript

-->

</script>
```

Podemos agregar dentro de las etiquetas <script></script> los elementos de comentario HTML <!-- y --> para lograr que si un browser antiguo no sabe cómo interpretar javascript, muestre el código comentado y no en pantalla. Si bien no vamos a lograr funcionalmente lo programado, al menos no se verá en pantalla el código.

Dentro de un evento

Dentro de una etiqueta HTML podemos escribir javascript en sus atributos eventos.

Ejemplo:

```
<p onclick="javascript:alert('javascript dentro de un html');"> Haga click aquí </p>
```

En un archivo independiente .js

Por medio de la etiqueta <script> podemos relacionar un archivo HTML con uno escrito en JavaScript logrando de esta forma tener separados la parte estructural de la funcional.

Se utiliza el atributo src=”archivo.js” para establecer cual va a ser el documento javascript que queremos vincular.

Ejemplo:

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

Ajax

AJAX representa las siglas de Asynchronous JavaScript And XML. No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-Técnología Servidor- en nuestro caso, Java-) que nos permiten hacer páginas de internet más interactivas.

La característica fundamental de AJAX es que nos permite actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar la página entera.

De modo similar podemos enviar información al servidor.

Algunos sitios web que utilizan AJAX

Google Maps

GMail

XMLHttpRequest

El propósito de esta interfaz es proporcionar contenido dinámico y actualizaciones asíncronas en páginas WEB mediante tecnologías construidas sobre ella como por ejemplo AJAX.

Ejemplo

La función validarUsuario() está asociada al evento onkeyupevent (se ejecuta cada vez que el usuario suelta una tecla) sobre el campo “idUsuario”

<input type="text" size="20" id="idUsuario" name="id" onkeyup="validarUsuario();">

Creamos el objeto ajaxRequest:

```
var ajaxRequest;  // Esta variable nos permitirá utilizar Ajax
function ajaxFunction(){
try{
  // Código para la mayoría de los browsers
  ajaxRequest = new XMLHttpRequest();
}catch (e){
  // Código exclusivo para Internet Explorer
  try{
     ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
  }catch (e) {
     try{
        ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
     }catch (e){
        // Algo falló!
        alert("Error!");
        return false;
     }
  }
}
}
```

Ahora vamos a escribir la función que utilizará Ajax para validar contra el servidor.

```
function validarUsuario() {
  ajaxFunction();
  // Acá indicamos cómo procesar la request una vez que ésta cambie de estado.
  ajaxRequest.onreadystatechange = processRequest;
  if (!target) target = document.getElementById("idUsuario");
  var url = "validate?id=" + escape(target.value);
  ajaxRequest.open("GET", url, true);
  ajaxRequest.send(null);
}
```

En el servidor, creamos un Servlet que retorne una respuesta a esa validación:

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                     throws IOException, ServletException
{
  String targetId = request.getParameter("id");

  if ((targetId != null) &&
      !accounts.containsKey(targetId.trim()))
  {
    response.setContentType("text/xml");
    response.setHeader("Cache-Control", "no-cache");
    response.getWriter().write("true");
  }
  else
  {
    response.setContentType("text/xml");
    response.setHeader("Cache-Control", "no-cache");
    response.getWriter().write("false");
  }
}
```

Finalmente, en nuestro código cliente, implementamos la función *processRequest:*

```
function processRequest()
```

```
{
 if (xmlhttp.readyState==4 && xmlhttp.status==200)
      {
      document.getElementById("divRespuesta").innerHTML=ajaxRequest.responseText;
      }
}
```

Componentes JavaBeans

Qué es un JavaBean

Es un mecanismo que permite el manejo de información proveniente de formularios

Debe seguir ciertas reglas, esencialmente respetar el encapsulamiento y poseer un constructor vacío

Cada formulario puede tener un bean asociado

Dónde utilizarlos

Se utiliza particularmente con los componentes de manejo de datos que forman parte de un formulario HTML

Cada atributo del bean deberá corresponder con un campo del formulario HTML

Los atributos deberán del bean deberán tener el mismo nombre que el campo en el formulario

Como crearlos

Para poder utilizarlos, se deberá crear dentro de un paquete una clase con todos los campos del formulario como atributos

Todos los atributos deberán ser privados, y tener métodos de acceso públicos

Los métodos de acceso deberán seguir la convención: comenzar con set o get y luego el nombre del atributo con la primera letra en mayúscula

Por ejemplo, setNombre() o getNombre(), donde “nombre” es el atributo

Deberá tener el constructor vacío

Se estila ubicarlos en un paquete llamados “beans”, y las clases en general terminan con la palabra Bean

jsp:useBean

Es el mecanismo que se utiliza para obtener el bean

Si no está instanciado, lo instancia

La forma de utilización es la siguiente:

<jsp:useBean id="identificador del bean dentro del jsp"

class="nombre completo de la clase" />

jsp:setProperty

Es el mecanismo que se utiliza para establecer un dato en un bean

Los datos se representan a través de propiedades

La forma de utilización es la siguiente:

<jsp:setProperty

name="identificador del bean dentro del jsp" property="nombre de la propiedad"

value=”un valor” />

Para llenar automáticamente el bean con los datos del formulario enviado, se debe hacer lo siguiente:

<jsp:setProperty

name="identificador del bean dentro del jsp" property="*" />

jsp:getProperty

Es el mecanismo que se utiliza para obtener un dato de un bean

Los datos se representan a través de propiedades

La forma de utilización es la siguiente:

<jsp:getProperty name="identificador del bean dentro del jsp" property="nombre de la propiedad" />

Ejemplo de uso

Creación de la clase FormBean:

package beans;

```
public class FormBean {

    public FormBean() {
    }

    private String nombre;
    private String apellido;

    public String getNombre() {
       return nombre;
    }

    public void setNombre(String nombre) {
       this.nombre = nombre;
    }

    public String getApellido() {
       return apellido;
    }

    public void setApellido(String apellido) {
    this.apellido = apellido;
    }

}
```

Para poder utilizarlos desde una página JSP, se deberá hacer lo siguiente:

```
<jsp:useBean id="alumno" class="beans.FormBean" />

<jsp:setProperty name="alumno" property="*" />



<h1>Bienvenido!</h1>

Nombre:<jsp:getProperty name="alumno" property="nombre" /><BR>

Apellido:<jsp:getProperty name="alumno" property="apellido" /><BR>
```