

DESARROLLO DE SERVICIOS TELEMÁTICOS

Tema 3

3er. curso
Graduado en Informática
Mención de Tecnologías de la Información

Autora: Lidia Fuentes



Lidia Fuentes Fernández

Programación Web del lado del cliente

- Introducción a la programación Web del lado del cliente
- Hojas de estilo
- Lenguajes script y otras tecnologías
- Aplicaciones de Internet enriquecidas (*Rich Internet Applications*)



3.1. Introducción a las Tecnologías Web del lado del cliente

- Permiten añadir funcionalidad a una aplicación ejecutada a partir de un navegador web
- Estas tecnologías no sirven para programar una aplicación completa, sólo para mejorar las páginas Web.
- Se caracterizan fundamentalmente por ser un código (etiquetas, lenguajes script, ...) que se interpreta y ejecuta por parte del navegador (o sea, en el cliente)

3



3.1. Introducción a las Tecnologías Web del lado del cliente

- Tecnologías base
 - Extensión etiquetas HTML
 - Lenguajes script
 - Los applets de Java

4



3.1. Introducción a las Tecnologías Web del lado del cliente

- Extensión de las etiquetas HTML
 - Cascading Style Sheets (CSS)
 - Mecanismo para definir estilos (ej: tipo de fuente, color, tamaño fuentes, etc.) en HTML
 - Separa la presentación de la estructura de una página HTML
 - Dynamic HTML (DHTML)
 - Define el concepto de manejador de eventos de usuario enviados por el navegador

5



3.1. Introducción a las Tecnologías Web del lado del cliente

- Lenguajes script
 - Permiten codificar manejadores de eventos de usuario (eventos DHTML)
 - Pueden generar contenido dinámicamente en el cliente, fundamentalmente relacionado con la apariencia y navegación entre páginas, responsabilidad del navegador
 - Permite incrementar la interactividad con el cliente sin gestionar peticiones/respuestas HTTP
 - Permite desarrollar cualquier aplicación que no necesite datos locales (ej: calculadora, conversión de euros a pesetas, etc.)
 - Hacer validación de los formularios en el cliente
 - Etc.

6



3.1. Introducción a las Tecnologías Web del lado del cliente

■ Lenguajes script

- Son lenguajes poco tipados y son interpretados por el navegador que lo ejecuta línea a línea
- Algunos browsers como Netscape e Internet Explorer de Microsoft han implementado lenguajes tipo scripts
 - Javascript
 - ActiveX
- Se transfiere en claro dentro de una página web y embebido dentro de un fichero html

```
<HTML>
<!-- Ejemplo javascript -->
<SCRIPT TYPE="text/javascript">
    function ...
</SCRIPT> ...
</HTML>
```

7



3.1. Introducción a las Tecnologías Web del lado del cliente

■ Applets de java

- Los applets son pequeñas aplicaciones java que residen en la máquina del servidor, se transmiten a través de la red, y se ejecutan automáticamente en el navegador (el browser) como si fuese parte del documento html.

8



Tecnologías del lado del cliente vs aplicaciones web

■ Ventajas

- ☐ Liberan recursos en el servidor
- ☐ Reducen el trasvase de información cliente-servidor y el tiempo de respuesta
- ☐ La presentación de los datos es ejecutada en el navegador, encargado de visualizarlos, entonces puede optimizarla

■ Inconvenientes

- ☐ Sobrecargan la máquina del cliente
- ☐ Problemas de seguridad
- ☐ Poca compatibilidad de los navegadores (dependen mucho del tipo de navegador)
- ☐ Limitaciones para desarrollar aplicaciones web completas

9



Limitaciones lenguajes script

- Sólo controlan la información dinámica que está basada en el entorno del cliente.
- Con la excepción de las cookies, el HTTP y el envío de formularios no están disponibles
- Y, como se ejecutan en el cliente, no pueden acceder a los recursos en el lado del servidor, como bases de datos, catálogos, información de precios, etc.
- **Solución de compromiso: Rich Internet Applications (RIA)**
 - ☐ Es un conjunto de tecnologías que pretenden combinar las ventajas de las aplicaciones web y de las aplicaciones de escritorio
 - AJAX/Javascript, Applets/Java Web Start, JSF, Flash/Adobe Flex, GWT, etc.

10



3.2. Hojas de estilo

■ CSS (Cascading Style Sheets)

- Nos permite controlar los estilos de presentación de una página web (ej: tipo de letra, color, espaciado)

■ Motivación

- Antes un estilo de fuente se definía como:
` Texto 1 `
- Si quiero aplicar el mismo estilo a varios textos
` Texto 2 `
` Texto 3 `
- Con las hojas de estilo, un estilo de presentación se define sólo una vez y puede usarse en diferentes ficheros HTML

11



3.2. Hojas de estilo

■ Hoja de estilo

- Es un conjunto de **reglas** que determinan cómo serán presentados los elementos de una página HTML
- Un regla puede aplicarse a:
 - Elementos representados por una misma etiqueta HTML
 - Un grupo de elementos que comparten el mismo atributo "class"
 - Elemento individual (identificado por "id")
- Estándar W3C
 - <http://www.w3.org/TR/CSS2/>

```
/* REGLA CSS */
selector, ... {
  property1: value1;
  property2: value2;
  compositeProperty: property1Value property2Value ... ;
}
```

12



Formato reglas CSS

■ Etiqueta HTML

- `<p> Hola Mundo </p>`

```
/* Ejemplo */
p {
    font-family: times new roman;
    font-size: 11pt;
}
```

■ Elementos de una clase

- `<p class="authorName">Autor de la pagina</p>`

```
/* Ejemplo */
.authorName {
    font-size: 10pt;
    color: blue;
}
```

■ Elemento con identificador "id"

- `<div id="navbar"> ... </div>`

```
/* Ejemplo */
#navbar {
    font-size: 9pt;
    color: white;
    background-color: black;
}
```

13



Formato reglas CSS

■ Varios selectores

- Se ponen separados por comas

```
/* Ejemplo */
p, h4 {
    font-family: courier new;
    font-size: 10pt;
}
```

■ Valores multivaluados

- El navegador los debe considerar en orden de preferencia

```
/* Ejemplo */
.authorName {
    font-family: verdana, tahoma, georgia;
}
```

■ Valores relativos de propiedades numéricas

```
/* Ejemplo */
em {
    font-size: 120%;
}
```

14



Asociar hojas de estilo

- Asociar una hoja de estilo a una página HTML con la
 - Etiqueta `<style>` en línea
 - Etiqueta `<style>` y directiva `@import`
 - Etiqueta `<link>`
- En línea (embebido en la página HTML)

```
<head><style type="text/css">
  H1 {
    text-align: center;
  }
  P {
    font-family: times new roman;
    font-size: 11pt;
  }
</style>
...</head>
```

15



```
<!DOCTYPE html>
<html>

<head>
<style type="text/css">
  H1 {
    text-align: center;
  }
  P {
    font-family: times new roman;
    font-size: 11pt;
  }
</style>
<meta http-equiv="Content-Type" content="text/html">
<title>JSP Page</title>
</head>

<body>
  <h1>Hello World!</h1>
  <p>Estilo CSS</p>
</body>

</html>
```





Asociar hojas de estilo

■ directiva @import

```
<head>
  <style type="text/css">
    @import url("main.css");
    ...
  </style>
</head>
```

```
/* main.css */
root {
  display: block;
}

body {
  color: purple;
  background-color: aquamarine;
}
```

17



Asociar hojas de estilo

■ directiva

```
<head>
  <style type="text/css">
    @import url("main.css");
    ...
  </style>
</head>
```

```
/* main.css */
root {
  display: block;
}

body {
  color: purple;
  background-color: aquamarine;
}
```



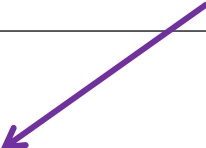
18



Asociar hojas de estilo

■ Etiqueta link

```
<head>
  <link rel="stylesheet" type="text/css" href="main.css" />
</head>
```



```
/* main.css */
root {
  display: block;
}

body {
  color: purple;
  background-color: aquamarine;
}
```

19



@import vs <link>

■ @import

- ☐ Es una especie de "include" de "C"
- ☐ Lo interpreta el módulo CSS para importar todos los estilos definidos en un fichero .css

■ Link

- ☐ El HTML interpreta la etiqueta <link>
- ☐ Se pueden importar estilos definidos en un sitio identificado por una URL
`href="http://www.hostnoexistente.com/estilo.css"`

20



Definir características de los enlaces

- `a { }` /*Para características generales a los enlaces*/
- `a:link { }` /*para el enlace, antes de ser visitado*/
- `a:visited { }` /*Para cuando ha sido visitado*/
- `a:hover { }` /*Cuando el ratón está sobre él*/
- `a:active { }` /*Cuando está siendo pulsado*/

```
a:link { ; }  
a:visited { ; }  
a:hover {color:red ; }  
a:active {text-decoration: underline ; }
```

21



Hojas de estilo en cascada

- Permiten combinar los estilos definidos en varios ficheros de estilo
- Para definir una cascada, los autores especifican una secuencia de elementos LINK y/o STYLE

22



3.3. Lenguajes script y otras tecnologías

- DHTML
- Javascript

23



3.3.1. HTML Dinámico

- Dynamic HTML (DHTML)
 - Es la denominación genérica al conjunto de tecnologías del lado del cliente que permiten variar dinámicamente el formato de las páginas Web
 - Describe las posibilidades de HTML en conjunto con los manejadores de eventos de lenguajes script (ej: JavaScript) y las especificaciones CSS de hojas de estilo
 - Los eventos se definen en el *Document Object Model* (DOM)
 - Ver <http://www.w3.org/TR/DOM-Level-3-Events/>

24



3.3.1. Lenguaje Javascript

- Definido por Netscape, que al igual que java es independiente de la plataforma de ejecución.
- JavaScript es un lenguaje script orientado a documento.
- Todo el código está dentro de funciones.
- Las funciones se codifican entre las etiquetas **<script>** y **</script>**
- La llamada a las funciones se hace a través de un evento de un elemento del documento (DHTML).

25



Javascript (alert)

- **LA VENTANA "ALERT"**
 - Ventana estándar para mostrar información en pantalla.
 - Se puede mostrar texto, variables y ambas cosas.
 - El diseño de la ventana ya está definido lo único que podemos hacer es mostrar la información en una o varias líneas.
 - Su sintaxis es:
 - `alert("texto de la ventana");`
 - `alert(variable);`
 - `alert("texto"+variable);`

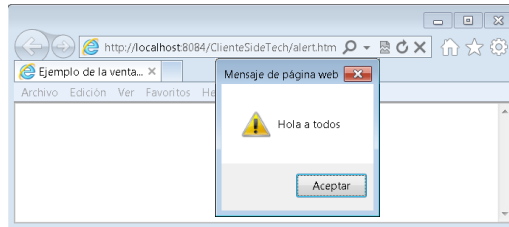
26



Ejemplo “alert”

```
<html>
<head>
<script>
  function hola()
  {
    alert("Hola a todos");
  }
</script>
<title>Ejemplo de la ventana alert</title>
</head>

<body onload=hola();>
</body>
</html>
```



27



Javascript (eventos)

- Las acciones que realiza el usuario las detectamos mediante eventos
- Se llama automáticamente a la función que tengamos asociada. Desde esta función realizaremos las acciones que tengamos desarrolladas

```
<elemento nombre_evento=nombre_funcion([parametros]);>
```

28



Javascript (eventos)

EVENTO	SE PRODUCE AL
onLoad	Terminar de cargar una página o frame (entrar).
onUnLoad	Descargar una página o frame (salir).
onAbort	Abortar la carga de una página.
onError	Producirse algún error en la carga de la página.
onMouseOver	Pasar el ratón por encima de un enlace, area o frame.
onMouseOut	Dejar de estar el ratón encima de un enlace, area o frame.
onMouseMove	Mover el ratón por el documento.
onKeyUp	Presionar una tecla.
onClick	Hacer click con el ratón.
onResize	Dimensionar la ventana del navegador.
onMove	Mover la ventana del navegador.
onChange	Modificar texto en un control de edición. Sucede al perder el foco.
onSelect	Seleccionar texto en un control de edición.
onFocus	Situar el foco en un control.
onBlur	Perder el foco un control.
onSubmit	Enviar un formulario.
onReset	Inicializar un formulario.

Ver <http://www.w3.org/TR/html4/interact/scripts.html>


29



Ejemplo “eventos”

```
<html>
<head>
<script>
  function press(){alert("No toque !!");}
  function foco(){alert("Foco en la primera Caja");}
  function tecla(){alert("Pulsada tecla");}
</script>
<title>Software de Comunicaciones</title>
</head>
<body>
  <input type="button" value="Press" onClick=press();>
  <input type="text" size="5" onFocus=foco();>
  <input type="text" size="5" onKeyPress=tecla();>
</body>
</html>
```

30

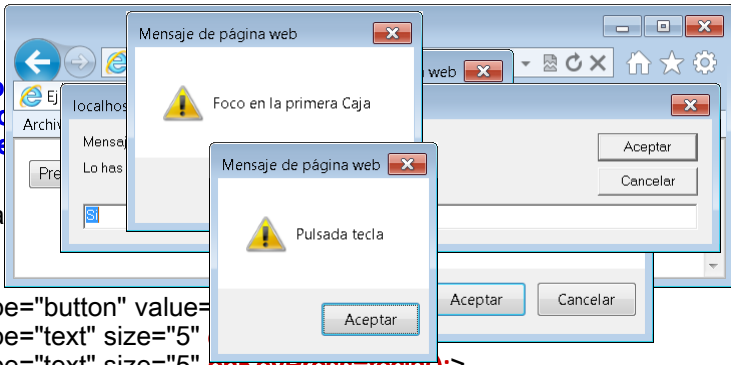

Lidia Fuentes Fernández

Ejemplo “eventos”


```

<html>
<head>
<script>
function p
function f
function t
</script>
<title>Softwa
</head>
<body>
<input type="button" value=
<input type="text" size="5"
<input type="text" size="5" onKeyPress=tecla();>
</body>
</html>

```



31


Lidia Fuentes Fernández

Javascript (texto)

- Se pueden usar todos los elementos de un formulario y enriquecerlos con javascript
 - Ej: recoger el valor de una caja de texto

FUNCIÓN	DESCRIPCIÓN
variable=nombre_caja.value;	Guarda el contenido de la caja
nombre_caja.value=valor o variable;	Muestra en la caja el valor
nombre_caja.value="";	Limpia el contenido de la caja
nombre_caja.focus();	Envía el foco a la caja

32



Ejemplo “caja de texto”

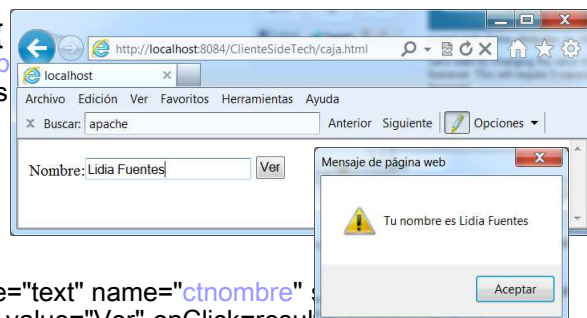
```
<html><head>
<script>
  function resultado() {
    var nombre=ctnombre.value;
    alert("Tu nombre es "+nombre);
    ctnombre.value="";
    ctnombre.focus();
  }
</script></head>
<body>
  Nombre:<input type="text" name="ctnombre" size="25">
  <input type="button" value="Ver" onClick=resultado();>
</body></html>
```

33



Ejemplo “caja de texto”

```
<html><head>
<script>
  function resultado() {
    var nombre=ctnombre.value;
    alert("Tu nombre es "+nombre);
    ctnombre.value="";
    ctnombre.focus();
  }
</script></head>
<body>
  Nombre:<input type="text" name="ctnombre" size="25">
  <input type="button" value="Ver" onClick=resultado();>
</body></html>
```



34



Javascript (sentencias de control)

- Se utilizan las típicas del lenguaje Java (if, switch, while, do-while, for,...)

```
<script>
function repetir() {
  var valor=1;
  while(valor<=10)
  {
    alert("Esto sale 10 veces:"+ valor);
    valor++;
  }
}
</script></head>
<body>
  <a href="Ejemplo.htm" onMouseOver=repetir();>ir a uno</a>
```

35



Javascript (animaciones)

- La animación en JavaScript puede ser de texto, imágenes o ambas cosas interactuando.
 - Dar un nombre al texto o a la imagen por medio del atributo **name** en caso de las imágenes e **id** en caso de texto.
 - Si la animación avanza en el tiempo: **setTimeout()**
 - Para modificar el estilo:

SINTAXIS	DESCRIPCIÓN
style="Position:absolute;top:pos;left:pos"	Posibilita el cambio de posición.
style="color:nombre_color"	Posibilita el cambio de color.
style="visibility:hidden o visible"	Posibilita mostrar y ocultar.

36



Ejemplo (animaciones)

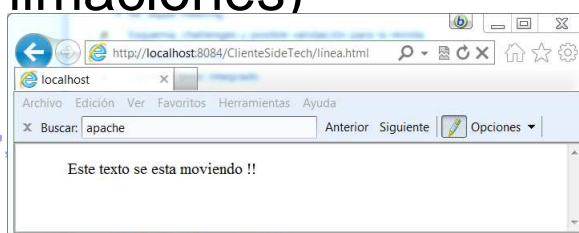
```
<html><head>
<script>
  var horizontal=12;
  window.setTimeout("mueve()",150);
  function mueve() {
    horizontal+=10;
    TEXTO1.style.left=horizontal;
    if(horizontal>=200)
      horizontal=12;
    window.setTimeout("mueve()",500);
  }
</script></head><body>
<p id="TEXTO1" style="position:absolute; top:16; left:12">
Esta línea se esta moviendo !!</p>
</body></html>
```

37



Ejemplo (animaciones)

```
<html><head>
<script>
  var horizontal=12;
  setTimeout("mueve()",
function mueve() {
  horizontal+=10;
  TEXTO1.style.left=horizontal;
  if(horizontal>=200)
    horizontal=12;
    setTimeout("mueve()",500);
  }
</script></head><body>
<p id="TEXTO1" style="position:absolute; top:16; left:12">
Esta línea se esta moviendo !!</p>
</body></html>
```



38



Ejemplo (animaciones)

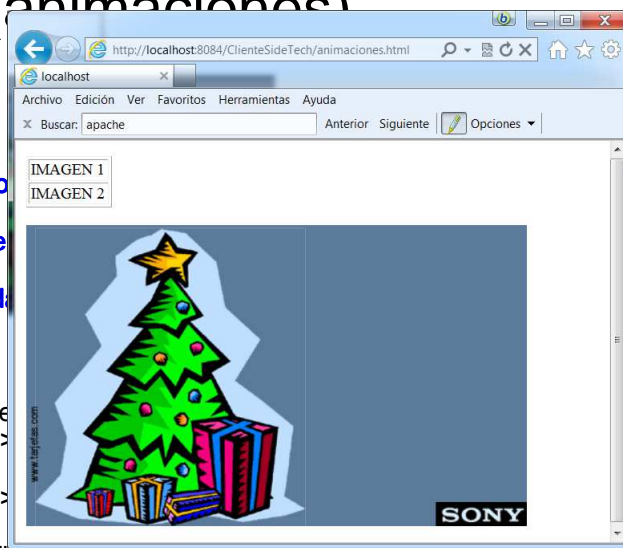
```
<html><head>
<script>
function imag(valor) {
    if(valor==1)
        img.src="imagen1.jpg"
    else
        img.src="navidad.gif"
    }
</script>
</head><body><table border="1" width="15%"><tr>
<td width="100%"><p onmouseover=imag(1);>IMAGEN 1</td>
</tr><tr>
<td width="100%"><p onmouseover=imag(2);>IMAGEN 2</td>
</tr></table>
<br>
</body></html>
```

39



Ejemplo (animaciones)

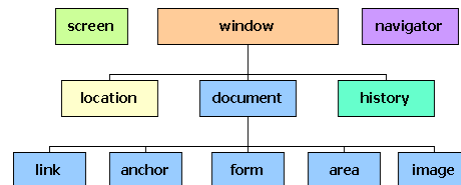
```
<html><head>
<script>
function imag(valor) {
    if(valor==1)
        img.src="image
    else
        img.src="navida
    }
</script>
</head><body><table
<td width="100%">
</tr><tr>
<td width="100%">
</tr></table>
<br>
</body></html>
```



40



Javascript (objetos predefinidos)



- Cuando se carga un documento en el navegador, se crean automáticamente una colección de Objetos predefinidos
- Son útiles para describir el documento sobre el que se trabaja, la ventana del navegador y todo tipo de elementos de las páginas web.
- Se agrupan en los objetos:
 - window, document, history, location, navigator y screen

41



Javascript (window)

- Nos permite definir las características de la ventana del navegador o de las ventanas que construyamos nuevas.

METODO	DESCRIPCIÓN	SINTAXIS
open	Abrir ventanas.	var=window.open("url","name","atrbts");
close	Cerrar ventanas.	var.close();
opener	Indica si se abrió.	var_boolean=var.opener SI devuelve true
closed	Indica si se cerró.	var_boolean=var.closed SI devuelve false
Location	Enlaza con una página.	var.Location="url";
print	Imprime el documento.	print();
alert	Abre ventanas alert.	alert(datos);
confirm	Abre ventanas confirm.	confirm(datos);
prompt	Abre ventanas prompt.	prompt(datos,"val inici");
status	Texto en barra estado.	status="mensaje";
showModalDialog	Crea ventana modal.	var=window.showModalDialog("url","atrbts");

42



Javascript (documento)

- Objeto dependiente de *window*, que contiene las propiedades para trabajar con el documento de la ventana.
- Sus métodos pueden ser usados también por *window*.
- Todos los métodos están listados en <https://developer.mozilla.org/en-US/docs/DOM/document>

METODO	DESCRIPCION	SINTAXIS
write	Escribe en el documento.	document.write(dato);
writeln	Escribe y salta de línea.	document.writeln(dato);
referrer	Uri del documento anterior.	var=document.referrer;
location	Uri del documento actual.	var=document.location;
lastModified	Fecha modificación.	var=document.lastModified;
document.forms*	Devuelve una lista de elementos FORM dentro del document actual	var=document.forms["f1"]
document.location	Devuelve la URI del documento actual	var=document.location;
document.body*	Devuelve el nodo BODY del documento actual	var=document.body;
images	Devuelve la colección de imágenes del documento	var=document.images[3];
...

43



Javascript (histórico)

- Objeto derivado de *window*, contiene todas las direcciones que se han ido visitando durante la sesión actual. Al ser un objeto derivado de *window*, este también puede utilizar sus métodos.
- Tiene 3 métodos:

METODO	DESCRIPCIÓN	SINTAXIS
back()	Vuelve a la página anterior.	window.history.back();
forward()	Nos lleva a la página siguiente.	window.history.forward();
go(valor)	Van donde le indique el número. Este puede ser: -1 como back. num lleva a pag X 1 como forward	window.history.go(valor);

44



Ejemplo “window”

```
<html><head><script>
var newWindow;
function abre() {
  newWindow=window.open("", "", "status=yes,height=200,width=300");
  if (newWindow != null) {
    var newContent="<HTML><HEAD><TITLE> Nueva ventana
    </TITLE></HEAD>";
    newContent += "<BODY><H1>Esta ventana es nueva y la he hecho
    yo.</H1></BODY></HTML>";
    newWindow.document.write(newContent); // escribir HTML en la nueva ventana
    newWindow.document.close();
  }
  newWindow.status="Ventana personal";
}
function cierra() { newWindow.close(); history.back();} </script></head>
<body onload=abre();>
  <input type="button" value="Cerrar" onClick=cierra();>
</body></html>
```

45



Ejemplo “ . . . ”

```
<html><head><script>
var newWindow;
function abre() {
  newWindow=window.open("http://www.uma.es/");
  if (newWindow != null) {
    var newContent="<HTML><HEAD><TITLE> Nueva ventana
    </TITLE></HEAD>";
    newContent += "<BODY><H1>Esta ventana es nueva y la he hecho
    yo.</H1></BODY></HTML>";
    newWindow.document.write(newContent);
    newWindow.document.close();
  }
  newWindow.status="Ventana personal";
}
function cierra() { newWindow.close(); history.back();} </script></head>
<body onload=abre();>
  <input type="button" value="Cerrar" onClick=cierra();>
</body></html>
```



46



Declaración de manejador

- Asociar un elemento HTML con un manejador de eventos

- `document.getElementById("id")`

```
<html>
<head></head><body>
<a id="link1" href="#" onclick="clickTheLink()">Texto del enlace</a>
</body></html>
```

Forma declarativa

```
<html>
<head>
<script>
Function clickTheLink() {
    // implementacion del manejador del evento click
}
window.onload = function() {
    document.getElementById("link1").onclick = clickTheLink;
};
</head><body>...</body></html>
<a id="link1" href="#">Texto del enlace</a>
....
```

evento



Modificación parcial página Web

- Propiedad para obtener y reemplazar el contenido dentro de un elemento HTML
 - Se aplica a todo el código HTML incluido dentro de las etiquetas de apertura y cierre del elemento
 - `document.getElementById("id").innerHTML="hola";`
- Modificar un atributo del elemento HTML

Cualquier atributo del elemento HTML seleccionado. Ejemplo:

```

→ doc.src="otrafig.jpg";
```

```
var doc = document.getElementById("id");
doc.innerHTML = "otro texto";
doc.style.backgroundColor = "#ffffcc";
doc.style.color = "ffcccc";
```

- Métodos para modificar un conjunto de elementos HTML
 - `document.getElementsByClassName("classname");`
 - `document.getElementsByTagName("tagname");`



Modularización de código javascript

- Javascript insertado dentro de las etiquetas `<script></script>`
 - Este código puede verlo y editarlo cualquier usuario
 - Difícil de modularizar y reutilizar
- Ficheros .js
 - En NetBeans NewFile->Web->Javascript File
 - Tiene extensión .js
 - En el fichero .html que usa el .js se incluye

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

49



Fichero .js

■ Etiqueta script

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

```
/* ventanas.js */
var newWindow;
function abre() {
  newWindow=window.open("", "", "status=yes,height=200,width=300");
  if (newWindow != null) {
    var newContent="<HTML><HEAD><TITLE> Nueva ventana </TITLE></HEAD>";
    newContent += "<BODY><H1>Esta ventana ... yo.</H1></BODY></HTML>";
    newWindow.document.write(newContent); // escribir HTML en la nueva ventana
    newWindow.document.close();
  }
  newWindow.status="Ventana personal";
}
function cierra() {
  newWindow.close();
  history.back();
}
```

50



Validación de formularios

■ Código de validación en el formulario

```
<form action="..." method="..." id="..." name="..."  
      onsubmit="return validarForm(this)"> ...</form>
```

■ Esquema de validación en javascript

```
function validarForm(formulario) {  
    if (condicion campo1 formulario) {  
        // si no se cumple la condicion  
        alert('Error: el campo1 debe ser ...');  
        return false;  
    }  
    else if (condicion campo2 formulario) {  
        alert('Error: el campo2 debe ser ...');  
        return false;  
    }  
    ...  
    // Todas las condiciones se han cumplido  
    return true;  
}
```

51



Ejemplo validar formulario

```
function validarForm(formulario) {  
    var login=formulario.loginName;  
  
    // valor no nulo o longitud del campo igual a 0  
    if (login.value == null || login.value.length==0) {  
        alert('Mandatory Field');  
        login.focus();  
        return false;  
    }  
    else if (...) {  
    }  
    ...  
    return true;  
}
```

52



3.4. Aplicaciones de Internet enriquecidas

- Rich Internet Applications (RIA)
 - Surgen como un esfuerzo por combinar las ventajas de las aplicaciones web y las aplicaciones de escritorio
 - No hay necesidad de instalaciones ni actualizaciones del lado del usuario

53



3.4. Aplicaciones de Internet enriquecidas

- Applets de Java
- Java Web Start
- AJAX

54



3.4.1. Applets

- Los applets son pequeñas aplicaciones gráficas que residen en la máquina del servidor, se transmiten a través de la red, y se ejecutan automáticamente en la máquina donde reside la aplicación cliente (el browser) como si fuese parte del documento html.
- Un applet va integrado dentro de un documento Web que al ser invocado y transportado a través de la red lleva consigo el código del applet (.class).
- Las applets no tienen ventana propia: se ejecutan en la ventana del browser (en un “panel” o pestaña).
- Aunque su entorno de ejecución es un browser, las applets se pueden probar sin necesidad de browser con la aplicación appletviewer del JDK de Sun.

55



Applets

- Puede heredar de Applet (uso de awt) o de JApplet (Swing)
- Métodos que controlan su ejecución:
 - init()
 - Se llama automáticamente al método **init()** en cuanto el browser o visualizador carga el **applet**.
 - start()
 - Se llama automáticamente en cuanto el **applet** se hace visible
 - stop()
 - Se llama de forma automática al ocultar el **applet**
 - destroy()
 - Se llama a este método cuando el **applet** va a ser descargada para liberar los recursos

56



Applets (página html)

<! Deprecated>

```
<APPLET CODE="miApplet.class" [CODEBASE="unURL"]
    [NAME="unName"]
    WIDTH="wpixels" HEIGHT="hpixels"
    [ALT="Mensaje si no se puede ejecutar el applet"]>
    [<PARAM NAME="MyName1" VALUE="valueOfMyName1">]
    [<PARAM NAME="MyName2" VALUE="valueOfMyName2">]
</APPLET>

<OBJECT codetype="application/java"[CODEBASE="unURL"]
    CLASSID="java:miApplet.class"
    WIDTH="wpixels" HEIGHT="hpixels">
    [<PARAM name="MyName1" VALUE="valueOfMyName1">]
    [Código HTML que aparece si el cliente no puede ejecutar applets Java]
</OBJECT>
```

57



```
import java.applet.*;
import java.awt.*;

public class SimpleApplet extends Applet {
    private String mensaje;
    private int x,y;
    private Dimension d;

    public void init() {
        mensaje=getParameter("Mensaje");
        d=this.getSize();
        x=d.width/2;
        y=d.height/2;
    }
    public void start() {
        repaint();
    }
    public void paint(Graphics g) {
        g.drawRect(0,0,d.width-1,d.height-1);
        g.drawString("MENSAJE => "+mensaje,x,y);
    }
    public boolean mouseUp(Event e,int x, int y) {
        this.x=x;
        this.y=y;
        repaint();
        return true;
    }
}
```


58

Lidia Fuentes Fernández

Despliegue (etiqueta APPLET)

```
<HTML>
<H3><HR WIDTH="100%">
  Este es mi primer applet
  <HR WIDTH="100%"></H3>
<P>
<APPLET
  code="simpleapplet.SimpleApplet"
  archive="SimpleApplet.jar"
  width=350 height=200>
  <param name = Mensaje value="Mueve este texto">
</APPLET>
</P>

<HR WIDTH="100%"><FONT SIZE=-1>
<I>Applet de la asignatura DST</I></FONT>
</BODY>
</HTML>
```



Incluir en panel de control->Programas
-> Java->Seguridad->Lista de
excepciones a sitios =
<http://localhost:8084> (o puerto del
servidor Web usado)

Lidia Fuentes Fernández

Crear Applets con NetBeans

- Desarrollar un aplicación de tipo Applet
 - ☐ Crear un proyecto Java
 - ☐ Crear una clase Applet
 - New File->Java->Applet (o JApplet si se usa Swing)
 - ☐ Codificar la clase Applet redefiniendo los métodos de Applet o de JApplet
 - ☐ Generar el .jar del proyecto (opción Clean and Build)
- Insertar un applet en un proyecto
 - ☐ En el proyecto donde se quiere usar el applet (ej: un servicio web) insertar el .jar (opción Properties->Packaging->Add Project)

60



Despliegue de applets

- Usando la etiqueta APPLET (restricciones de seguridad severas)
- Usando el fichero .jnlp (mejora sus posibilidades de acceso a través de la API JNLP)


61



Despliegue .jnlp con NetBeans

- En el proyecto Java donde reside el applet configurar el despliegue por Web Start
 - Propiedades->Application->Web Start->Applet Descriptor (agregar parámetros si es necesario)
 - Clean and Build (regenerar todo el proyecto)
 - Se crean los ficheros
 - launch.html (ejemplo para lanzar el applet con .jnlp)
 - launch.jnlp
 - Verlos en la vista Files
 - <Nombre proyecto>/dist/launch.*

62


Lidia Fuentes Fernández


Despliegue (script de launch.html)

```

<body>
  <h3>Este es mi primer applet (despliegue con JNLP)</h3>
  <script src="http://java.com/js/deployJava.js"></script>
  <script>
    var attributes = {
      code:      "simpleapplet.SimpleApplet",
      archive:   "SimpleApplet.jar",
      width:    350,
      height:   200
    };
    var parameters = {
      Mensaje: "Mueve este texto",
      jnlp_href: "launch.jnlp"
    };
    var version = "1.6"; <!-- Version de Java -->
    deployJava.runApplet(attributes, parameters, version);
  </script>
  ...

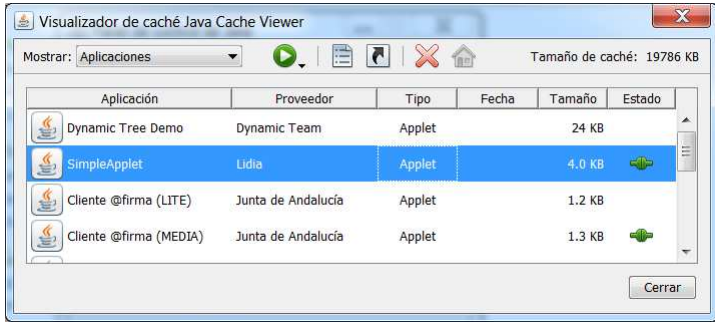
```

63


Lidia Fuentes Fernández

Applets

- Panel de control -> Programas -> Java -> General -> Archivos temporales de Internet -> Ver



Aplicación	Proveedor	Tipo	Fecha	Tamaño	Estado
Dynamic Tree Demo	Dynamic Team	Applet		24 KB	
SimpleApplet	Lidia	Applet		4.0 KB	
Cliente @firma (LITE)	Junta de Andalucía	Applet		1.2 KB	
Cliente @firma (MEDIA)	Junta de Andalucía	Applet		1.3 KB	

64



Applets (seguridad)

- Por la propia naturaleza “abierta” de Internet, las **applets** tienen importantes restricciones de seguridad: sólo pueden leer y escribir ficheros en el servidor del que han venido, sólo pueden acceder a una limitada información sobre el ordenador en el que se están ejecutando, no pueden crear sockets, etc.
- Con ciertas condiciones, las **applets** “de confianza” (**trusted applets**) pueden pasar por encima de estas restricciones.

65



Applets (seguridad)

- Dos opciones:
 1. Creación, exportación e importación de un [selfCertificate](#)
Creación y firma de un fichero [Jar](#) (el applet)
 2. Configuración mediante la herramienta [PolicyTool](#)

66



Applets (seguridad)

■ Creación de un "autocertificado"

```
C:\Documents and Settings\usuario1>keytool -genkey -alias dst
Enter keystore password: XXXX
.....
Is <CN=Desarrollo de Servicios Telematicos, OU=LCC, O=UMA, L=M, ST=M, C=ES>
correct?
[no]: yes
Enter key password for <dst>
(RETURN if same as keystore password):
C:\Documents and Settings\usuario1>keytool -selfcert -v -alias dst
```

■ Exportar/Importar un "autocertificado"

```
C:\Documents and Settings\usuario1>keytool -export -alias dst -file dst.cer
Enter keystore password: dst123
Certificate stored in file <dst.cer>
```

CLIENTE (opción 1):

```
C:\Documents and Settings\usuario1> keytool -import -alias dst -file dst.cer
(y ejecuta con doble click el fichero dst.cer)
```

67



Applets (seguridad)

■ Crear un fichero jar a partir de un applet

```
C:\Documents and Settings\usuario1> jar cvf hola.jar HolaApplet.class
```

■ Firmar el fichero jar con el applet

```
C:\Documents and Settings\usuario1> jarsigner -signedjar hola-sign.jar
hola.jar dst
```

CLIENTE (opción 2):

El Java plug-in para los navegadores incluye la carga de certificados (verlo en windows en el panel de control)

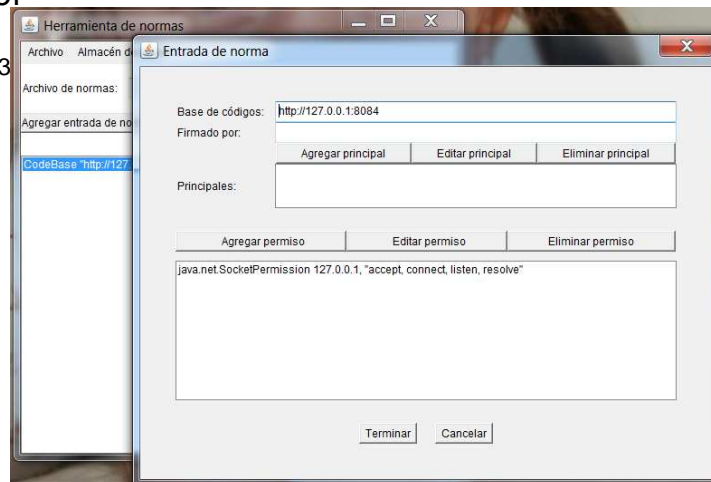
68



Applets (seguridad)

- Dar permisos a applets con la herramienta policytool

CLIENTE (opción 3
(no usar
certificados)



3.4.2. Java Web Start

Simplifica el despliegue de aplicaciones Java con applets (se transforman en una aplicación)

- Java Web Start automatiza el proceso de actualización de versiones de applets sin necesidad de cargar el applet de cada vez
- En cuanto a la seguridad, el Java Web Start preguntará al cliente para obtener los permisos necesarios para descargar la aplicación de la web
- Los ficheros que se cargan tienen extensión .jnlp



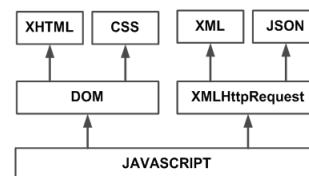
Java Web Start API

- Permite buscar, abrir y modificar ficheros locales al cliente (*FileOpenService*, *FileSaveService*, ...)
- Permite acceder al portapapeles del sistema
- Permite acceder a las opciones de impresión (*PrintService*)
- El servicio de persistencia (*PersistenceService*) permite almacenar datos locales aunque el entorno de ejecución sea no seguro (como las cookies)
- Puede usarse el servicio de descarga de aplicaciones (*DownloadService*, *DownloadServiceListener*) para controlar como se descarga la aplicación y como se “cachea”
- Es posible controlar como se gestionan los argumentos de una aplicación cuando hay varias instancias en ejecución (*SingleInstanceService*)

71



3.4.3. AJAX



- AJAX (Asynchronous JavaScript y XML)
 - Conjunto de técnicas que permiten a los navegadores comunicarse asincrónicamente con los servidores web
 - Descargar un contenido para ser insertado en la página actual
 - Transmitir información para almacenarse en el servidor
 - ...
 - Esta interacción se produce sin necesidad de recargar la página actual
 - Está en la línea de los objetivos Web 2.0

72



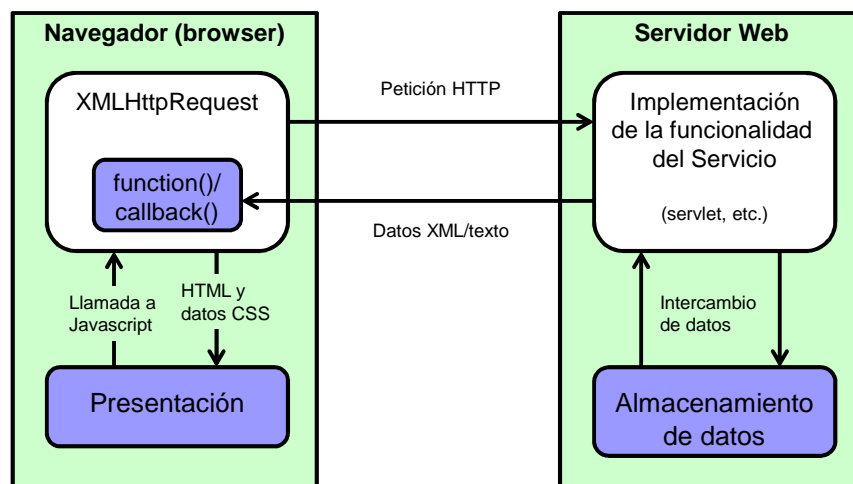
El objeto XMLHttpRequest

- La clave de AJAX es el objeto XMLHttpRequest
- El objeto XMLHttpRequest es capaz de transmitir una petición HTTP de forma asíncrona (no se recarga la página)
- Se puede considerar un cliente HTTP ejecutándose en segundo plano (dentro del módulo de procesamiento Javascript)
- Tecnologías alternativas
 - Flex (descendiente de Flash)
 - OpenLaszlo (también descendiente de Flash pero open source)

73



Flujo de comunicación



74



Métodos (Funciones)

XMLHttpRequest

Método	Descripción
<code>open(method, URL[, async])</code>	Abre una conexión con esa URL mediante un método (GET o POST) con el valor opcional <code>async</code> : "true" (petición asíncrona) "false" (petición síncrona)
<code>send(contenido)</code>	Envía datos a través de una conexión establecida
<code>abort()</code>	Cancela la petición en curso
<code>getAllResponseHeaders()</code>	Devuelve todas las cabeceras de la respuesta HTTP como una cadena
<code>getResponseHeader(headName)</code>	Devuelve el valor de una cabecera
<code>setRequestHeader(headName, value)</code>	Establece el valor de una cabecera de la petición HTTP

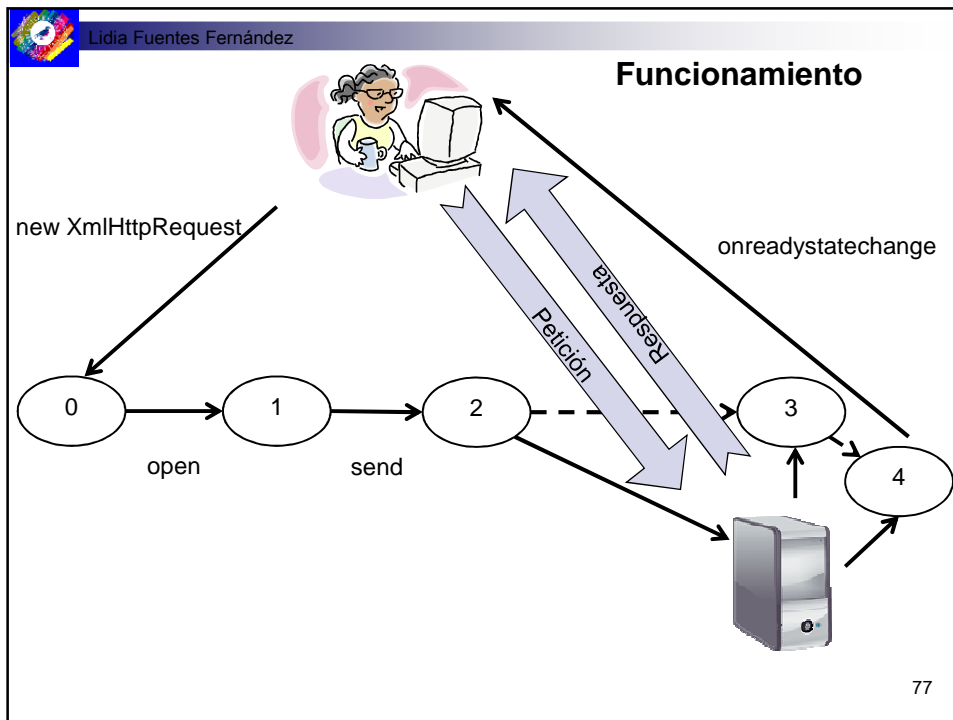
75



Atributos (Propiedades)

XMLHttpRequest

	Atributo	Descripción
Estado de la petición	<code>onreadystatechange</code>	Función que se ejecuta al finalizar una petición asíncrona
	<code>readyState</code>	Estado de la petición, puede valer desde 0 hasta 4 0: no inicializada. 1: conexión abierta. 2: petición enviada. 3: recibiendo datos. 4: petición completada y respuesta lista.
Control de la operación respuesta	<code>responseText</code>	Datos devueltos por el servidor en formato cadena.
	<code>responseXML</code>	Datos devueltos por el servidor en forma de documento XML
	<code>status</code>	Código de respuesta HTTP enviado por el servidor, del tipo 404 (documento no encontrado) o 200 (OK).
	<code>statusText</code>	Mensaje de respuesta HTTP enviado por el servidor junto al código (status), para el caso de código 200 contendrá "OK"



Lidia Fuentes Fernández

Implementación AJAX

- Creación del objeto


```

if (window.XMLHttpRequest) { // Mozilla/Chrome
    hrq = new XMLHttpRequest();
}
else if (window.ActiveXObject) { // Internet Explorer
    hrq = new ActiveXObject("Microsoft.XMLHTTP");
}
      
```
- Petición de información (HTTP Request)


```

hrq.open("POST", "http://url/", true);
hrq.setRequestHeader('Content-Type', 'application/
                               x-www-form-urlencoded');

hrq.send("nombre=DST");
hrq.onreadystatechange = function() { // callback
    if (hrq.readyState == 4) {
        if (hrq.status == 200) {
            // Proceso datos
        } else { alert(hrq.statusText); }
    }
};
      
```



Ejemplo "SecretWord"

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" src="javascript4ajax.js"></script>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ejemplo AJAX</title>
  </head>
  <body>
    <h1>Ejemplo sencillo con AJAX</h1>
    <form name="f1">
      <p>Palabra secreta: <input name="word" type="password">
        <input value="Adivinar" type="button"
          onclick='JavaScript:xmlhttpPost("SecretWord")'></p>
      <div id="result">Zona donde se visualiza el resultado AJAX</div>
    </form>
  </body>
</html>
```

79



```
function xmlhttpPost(strURL) {
  var hrq = false;
  if (window.XMLHttpRequest) { // Mozilla/Chrome
    hrq = new XMLHttpRequest();
  }
  else if (window.ActiveXObject) { // Internet Explorer
    hrq = new ActiveXObject("Microsoft.XMLHTTP");
  }
  hrq.open('POST', strURL, true);
  hrq.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
  hrq.onreadystatechange = function() { // Callback
    if (hrq.readyState == 4) { // Solo una comprobacion
      updatepage(hrq.responseText);
    }
  }
  hrq.send(getquerystring());
}

function getquerystring() {
  var form = document.forms['f1'];
  var word = form.word.value;
  var qstr = 'w=' + escape(word); // NOTA: no '?'
  return qstr;
}

function updatepage(str){
  document.getElementById("result").innerHTML = str;
}
```

80

Lidia Fuentes Fernández

```

function xmlhttpPost(strURL) {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("POST", strURL, true);
    xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == XMLHttpRequest.DONE) {
            // XMLHttpRequest.DONE == 4
            if (xmlhttp.status != 0) {
                alert("Error: " + xmlhttp.status);
            } else {
                // xmlhttp.status == 200
                document.getElementById("p1").innerHTML += xmlhttp.responseText;
            }
        }
    };
    xmlhttp.send("palabra=" + document.getElementById("p2").value);
}

function updateWord() {
    var f = document.getElementById("form");
    var w = document.getElementById("word");
    w.value = f.elements["palabra"].value;
    return true;
}

function updateWord() {
    document.getElementById("p1").innerHTML = "La palabra secreta es " + document.getElementById("word").value;
}

```

Ejemplo sencillo con AJAX

Palabra secreta:

Servlet SecretWord at /MyAjaxApp

La palabra secreta es aãkldkcnoe

81

Lidia Fuentes Fernández

Servlet SecretWord

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    String secretword=request.getParameter("w");

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet SecretWord</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet SecretWord at " + request.getContextPath() + "</h1>");
        out.println("<p>La palabra secreta es <b> " + secretword + "</b></h2>");

        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}

```

82



Bibliotecas Ajax

- Existen muchas bibliotecas y frameworks de funciones javascript y AJAX
 - Ocultan el uso del objeto `xmlhttpPost`
 - Facilitan la programación con javascript AJAX y aseguran la compatibilidad con los navegadores más modernos
 - Las peticiones se hacen mediante la invocación del método `Ajax.Request`
 - Ejemplo: `prototype.js`
 - Descargar de <http://prototypejs.org/>

83



```
<html>
  <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript">
      window.onload = function() {
        $("userName").onblur = function() {
          new Ajax.Request("checkUserName?userName=" + $("userName").value,
            { onSuccess : function(transport) {
              if (transport.responseText == "present") {
                $('message').innerHTML = "Choose another name!"
              }
              //responseText == absent
            else {
              $('message').innerHTML = "User name accepted!"
            }
          }}});
        }
      }
    </script>
  </head>
  <body>
    <form>
      <input id="userName" type="text"/>
      <input type="submit" value="Enter Name" />
      <div id="message"></div>
    </form>
  </body>
</html>
```

84

Lidia Fuentes Fernández

Servlet checkUserName

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    String loginName = request.getParameter("userName");

    response.setContentType("text/text;charset=UTF-8");
    PrintWriter out = response.getWriter();

    if ("Lidia".equalsIgnoreCase(loginName)) {
        out.print("present");
    }
    else
        out.print("absent");

    out.close();
}

```

Lidia Fuentes Fernández

AJAX vs servicios Web

- Es más rápido (no hay necesidad de recargar la página para visualizar un nuevo contenido)
- Limitaciones
 - Si la conexión con el servidor es lenta, las ventajas de AJAX disminuyen (lo de menos sería descargar una página web)
 - Si se carga mucho el navegador con muchas peticiones “pesadas”, las ventajas de AJAX también disminuyen
 - El patrón de diseño MVC es más difícil de implementar solamente con AJAX
 - No funciona si el usuario tiene desactivado el JavaScript en su navegador.
 - Una petición AJAX solamente puede enviarse al servidor Web que ha originado la página Web
 - Solución: usar servicios proxy

86



Usabilidad: desventajas

- Se pierde el concepto de volver a la página anterior.
- Si se guarda en favoritos no necesariamente al visitar nuevamente el sitio se ubique donde nos encontrábamos al grabarla.
- La existencia de páginas con AJAX y otras sin esta tecnología hace que el usuario se desoriente.

87



Frameworks front-end avanzados

- Existe una gran cantidad de frameworks que ofrecen bibliotecas javascript con funcionalidades similares
 - Ver en TodoMVC
- Su uso depende de las modas y es muy cambiante
 - Difícil migrar de un framework a otro

