Tema 4

Librerías Javascript. jQuery

jQuery
write less, do more.

Librerías Javascript.
jQuery

1.

Introducción a las librerías Javascript

Qué son y por qué usarlas

Problemas con Javascript

- "Carencias" del lenguaje
  - Falta de mecanismo de gestión de dependencias y paquetes
  - No existencia de clases
- Incompatibilidades entre navegadores
  - Generalmente con Explorer, aunque los demás también presentan ligeras diferencias
- Productividad
  - El DOM es un estándar ampliamente soportado y muy potente, pero tedioso de usar

Librerías Javascript

- Ventajas:
  - Permiten una mayor productividad, simplificando las tareas más comunes
    - Trabajo con DOM
    - AJAX
    - Animaciones
    - Widgets o componentes gráficos
  - "Amplían" el propio lenguaje Javascript
    - POO basada en clases (p.ej. prototype.js)
  - Encapsulan las incompatibilidades entre navegadores
- Inconvenientes:
  - Introducimos en nuestro código dependencias de APIs no estándar
- Muchas veces se habla de ***frameworks*** para recalcar la idea de que además aportan una filosofía o un estilo de programación propio

Ejemplo

- AJAX en prototype.js

```
new Ajax.Request('/una_url',
  {
    method:'get',
       //podemos pasar parámetros HTTP de manera sencilla.
       parameters: {param1: 'valor1', param2: 'valor2'},
       //El parámetro 'transport' no es ni más ni menos que el XMLHttpRequest.
       //Si la respuesta del servidor es de tipo JSON, automáticamente se llama a
       //eval() y se obtiene el resultado en el segundo parámetro
    onSuccess: function(transport,json){
          if (json)
             alert("me han enviado un objeto con JSON!!);
          else
             alert("La respuesta del servidor es: " + transport.responseText);
    },
    onFailure: function(){ alert('Error') }
  });
```

Algunas librerías

- http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks
- **"Todoterreno"**
  - Prototype (http://prototype.js.org)
  - jQuery (http://jquery.com)
  - MooTools (http://mootools.net)
  - Closure (http://code.google.com/intl/es-ES/closure/)
  - ...
- **Widgets**:
  - DOJO (http://dojotoolkit.org)
  - ExtJS (http://www.sencha.com/products/js/)
  - YUI (http://developer.yahoo.com/yui/2/)
  - jQuery UI (http://jqueryui.com)
- **Animaciones**
  - Scriptaculous (http://script.aculo.us)
  - jQuery UI

Frameworks avanzados

- Intentan hacer que la programación de aplicaciones web sea como la de aplicaciones de escritorio
- Algunos ejemplos
  - **GWT** - Google Web Toolkit: un poco peculiar, ya que se programa en Java y el resultado se traduce automáticamente a Javascript
  - **Capuccino**
  - **SproutCore** (Apple): *framework* tras muchas aplicaciones web de Apple (p.ej. iCloud.com)

2

Introducción a jQuery

Conceptos básicos de uso

¿Qué es jQuery?

- Librería javascript que simplifica tareas comunes y proporciona una capa de compatibilidad entre navegadores.
- Facilidades
  - Manejo avanzado del DOM
  - Eventos
  - Efectos visuales
  - AJAX
  - Widgets (jQuery UI)
  - Es extensible mediante plugins, de los cuales hay una gran cantidad: http://plugins.jquery.com/

Incluir jQuery en nuestra web

- jQuery es físicamente un único fichero .js. Basta con bajarlo y referenciarlo en todos nuestros HTML

```
<script type="text/javascript" src="js/jquery-1.4.3.min.js"/>
```

- Se puede descargar de jquery.com
  - Versión *minified*: código fuente comprimido y sin tabular para reducir el tamaño. Recomendable en producción
  - Versión *uncompressed*: código tabulado, recomendable para depurar.
- También está disponible on-line. Así si el usuario ya ha navegado por otro sitio que la usa, se tomará de cache

```
<script type="text/javascript"
        src="http://code.jquery.com/jquery-1.4.3.min.js"/>
```

La función $

- A la mayoría de funcionalidades de jQuery se accede a través de una única función: jQuery o $ (son equivalentes)
  - Al "exponer" una única función se reduce el riesgo de colisiones con nuestras propias funciones
  - $ es un nombre válido en Javascript para una función y tiene la ventaja de ser corto y fácil de recordar
- Estructura típica de una sentencia jQuery: selector/comando/parámetros

```
//Al poner el # estamos seleccionando el nodo por su id
//el comando css nos permite cambiar propiedades CSS
$('#parrafo1').css('color','red')

//por cierto, el equivalente en DOM estándar sería
document.getElementById("parrafo1").style.color = "red"
```

Encadenar comandos

- La mayoría de comandos devuelven el resultado de aplicarlos, por lo que se pueden encadenar. De hecho, es el estilo típico de escribir código jQuery

```
//addClass coloca un atributo class en el nodo
//show lo muestra si estaba oculto, con una animación
$("#parrafo1").addClass("destacado").show("slow")

//es muy común ver el código así:
$("#parrafo1")
  .addClass("destacado")
  .show("slow")
```

Librerías Javascript.
jQuery

3

Selectores

Seleccionando nodos y ejecutando operaciones básicas sobre ellos

Selectores jQuery

- Se usa la sintaxis de CSS3 para seleccionar nodos
  - La selección la implementa jQuery, así que aunque nuestro navegador no implemente CSS3, funcionará
  - No vamos a ver la sintaxis exhaustivamente, solo unos cuantos ejemplos
- Sintaxis básica: CSS1

```
//ejemplos
//todos los párrafos
$('p')
//todos los párrafos con class="destacado"
$('p.destacado')
//todas las listas, ordenadas o con viñetas
$('ol, ul')
//todas las etiquetas. Cuidado: este código es muy lento
$('*')
```

¿Qué devuelve un selector jQuery?

- No devuelve directamente los nodos del DOM, sino una clase propia (un "wrapped set" de elementos)
  - Podemos usar los métodos de jQuery directamente sobre estos "nodos"

```
//'css' cambia el valor de una propiedad CSS
$('#pie').css('color','gray')
```

  - Podemos acceder a los nodos "reales" del DOM por posición, bien con notación de array o bien con el método get()

```
//cambiar el href del primer enlace de la página
//Como queremos usar el API DOM tenemos que acceder al nodo
$('a')[0].href = 'http://www.ua.es'
//idem, con get(). Podemos saber el tamaño con size()
$('a').get(0).href = 'http://www.ua.es'
alert('Hay ' + $('a').size() + ' enlaces')
```

Bucles implícitos y explícitos

- Implícito
  - Cuando un selector engloba a varios nodos, el comando jQuery implícitamente hace un bucle por todos los nodos

```
//Esto ocultaría TODOS los párrafos
$('p').hide()
```

- Explícito
  - Podemos movernos por el conjunto con **each**(), pasándole una función a ejecutar para cada elemento. La función recibirá automáticamente como parámetro el índice del elemento actual dentro del conjunto. Con **this** podemos acceder al elemento.

```
$('a').each(function(i){
        alert("el enlace nº " + i + " apunta a " + this.href)
})
```

Selectores jQuery: atributos

- Tomado de CSS2, que permite filtrar la selección por el valor de los atributos. jQuery amplía la sintaxis

```
//Ejemplos
//todos los campos de password
$('input[type=”password”]')
//todos los campos que no sean de password
$('input[type!=”password”]')
//todos los enlaces cuyo href comience por http://
$('a[href^=”http://”]')
//todos los enlaces cuyo href termine por .com
$('a[href$=”.com”]')
//todas las imágenes cuya URL contiene la palabra jquery
$('img[src*=”jquery”]')
//comprobar si existe un atributo, el valor es indiferente
$(‘form[encoding]’)
```

Selectores jQuery: posición

- Permiten seleccionar nodos por su posición en el árbol o en relación a otros nodos
- Algunos son de CSS3, otros propios de jQuery

```
//el primer div. El último sería :last
$('div:first')
//todas las filas de tabla impares. Para las pares, :even
$('tr:odd')
//el último li dentro de las listas ordenadas
//(si hay varios <ol>, esto se cumplirá para cada uno de ellos)
$('ol li:last-child')
//las filas de tabla múltiplo de 3: la 3, 6, 9, 12..
$('tr:nth-child(3n)')
//los elementos de lista impares (pero solo los de listas <ol>)
$('ol li:nth-child(odd)')
//los elementos de lista más allá del quinto (sin incluirlo)
//para menor, usar :lt, y para igual, :eq
$('li:gt(5)')
```

Selectores jQuery: otros filtros

- Algunos son de CSS3, otros propios de jQuery

```
//De todos los botones de radio quedarnos con los marcados (CSS3)
$('input[type=”radio”]:checked')
//$('input[type=”radio][checked]') no nos sirve porque
//seleccionaría los marcados inicialmente, pero no AHORA
//campos de formulario deshabilitados (CSS3, también existe :enabled)
$('input:disabled')
//cualquier cabecera: h1,... h6
$(':header')
//podemos seleccionar solo un tipo de campo
(:checkbox, :radio, :reset, :submit, :text, :password, :file) o todos (:input)
Por ejemplo, $(':password') y
$('input[type=”password”]') serían lo mismo
//podemos combinar todo esto
$(':checkbox:checked:enabled')
//el filtro :not es la negación. Campos NO checkbox (CSS3)
$('input:not(:checkbox)')
```

Librerías Javascript. jQuery

4

Navegación y manipulación del DOM

Simplificando el API del DOM estándar

Navegar por el árbol

- Recordar que en el DOM estándar, dado un nodo podemos acceder a sus "parientes cercanos" (parentNode, children, firstChild, siblings,...)
  - En jQuery podemos hacer lo mismo con un conjunto de nodos resultante de aplicar un selector

```
//devuelve los hijos de cada uno de los nodos seleccionados (todos juntos)
//o sea, tendríamos todos los <li> de todas las listas no ordenadas
$('ul').children()
//devuelve todos los <div> que contienen párrafos
$('p').parent('div')
//devuelve el "ancestro" más inmediato
//averiguamos fácilmente qué <div> contiene a la imagen
$('#imagen').closest('div')
```

Acceso básico al DOM

- **Añadir/eliminar clases**: addClass(nom), removeClass(nom)
  - Un elemento puede tener varias clases separadas por espacios. addClass añade la clase, no elimina la actual

```
//Todos los párrafos pasan a tener la class 'destacado'
$('p').addClass('destacado')
```

- **Cambiar CSS**: css(prop, val). No importa si el estilo actual está en un atributo style, en la cabecera del HTML o en un archivo CSS aparte

```
//Todos los párrafos pasan a tener color rojo
$('p').css('color','red')
```

  - En lugar de un valor concreto se le puede pasar una función que cambie el valor a partir del actual (que recibirá como parámetro)

```
//el primer parámetro es el índice del elemento en el conj.
//el segundo es el valor actual de la propiedad CSS
$('a').css('width',function(i,actual) {
        return parseInt(actual) + 1 + "px"
    })
```

Manipulación del DOM

- Modificar el contenido de un elemento

```
//Esto es como cambiar el innerHTML
$(#cabecera).html("<h1>hola</h1>")
//También se puede simplemente leer el valor actual
var cab = $(#cabecera).html()
//también podemos acceder al texto, sin etiquetas
//dado <p id="par">Hola <b>amigos <em>míos</em></b></p>
alert($(#par).text()) //Hola amigos míos
```

Manipulación del DOM: insertar/mover

- Para **añadir al contenido de un nodo**: al final: append(), al comienzo: prepend()
  - Si el argumento es un selector, estamos moviendo los nodos afectados

```
//Al final de #cabecera aparecería el nuevo contenido,
//pero el antiguo no se borraría
$('#cabecera').append('<h1>hola</h1>')
//Esto MOVERÍA el elemento con id=”saludo” al final de cabecera
$('#cabecera').append($('#saludo'))
```

- Para **añadir al mismo nivel** que un nodo: before(), after()

```
//Al final de la lista con id=”lista” aparecería un nuevo <li>
$('#lista li:last').after('<li>Ahora soy el último</li>')
```

- appendTo(), prependTo(), insertBefore() e insertAfter() son idénticos pero primero se pone el nuevo nodo y como argumento el selector del sitio donde insertar/mover

Manipulación del DOM: eliminar,copiar

- remove([selector]), empty()

```
//Elimina todos los párrafos
$('p').remove()
//Elimina los párrafos pero solo los de la class 'nota'
$('p').remove('.nota')
//Esto borraría el contenido de los párrafos
//(pero los propios <p> seguirían estando)
$('p').empty()
```

- clone()

```
//Hace una copia de todas las imágenes y las mete dentro
//del elemento con id="galeria"
$('img').clone().appendTo('#galeria')
```

Creación “estructurada” de HTML

- Aparte de como selector, $ también se puede usar para crear fragmentos de HTML
  - Si se le pasa un solo parámetro, es el HTML en forma de cadena

```
$('<input type="button" value="clicame">').appendTo('body')
```

  - Se le puede pasar otro parámetro con los atributos en formato JSON

```
$('<input>',
   { type:'button',
     value:'Clicame',
     click:function() {
              alert('¡hola!')
           }
   })
   .appendTo('body')
```

Librerías Javascript.
jQuery

5.

Eventos

¡Código compatible con IE sin esfuerzo!

Gestión de eventos

- jQuery tiene un modelo de eventos parecido al DOM2 Events, pero al ser propio proporciona compatibilidad con Explorer
  - Para un elemento y un evento podemos tener varios *listeners*, en contraste con un solo *handler* (modo "tradicional"). Los eventos solo suben (en DOM2 primero bajan y luego suben)
  - Añadir listener: **bind**(evento,listener)

```
$('#boton').bind('click', function() {alert('hola')})
        function quetal() {alert('que tal')}
         $('#boton').bind('click', quetal)
```

  - Eliminar listener: **unbind**(evento,[listener])

```
//Elimina todos los listeners para el evento click
$('#boton').unbind('click'})
//Para poder eliminar un listener tenemos que haberle
//dado nombre a la función
$('#boton').unbind('click', saludo)
```

Información sobre el evento

- Igual que en DOM2 Events, el listener recibe automáticamente un objeto con las propiedades del evento (¡ahora también en IE!)
  - Las propiedades del evento son propias de jQuery. Consultad la documentación de jQuery para la referencia

```
$('#imagen').bind('click',
   function(e) {
    alert('click en ' + e.pageX + "," + e.pageY)
   })
```

  - Para prevenir acción por defecto: evento.preventDefault(). Para parar la propagación "arriba" evento.stopPropagation(). Si devolvemos false, estamos haciendo ambos

```
//si el campo con id="texto" está vacío, anular el envío del formulario
//con id="formu"
$('#formu').bind('submit', function(e) {
   if ($('#texto').val()==""))
        e.preventDefault()
   })
```

Algunas cosas más sobre eventos

- En lugar de bind **se puede usar el nombre específico del evento** para definir el listener. Así el código queda más claro y es más corto

```
$('#boton').click(function() {alert("click!")})
```

- Para **disparar un evento** se puede usar trigger(evento)
  - La mayoría de propiedades del evento estarán vacías, ya que no es un evento "de verdad"

```
//esto es como si hubiéramos hecho click sobre el botón
$('#boton').trigger('click')
```

live()

- Definir un manejador de evento para todos los nodos actuales **y futuros** que encajen con un selector
  - **bind** solo contempla los actuales, pero no los añadidos dinámicamente tras su definición

```
$('input[type=”button”]').live(‘click’,function() {alert(“hola”)})
<div id=”miDiv”></div>
//con el “bind” normal este botón no tendría listener para click
$('<input>',
   { type:'button',
     value=‘Clica aquí’})
.appendTo($(‘#miDiv’))
```

Ejecutar código cuando la página esté lista

- Evento "**ready**": el DOM está listo para ser manipulado
  - El equivalente estándar ("onload" del body) se espera a que se carguen imágenes y otros

```
$(document).ready( function() { alert('El DOM se ha cargado') })
```

- Hay un equivalente al "onload" que es **load**(). Se puede usar no solo con la página completa (aquí window) sino también con imágenes, por ejemplo

```
<img id="imagen" src="una_imagen.jpg">
<script>
   $('#imagen').load( function() { alert('Se ha cargado la imagen') })
   $(window).load(function() { alert('Este es el equivalente al
                                      body="onload"')}
</script>
```

Librerías Javascript.
jQuery

6.

Efectos y animaciones

Animaciones

- Mostrar (aumentando tamaño y opacidad)/ocultar: show(), hide(), toggle(). Admiten como parámetro un nº de milisegundos o bien 'slow', 'normal', 'fast'. Sin parámetro no hay animación gradual

```
$('#boton')
  .click(function() {
            $('#formulario').toggle('slow')
})
```

- Otras animaciones: fadeIn, fadeOut (cambia solo opacidad, no tamaño), slideUp, slideDown (solo tamaño)
- jQuery UI añade algunas animaciones adicionales

Animaciones personalizadas

- **Se puede animar cualquier propiedad CSS**. Se especifica el valor final para la propiedad, y jQuery la interpola (como en las transiciones de CSS3). En jQuery deben ser propiedades numéricas

```
$('#imag')
      //si el elemento no está posicionado no podemos
      //cambiar sus coordenadas
      .css('position', 'relative')
      .animate(
            //propiedades a animar, en formato JSON
            {opacity:0, left:300},
            //tiempo en ms o 'fast', 'normal' o 'slow'
            'slow',
            //"easing", o "ritmo" al que se hace la transición
            'swing', //puede ser 'linear' o 'swing'
            //función que se llama cuando termina la animación
            function() {$('#imag').remove()}
 )
```

Librerías Javascript.
jQuery

7.

AJAX

Simplificando la petición AJAX y el procesamiento de la respuesta

AJAX

- **$.ajax** permite hacer una petición AJAX con una sintaxis más amigable y concisa que la del XMLHttpRequest nativo

```
$.ajax({
 type: 'GET',  //por defecto es GET, por tanto esto es superfluo
 url: 'getCotizacion.php',
 //Si el AJAX es asíncrono. Por defecto se asume 'true'
 async: false,
 //parámetros de la petición
 data: {empresa: 'AAPL', moneda:'eur' },
 //tipo de datos que esperamos del servidor: 'xml'/'html'/'json'..
 dataType: 'text',
 //callback en caso de éxito
 success: function(respuesta) {
      alert('El servidor me dice: ' + respuesta)
 },
 //callback en caso de error
 error: function() {
      alert('Se ha producido un error');
 }
});
```

Tipo de datos esperado del servidor

- Si sabemos que nos envían JSON podemos usar **$.getJSON**(url, params, callback)
  - Al *callback* se le llama con el JSON ya evaluado con un *parser* JSON

```
$.getJSON(
      'getCotizacion.php',
      {empresa: 'AAPL', moneda:'eur' },
      function(obj) {
            //'obj' es el resultado de eval() de la resp. del servidor
            ...
      });
```

- Si sabemos que nos envían HTML podemos usar **load()** para cargar la respuesta en algún punto del DOM

```
$('#resultado').load(
      'getCotizacionHTML.php',
      {empresa: 'AAPL', moneda:'eur' }
 )
```

Tomar parámetros de un formulario

- Si los parámetros de la petición AJAX se corresponden con los campos de un formulario, no es necesario que los pasemos "a mano", uno a uno. **serialize**() lo hace por nosotros

```
$.ajax({
 type: 'GET',
 url: 'getCotizacion.php',
 //parámetros de la petición: los campos del formulario id="formu"
 data: $('#formu').serialize(),
 //tipo de datos que esperamos del servidor
 dataType: 'json',
 //callback en caso de éxito
 success: function(obj) {
      //recordar que obj es el result. de eval() del JSON recibido
 }
})
```

Referencias

- El sitio web de jQuery incluye una extensa documentación y muchos tutoriales, algunos en castellano.
- Libros recomendados disponibles en Safari O'Reilly (http://proquestcombo.safaribooksonline.com)
  - ***jQuery in Action, 2nd ed***., Bear Bibeault; Yehuda Katz. Ed. Manning. *Exhaustivo y bastante detallado. Se puede usar como tutorial o como referencia*
  - ***jQuery Novice to Ninja***, Earle Castledine; Craig Sharkie, SitePoint. *Va desarrollando un ejemplo de aplicación desde el principio, incorporando jQuery progresivamente*