JQuery

JQuery

- Es la librería JavaScript que ha irrumpido con más fuerza como alternativa a Prototype.
- También ha sido programada de forma muy eficiente y su versión comprimida apenas ocupa 20 KB.
- La documentación de jQuery es muy completa e incluye muchos ejemplos.
  http://docs.jquery.com/

JQuery

- **Funciones y métodos básicos**
  - La función dollar ($()). Es más que un simple atajo mejorado de la función **document.getElementById**().
  - La cadena que se pasa como parámetro puede hacer uso de Xpath o de CSS. Además, separando expresiones con un carácter "," se puede seleccionar un número ilimitado de elementos.

JQuery: Selectores

- Algunos ejemplos de expresiones

```
// Selecciona todos los div de la página que tengan al menos un parrafo
$(‘div[p]')

// Selecciona el elemento cuyo id sea “header"
$('#header')

// Selecciona por atributo
$('input[name=“nombre”]')

// Selecciona de elemento por nombre de clase CSS
$(‘span.classCSS');

//Selección anidada CSS
$(‘seccion#principal div.resaltado a');
```

JQuery: Pseudo-Selectores

- :**button** - Selecciona elementos <button> o <input> con el atributo type='button'
- :**checkbox** - Selecciona elementos <input> con el atributo type='checkbox'
- :**checked** - Selecciona elementos <input> con el atributo type='checkbox‘, que estén marcados.
- :**disabled** - Selecciona elementos que están deshabitados
- :**enabled** - Selecciona elementos que están habilitados

JQuery: Pseudo-Selectores

- :**file** - Selecciona elementos <input> con el atributo type='file'
- :**image** - Selecciona elementos <input> con el atributo type='image'
- :**input** - Selecciona elementos <input>, <textarea> y <select>
- :**password** - Selecciona elementos <input> con el atributo type='password'

JQuery: Pseudo-Selectores

- :**radio** - Selecciona elementos <input> con el atributo type='radio'
- :**reset** - Selecciona elementos <input> con el atributo type='reset'
- :**selected** - Selecciona elementos <options> que están seleccionados
- :**submit** - Selecciona elementos <input> con el atributo type='submit'
- :**text** - Selecciona elementos <input> con el atributo type='text'

JQuery: Pseudo-Selectores

- :**first** – Selecciona el primer elemento hijo
- **:odd** – Selecciona los elementos hijos que estén en posición impar
- **:gt[num]** – Selecciona los elementos en posiciones mayores que **num**.
- **:animated** – Selecciona los elementos animados

JQuery: Pseudo-Selectores

- Algunos ejemplos de aplicación de los psudo-selectores

```
// obtiene todos los elementos inputs dentro del formulario #myForm
$('#myForm :input');

// selecciona el primer elemento <a> con la clase 'external'
$('a.external:first');

// selecciona todos los elementos <tr> impares de una tabla
$('tr:odd');

// selecciona todos los divs visibles
$('div:visible');

 // selecciona todos los divs excepto los tres primeros
$('div:gt(2)');
```

JQuery: DOM Selección

- Una vez se han seleccionado los nodos del arbol DOM, estos se tendrán que procesar
- Para ello jQuery permite una vez seleccionados los elementos con los selectores, aplicarles funciones de **filtrado** y **búsqueda**.
  - **contains**() limita los elementos seleccionados a aquellos que contengan en su interior el texto indicado como parámetro

```
// Sólo obtiene los párrafos que contengan la palabra 'importante'
$('p').contains('importante');
```

JQuery: DOM Selección

- **not**() elimina de la selección de elementos aquellos que cumplan con el selector indicado

```
// Selecciona todos los enlaces de la página, salvo el que tiene una 'class' igual a 'especial'
$('a').not('.especial');

// La siguiente instrucción es equivalente a la anterior
$('a').not($('.especial'));
```

- **filter**() es la inversa de **not**(), ya que elimina de la selección de elementos aquellos que no cumplan con la expresión indicada.

JQuery: DOM Selección

- **end**(), permite volver a la selección original de elementos después de realizar un filtrado de elementos.

```
$('a').filter('.pinchame').click(function(){
        alert('Estás abandonando este sitio web');
}).end().filter('ocultame').click(function(){
        $(this).hide();
        return false;
}).end();
```

- El ejemplo obtiene todos los enlaces de la página $('a') y aplica diferentes funciones manejadoras del evento click en función del tipo de enlace.

JQuery: DOM Selección

- **html**(), permite tanto definir el contenido HTML del nodo seleccionado, como seleccionar el contenido HTML del nodo seleccionado.

```
$('h1').html('hello world');
$('h1').html();
```

- **eq**() permite seleccionar un elemento de un Array por su posición en el Array.

```
$('ul li').eq(5); // el sexto item de una lista desordenada
```

- **first**(), selecciona el primer nodo que coincida con la selección.

JQuery: DOM Selección

- **parent**() permite seleccionar el elemento contenedor.

```
// seleccionar el contenedor de un div en estado visible
$('div:visible').parent();
```

- **children**(), permite obtener los hijos del elemento definido.

```
// seleccionar todos los elementos hijos de #myList
$('#myList').children();
```

- **siblings**() permite seleccionar los hermanos.

```
// seleccionar todos los items hermanos del elemento <li>
$('li.selected').siblings();
```

JQuery: DOM Iteración

- También se proporcionan funciones para recorrer los elementos seleccionados.
  - **next**(), permite obtener el siguiente elemento definido.

```
// seleccionar el elemento párrafo mas próximo, a H1
$('h1').next('p');
```

JQuery: DOM Iteración

- **each**(), permite recorrer arrays y objetos.
- El primer parámetro es el objeto que se quiere recorrer y el segundo es el código de la función que lo recorre (a esta función se le pasa el índice y el valor):

```
// Recorrer arrays y objetos
var variable = ['a', 'e', 'i', 'o', 'u'];
//var variable = { id: '12DW2', precio: 12.34, cantidad: 5 };

$.each(variable , function(i, n){
		alert('Vocal número ' + i + " = " + n);
});
```

JQuery: DOM Manipulación

- Existen funciones, que permiten mover, insertar y borrar nodos de un árbol DOM.
  - **appendTo**() y **append**(), permiten mover un elemento a otro lugar dentro del mismo u otro árbol DOM.

```
// hacer que el primer item de la lista sea el último
var $li = $('#myList li:first').appendTo('#myList');

// otro enfoque para el mismo problema
$('#myList').append($('#myList li:first'));
```

JQuery: DOM Manipulación

- **clone**(), permite crear una copia de un nodo existente.

```
// copiar el primer elemento de la lista y moverlo al final de la misma
$('#myList li:first').clone().appendTo('#myList');
```

- **remove**(), permite borrar por completo un nodo existente.
- **detach**(), permite eliminar el elemento de su posición, pero mantiene el elemento por si se necesita reinsertar.
- **empty**(), vacía un nodo.

JQuery: DOM Manipulación

- Se pueden crear nuevos elementos, de una forma sencilla, gracias al selector $().

```
$('<p>Un nuevo párrafo</p>');
$('<a/>', {
        html : 'Un <strong>nuevo</strong> enlace',
        'class' : 'new',
        href : 'foo.html'
});
```

JQuery: CSS

- Ofrece la posibilidad de obtener y asignar propiedades de CSS de forma sencilla.

```
// devuelve una cadena de caracteres con el tamaño del texto (Ej: "19px“)
$('h1').css('fontSize');

// también funciona
$('h1').css('font-size');

// establece una propiedad individual CSS
$('h1').css('fontSize', '100px');

// establece múltiples propiedades CSS
$('h1').css({
        'fontSize' : '100px',
        'color' : 'red'
});
```

JQuery: CSS

- Aunque se permita la opción de establecer propiedades de estilos de forma directa, habrá que evitarlo y acudir a clases CSS.

```
var $h1 = $('h1');

//Añadir una clase CSS big, a los nodos H1
$h1.addClass('big');

//Borrar una clase CSS big, de los nodos H1
$h1.removeClass('big');

//Cambiar la clase CSS de los nodos H1, por big
$h1.toggleClass('big');

//Comprobar si los nodos H1 tienen la clase CSS big
if ($h1.hasClass('big')) { ... }
```

JQuery: Atributos

- Se pueden obtener y establecer los valores de los atributos de las etiquetas.

```
//Lectura de un atributo
$('a').attr('href‘);

//Establecimiento de un atributo
$('a').attr('href', 'allMyHrefsAreTheSameNow.html');

//Establecimiento de multiples atributos
$('a').attr({
        'title' : 'all titles are the same too',
        'href' : 'somethingNew.html'
});
```

JQuery: Eventos

- Ofrece la posibilidad de acceder al árbol DOM, cuando solo los elementos HTML se han cargado, sin esperar a todo el contenido, al contrario que el evento ***onload*** de ***window***.

```
$(document).ready(function() {
...
});
```

- Las aplicaciones jQuery pueden iniciarse más rápidamente que las aplicaciones JavaScript tradicionales.

JQuery: Eventos

- Se pueden lanzar los evento de forma programática.

```
// Ejecuta el evento 'onclick' en todos los párrafos de la página
$('p').click();

// Ejecuta el evento 'mouseover' sobre un 'div' con id 'menu'
$('div#menu').mouseover();
```

JQuery: Eventos

- **toggle**(), permite ejecutar dos funciones de forma alterna cada vez que se pincha sobre un elemento:

```
$("p").toggle(function(){
        alert("Me acabas de activar");
},function(){
        alert("Me acabas de desactivar");
});
```

- Las pulsaciones (click) impares, se ejecuta la primera función y las pares se ejecuta la segunda función.

JQuery: Eventos

- **on**() y **off**() permiten asociar/desasociar eventos a componentes:

```
$('p').on({
        'click': function() {
                console.log('clickeado');
        },
        'mouseover': function() {
                console.log('sobrepasado');
        }
});
$('p').off('click');
```

JQuery: Eventos

- **bind**(), asocia a los elementos actuales el evento correspondiente, pero no a los elementos que se inserten programáticamente:

```
$(document).ready(function(){
        $("div a").bind("click",function(){
                alert('Has hecho click');
        });
});
```

JQuery: Eventos

- **live**(), asocia a los elementos actuales y futuros, que cumplan con el selector, el evento correspondiente:

```
$(document).ready(function(){
        $("div a").live("click",function(){
                alert('Has hecho click');
        });
});
```

JQuery: Eventos

- **delegate**() idéntico a **live()**, salvo, que este permite ser aplicado sobre sentencias encadenadas de JQuery

```
$(document).ready(function(){
        $("#idDiv").children("table").delegate("tr","click",function(){
                alert('Has hecho click');
        });
});
```

- En ambos dos casos, es conveniente hacer uso de **event.preventDefault**() y **event.stopInmediatePropagation**().

JQuery: Efectos

- Todas las funciones relacionadas con los efectos visuales permiten indicar dos parámetros opcionales:
  - El primero es la duración del efecto.
  - El segundo parámetro es la función que se ejecuta al finalizar el efecto visual.

```
// Oculta todos los enlaces de la página
$('a').hide();

// Muestra todos los 'div' que estaban ocultos
$('div:hidden').show();
```

JQuery: Ajax

- El método principal para realizar peticiones AJAX es **$.ajax**()
- A partir de esta función básica, se han definido otras funciones relacionadas, de más alto nivel y especializadas en tareas concretas: **$.get(), $.post(), $.load()**, etc.
- La sintaxis de **$.ajax**() es muy sencilla:

```
$.ajax(opciones);
```

JQuery: Ajax

- Dentro de opciones, se incluye toda la información necesaria, como
  - Url
  - Method HTTP
  - Parámetros.
  - Callbacks de exito y error.

```
$.ajax({
        url: '/ruta/hasta/pagina.php',
        type: 'POST',
        async: true,
        data: 'parametro1=valor1&parametro2=valor2',
        success: procesaRespuesta,
        error: muestraError
});
```

JQuery: Ajax

- Opciones para el método $.ajax()

Opción	Descripción
`async`	Indica si la petición es asíncrona. Su valor por defecto es `true`, el habitual para las peticiones AJAX
`beforeSend`	Permite indicar una función que modifique el objeto `XMLHttpRequest` antes de realizar la petición. El propio objeto `XMLHttpRequest` se pasa como único argumento de la función
`complete`	Permite establecer la función que se ejecuta cuando una petición se ha completado (y después de ejecutar, si se han establecido, las funciones de `success` o `error`). La función recibe el objeto `XMLHttpRequest` como primer parámetro y el resultado de la petición como segundo argumento
`contentType`	Indica el valor de la cabecera `Content-Type` utilizada para realizar la petición. Su valor por defecto es `application/x-www-form-urlencoded`
`data`	Información que se incluye en la petición. Se utiliza para enviar parámetros al servidor. Si es una cadena de texto, se envía tal cual, por lo que su formato debería ser `parametro1=valor1&parametro2=valor2`. También se puede indicar un array asociativo de pares clave/valor que se convierten automáticamente en una cadena tipo *query string*
`dataType`	El tipo de dato que se espera como respuesta. Si no se indica ningún valor, jQuery lo deduce a partir de las cabeceras de la respuesta. Los posibles valores son: `xml` (se devuelve un documento XML correspondiente al valor `responseXML`), `html` (devuelve directamente la respuesta del servidor mediante el valor `responseText`), `script` (se evalúa la respuesta como si fuera JavaScript y se devuelve el resultado) y `json` (se evalúa la respuesta como si fuera JSON y se devuelve el objeto JavaScript generado)

JQuery: Ajax

- Opciones para el método $.ajax()

`error`	Indica la función que se ejecuta cuando se produce un error durante la petición. Esta función recibe el objeto `XMLHttpRequest` como primer parámetro, una cadena de texto indicando el error como segundo parámetro y un objeto con la excepción producida como tercer parámetro
`ifModified`	Permite considerar como correcta la petición solamente si la respuesta recibida es diferente de la anterior respuesta. Por defecto su valor es `false`
`processData`	Indica si se transforman los datos de la opción `data` para convertirlos en una cadena de texto. Si se indica un valor de `false`, no se realiza esta transformación automática
`success`	Permite establecer la función que se ejecuta cuando una petición se ha completado de forma correcta. La función recibe como primer parámetro los datos recibidos del servidor, previamente formateados según se especifique en la opción `dataType`
`timeout`	Indica el tiempo máximo, en milisegundos, que la petición espera la respuesta del servidor antes de anular la petición
`type`	El tipo de petición que se realiza. Su valor por defecto es GET, aunque también se puede utilizar el método POST
`url`	La URL del servidor a la que se realiza la petición

JQuery: Ajax

- **$('selector').load()**, que carga en el elemento seleccionado, el resultado de la petición al servidor.
- Es idéntica a la función **Ajax.Updater()** del framework **Prototype**.

```
$('#resultado').load('resultado.html');
```

JQuery: Ajax

- **$('selector').loadIfModified**(), carga la respuesta del servidor en el elemento sólo si esa respuesta es diferente a la última recibida.

```
$('#resultado').loadIfModified('resultado.html');
```

Jquery: Navegador

- jQuery detecta automáticamente el tipo de navegador en el que se está ejecutando y permite acceder a esta información a través del objeto $.**browser**:
  - $.**browser.msie**; // 'true' para Internet Explorer
  - $.**browser.mozilla**; // 'true' para familia Firefox
  - $.**browser.opera**; // 'true' familia Opera
  - $.**browser.safari**; // 'true' familia Safari

Jquery: Utillería

- $.**trim**(), borra los espacios en blanco del principio y final.

```
$.trim('        varios espacios en blanco          ');
// devuelve 'varios espacios en blanco'
```

- $.**inArray**(), devuelve el índice de un valor en un array, o -1 si el valor no se encuentra en el array.

```
var myArray = [ 1, 2, 3, 5 ];

if ($.inArray(4, myArray) !== -1) {
        console.log('valor encontrado');
}

//En este caso no se muestra el mensaje por consola, al no estar 4 dentro del Array.
```

Fin