Empezando con jQuery

Breves anotaciones para 2º DWEB

APS-IZV
dic 2018

Empezando con jQuery

Conocimientos previos: Repasar selectores **CSS** utilizando la presentación proporcionada.

Introducción a jQuery: **codeschool_try_jquery.pdf**

jQuery es una librería JavaScript de código abierto que proporciona docenas de métodos para facilitar la programación. Las funciones jQuery funcionan en los distintos navegadores. Para incluir jQuery en tus páginas web hay que descargar un fichero con la librería de la siguiente página (puedes descargar la versión no-comprimida para estudiar el código, si es que te apetece):

www.jQuery.com

Una vez descargado el fichero se puede incluir en la página web con una línea como la siguiente:

```
<script src=" jquery-3.3.1.min.js"></script>
```

Si tenemos el fichero en nuestro equipo o en un servidor web local, podemos desarrollar aplicaciones jQuery sin tener conexión a internet. Para aplicaciones en producción tenemos que poner el fichero en el servidor web.

Otra forma de incluir la librería jQuery en tus aplicaciones web es obtener el fichero de un CDN (Content Delivery Network) que es un servidor web que aloja software open-source. Para obtener las librerías jQuery podemos usar sitios CDN como Google, Microsoft o jQuery. La ventaja es que podemos utilizar la última versión sin tener que cambiar la URL, la desventaja es que tenemos que estar conectados a internet. Así es como se incluye jQuery desde un CDN:

```
<script src="//code.jquery.com/ jquery-3.3.1.min.js"></script>
```

Para bajarlo de Google (es buena idea ya que muchos usuarios ya lo tendrán en caché):

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

O del CDN de Microsoft:

```
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>
```

Para escribir nuestro código javaScript que incluye funciones jQuery, usamos como hasta ahora un fichero externo con extensión .js que incluimos en la página html con una etiqueta <script> después de la etiqueta script anterior.

Introducción

jQuery usa selectores CSS para seleccionar uno o varios elementos HTML a los que aplicar determinados métodos (acciones sobre los elementos seleccionados). Permite escribir aplicaciones con menos líneas de código. Aquí tenéis un ejemplo:

El código .css sería el mismo en ambos casos:

```
h2 {
   background: url(./imagenes/mas.jpg) no-repeat left center;
}
h2.menos {
   background: url(./imagenes/menos.jpg) no-repeat left center;
}
div {
  display: none;
}
div.abierto {
   display: block;}
```

//javaScript

```
(function() {
  let cambiar = function(e) {
    e.currentTarget.classList.toggle("menos");
    e.currentTarget.nextElementSibling.classList.toggle("abierto");
  }
  let elementosH2 = faqs.querySelectorAll("#faqs h2");
  for (var i = 0; i < elementosH2.length; i++) {
    elementosH2[i].addEventListener('click',cambiar);
  }
}})();
```

// jQuery

```
(function(){
  var manejaEvento = function(e) {
    $(this).toggleClass('menos');
    $(this).next().toggleClass('abierto');
  }
  $("#faqs h2").on('click', manejaEvento);
})();
```

Seleccionar elementos: $("selector")

El selector $ (o jQuery) permite selecccionar elementos HTML basándonos en su id, clase, tipo, atributos, valores de atributos... Para ello se utiliza el símbolo dolar y los paréntesis: $(), o bien jquery(). El resultado es una lista de elementos "viva" que se modificará automáticamente conforme se modifique el DOM.

```
jQuery("h1");
$("h2");
jQuery("p");
$("a");
```

Seleccionar elementos por el tipo de etiqueta:

```
$("li"); // Todos los elementos <li>
```

Seleccionar elementos por ID:

```
$("#contenedor");
```

Seleccionar elementos por class:

```
$(".articulos");
$("p.intro"); // Selecciona los elementos <p> con clase 'intro'
```

Seleccionar descendientes

```
$("#destino li"); //Elementos <li> que están dentro de un elemento con ID="destino"
```

Seleccionar hijos directos (no descendientes de cualquier nivel). Padre e hijos se separan con el símbolo >

```
$("#destino > li"); //Elementos <li> que son hijos de un elemento con ID "destino"
```

Seleccionar hermanos adyacentes

```
$("h2 + div"); //Elementos <div> que son hermanos adyacentes de elementos <h2>
```

Seleccionar hermanos en general

```
$("ul ~ p");  //Elementos <p> que son hermanos de elementos <ul>
```

Seleccionar elementos con múltiples selectores (se separan con ,):

```
$(".promo, #france");
$("#faqs li, div p");
```

Ver más ejemplos de selectores al final del tema (Algunos ***selectores*** jQuery)

Llamar a métodos jQuery

Una vez que hemos seleccionado el elemento o los elementos obtenemos un "objeto jquery" al que le aplicamos un método:

$("selector").metodo(parametros)

De esta forma aplicamos el mismo método a todos los elementos seleccionados sin tener que recorrer un nodeList. Por ejemplo podemos asignar a todos un manejador de eventos, cambiar su contenido, etc.

Algunos ejemplos

Obtener/modificar el texto de un elemento:

```
$("h1").text(); // Devuelve todo el texto contenido en todos los elementos h1
$("h1").text("Hola"); //Podemos pasarle un string o un número
$('#total').text(price * quantity);
$("li").text("Granada"); // Modifica el texto de todos los li
```

Obtener/modificar el valor de un cuadro de texto u otro control de un formulario:

```
var euros = $("#euros").val(); // Obtiene el valor
$("#euros").val("34"); // Asigna un nuevo valor
```

Obtener el siguiente hermano de un elemento

```
$("nombre").next().text("El nombre es obligatorio");
```

Enviar un formulario:

```
$("#formulario1").submit();
```

Poner el foco en un elemento del formulario:

```
$("#email1").focus();
```

Manejo de eventos

Asegurarnos de que el DOM está totalmente cargado

Escuchar la señal que indica que el DOM está "preparado" y entonces ejecutar el código:

```
$(document).ready(function() {
      // manejador para el evento  .ready()
});
```

Equivalente a la siguiente forma corta:

```
$(function() {
  // manejador para el evento  .ready()
});
```

Vincular manejadores de eventos

.on(<eventos>[, selector] [, data],<manejador de evento>)

<eventos> es una lista de tipos de eventos separada por espacios (a varios eventos podemos asociar la misma función)
<selector> es un string para filtrar descendientes
<data> datos que se pasan al manejador y que los recogerá en **event.data**
<manejador de eventos> es la función a ejecutar cuando se produzca el evento

```
$('button').on('click', function() {
	// código a ejecutar cuando cualquier el botón es pulsado
});
$('.vacation').on('click', 'button', function() {
	// código a ejecutar cuando se produzca un click en un botón que sea
	// descendiente de un elemento con clase .vacation
});

// Utilizando una function con nombre
function notify() {
  alert( "clicked" );
}
$("button").on( "click", notify );

//Pasar datos al manejador de eventos:
function greet(event) {
  alert("Hello " + event.data.name );
}
$("button").on("click", {name: "Karl"}, greet );
$("button").on("click", {name: "Addy"}, greet );
// Asociar a varios eventos la misma función manejadora:
$("#join_list").on('click mouseenter', funcionManejadora);
```

Desvincular controladores de eventos

.off()

```
$('p').off('click'); // Desvincula todos los controladores del evento click
$('p').off('click', bar); // desvincula el controlador bar del evento click
```

this

$(this)

Sería el equivalente a e.currentTarget (elemento al que se asignó el manejador de eventos). Pero ojo, si queremos utilizar métodos jQuery, no los podemos aplicar a e.currentTarget que sería un objeto tipo htmlElement y no un objeto jQuery.

```
$(document).ready(function() {
	$('button').on('click', function() {
		var price = $('<p>From $399.99</p>');
		$(this).after(price);
		$(this).remove();
	});
});
```

Si tenemos el siguiente fichero index.html:

```
<li class="vacation onsale"data-price='399.99'>
	<h3>Hawaiian Vacation</h3>
	<button>Get Price</button>
	<ul class='comments'>
		<li>Amazing deal!</li>
		<li>Want to go!</li>
	</ul>
</li>
```

Y queremos asignar una función al evento click del botón, de forma que aparezca el precio y desaparezca el botón, si lo hacemos de la siguiente forma (usando this) funcionará para cada uno de los botones del documento:

```
$('button').on('click', function() {
		var vacation = $(this).closest('.vacation');
		var amount = vacation.data('price');
		var price = $('<p>From $'+amount+'</p>');
		vacation.append(price);
		$(this).remove();
	});
```

El manejador anterior se asigna a cualquier botón del documento. Si queremos que sólo se asigne a los botones que estén "dentro" de un elemento con clase "vacation":

```
$('.vacation').on('click', 'button', function(){ …});
```

Eventos de ratón

click

dblclick

focusin

El elemento (o cualquier elemento dentro de él) obtiene el foco.

focusout

El elemento (o cualquier elemento dentro de él) pierde el foco.

mousedown

Cuando el puntero del ratón está sobre el elemento y el botón es presionado.

mouseup

Cuando el puntero del ratón está sobre el elemento y el botón es soltado.

mousemove

Cuando el puntero del ratón se mueve dentro del elemento.

mouseout

Cuando el puntero del ratón abandona el elemento o cualquiera de sus hijos.

mouseover

Cuando el puntero del ratón entra en el elemento (o en sus hijos).

mouseleave

Cuando el puntero del ratón deja un elemento.

mouseenter

Cuando el puntero del ratón entra en el elemento. Ejemplo:

```
function showTicket () {
	$(this).closest('.confirmation').find('.ticket').slideDown();
}
$(document).ready(function() {
	$('.confirmation').on('click', 'button', showTicket);
	$('.confirmation').on('mouseenter', 'h3', showTicket);
});
```

.hover()

Tiene uno o dos manejadores asociados que se ejecutarán cuando el puntero del ratón entre y salga de los elementos seleccionados. Si tiene un solo manejador será el que se ejecute en ambos casos.

Eventos de teclado

.keypress()

.keydown()

.keyup()

Si pulsamos y soltamos una tecla, primero se produce un evento keydown, keypress y por último un keyup.
Si hacemos una pulsación prolongada de una tecla se produce un keydown y un keypress repetidos mientras se mantiene pulsada la tecla . Cuando se suelta la tecla y se produce un keyup.
En el caso de las teclas CTRL, Mayúsculas o ALT, se producen múltiples keydown hasta que se suelta la tecla y se produce un keyup. Es decir, al pulsar una de estas teclas no se produce el evento keypress.
En el c9 tienes un ejemplo que muestra los códigos en en la carpeta *T6-jQuery/eventoTeclado*.

Evitar la acción por defecto de los eventos

```
evento.preventDefault();
```

Recorrer y manipular el DOM

Atravesar el DOM usando filtros:

.find()

Busca los elementos **descendientes** de cada uno de los elementos del conjunto actual (filtrados por un selector, un objeto jQuery o un elemento).

```
$("#destino").find("li");
```

Lo anterior es equivalente a `$("#destino li")`, pero es más rápido de ejecutar.

```
var spans = $( "span" );
$( "p" ).find( spans ).css( "color", "red" );
```

.first()

```
$("li").first();
```

.last()

```
$("li").last();
```

.filter()

Reduce el conjunto de elementos coincidentes a aquellos que coinciden con el selectordel filtro.

```
$('.vacation').filter('.onsale'); // equivalente a $('.vacation.onsale') pero más
                                  // rápido
```

Pasearse por el DOM:

Para referirnos a los elementos diferentes al primero y último podemos usar una técnica llamada "Walking the DOM" . Encadenando métodos podemos referirnos a distintos elementos. Por ejemplo:

```
$("li").first();
$("li").first().next();
$("li").first().next().prev(); // primer elemento
```

*Pasearse por el DOM "**hacia arriba**":*

.parent([selector])

```
$("li").first().parent(); //selecciona el elemento "padre" del primer li
$("li").parent('.saludo');
```

Se puede poner un selector como en el ejemplo anterior que daría la lista de los padres de los elementos li, pero sólo aquellos padres con la clase *saludo*.

.parents([selector])

```
$("li").parents(); //selecciona todos los ancestros de cada elemento del conjunto
de li (analiza varios niveles desde cada li)
```

.closest([selector])

```
$("li").closest(".viaje"); //para cada elemento li, selecciona su primer ancestro
que tiene la clase "viaje"
```

*Pasearse por el DOM "**hacia abajo**":*

.children()

```
$("#destino").children("li"); // equivalente a $("#destino > li");
```

.siblings()

Selecciona todos los elementos hermanos del seleccionado

Añadir y borrar nodos

Crear un nodo:

```
$(document).ready(function() {
      var price = $('<p>From $399.99</p>');
});

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

De esta forma se crea el nodo pero **no se añade al DOM.**

Clonar un nodo

.clone()

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

Añadir un nodo al DOM

Tenemos los siguientes métodos:

.append(<element>)

```
$('.vacation').append(price);//Añade el nodo price como último hijo de .vacation
```

.appendTo(<element>)

```
price.appendTo($('.vacation')); //Añade el nodo price como último hijo de .vacation
```

.prepend(<element>)

```
$('.vacation').prepend(price);//Añade el nodo price como primer hijo de .vacation
```

.prependTo(<element>)

```
price.prependTo($('.vacation'));
```

.after(<element>)

```
$('.vacation').after(price); // Inserta el nodo price después de .vacation
```

.insertAfter(<element>)

```
price.insertAfter($('.vacation'));
```

.before(<element>)

```
$('.vacation').before(price); // Inserta el nodo price antes de .vacation
```

.insertBefore(<element>)

```
price.insertBefore($('.vacation'));
```

Eliminar un nodo del DOM

```
$('button').remove();
```

Manipular clases

.addClass(<class>)

```
$('.vacation').filter('.onsale').addClass('brillante');
```

.removeClass(<class>)

```
$('.brillante').removeClass('brillante');
```

.toggleClass()

```
$('#vacations').on('click', '.vacation', function() {
      $(this).toggleClass('brillante');
});
```

.hasClass(<class>)

Devuelve verdadero o falso

```
$(document).ready(function() {
$('#vacations').on('click', '.vacation', function() {
      $(this).toggleClass('highlighted');
      if($(this).hasClass('highlighted')) {
            $(this).animate({'top': '-10px'});
      } else {
            $(this).animate({'top': '0px'});
      }
      });
});
```

Manipular atributos

.attr()

```
$('#myDiv a:first').attr('href', 'newDestination.html');
$('#myDiv a:first').attr({ //Varios atributos como un objeto
        href : 'newDestination.html',
        rel : 'super-special'
```

```
});
```

Manipular propiedades

.prop()

Obtiene el valor de una propiedad del primer elemento del conjunto de elementos al que se aplica el método. O bien, asigna una o más propiedades a cada elemento del conjunto.

Attributes vs. Properties

La diferencia entre *atributos* y *propiedades* puede ser importante en situaciones concretas. Antes del **jQuery 1.6**, el método `.attr()` a veces tenía un comportamiento inconsistente. A partir de **jQuery 1.6**, el método `.prop()` proporciona una manera explícita de obtener valores de propiedades, mientras que `.attr()` devuelve atributos, que no siempre se corresponden con las propiedades.

Por ejemplo, el atributo `checked` no se corresponde con la propiedad `checked`. El atributo realmente corresponde con la propiedad `defaultChecked` y debe ser usado sólo para poner el valor inicial del checkbox. El valor del atributo `checked` no va a cambiar con el estado del checkbox, mientras que la propiedad `checked` si lo hará.

Mostrar/ocultar elementos

En los siguientes métodos es posible normalmente indicar la duración en milisegundos (velocidad a la que aparecen o desaparecen los elementos) y también podemos indicar una función callback que se ejecutará cuando la animación termine.

.slideDown()

Muestra los elementos con un movimiento deslizante hacia abajo. Tiene posibles parámetros para configurar el movimiento (mirar la página jquery.com)

```
$(this).closest('.confirmation').find('.ticket').slideDown();
```

Previamente .ticket estaba oculto (con .css):

```
.ticket {
	display:none;
}
```

.slideUp()

Muestra los elementos con un movimiento deslizante hacia arriba.

.slideToggle()

Visualiza u oculta los elementos seleccionados

```
$(document).ready(function() {
  $("#tour").on("click", "button", function() {
$(".photos").slideToggle(); //Al pulsar el botón aparecen/desaparecen las fotos
  });
});
```

.fadeIn()

Visualiza los elementos coincidentes con el selector desvaneciéndolos hacia opacos.

.fadeOut()

Oculta los elementos coincidentes con el selector desvaneciéndolos hacia transparentes.

```
$('#startup_message').fadeOut(5000);
$('#startup_message').fadeOut(5000).slideDown(1000);//Podemos encadenar métodos
```

.fadeTo()

Ajusta la opacidad de los elementos seleccionados. Hay que especificar la duración y la opacidad.

```
$('#startup_message').fadeTo(5000, .2).fadeTo(100, 1);
// O el callback equivalente:
$('#startup_message').fadeTo(5000, .2,
	function() {
```

```
                $(this).fadeTo(1000, 1);
        }
};
```

.fadeToggle()

Visualiza u oculta los elementos animando su opacidad (cambia en cada click)

```
$(this).closest('.vacation').find('.comments').fadeToggle();
```

.show()

Se corresponde a .css('display', 'block')

.hide()

Se corresponde a .css('display', 'none')

Cambiar estilos

En un fichero .css externo

```
.brillante {
        background-color:#563;
        border-color: 1px solid #967;
}
```

Y en .js:

```
$(this).addClass('brillante');
```

Animaciones

.animate({propiedades} [, duracion] [, callback])

Realiza una animación personalizada de un conjunto de propiedades CSS. Por ejemplo (sacado de api.jquery.com/animate)

http://jsfiddle.net/apsierra/s9XRX/1/

Ejemplos útiles de uso de jQuery

Ejecutar una función por cada elemento de una selección

Se ejecuta una función por cada elemento de la selección. La función recibe como argumento el índice del elemento actual y el mismo elemento. Dentro de la función, se puede hacer referencia al elemento DOM a través de `this`.

```
$('#paisaje li').each(function(indice, elemento) {
//elemento es un objeto HTMLelement
  alert('El elemento ' + indice + 'contiene el texto: ' + $(elemento).text());
});
```

También existe la función $.each(array, callback) que puede usarse para iterar sobre un array o sobre un objeto cualquiera. Si el objeto es un array o un objeto array-like que tiene una propiedad length, se itera por un índice numérico, desde 0 hasta length-1. Si se trata de objetos de otro tipo, se itera a través de los nombres de sus propiedades.

```
$.each([ 52, 97 ], function( index, value ) { // $.each(array,callback)
  alert( index + ": " + value );
});
```

Comprobar el tipo de un determinado valor

```
jQuery.isFunction(myValue);
jQuery.isPlainObject(myValue);)
jQuery.isArray(myValue);
jQuery.isNumeric(16);
```

Atributos data-

En HTML 5 podemos utilizar atributos ***data-loquesea***:

```
<li class="vacation onsale" data-price="399.99">
```

en jQuery tenemos el método .data() que permite obtener/asignar valor de ese atributo:

```
var precio = $('.vacation').first().data('price'); //Devuelve un string
$('.vacation').first().data('price', '350');
$('#myDiv').data('keyName', { foo : 'bar' });
```

Convertir un string en un número (signo +):

```
var precio = +$('.vacation').first().data('price');
```

Depurar

La propiedad .length indica el número de nodos que contiene una selección. Si ese número es 0, la selección no es correcta. Para comprobarlo podemos escribir por ejemplo:

```
alert($('button').length);
```

Existen herramientas que nos permiten depurar jQuery, probad las que encontréis en internet. Podéis usar una extensión de Chrome llamada "jQuery Debugger" que permite inspeccionar los selectores. En Firefox tenemos FireQuery.

JQuery para trabajar con formularios

Jquery proporciona selectores, métodos y manejadores de eventos que hacen más fácil trabajar con formularios. Sin embargo no proporciona formas de validar datos.

En la siguiente tabla vemos los **selectores** que podemos usar para formularios con jQuery. Permiten seleccionar varios tipos de controles así como seleccionar controles deshabilitados, habilitados, chequeados y seleccionados.

:input	Todos los elementos del formulario: input, select, textarea, button
:text	Todos los cuadros de texto (elementos input con type= “text”)
:radio	Todos los radio buttons (elementos input con type=”radio”)
:checkbox	Todos los Check boxes (elementos input con type=”checkbox”)
:file	Todos los campos input con type=”file”
:password	Todos los campos input con type=”password”
:submit	Todos los elementos buttons y submit (campos input con type=”submit” y elementos button)
:reset	Todos los botones reset (input con type=”reset”)
:image	Todos los botones image (input con type=”image”)
:button	Todos los botones (elementos button y elementos input con type=”button”
:disabled	Todos los elementos deshabilitados (elementos que tienen el atributo disabled).
:enabled	Todos los elementos habilitados (elementos que no tienen el atributo disabled).
:checked	Todos los checkboxes y radio buttons que están marcados
:selected	Todas las opciones en elementos select que están seleccionadas

val()

Método para obtener y asignar valores a un control. Obtiene el valor de un cuadro de texto u otro campo en un formulario. Para modificarlo usamos val(valor).

trim()

Método para borrar todos los espacios al principio y al final de un string:

```
var nombre = $('#nombre').val().trim();
$('#nombre').val(nombre);
```

Ejemplos:

1. Obtener el valor del elemento marcado de un radio button. Se seleccionan todos los elementos con el atributo name='contacto' (incluye todos los radiobutton porque tendrán el mismo *name*). Después se usa el selector :*checked* para obtener el radio button que está seleccionado (sólo uno puede estar seleccionado). Para un radio button, el método *val()* devuelve el valor del atributo value. No se pone espacio antes de :*checked*.

```
var radioButton = $('input[name='contacto']:checked').val();
```

2. Obtener un array con los elementos seleccionados de una lista. En este caso usamos el selector :*selected* para obtener todas las opciones seleccionadas de una lista con id='listaSelect'. Se pone un

espacio antes de :*selected* porque se seleccionan los elementos **descendientes** de la lista.

```
var opcionesSelect = [];
opcionesSelect = $('#listaSelect :selected');
```

Eventos de formulario

blur(manejador)

El manejador se ejecuta cuando el foco abandona el elemento seleccionado.

focus(manejador)

El manejador se ejecuta cuando el foco entra en el elemento seleccionado.

select(manejador)

El manejador se ejecuta cuando el usuario selecciona texto en un campo de texto o textarea.

submit(manejador)

El manejador se ejecuta cuando ocurre el evento submit, es decir, cuando el usuario hace click en un botón submit o cuando el usuario mueve el foco a un botón submit y presiona la tecla enter. Pero también ocurre cuando el método submit() se usa para lanzar el evento.

change(manejador)

El manejador se ejecuta cuando el valor del elemento seleccionado ha cambiado.

Podemos utilizar los siguientes métodos para **iniciar eventos** desde el propio código en un elemento determinado. Por ejemplo, si llamamos al métodos focus() en un cuadro de texto, el foco se moverá al cuadro de texto y el evento focus se lanzará. Si no se le ha asignado ningún manejador de eventos al evento focus, no se procesará el evento.

blur()

Quita el foco del elemento seleccionado y lanza el manejador del evento blur.

focus()

El foco se mueve al elemento al que se le aplica el método.

select()

Lanza el evento select.

submit()

Lanza el evento submit.

change()

Lanza el evento change.

Ejemplos:

1. Un manejador que habilita o deshabilita radio buttons cuando un check box es seleccionado o deseleccionado:

```
$('#contacto).on("change", function(){
                if    ($('#contacto).prop('checked')) {
                      $(':radio').prop('disabled', false);
                }
                else {
                      $(':radio').prop('disabled', true);
                }
          });
```

2. Un manejador de eventos que lanza el evento submit después de validar algunos datos

```
$(document).ready(function() {
   $('#envio).click(  // envio es un botón normal, no un botón submit
          function(){
                // Código de validación iría aquí
                $('#formulario').submit();
          }
   );
});
```

Nota: Podemos usar también un manejador de eventos para el evento submit del formulario para validar los datos antes de enviarlos al servidor. En este caso, si cualquier dato no es válido, hay que usar el método

preventDefault() del objeto event para cancelar el envío de datos al servidor (igual que hacíamos en javaScript).
Podemos encontrar numerosos plugins jQuery para validar formularios, entre ellos:
http://jqueryvalidation.org/

Ejercicio: Realizar un formulario que pida dos direcciones de e-mail y las valide al pulsar el botón de envío. Si están vacías debe mostrar un error y si no coinciden también. Los errores aparecerán a la derecha de los campos de texto.

Algunos selectores jQuery

[atributo]	Todos los elementos con ese atributo
[atributo=valor]	Todos los elementos con ese atributo y valor. `Ej: $("input[type=text]")`
:contains(texto)	Todos los elementos que contengan el texto especificado
:empty	Todos los elementos que no tengan hijos (incluso nodos de texto)
:eq(n)	El elemento del índice n dentro del conjunto (empieza en 0). Ej: `$("#faqs p:eq(2)")` `//devuelve el tercer elemento <p> descendiente del elemento "faqs"`
:even	Todos los elementos con un índice par dentro del conjunto. Ej: `$("tabla > tr:even")`
:first	El primer elemento dentro del conjunto
:first-child	Todos los elementos que son el primer hijo. Ej: `$("li:first-child")` `//Selecciona los elementos li que son el primer hijo de su elemento padre`
:gt(n)	Todos los elementos de la selección que tienen un índice mayor que n
:has(selector)	Todos los elementos que contienen el elemento especificado por el selector
:header	Todos los elementos que son cabeceras (h1, h2..)
:hidden	Todos los elementos ocultos
:last	El último elemento de la selección
:last-child	Todos los elementos que son el último hijo de sus padres.
:lt(n)	Todos los elementos de la selección que tienen un índice menor que n
:not(selector)	Todos los elementos que no son seleccionados por el selector
:nth-child(n)	Todos los elementos que son el hijo n de sus elementos padres
:odd	Todos los elementos impares del conjunto seleccionado
:only-child	Todos los elementos que son hijos únicos de su elemento padre
:parent	Todos los elementos que son padres de otros elementos, incluyendo nodos de texto
:text	Todos los elementos input con de tipo texto
:visible	Todos los elementos que están visibles

jQueryUI (interfaz de usuario en jQuery)

Visitar la página http://jqueryui.com/
Es una librería que añade usos a jQuery proporcionando características de alto nivel que puedes usar escribiendo muy poco código. Para usarla necesitamos bajarnos la librería completa o bien una a medida de lo que necesitemos utilizar en nuestra página.
Para ver cómo funcionan las interfaces y su cómo utilizarlas en nuestra página: https://jqueryui.com/demos/
Aquí elegimos la herramienta que queremos y se nos muestran varias maneras de utilizarla.
Hay cuatro tipos de características que jQuery UI proporciona:
themes: proporcionan el formato de los widgets, y se implementan mediante una hoja de estilos CSS que es parte de la descarga de jQuery UI. Se trata de una serie de temas predefinidos así como una aplicación llamada ThemeRoller que te permite crear un tema a medida.
widgets: son elementos como acordeones, barras de progreso, tooltips, calendarios, etc.
interactions: dan características a los elementos como draggable, droppable, resizable, etc.
effects: animaciones de color, transiciones de clases, etc

Interactions

Se pueden aplicar a cualquier elemento HTML. Por ejemplo, puedes hacer una lista desordenada "sortable", o puedes hacer una fila de una tabla "selectable". Para estos usos es necesario incluir "interaction" en tu fichero de descarga.

- Draggable
- Droppable
- Resizable
- Selectable
- Sortable

Widgets

- Accordion
- Autocomplete
- Button
- Checkboxradio
- Controlgroup
- Datepicker
- Dialog
- Menu
- Progressbar
- Selectmenu
- Slider
- Spinner
- Tabs
- Tooltip

Effects

Añade transiciones animadas. Se pueden usar con interacciones y widgets. Por ejemplo, un efecto puede usarse para controlar cómo una nueva tabla es visualizada en un widget "Tabs". Además, los efectos jQuery UI pueden usarse con elementos HTML que no son parte de jQuery UI.

- Add Class
- Color Animation
- Easing
- Effect
- Hide
- Remove Class
- Show
- Switch Class
- Toggle
- Toggle Class

Cómo construir y usar un "jQuery UI custom download"

Antes de descargar jQuery UI hay que construir la descarga. Esto nos permite seleccionar los componentes que vamos a necesitar para hacer el fichero a descargar lo más pequeño posible.

Hay una página llamada Download builder http://jqueryui.com/download/que permite elegir los elementos que queremos descargar seleccionándolos con el ratón. Si intentamos deseleccionar un elemento

que es requerido por otro componente, nos avisa mediante un mensaje de dicha dependencia. Para la mayoría de los sitios web, deseleccionaremos la mayoría de las interacciones y los efectos. Si, por ejemplo, vamos a usar sólo unos pocos widgets podemos deseleccionar el resto.

Después de seleccionar los componentes que queremos, podemos seleccionar un tema de la lista o crear uno a medida mediante la herramienta ThemeRoller http://jqueryui.com/themeroller/ .
Cuando estamos seguros de que tenemos todo lo que queremos, hacemos clic en el botón *Download* y obtenemos un fichero .zip que contiene una serie de ficheros y carpetas, entre ellos el fichero index.html.

El fichero index.html puede ser usado como demostración de cómo quedarán nuestras páginas con los componentes que hemos elegido.

Si abrimos el fichero index.html en un editor de textos veremos que enlaza con unas pocas dependencias: tu tema, jQuery y jQuery UI. Generalmente necesitarás incluir esos tres ficheros en cualquier página que use jQuery UI. Ejemplo:

```
// tema css:
<link rel="stylesheet"href="css/themename/jquery-ui.custom.css" />
// Librería jQuery:
<script src="external/jquery/jquery.js"></script>
// jQueryUI
<script src="jquery-ui.js"></script>
// A continuación tendremos que incluir nuestro fichero javascript:
<script src=”miScript”></script>
```

Una vez incluidos los ficheros necesarios, podemos añadir widgets jQuery a la página. Por ejemplo:

HTML:

```
1  <input type="text" name="date" id="date" />
```

JavaScript:

```
1  $( "#date" ).datepicker(); //El elemento #date abrirá un calendario
```

Ejemplo tabs: http://jsfiddle.net/apsierra/98pky/5/
Ejemplo acordeón: http://jsfiddle.net/apsierra/D5Gbr/1/

Cómo funciona jQuery UI

Inicialización

Debemos introducir un ciclo de vida completo para el widget. El ciclo de vida empieza cuando el widget es inicializado. Para inicializarlo, simplemente llamados al plugin en uno o más elementos:

```
$("#elem").tabs();
```

Esto inicializará cada elemento del objeto jQuery (en este caso el elemento con id="elem"). Podemos pasar un conjunto de **opciones** durante la inicialización para sobreescribir las opciones por defecto:

```
$("#elem").tabs({ collapsible: true });
```

Las opciones que no pasamos, tendrán su valor por defecto. Las opciones son parte del estado del widget, podemos poner opciones después de la inicialización también.

Métodos

Podemos preguntar por el estado del widget o realizar acciones en él llamando a un método. Por ejemplo, para llamar al método ***disable(index)*** de nuestro widget tabs:

```
$("#elem").tabs("disable", 1); // Deshabilita el panel número 1
```

Métodos comunes

Cada widget tiene su propio conjunto de métodos según la funcionalidad que proporciona, pero hay algunos comunes:

option(optionName, value)

Obtiene o modifica las opciones después de la inicialización:

```
$("#elem").tabs("option"); //Obtiene un objeto que representa las opciones del tabs
$("#elem").tabs("option","disabled"); //Obtiene el valor asociado a la opción"disabled"
$("#elem").tabs("option","disabled", true); //asigna true a la opción"disabled"
```

disable

Deshabilita todos los tab.

```
$("#elem").tabs("disable");
```

enable

Habilita todos los tab

```
$( "#elem").tabs("enable");
```

destroy

Destruye el widget, termina su ciclo de vida. Vuelve el elemento al estado anterior a la inicialización del widget.

```
$("#elem").tabs("destroy");
```

widget

Devuelve un objeto jQuery que contiene el elemento contenedor del widget.

```
$("#elem").tabs("widget"); //Devolverá el elemento con id="elem"
```

Eventos

Todos los widgets tienen eventos asociados con su comportamiento para notificarnos cuando ha cambiado su estado. Por ejemplo:

```
$( ".clase" ).tabs({
      activate: function( event, ui ) {}
});
```

Donde:

event es el evento (tabsactivate)

ui es un objeto con las propiedades:

newTab (objeto jquery conteniendo el Tab seleccionado)

oldTab

newPanel (objeto jquery conteniendo el panel seleccionado)

oldPanel