OpenWebinars: JQuery

Desarrollar webs más interactivas

Anastasia Macovei

Junio 2018

¿Qué es jQuery?

Nociones básicas

Conocimientos

- **árbol HTML**: representación en forma de árbol jerárquico cuya raíz es la etiqueta <html>
- **Script**: elemento ejecutable dentro de una página HTML
- **Evento**: acción que sucede en el navegador, ya sean propios del navegador o las interacciones con los usuarios.
- **Ajax**: comunicación asíncrona que permite actualizar ciertas partes de la página sin recargar la página

Jquery: `write less, do more`
Escribimos menos haciendo más y permite manejar nuestro árbol DOM.

Características:
- manipular el DOM
- Selectores avanzados tanto de CSS como jQuery
- Eventos de la página
- Funciones de parseo de json
- Ajax
- Compatible con todos los navegadores

Desventajas:
- el rendimiento no es óptimo porque accedemos continuamente al DOM
- Se ha abandonado por BootStrap 5 y gitHub
- Hay nuevos frameworks (Angular, React, Vue)

DOM

DOM

Es una api para documentos HTML y XML. Comunicamos la página web con los scripts que maneja. Todas las páginas web se pueden representar en forma de árbol cuyo padre es HTML.
Hay distintos nodos: antecesor, descendiente, padres e hijos, así como hermanos.

Con *API DOM* referenciamos objetos, modificamos propiedades y funciones, así como responder a eventos.

Instalación

https://jquery.com/download/ -> versión comprimida o no.

```
<script src='js/jquery_3.5.1.js'></script>
```

- Para ver que haya funcionado nuestra librería ponemos en el navegador $() y devuelve un objeto.
- Siempre pondremos el script al final del body, antes de que cierre.
- Añadir directamente el CDN ->

```
<script src="https://code.jquery.com/jquery-3.5.1.js"
integrity="sha256-QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc="
crossorigin="anonymous"></script>
```

Selectores jQuery

- seleccionar qué elemento quiero cambiar o a cuál quiero hacerle un evento
- podemos usar los selectores CSS3, pero también selectores propios.

Selectores CSS:

* : selector universal
- etiqueta
- id
- .clase
- Elemento que está dentro de otro elemento
- Elemento 1 y elemento 2
- Hermanos

Podemos seleccionar con la función . y cogemos el índice del que queramos seleccionar:

```
$("img").eq(1).hide()
```

Selectores CSS más usados:

- :hover
- :visited
- :first-of-type
- :last-of-type
- nth-child()
- :required
- ::after
- ::before

Ejemplo: queremos ocultar el tercer hijo de ul:

```
$("ul.iconized li:nth-child(3)").hide()
```

Ejercicios usando selectores CSS

- Seleccionar la primera columna del cuerpo de la tabla (dos maneras)

```
$("tbody tr td:first-child").hide()
$("tbody tr td:first-of-type”).hide()
```

- El primer elemento de la lista (de dos maneras).

```
$(".iconized li:nth-child(1)").hide()
$(".iconized li”).eq(0).hide()
```

- El campo del formulario anterior al botón (de dos maneras).

```
$("input").eq(1).hide()
$("input:last-of-type").hide()
```

- El último elemento de la lista (de dos maneras).

```
$(".iconized li:last-child").css("color", "red");
$(".iconized last-of-type").css("color", "blue");
```

- La última imagen de mi página (de dos maneras)

```
$(“card img:last-child").hide()
$("img").eq(5).hide()
```

Selectores jQuery más usados:

- :eq(index)
- :has //ejemplo: $("form").has("input[type='password']");
- :contains
- :even / :odd -> para filas pares e impares
- :input -> se seleccionan todos los elementos de un formulario
- :gt / :lt ->rango de lo que se quiere seleccionar
- :first / :last -> .first() .last() actualmente
- :button, :file, :radio, :reset
- :submit, :text, :checkbox
- :password, :image
- :parent ->que tiene algún hijo dentro o más etiquetas, o texto

Método de jQuery: .each() & $(this)

Muchos elementos pero distintas tareas cuando queremos hacer cosas muy simulares pero ligeramente distintas.

```
 $(".iconized li").each(function(index){ //se ejecutará 3
veces porque tenemos tres elementos así
    console.log("El elemento " +index + " contiene" + $
(this).text())
    if(index % 2 == 0){
      $(this).css("color", "yellow")
    }
    else{
      $(this).css("color", "blue")
    }
    })
```

Ejercicios prácticos

• Celdas de la tabla que contengan un texto determinado.

```
$("td:contains('B')").css("color", "green")
```

• Celdas de la primera columna de la tabla que contengan un texto.

```
$(“td:nth-child(1):contains('A')")
```

• Todos los campos y los botones del formulario.

```
$("input")
$("form span”)
```

• Todos los elementos de cabecera.

```
$(“:header”).hide()
```

• Las dos últimas imágenes (de las que están en fila de cuatro

```
$("img:eq(2), img:eq(3)")
```

Ejercicio con each() y this()

```
 $(function () {
      var total = 0;
      var numElementos = 3;

      $("tbody tr").each(function (index) {
        console.log("ÍNDICE" + "------>" + index);

        let examen1 = $("tbody tr:nth-child(" + (index + 1) +
") td").eq(1).text();
        console.log("Fila 1" + "-->" + examen1);
        let examen2 = $("tbody tr:nth-child(" + (index + 1) +
") td").eq(2).text();
        console.log("Fila 2" + "-->" + examen2);
        let examen3 = $("tbody tr:nth-child(" + (index + 1) +
") td").eq(3).text();
        console.log("Fila 3" + "-->" + examen3);
        let total = (parseFloat(examen1) + parseFloat(examen2)
+ parseFloat(examen3)) / numElementos;
        $("tbody tr:nth-child(" + (index + 1) + ")
td").eq(4).text(total);
        (total >= 5) ? $(this).css("color", "green") : $
(this).css("color", "red");

      })
    })
```

CSS: atributos y estilos

- a través de la función **.css (“propiedad”, “valor”)**
- Es importante recordar que se accede solo al primer elemento del selector
- También podemos añadir un **.prop ->** por ejemplo hacer un incremento en el width de la propiedad que tenemos
- En el caso de tener varias propiedades devolveremos un array
- Funciones adicionales
  - .width()
  - .height()

Funciones comunes que podemos utilizar de jQuery:

- addClass()
- removeClass()
- toggleClass() ->añade o quita clases: de sí a no y de no a sí
- hasClass() -> devuelve true si cualquiera de los elementos seleccionados tiene esa clase

Funciones para añadir atributos y modificar sus propiedades:

- es importante diferenciar entre atributo y propiedad
- Los atributos nos dan información adicional sobre la etiqueta
- Las propiedades nos dan información sobre el valor actual. No es necesario que esté en el HTML.
- función .attr() -> var valor = $(“selector”).attr(“href”)

Ejemplo:

```
$("img").attr("src", "./img/c1.jpg")
```

- remoteAttr()
- prop() y removeProp()

Ejemplo de todo lo anterior con atributos y propiedades:

```
//Si tenemos el siguiente elemento en la página
        // <input id="ej" type="text" value="Pepe">

        console.log($("#ej").attr("value"));
        //Salida -> Pepe

        console.log($("#ej").prop("value"));
        //Salida -> Pepe

        console.log($("#ej").attr("disable"));
        //Salida -> undefined -> No está en el HTML

        console.log($("#ej").attr("disable"));
        //Salida -> false . No está pero es una propiedad del DOM

        //Modificamos el valo del elemento del DOM
        $("#ej").val("Manuel");

        console.log($("#ej").attr("value"));
        //Salida -> Pepe, el valor que tengo en el HTML

        console.log($("#ej").prop("value"));
        //Salida -> Manuel, el valor actual de la propiedad
```

Valores y contenidos

Las funciones que se usan:

- **.empty**() -> vaciar la etiqueta con todos los hijos que contiene incluido el contenido.
- **.html**() -> cambiamos el html de nuestra página tanto para obtener como para fijar

```
$(".iconized").html(function(index, oldtext){
        $(this).html(oldtext + "<li>Nuevo </li>")
      })
```

- .**text**() -> se obvia las etiquetas y representa el texto que corresponde

- .**append**(), .**prepend**(), .**appendTo**(), **prependTo**() -> antes o después de la apertura con prepend
- **wrap**(), **unwrap**(), **wrapAll**(), **wrapInner**() -> añade cierta estructura a una etiqueta html
- **val**() -> obtener el valor

Ejemplo:

```
var nombre = $("input").eq(0).val(function (index, valor) {
        return valor + "nuevo";
      })
```

Ejercicios prácticos

• Jugando con las clases modifica los estilos de la primera y última celda de cada fila.

```
$("tbody td tr:first, tbody td tr:last").addClass("celda_destacada")
```

• Todas las imágenes deben mostrar el logo de Ow.

```
$("img").attr("scr", ".img/logo-openwebinars.png").css("backround-color", "blue"
```

• Añade a la tabla 10 filas iguales al final (en las filas de cada celda el número de la fila)

```
for (i = 0; i < 10; i++) {
        $("table tbody").append("<tr> <td>" + i + "</td></tr" +i + "<tr>
            <td>" + i)
      }
```

• Convierte el campo de Contraseña en un campo de fecha.

```
$(":password").attr("type", "date")
```

• Establece valores por defecto en todos los campos del form

```
$("input").eq(0).val("Hola")
```

• Modifica varias propiedades de las etiquetas de cabecera con una única línea.

```
$(":header").css({
        "color": "green",
        "background-color": "yellow",
        "font-size": "40px"
      })
```

• Añadir índice a los elementos de lista respetando el contenido.

```
$(".iconized li").html(function(index, texto){
        return texto + ":" + index
      })
```

• Envolver todas las celdas de la tabla dentro de un div.

```
$("td").wrap("<div></div>")
```

• Darle al div anterior una clase que produzca un cambio en los estilos.

```
 $("td > div").addClass("nueva_clase")
```

Eventos en jQuery

Se capturan eventos y se responden a ellos. La propagación de los eventos se llama 'event bubbling' de hijos a padres.

Hay que tener en cuenta que se puede hacer dos tipos de capturas de eventos:

- **Asociación directa:** cuando el evento sucede en elemento o hijo
- **Asociación delegada:** cuando el evento sucede en hijos que cumplen selector

También existe la función análoga **.one** (cuando se captura el evento, se elimina). Se diferencia con on porque cuando se hace la captura se elimina.

- eventos que tienen nombres, como por ejemplo **.click** -> siempre tendrá que ser asociación directa

Eventos en jQuery:

- change()
- click()
- dblclick()
- focus()
- focusout()
- hover()
- keypress()
- keydown()
- keyup()
- mousedown()

El objeto Event

Es un objeto javascript con información adicional sobre el mismo. Tiene métodos propios que permite evitar la propagación del evento hacia arriba.

Propiedades y funciones:

- e.currentTarget()
- e.delegatedTarget()
- e.pageX ->posición de la pantalla de dónde se ha producido el evento
- e.pageY->
- e.preventDefault()
- e.stopPropagation()
- e.Target()
- e.timeStamp
- e.type ->nombre del evento que se ha producido
- e.which
- e.metaKey

Eventos de ratón y teclado

En relación a los eventos de teclado:

- keydown(). Incluye las teclas especiales
- keyup(). Aquí no incluimos teclas especiales

- keypress() -> no es lo mismo que keydown(). No es case sensitive

Eventos de ratón:

- mousedown()
- mouseenter() -> solo en el elemento
- mouseleave() -> solo cuando se sale del elemento
- mousemove() -> se dispara cuando entra en un elemento o sus hijos
- mouseout() -> del elemento y sus hijos
- mouseover()
- click()
- dbllick()
- mouseup()

Ejemplo práctico de las funciones:

```
$(function () {
            var keydownVeces = 0;
            var keypressVeces = 0
            var entrandoEL = 0;
            var saliendoEL = 0;
            var entrandoOO = 0;
            var saliendoOO = 0;

            $('#elemento').keydown(function (event) {
                $('#keydownveces span').html(++keydownVeces);
                $('#teclakeydown span').html(event.which);
            });

            $('#elemento').keypress(function (event) {
                $('#keypressveces span').html(++keypressVeces);
                $('#teclakeypress span').html(event.which);
            });

            $(".zona").mouseenter(function (event) {
                $("p#entrada1 span").html(++entrandoEL);
            });
```

```
        $(".zona").mouseleave(function (event) {
            $("p#salida1 span ").html(++saliendoEL);
        });

        $(".zona").mouseover(function (event) {
            $("p#entrada2 span").html(++entrandoOO);
        });

        $(".zona").mouseout(function (event) {
            $("p#salida2 span ").html(++saliendoOO);
        })
    })
```

Funciones avanzadas para eventos

Además de capturar respuestas tenemos funciones avanzadas. Se puede controlar la propagación del mismo. Si tenemos la captura de evento con el selector y el .on, si queremos evitar el comportamiento por defecto lanzaríamos .prevenDefault().

Para evitar la propagación hacia arriba, haríamos: .stopPropagation() -> no lo propagaría hacia el padre.

Para el resto: .stopImmediatePropagation()

Ejemplo:

```
$("a").click(function (event) {
            event.preventDefault();
            alert("No me voy a ningún sitio");
            //Si algún padre tuviera algún handler no se ejecutaría
            event.stopImmediatePropagation();
        });
```

También podemos conectar y desconectar Handlers con **.on**() y **.off**()

Por otro lado, podemos simular un click, por ejemplo, con los **triggers**. Con trigger se ejecutan todo los handles asociados.

O también creamos nuestro propios eventos.

Movimiento por el DOM

Toda página HTML tiene una estructura similar: padres, hermanos e hijos. JQuery proporcionada funciones para moverse fácilmente por el árbol a partir de los elementos seleccionados.

Algunas de las más típicas son:

- children()
- next(), nextAll(), nextUntil()
- prev(), prevAll(), prevUntil()
- parent(), parents(), parentsUntil()
- siblings()
- closest()

Filtrar en el árbol

jQuery también nos permite filtrar los selectores que queramos con las siguientes funciones:

- **find()** -> seleccionar los hijos de los elementos seleccionados que cumplen con ciertas condiciones
- **filter()->** de todos los seleccionados nos quedamos los que cumplen con otro criterio. También se puede usar con una función con índex.
- **slice()** -> parte todos los elementos de un selector por un indice y otro

Insertar nuevos nodos en el árbol

Funciones más utilizadas:

- **after(), before()** -> se selecciona el elemento que queramos y después la lista de contenidos que queremos añadir y los añade antes o después. El contenido puede ser HTML, texto, objeto de jQuery. También se le pueden pasar estas funciones con un índice
- **insertAfter(), insertBefore()** -> se inserta antes o después
- **clone()** -> se clonan los elementos seleccionados

Para eliminar/reemplazar elementos del DOM:

- **remove()** -> borramos del DOM todos los elementos seleccionados

- **detach()** -> funciona igual que remove pero va a devolver lo que se ha borrado y de esta manera podremos guardarlo.
- **replaceAll** -> reemplazamos con el conjunto de los seleccionados
- **replaceWith()** -> igual que el anterior pero el origen y el objetivo está al revés

Animaciones y efectos

Jquery proporciona los siguientes efectos:

Display:

- hide()
- show()
- toggle()

Sliding(cortinilla)

- slideDown()
- slideUp()
- slideToggle()

Fading(desvanecimiento)

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

Se le puede expresar la duración que queramos de la animación o también una animación con una función.

Tenemos que tener en cuenta la cola de efectos, de tal manera que cuando acaba un efecto, empieza otro, si tuviéramos más de un efecto en un selector. Para manejarla tenemos:

- **.stop()** -> finaliza la animación
- **.finish()** -> finaliza la actual y elimina las que están esperando y también las finaliza.

- **queue('fx')** -> para obtener la cola de efectos
- **clearQueue**()

Animaciones

Son nuestros propios efectos. Animar un elemento en mi página web es modificar en el tiempo las propiedades CSS. No todas las propiedades son animables. Con jquery se pueden animar aquellas que tengan valores numéricas (o unidades numéricas). Ejemplo de las que son animables:

- width / height()
- Top / left / right / button
- Opacity
- BorderWith
- Padding
- Margin
- fontSize
- lineHeight

Las que no son animables pero sí podemos hacerlo con otras extensiones:

- Color
- Background-color
- Display
- Float

La función .animate() tiene muchas funciones a su vez. Es muy flexible. Ejemplo:

```
//Especificando los propiedades
   // Valores especiales 'show', 'hide' 'toggle'
   // Incremento con respecto al valor actual += y -=
   $("some_selector").animate({
       prop1 : valor1,
       prop2: valor2,
       …
       proprN : valorN
   });
```

```
    //Especificando los propiedades  y duración
    $("some_selector").animate({
        prop1 : valor1,
        prop2: valor2,
        …
        proprN : valorN
    }, duración_milisegundos);
```

Con **.animate()** si no le pone nada tendremos una duración de 4000 ms.

Ajax

Ajax hace referencia a Javascript asíncrono conjuntamente con varias tecnologías existentes (HTML, CSS, javascript, XML, etc). Para no tener que volver a recargar la página.

En cuanto a las tecnologías relacionadas tenemos a JSON y XML.

Ajax hace que trabajar con estas tecnologías que sea mucho más fácil porque proporciona funciones como las siguientes:

- $.ajax()
- $.ajaxGetJSON()
- $,ajaxSetup()
- $.ajaxGet()
- $.ajaxPost()

En nuestro .ajax() tenemos las más simples aunque existen muchas más:

- async
- Method
- statusCode
- Success
- Complete

jQuery UI

Es un conjunto seleccionado de efectos, componentes y temas construidos sobre jQuery.

Se puede instalar de varias maneras (local o remota). O instalación personalizadas con distintos temas.

Proporciona elementos:

- Competentes o Widgets
- Efectos
- Interacciones -> lo más interesante, como por ejemplo redimensionar elementos etc.

Componentes: elementos que son comunes que constan de estructura, estilo y comportamiento que están bien documentos. Ejemplos de los que proporciona jQuery UI:

- acordeón
- Autocompletado
- Botones
- datePicker
- Dialog
- Menú
- ProgressBar

Para utilizar un componente iríamos a la página oficial de jQuery UI y buscaríamos en widgets lo que quisiéramos usar en nuestra página, y en **source** tendríamos el html que queramos pegar en nuestra web.

Después, en jQuery UI tendríamos que usar las funciones correspondientes como: **menu**(), **tabs**(), entre otras.

Efectos:

- Más funcionalidades, así como tipos de efectos.
- animaciones de colores
- Transiciones

Funciones:

- .addClass()
- effect()
- switchClass()
- easing()
- toggleClass()

En cuanto a las interacciones avanzadas, con jQuery UI es lo más útil. Son eventos avanzados realizadas por los usuarios de manera frecuente. Con jQuery UI podemos:

- draggable()
- dropable()
- resizable()
- selectable()
- sortable()

CURSO FINALIZADO