DESARROLLO DE APLICACIONES FULL STACK JAVA TRAINEE

Módulo 3
Fundamentos de Desarrollo Web

MÓDULO 3 – FUNDAMENTOS DE DESARROLLO WEB

Objetivo de la jornada

- Conocer las posibilidades que brinda la librería jQuery, cómo obtenerla y agregarla adecuadamente a un proyecto
- Entender cómo manipular objetos a través de jQuery
- Analizar opciones de librerías que interactúan con jQuery

Parte 8: jQuery básico

8.1.- ¿Qué es jQuery?

JQuery es una biblioteca de JavaScript rápida, liviana y con una cantidad importante de funcionalidades. Permite que acciones tales como la manipulación de documentos HTML, el manejo de eventos y las animaciones sean bastante más simples de implementar gracias a la existencia de APIs compatibles con los navegadores más recurrentes. Además de ser una herramienta que se adapta a muchos escenarios, tiene opciones de interacción con muchos plugins externos.

En general, lo que busca jQuery es simplificar la forma en la que se desarrollan aplicaciones web, ya que al usar esta alternativa se utiliza menos código y, por ende, menos tiempo que una aplicación creada en base a JS puro o nativo. Esto la ha convertido entre los programadores en una herramienta muy popular, no olvidando por cierto que los conceptos involucrados en este tipo de plataformas evolucionan constantemente.

JQuery permite acceder a distintos elementos del DOM (textos, imágenes, enlaces, etc.), cambiar los estilos CSS en un sitio de forma dinámica o hacer peticiones asincrónicas a través de Ajax usando instrucciones simples y precisas. Es importante considerar, por cierto, que siempre se puede realizar las mismas acciones u obtener similares resultados en un sitio usando solo JavaScript, pero muchos programadores usan jQuery en gran medida por la simplicidad del código. De hecho, el lema de jQuery es “escribe menos, haz más”, partiendo de la premisa que al escribir menos código, lo errores serán menos frecuentes.

Tanto en la documentación oficial de jQuery [2] como en su sitio oficial [1] se puede encontrar mucha información respecto de sus funcionalidades. Siempre es recomendable revisar la documentación oficial de una plataforma, lo cual aplica en todo tópico nuevo que un programador tiene oportunidad de abordar.

Ya se había mencionado que, si bien en sus inicios JavaScript había sido implementado en todos los navegadores de uso masivo, tenía como inconveniente que los sitios no lograban tener comportamientos unificados frente a situaciones similares. Hoy en día, por cierto, el lenguaje se ha estandarizado minimizando las diferencias visuales y acciones divergentes, con compatibilidad casi total. jQuery, por su parte, cuenta con la lógica necesaria para adecuarse a cada navegador y versión existente de JavaScript.

8.2.- ¿Cómo agregar jQuery a un proyecto?

Al ser jQuery una biblioteca de JavaScript, es simplemente un fichero o archivo con extensión ".js", con instrucciones y acciones basadas en código implementado en dicho lenguaje. El código incluido en dicho archivo pone a disposición del programador una serie de funciones prefrabricadas que se pueden utilizar libremente.

Para usar jQuery solo se debe descargar la librería desde el sitio oficial y añadirlo a un proyecto HTML, al igual que se haría con cualquier documento con esta extensión. Al revisar el sitio, se podrá notar que existen dos versiones: la minimizada (jquery.min.js) y la "no comprimida" (jquery.js); desde el punto de vista funcional son idénticos, pero se sabe que la versión minimizada ocupará mucho menos espacio ya que elimina caracteres sobrantes (espacios, saltos de línea, comentarios, nombres de variables reducido) que no son aporte en lo funcional, solo desde la visualización. El elegir una versión minimizada, entonces, ayudará a que la carga de los sitios sea más eficiente, pero con un código ininteligible; no se debe olvidar, por cierto, que las librerías son "cajas negras" para estos efectos, ya que no es relevante su contenido en sí, sino el aporte o funcionalidades que puedan contener. Desde otra perspectiva, se puede considerar que una versión comprimida o minimizada se usa solo en fase de producción, mientras que la versión no comprimida se usa en fases de desarrollo de un proyecto web, aunque esto no debe ser considerado una imposición.

Una vez que se ha obtenido la librería jQuery, se debe agregar a un documento HTML de forma similar a lo que se hace con cualquier documento JavaScript usando la etiqueta <script>. En el ejemplo que se muestra más adelante, el atributo src tiene el valor de "jquery.js"; con esto se asume que la librería está descargada en el mismo directorio en el que se encuentra el sitio HTML que lo referencia.

```
<script src="jquery.js"></script>
```

Ilustración 1: Agregando jQuery a un sitio HTML

Otra forma de utilizar jQuery consiste en incluirlo a través de un CDN (Content Delivery Network o Red de entrega de contenidos, en español). Un CDN es una red de computadores o servidores que contienen copias redundantes de archivos, a fin que los clientes puedan acceder a ellos bajo un modelo de alta disponibilidad. Desde el sitio de jQuery se puede encontrar la dirección que se debe agregar a la etiqueta <script> para incluir la librería a través de un CDN.

En cuanto a los CDN para jQuery, es necesario destacar que empresas como Google o Microsoft prestan dicho servicio. En caso que se desee usar el CDN directo de jQuery se debe usar la siguiente instrucción.

```
<script src="https://code.jquery.com/jquery-
3.2.1.js"></script>
```

Ilustración 2: Agregando jQuery a un sitio HTML a través del CDN oficial

Lo anterior, por supuesto, implica que el equipo cliente tiene una conexión activa a Internet; de no ser así, las funcionalidades de la librería no podrán ser usadas. Siempre está la opción, por lo demás, de descargar el archivo directamente tal como indica el ejemplo anterior. Si se desea agregar la librería directamente del CDN de Google, se debe incluir el siguiente comando en el sitio.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jq
uery.min.js"></script>
```

Ilustración 3: Agregando jQuery a un sitio HTML a través del CDN de Google

8.3.- Manipulando elementos del DOM con jQuery

El término DOM proviene del idioma inglés, y significa “Document Object Model”, que traducido es el Modelo de Objetos del Documento. El DOM es una representación en estructura en modo árbol de todos los elementos de una página web. Su interacción con jQuery simplifica la sintaxis para buscar, seleccionar y manipular los elementos del DOM; se puede, por ejemplo, encontrar un elemento con una determinada propiedad, modificar un atributo específico o hacer que se asocie a un evento tal como presionar un botón en un ratón.

Es necesario considerar que no es posible interactuar de forma segura con el contenido de una página hasta que ésta no se encuentre preparada para su manipulación. jQuery

permite detectar este estado a través de la declaración $(document).ready() de forma tal que el bloque se ejecutará sólo una vez que la página este disponible.

```
$(document).ready(function() {
    console.log('el documento está preparado');
});
```

Ilustración 4: El bloque $(document).ready()

Existe además una forma abreviada para la instrucción incluida en la ilustración anterior, la cual se puede encontrar en algunos sitio; sin embargo, no es recomendable usarla en caso que esté escribiendo código para gente que no conoce jQuery.

```
$(function() {
    console.log('el documento está preparado');
});
```

Ilustración 5: Forma abreviada del bloque $(document).ready()

Por otro lado, existe la posibilidad de pasarle a $(document).ready() una función nombrada en lugar de una anónima. La sintaxis correspondiente es la que se indica a continuación.

```
function readyFn() {
    // código a ejecutar cuando el documento esté listo
}

$(document).ready(readyFn);
```

Ilustración 6: Pasar una función nombrada en lugar de una función anónima

La biblioteca soporta gran parte de los selectores establecidos en CSS3 y varios más no estandarizados. En la documentación de la plataforma [2] se puede encontrar una completa referencia sobre los selectores de la biblioteca. Algunas técnicas para seleccionar elementos son las siguientes:

Selección de elementos en base a su id

```
// notar que los IDs deben ser únicos por página
$('#myId');
```

Ilustración 7: Selección de elementos en base a su id

Selección de elementos en base al nombre de clase

```
// si se especifica el tipo de elemento,
// se mejora el rendimiento de la selección
$('div.myClass');
```

Ilustración 8: Selección de elementos en base al nombre de clase

Selección de elementos por su atributo

```
// considere que puede ser muy lento
$('input[name=first_name]');
```

Ilustración 9: Selección de elementos por su atributo

Selección de elementos en forma de selector CSS

```
$('#contents ul.people li');
```

Ilustración 10: Selección de elementos en forma de selector CSS

Pseudo-selectores

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

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

// selecciona todos los elementos del tipo input
// dentro del formulario #myForm
$('#myForm :input');

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

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

// selecciona todos los divs actualmente animados
$('div:animated');
```

Ilustración 11: Pseuso-selectores

La elección de selectores correctos es un punto importante cuando se desea mejorar el rendimiento del código. Una pequeña especificidad — por ejemplo, incluir el tipo de elemento (como div) cuando se realiza una selección por el nombre de clase — puede ayudar bastante. Es recomendable entonces dar algunas "pistas" a jQuery sobre en que lugar del documento puede encontrar lo que desea seleccionar. Por otro lado, demasiada especificidad puede ser perjudicial. Un selector como #miTabla thead tr th.especial es un

exceso, lo mejor sería utilizar #miTabla th.especial. jQuery ofrece muchos selectores basados en atributos, que permiten realizar selecciones basadas en el contenido de los atributos utilizando simplificaciones de expresiones regulares. Estos tipos de selectores pueden resultar útiles pero también ser muy lentos; cuando sea posible, es recomendable realizar la selección utilizando IDs, nombres de clases y nombres de etiquetas.

```
// encontrar todos los <a> cuyo atributo rel
// terminan en "thinger"
$("a[rel$='thinger']");
```

Ilustración 12: Selección de elementos por aproximación de atributos

Comprobar selecciones

```
if ($('div.foo').length) { ... }
```

Ilustración 13: Evaluar si una selección posee elementos

Almacenar selecciones

```
var $divs = $('div');
```

Ilustración 14: Almacenar selecciones

Refinamiento de selecciones

```
// el elemento div.foo contiene elementos <p>
$('div.foo').has('p');

// el elemento h1 no posee la clase 'bar'
$('h1').not('.bar');

// un item de una lista desordenada
// que posse la clase 'current'
$('ul li').filter('.current');

// el primer ítem de una lista desordenada
$('ul li').first();

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

Ilustración 15: Refinamiento de selecciones

Refinamiento de selecciones

- **`:button`** Selecciona elementos **`<button>`** con atributo **`type='button'`**

- **:checkbox** Selecciona elementos **<input>** con atributo **type='checkbox'**
- **:checked** Selecciona elementos **<input>** del tipo **checkbox** seleccionados
- **:disabled** Selecciona elementos del formulario que están deshabitados
- **:enabled** Selecciona elementos del formulario que están habilitados
- **: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'**
- **: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'**

Utilizando pseudo-selectores en elementos de formularios

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

Ilustración 16: Selectores en formularios

Encadenamiento

```
$('#content').find('h3').eq(2).html('nuevo texto para el
tercer elemento h3');
```

Ilustración 17: Encadenamiento

```
$('#content')
    .find('h3')
    .eq(2)
    .html('nuevo texto para el tercer elemento h3');
```

Ilustración 18: Encadenamiento con formato

Restablecer selección original

```
$('#content')
    .find('h3')
    .eq(2)
        .html('nuevo texto para el tercer elemento h3');
    // reestablece selección de elementos h3 en #content
```

```
    .end()
    .eq(0)
        .html('nuevo texto para el primer elemento h3');
```

Ilustración 19: Restablecer selección original

Moverse a través del DOM usando métodos de recorrido

```
// seleccionar el inmediato y próximo elemento <p>
// con respecto a H1
$('h1').next('p');

// seleccionar el elemento contenedor a un div visible
$('div:visible').parent();

// seleccionar el elemento <form> más cercano a un input
$('input[name=first_name]').closest('form');

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

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

Ilustración 20: Recorrido a través del DOM con funciones

Interactuando con una selección

```
$('#myList li').each(function(idx, el) {
    console.log(
        'El elemento ' + idx +
        'contiene el siguiente HTML: ' +
        $(el).html()
); });
```

Ilustración 21: Ejemplo de interacción con una selección

8.4.- Eventos en jQuery

Un evento es una acción específica que un visitante de un sitio puede realizar sobre una página web que tiene una interacción directa con el navegador; representa el momento en el que se ejecuta la tarea. Ejemplos de eventos son:

- hacer clic sobre un contenedor
- posicionar el mouse sobre un elemento específico
- seleccionar un checkbox

En el siguiente cuadro se muestran ejemplos de eventos por cada categoría de contenedor de un elemento.

Evento	Descripción
blur()	Conecta / desencadena el evento de desenfoque
change()	Conecta / activa el evento de cambio
click()	Conecta / activa el evento de clic
dblclick()	Conecta / activa el evento de doble clic
event.currentTarget	El elemento DOM actual dentro de la fase de burbujeo del evento
event.data	Contiene los datos opcionales transferidos a un método de evento cuando el controlador de ejecución actual esta enlazado
event.delegateTarget	Devuelve el elemento donde se ha conectado el controlador de eventos jQuery actualmente llamado
event.isDefaultPrevented()	Devuelve si event.preventDefault () fue llamado para el objeto de evento
event.isImmediatePropagationStopped()	Devuelve si event.stopImmediatePropagation () se llama para el objeto de evento
event.isPropagationStopped()	Devuelve si event.stopPropagation () fue llamado para el objeto de evento
event.namespace	Devuelve el espacio de nombres especificado cuando se desencadena el evento
event.pageX	Devuelve la posición del ratón en relación con el borde izquierdo del documento
event.pageY	Devuelve la posición del mouse en relación con el borde superior del documento
event.preventDefault()	Evita la acción predeterminada del evento

Evento	Descripción
event.relatedTarget	Devuelve el elemento entrado o salido en el movimiento del ratón.
event.result	Contiene el ultimo / anterior valor devuelto por un manejador de eventos disparado por el evento especificado
event.stopImmediatePropagation()	Evita que se llamen a otros controladores de eventos
event.stopPropagation()	Evita que el evento burbujee el árbol DOM, evitando que los manejadores de origen sean notificados del evento.
event.target	Devuelve el elemento DOM que ha desencadenado el evento.
event.timeStamp	Devuelve el número de milisegundos desde el 1 de enero de 1970, cuando se desencadena el evento
event.type	Devuelve que tipo de evento se ha activado
event.which	Devuelve la tecla del teclado o el botón del ratón para el evento
focus()	Conecta / activa el evento de enfoque
focusin()	Conecta un controlador de eventos al evento focus
focusout()	Conecta un controlador de eventos al evento de enfoque
hover()	Conecta dos manejadores de eventos al evento hover
keydown()	Conecta / activa el evento keydown
keypress()	Conecta / activa el evento de pulsaciones de teclas
keyup()	Conecta / activa el evento keyup
mousedown()	Conecta / activa el evento mousedown
mouseenter()	Conecta / activa el evento mouseenter

Evento	Descripción
mouseleave()	Adjunta / activa el evento mouseleave
mousemove()	Adjunta / activa el evento mousemove
mouseout()	Conecta / activa el evento mouseout
mouseover()	Conecta / activa el evento mouseover
mouseup()	Adjunta / desencadena el evento mouseup
off()	Elimina los manejadores de eventos conectados con el método on ()
on()	Agrega los controladores de eventos a elementos
one()	Agrega uno o más controladores de eventos a elementos seleccionados. Este controlador se lo se puede activar una vez por elemento
$.proxy()	Toma una función existente y devuelve una nueva con un contexto particular
ready()	Especifica una función a ejecutar cuando el DOM está completamente cargado
resize()	Adjunta / activa el evento de cambio de tamaño
scroll()	Conecta / activa el evento de desplazamiento
select()	Conecta / activa el evento seleccionado
submit()	Adjunta / activa el evento de envío
trigger()	Activa todos los eventos vinculados a los elementos seleccionados
triggerHandler()	Dispara todas las funciones enlazadas a un evento especificado para los elementos seleccionados

A continuación se muestran algunos ejemplos de eventos usando jQuery.

Cambio de estilo en pérdida de foco

```
$("input").blur(function(){
  $(this).css("background-color", "#ffffff");
});
```

Ilustración 22: Evento blur()

Múltiples eventos desencadenados

```
$("p").on({
  mouseenter: function(){
    $(this).css("background-color", "lightgray");
  },
  mouseleave: function(){
    $(this).css("background-color", "lightblue");
  },
  click: function(){
    $(this).css("background-color", "yellow");
  }
});
```

Ilustración 23: Eventos múltiples desencadenados

8.5.- Integrando jQuery y Ajax

AJAX es una herramienta que permite intercambiar datos con un servidor, y actualizar al mismo tiempo partes de una página web sin tener que recargar el sitio completo. La sigla viene como siempre del idioma inglés, y significa “Asynchronous JavaScript and XML”, que en español significa “Javascript asincrónico y XML”.

En términos simples, AJAX dice relación con cargar datos en segundo plano y mostrarlos en un sitio, sin cargar todo el sitio. Ejemplo de aplicaciones que usan AJAX son Facebook, Youtube, Google Maps, Google Chrome, entre otros.

jQuery posee varios métodos para agregar funcionalidad a través de AJAX. Con estos métodos es posible solicitar texto, HTML, XML o JSON desde un servidor remoto usando HTTP Get y HTTP Post. Toda esta información recibida puede ser cargada en un sitio, aplicando incluso estilo o bien dándole un formato determinado. Sin jQuery, la implementación de AJAX puede ser un poco complicada, dado que la implementación puede depender en algún punto del navegador que se está usando; esto implica tener código adicional para diferentes navegadores. Sin embargo, el equipo de desarrollo de

jQuery ha hecho las adaptaciones necesarias para que se pueda escribir una sola línea de código funcionalidades que antes se debían hacer con múltiples líneas.

Para entender como interactúan ambas metodologías, se mostrará un ejemplo. Imagine el archivo ajax_info.txt; debe estar cargado en la misma ubicación que la página que lo referenciará.

```
<h1>AJAX</h1>
<p>AJAX no es un lenguaje de programación.</p>
<p>AJAX es una técnica para acceder a servidores web
desde una página web.</p>
<p>AJAX significa JavaScript asincrónico y XML.</p>
```

Ilustración 24: Archivo ajax_info.txt

En el primer ejemplo se usará JavaScript para la carga; el ejemplo quedará como se muestra a continuación:

```
<!DOCTYPE html>
<html>
    <head>
        <script language="javascript">
            function loadDoc() {
                alert("Hola");
                var xhttp = new XMLHttpRequest();
                xhttp.onreadystatechange = function() {
                    if (this.readyState == 4 && this.status == 200) {
                        document.getElementById("demo").innerHTML = this.responseText;
                    }
                };
                xhttp.open("GET", "ajax_info.txt", true);
                xhttp.send();
            }
        </script>
    </head>
    <body>
        <div id="demo">
        <h2>AJAX cambiará este texto</h2>
        <button type="button" onclick="loadDoc()">Cambiar Contenido</button>
        </div>
    </body>
</html>

```

Ilustración 25: Utilizando AJAX a través de JavaScript

Como se puede ver, el código JavaScript presenta una serie de elementos que pueden ser molestos en el desarrollo, y que son prácticamente una “caja negra” para el usuario. Independiente de lo anterior, lo que se busca es actualizar el texto de un contenedor al presionar un botón, usando como fuente un archivo de texto.

Como alternativa al ejemplo anterior está jQuery, el que a través de funciones bastantes más simples logra el mismo resultado.

Ilustración 26: Integración de jQuery y AJAX

Parte 9: Plugins

9.1.- ¿Qué es plugin y cuándo usarlo?

Los plugins son los complementos que jQuery pone a disposición de los desarrolladores, con la finalidad de ampliar las funcionalidades del framework.

En la práctica se deben usar bajo las siguientes consideraciones:

- para resolver problemas complejos
- para resolver una necesidad específica
- para generar rutinas que puedan impactar en el mediano plazo en cualquier sitio web

En términos simples, un plugin es una función que se añade al objeto jQuery, para que a partir de ese momento responda a nuevos métodos.

9.2.- Plugins más comunes

En el mercado existen muchos complementos para jQuery, de uso tanto pagado como gratuito. Dentro de los que siguen vigentes y que se han actualizado con el paso del tiempo, se pueden destacar los siguientes.

Fancybox

Permite mostrar varios tipos de medios en una ventana emergente [5]. Acepta igualmente imágenes, videos y mapas, pudiendo adaptar sus dimensiones. Es de uso gratuito para proyectos de código abierto, o licenciado para efectos comerciales.

jQuery File Upload

Es un complemento con posibilidad de seleccionar múltiples archivos para subir a un sitio, con barra de progreso, validación, audio y video [6]. Otra ventaja es su amplia compatibilidad con diversos lenguajes de programación.

Select2

Este plugin provee un casilla de selección personalizable, con opción para búsqueda de datos, obtención de datos desde fuentes remotas, entre otras funciones. Dentro de las características que destaca el proveedor [7], se deben considerar la traducción a múltiples

idiomas, la posibilidad de ser usado en los navegadores más comunes y la integración de AJAX para la transferencia de gran cantidad de ítems en forma de lista. Es un proyecto de código abierto, y se puede acceder desde el sitio al repositorio en GitHub.

Datatables

Permite ordenar datos en una tabla, con paginación incluida, búsqueda, ordenamiento de registros, entre otras opciones [8]. Es de uso gratuito para todo efecto.

Este plugin es ampliamente usado, y tiene una red importante de desarrolladores que adhiere mejoras de forma continua.

Bootstrap

Y finalmente, no se puede dejar de mencionar esta herramienta. Si bien no es plugin en si mismo, entrega importantes funcionalidades [9]. Bootstrap es un conjunto de herramientas que permiten construir diseños adaptables bajo el concepto “mobile-first”. Cuenta con su propio CDN, y hace uso de elementos derivados tanto de jQuery como de CSS. Dada la relevancia que representa esta plataforma, es totalmente recomendable dar un vistazo por su sitio oficial.

Anexo: Referencias

[1] jQuery
Referencia: https://jquery.com/

[2] jQuery API
Referencia: https://api.jquery.com/

[3] Ejemplos de jQuery
Referencia: https://www.w3schools.com/jquery/default.asp

[4] AJAX
Referencia: https://learn.jquery.com/ajax/

[5] Fancybox
Referencia: http://fancyapps.com/fancybox/3/

[6] jQuery File Upload
Referencia: https://blueimp.github.io/jQuery-File-Upload/

[7] Select2
Referencia: https://select2.org/

[8] Datatables
Referencia: https://datatables.net/

[9] Bootstrap
Referencia: https://getbootstrap.com/