Unidad 3 / Escenario 5

Lectura Fundamental

Librería jQuery

Contenido

Palabras Claves: Librería jQuery, Librería de JavaScript, Manipulación del DOM, Eventos

Implementar el comportamiento de los elementos en una página Web se realiza a través de código JavaScript que principalmente debe recuperar los elementos a partir del DOM permitiendo luego poder acceder y/o modificar sus atributos, así como agregar manejadores, entre otras opciones más.

1. jQuery

Una de los beneficios de utilizar jQuery en lugar de escribir código JavaScript tradicional está en que simplifica y facilita la lectura al mismo tiempo que su código se ejecuta más rápido que su contraparte en JavaScript plano. La librería jQuery utiliza métodos modernos del DOM para optimizar la recuperación de elementos entre otras tareas.

En una página se reúne código HTML junto con reglas CSS y funcionalidad JavaScript, tal como se menciona en **??** estas tres partes deberían estar separadas dentro del documento la figura 1 muestra la ubicación de cada una de las partes dentro del documento HTML.

```
<html>
    <head>
        <!-- 1. Estilo (CSS)-->
    </head>
    <body>
        <!-- 2. Estructura (HTML) -->
        <!-- 3. Comportamiento
     (JavaScript)     -->
    </body>
</html>
```

Figura 1: CSS y JavaScript dentro de una página HTML

2. Configurar jQuery

2.1. Utilizando un CDN

Una de la maneras más sencillas de incluir la librería jQuery dentro de una página HTML es por medio de un CDN (Content Delivery Network) en este caso MaxCDN. Basta con ingresar la siguiente etiqueta `script` dentro de la etiqueta `head` o `body` de nuestra página HTML.

```
<script
  src="http://code.jquery.com/jquery-3.2.1.js"
  integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="
  crossorigin="anonymous"></script>
```

Agregaremos jQuery una página HTML que contiene únicamente un encabezado con el mensaje ‘Welcome to

jQuery'. Para verificar el correcto funcionamiento de jQuery utilizaremos la función `jQuery` definida en la librería que permite consultar elementos en el DOM de la página y agregaremos algunas reglas para estilizar el encabezado.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script
  src="http://code.jquery.com/jquery-3.2.1.js"
  integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="
  crossorigin="anonymous"></script>
<script type="text/javascript">
    jQuery(() =>{
        jQuery('h1').css('color','blue').css('font-size','18pt')
    });
</script>
</head>
<body>
<h1>Welcome to jQuery</h1>
</body>
</html>
```

En el código anterior se utilizó jQuery para modificar el color y el tamaño del encabezado a azul y 18pt respectivamente, aquí la función `jQuery` permitió manipular el encabezado luego de buscarlo por el nombre del elemento `h1`. Comúnmente se prefiere utilizar el alias `$` en lugar del identificador `jQuery`, lo que haría que nuestro código JavaScript se viera de la siguiente manera:

```
    $(() =>{
        $('h1').css('color','blue').css('font-size','18pt')
    });
```

3. Acceder al DOM con jQuery

Al utilizar JavaScript es común querer acceder a los elementos del DOM para consultarlos, modificarlos o agregar nueva funcionalidad, por esta razón jQuery proporciona los mecanismos necesarios para realizar estas tareas de una manera sencilla y clara, estas herramientas son los métodos para recorrer (traversal methods) y los selectores (selectors). Cabe aclarar que el objeto jQuery retornado por la consulta permite acceder a los elementos correspondidos por el selector, sin embargo, estos no son los elementos DOM convencionales y por tal razón no proporcionan los mismos métodos.

Para consultar elementos del DOM jQuery proporciona la función `$()` que recibe un selector CSS y retorna un nuevo objeto de jQuery que permite acceder a los elementos retornados por la búsqueda, así como enlazar eventos a estos elementos o agregar efectos o encadenar modificaciones. El símbolo `$` es solo un alias de `jQuery` su único propósito es el de reducir el código, aunque al principio parece algo extraño dado que el símbolo se asemeja más

un operador en poco tiempo el uso del símbolo se vuelve claro y generalmente se prefiere en lugar del nombre `jQuery`.

Las consultas a elementos del DOM se hacen por medio de los selectores CSS especificando por identificador (ID), nombre de etiqueta o clase, pueden utilizarse individualmente como también de manera combinada.

Utilizar la función `jQuery` con su alias `$` para encontrar todos los elementos de tipo `input` en una página puede lograrse invocando la función:

```
$('input')
```

note que en el argumento `'input'` se especifica únicamente el nombre del elemento, lo que significa que basta con poner el nombre de un elemento HTML para llevar a cabo la consulta. Si deseamos encontrar elementos por su ID en lugar de consultar por el nombre del elemento debemos anteponer el símbolo `#` al ID de la misma manera en que se hace en CSS. Dado que los identificadores en un documento deberían ser únicos, la consulta por el identificador deberá retornar un único elemento. La siguiente invocación de la función `jQuery` permite buscar y recuperar del DOM un elemento por el identificador `'password'`:

```
$('#password')
```

La tercera manera para consultar elementos del DOM es hacerlo mediante la clase CSS, para ello se anteponte el símbolo punto `(.)` al nombre de la clase tal como se especifica en las reglas CSS. Por ejemplo:

```
$('.anyclass')
```

Dado que el resultado de invocar la función `jQuery` es un objeto de jQuery que incluye los elementos resultado de la consulta, este objeto proporciona diferentes métodos que actúan sobre el conjunto de elementos sin necesidad de recorrer explícitamente la lista. La librería jQuery soporta casi todos los selectores incluidos en las especificaciones CSS desde la 1 a la 3. Dado que la consulta se hace a través de selectores un buen conocimiento de estos le permitirá utilizar más efectivamente la librería y podrá facilitar las consultas de elementos específicos o tediosos de buscar de la manera tradicional sin jQuery.

4. Métodos para Recorrer el DOM

A través del objeto jQuery que retorna la función `$()` cuando se consulta por un selector CSS dado, es posible invocar el método `filter()` que permite seleccionar del conjunto de elementos devueltos por la consulta aquellos elementos que cumplan una condición dada. El filtro se puede especificar a través de un selector CSS o pasando una función que lleva a cabo una validación para un elemento dado, esta función es invocada para cada uno de los elementos retornados por la consulta y de acuerdo al valor devuelto el elemento es removido o permanece en la lista. Note que este método es invocado posteriormente a una consulta previa y de esta manera realiza un recorrido sobre el conjunto de elementos resultantes.

La sentencia `$('input')` busca y retorna todos los elementos `input` del DOM, los cuales podrían incluir cajas de texto, botones de opción, casillas de selección entre otros. Utilizando el método `filter()` es posible obtener únicamente aquellos que sean de tipo `checkbox`, así:

```
$('input').filter('[type=checkboxes]')
```

Los métodos se pueden encadenar lo que permite llevar a cabo un filtro después de otro. Con la siguiente sentencia es posible seleccionar no solo los elementos `input` de tipo `checkbox` sino además lo que estén seleccionados.

```
$('input').filter('[type=checkbox]').filter(':checked')
```

Por supuesto esto puede ser simplificado mediante la sentencia:

```
$('input[type=checkbox]:checked')
```

Sin embargo, pasar una función al método `filter()` en lugar de un selector CSS, permite hacer validaciones más elaboradas o que no pueden realizarse a través de selectores CSS.

5. Eventos

HTML define un conjunto de acciones semánticas incorporadas que no requieren el uso de JavaScript, por ejemplo, recargar páginas como resultado de enviar un formulario al hacer clic en un elemento `input` de tipo `submit` o al hacer clic sobre un enlace. Sin utilizar JavaScript, las páginas HTML carecen de las características que frecuentemente encontramos en aplicaciones de escritorio, no habría forma de realizar acciones específicas de acuerdo a la interacción que tienen los usuarios con las páginas. Por medio de JavaScript es posible definir el comportamiento de la interfaz gráfica, para ello se definen manejadores de eventos, esto son, códigos que son ejecutados en respuesta a un evento, un suceso que puede ser generado por el sistema o bien por la interacción del usuario, mostraremos como utilizar jQuery para definir estos manejadores.

El código que será ejecutado para manejar el evento, dar respuesta a un suceso, es encapsulado en una función, cuya referencia es asignada a una de las propiedades sobre el elemento. Un botón tiene el atributo `onclick` que espera sea indicada el nombre de una función JavaScript que será invocada cuando un evento de tipo clic ocurra sobre el elemento. De igual manera un elemento formulario (`form`) tiene atributos `onsubmit` y `oncancel` que hacen referencia a eventos en que el formulario ha sido enviado o se ha cancelado el envío.

Cuando un manejador de evento es lanzado una instancia de un objeto llamado `Event` es pasado al manejador como primer argumento. Esto objeto contiene información útil sobre el evento ocurrido, generalmente incluye información sobre cual elemento lanzó el evento, las coordenadas de eventos del mouse, entre otras.

La librería jQuery proporciona un modelo de eventos que permite agregar manejadores de eventos a los elementos del DOM a través de su método `on()`. Este método permitirá agregar varios manejadores a un mismo elemento invocando el método directamente sobre el objeto resultante de una consulta.

Suponga que ha desarrollado una página que muestra un listado de problemas de una competencia de programación, por simplicidad cada problema consta de un identificador, título y descripción, esta última no es mostrada en la tabla debido a lo extensa que podría ser un descripción. Para ello se emplea una tabla en la cual cada una de las filas corresponde a la información de un problema y una de las columnas habilita la opción de eliminar problema a través de un botón. Un ejemplo de esta página se puede observar en la figura 2.

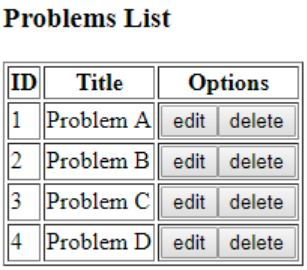


Figura 2: Tabla con lista de problemas y opciones de editar y eliminar.

Un comportamiento no deseado en la página es que permita la eliminación con tan solo hacer clic sobre el botón de eliminar sin mostrar un mensaje de confirmación. Utilizando la función jQuery para consultar todos los elementos `input` de tipo `button` cuyo atributo `value` sea 'delete' y el método `on()` se puede agregar un manejador para el evento clic que muestre un mensaje de confirmación. Esto se puede lograr de la siguiente manera:

```
$(()=>{
    $('input[type=button][value=delete]').on('click', (e)=>{
        return confirm('Are you sure you want delete this problem?');
    });
})
```

En el código anterior se invoca el método `on()` pasándole como argumentos la cadena `'click'` que significa que el manejador se asignará al evento clic. Por otra parte el manejador está siendo especificado a través de la definición de una función flecha (arrow function) la cual es anónima y cuya sintaxis consta de dos partes separadas por los símbolos =>, la parte de los parámetros y el bloque de la función. Más sobre las funciones flecha, puede ser encontrado en Mozilla Developer Network y colaboradores individuales, 2017.

Cuando `false` es retornado en un manejador de eventos agregado por medio de la función `jQuery` es equivalente a invocar los métodos `preventDefault()` y `stopPropagation()`. En cambio, al retornar `false` en un manejador agregado utilizando métodos nativos es equivalente a invocar únicamente `preventDefault()`.

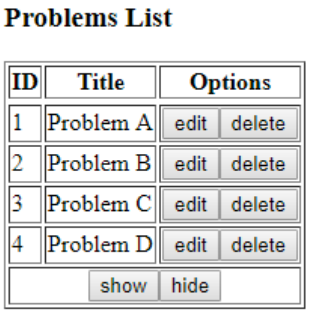
6. Animaciones y Efectos

Utilizando jQuery es posible crear animaciones y efectos. La librería proporciona los métodos `show()` y `hide()` que como su nombre lo sugiere muestran o hacen visible y ocultan elementos respectivamente. Para ocultar un elemento jQuery modifica el valor de la propiedad `style.display` haciéndola `none`. Si se invoca a un conjunto de elementos retornados por una consulta se ocultará cada uno de los elementos y en caso que alguno ya se encuentre oculto se mantendrá igual. De similar manera, cuando jQuery muestra un elemento modifica el atributo `style.display` con el valor `block` o `inline`, el valor a elegir depende del valor que el elemento tenía antes de que fuera ocultado o en caso de no haber sido explícitamente puesto del valor por defecto del elemento para ese atributo.

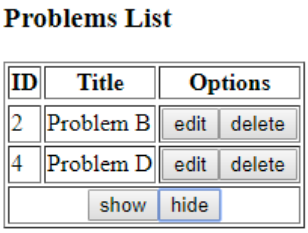
Como ejemplo, suponga que en el anterior código HTML del listado de problemas deseamos permitir que las filas pares se muestren u oculten. Para ello agregamos un par de botones al final de la tabla uno para ocultar y otro mostrar las filas, agregaremos un manejador para el evento clic en cada uno de los botones, consultaremos las columnas pares e invocaremos el método `show()` y `hide()` respectivamente.

```
$(()=>{
    $('input[type=button][value=show]').on('click', (e)=> {
        $('tbody tr:even').show();
    });
    $('input[type=button][value=hide]').on('click', (e)=> {
        $('tbody tr:even').hide();
    });
})
```

Esto permitirá que al hacer clic en el botón hide se oculten las filas pares de la tabla y con el botón show se muestren. Note que los métodos `show()` y `hide()` son aplicados a todos los elementos resultantes de la consulta.



(a) Tabla con todas las filas, al iniciar o luego de hacer clic en el botón show.



(b) Tabla con las filas pares ocultas, luego de hacer clic en el botón hide.

Figura 3: Tabla con botones que permiten ocultar y mostrar las filas pares.

Referencias

Bibeault, B., Katz, Y., & De Rosa, A. (2015). *Jquery in action.* "Manning Publications Co."

Flanagan, D. (2011). *Javascript: the definitive guide: activate your web pages.* " O'Reilly Media, Inc."

Freeman, E. T. & Robson, E. (2014). *Head first javascript programming: a brain-friendly guide.* " O'Reilly Media, Inc."

Mozilla Developer Network y colaboradores individuales. (2017). Arrow functions - javascript — mdn. Recuperado desde https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

INFORMACIÓN TÉCNICA

Módulo: Desarrollo de Front-End
Unidad 3: Librerías jQuery y jQuery UI
Escenario 5: Librería jQuery

Autor: Diego Satoba

Asesor Pedagógico: Ingrid Ospina Posada
Diseñador Gráfico: Catalina López
Asistente: Angie Laiton