Tema 7: jQuery.

Fecha entrega: 14-01-2019

Autor: Guillermo Boquizo Sánchez

Códigos manual jQuery. Parte I, II y III.

Entrega de los códigos del manual jQuery. Parte 1,2 y3.
En un documento pdf responde a las siguientes preguntas:

1. En una línea, define qué es jQuery.
2. Identifica la última versión jQuery
3. Indica las diferencias entre la versión DEVELOPMENT, PRODUCTION y SLIM.
4. Indica la línea donde introduces todo el código de la librería jQuery.
5. Indica qué es el jQuery CDN.
6. Indica brevemente y con tus palabras las ventajas del CDN de jQuery.
7. Indica la línea donde introduces las últimas versiones de al menos dos jQuery CDN.
8. Indica cómo jQuery ejecuta un código cuando el árbol DOM está totalmente cargado. Indica el equivalente en javaScript.
9. Función $ o función jQuery. Indica brevemente los argumentos que puedes enviarle. Añádele a la explicación un breve código de ejemplo (distinto al del manual)
10. Indica cómo puedes reemplazar el clásico $(document).ready(){...} con jQuery
11. En una línea, explica qué hace el método each() de jQuery. Explica qué es la iteración implícita.
12. Indica el argumento que ha de enviársele al método each().
13. Englobado en el contexto del each:
    1. Explica la utilidad de la palabra reservada this.
    2. Indica cómo se utiliza el índice de la iteración.
    3. Explica la utilidad de return false.
    4. Indica la diferencia entre return true y no ponerlo. Explícalo mediante un trozo de código.
14. Indica las diferencias y semejanzas entre el método size() y la propiedad length. Indica las ventajas e inconvenientes de utilizar uno u otra.
15. Indica qué hace el método data().
16. Tipos de datos admitidos por data()
17. Indica qué significa que data() almacena valores por referencia.
18. Cuántos objetos se crean si data() opera sobre un conjunto de elementos.
19. Indica qué hace el método removeData().
20. Identifica con tu propio código (en una línea a ser posible) los distintos tipos de selectores. Indica cómo se recogen en una variable.

1

jQuery es un framework o librería de JavaScript, un producto que actúa como base para la programación avanzada de aplicaciones, aportando una serie de funciones o códigos para realizar tareas habituales. El framework consiste en una librería de código con procesos y rutinas ya preparados para su uso.

2

La última versión disponible de jQuery es la 3.3.1.

3

La versión **DEVELOPMENT** o de desarrollo tiene como finalidad ser empleada en la etapa de desarrollo del código, por lo cual no posee minificación, presenta una buena estructuración y un buen formateo del código por si es necesario su aprendizaje por parte del desarrollador.

La versión **PRODUCTION** o de producción sí presenta minificación, con la finalidad de ocupar cuantos menos bytes mejor para evitar sobrecargas en la visualización de la página, mejorar la eficiencia del código, aun a costa de la legibilidad del mismo.

La versión **SLIM**, para terminar, excluye alguna funcionalidad, como por ejemplo ajax o los módulos de efectos, de manera que el espacio que ocupa la librería es menor, está pensada para proyectos rápidos y simples, y se encuentra disponible tanto en su versión de producción como en su versión de desarrollo.

4

En el head, podemos cargarlo de dos maneras, dependiendo de si se hace o no uso del CDN:

```
//Usando CDN
<script
     src="https://code.jquery.com/jquery-3.3.1.mi
     n.js"
     integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZ
     xltwRo8QtmkMRdAu8="
     crossorigin="anonymous">
</script>

//Mediante importación de la librería
<script
     type=”text/javascript”
     src=”rutaDelJS”>
</script>
```

5

CDN (Content Delivery Network o Red de Distribución de Contenido) es un servicio de jQuery que nos permite incluir las librerías de código del framework desde servidores de terceros. Este conjunto de servidores, con diferentes localizaciones, contienen copias locales de los contenidos de los clientes.

Cuando se emplea CDN, en vez de enlazar con los scripts de jQuery alojados en nuestro sitio, se linka con una URL de otro dominio que los aloja por nosotros. El uso de esta tecnología permite una mayor velocidad de entrega de nuestro sitio.

6

El uso de servidores CDN presenta las siguientes ventajas:

1. Mayor velocidad de entrega:

   Los servicios CDN están ofrecidos por grandes empresas, con replicación de servidores y diversas localizaciones de entrega a lo largo del mundo. Posiblemente Google, o cualquiera de los otros proveedores CDN, pueda enviar el script jQuery más rápido que el servidor local propio, y lo distribuya desde una localización más cercana a la red del cliente que lo visita.

2. Cacheado probable:

   Es muy probable que la persona que visita un sitio ya tenga cacheado el script jQuery tras visitar otro sitio que también esté usando CDN. Por ello, quizá no tenga ni que esperar a la descarga del framework y utilice la copia cacheada de su navegador.

No obstante, el manejo de CDN también presenta algún inconveniente:

1. Precisa de una conexión a internet para conectarse a CDN:

   Durante el desarrollo del sitio, si se está en modo offline, se precisa de acceso a la copia local de las librerías para que la web funcione.
   Si se precisa hacer pruebas en entornos sin internet, se requiere del mismo modo la copia local de la librería.

2. Pérdida de control:

   No se tendrá acceso al script para su modificación, en caso de necesidad, al encontrarse alojado en otro servidor.

7

En el head:

```
<script
      src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.
      1/jquery.min.js">
</script>

<script
      src="https://code.jquery.com/jquery-3.3.1.min.js"
      integrity="sha256-FgpKJQlLNfOu91ta32o/NMZxltwRo8QtmkMR
dAu8="    crossorigin="anonymous">
</script>
```

8

```
      //jquery
      $(document).ready()
      $(handler)

      //Vanilla JS
      window.addEventListener(handler)
      document.addEventListener(handler)
```

9

```
      $(“#id”) //selector, selecciona aquellos elementos del
      árbol DOM
            coincidentes
      $(“<p>Prueba creación</p>) //crea un elemento HTML

      $(“span”, $(this)) //Cambia el contexto del elemento.
```

10

```
$(function());
```

11

El método .each() de jQuery se emplea para recorrer en un objeto jQuery, de manera similar a la empleada en Vanilla JS cuando se hace uso del forEach(). Este método realizará las acciones determinadas con todos los elementos coincidentes con una selección dada en la función jQuery.

La mayoría de los métodos jQuery que devuelven objetos jQuery también iteran sobre el conjunto de elementos de la collection jQuery.

Este proceso, conocido como iteración implícita, es una característica propia de jQuery a través de la cual se realiza un recorrido no declarado, implícito, para añadir determinada funcionalidad a todos los elementos de dicha collection. Cuando esto ocurre, no es necesario explicitar la iteración con el método .each(). Así:

```
//itera de manera implícita sobre todos los li y les añade la clase bar.
$( “li” ).addClass( “bar” )
```

12


```
index:
      each(index)
element:
      each(element) //si se desea hacer una
                        modificación sobre algún elemento
```

13

1. La palabra reservada “this” tiene como utilidad referenciar al elemento del DOM que lo lanza.
2. .each(index):

   Recorre todos los elementos con el selector indicado.
3. Tiene como finalidad detener por completo la iteración.
4. Al devolver “return true” se pasa directamente a la siguiente iteración del bucle. Si no se indica, se sigue comprobando las demás líneas de código hasta llegar a una nueva iteración.

```
let palabra = coincidencia
if(palabra === “coincidencia”)
     return true;
```

14

El método size() devuelve el número de elementos seleccionados mientras que la propiedad length devuelve y almacena dicho número, ganando en rapidez, por lo que se aconseja.

Tanto size() como length son equivalentes y van a devolver un mismo valor, pero size(), internamente, realiza una llamada a una function que accede a la propiedad .length.

En jQuery 1.8 se consideró deprecated .size(), por razones de rendimiento y duplicidad en la funcionalidad, y fue eliminado en jQuery 3.0, por lo que se desaconseja su uso en versiones modernas de jQuery.

15

El método .data() permite asociar cualquier tipo de dato a elementos del DOM, de modo que se eviten referencias circulares y agujeros de memoria y puedan ser reutilizados posteriormente en el script. A su vez, también permite la lectura de datos previamente asociados con elementos DOM

Presenta la siguiente sintaxis:

A la hora de almacenar:

Pueden establecerse distintos valores para un elemento individual, y ser recuperados posteriormente.

```
$( "body" ).data( "foo", 52 );
$( "body" ).data( "bar", { isManual: true } );
$( "body" ).data( { baz: [ 1, 2, 3 ] } );
$( "body" ).data( "foo" ); // 52
$( "body" ).data(); // { foo: 52, bar: { isManual: true }, baz: [ 1, 2, 3 ] }
```

Ejemplo de uso:

```
$( '#user_mail' ).data( 'status', {
  to : 'info@anyaddres.com',
  subject : 'My Subject',
   mailContent : 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.',
  response : 'send'
} );
```

A la hora de realizar la lectura:


```
var elem = document.createElement( "span" );
$( elem ).data( "foo" ); // undefined
$( elem ).data(); // {}

$( elem ).data( "foo", 42 );
$( elem ).data( "foo" ); // 42
$( elem ).data(); // { foo: 42 }
```

Invocando a .data() sin parámetros se devuelve un objeto JavaScript que contiene cada valor almacenado como propiedad. El objeto puede ser usado directamente para obtener valores de dichos datos.

16

Como conjunto de pares clave-valor, cada clave será de tipo String, mientras que cada valor puede ser de cualquier tipo JavaScript con la excepción de undefined.

Además, .data también soporta cualquier tipo de objeto.

17

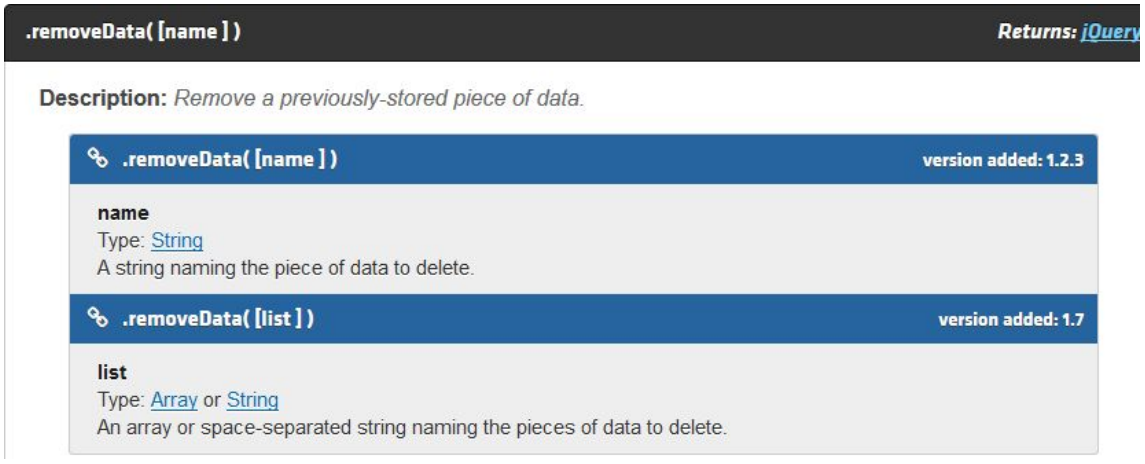
.data() almacena valores por referencia, esto es, no se realizarán copias independientes del objeto a almacenar, sino que dicho objeto permanece inmutable y lo que se asigna como dato es una referencia a ese único objeto.
De este modo, se garantiza un modo eficiente del uso de la memoria.

18

Se guarda un dato por cada elemento del objeto jQuery seleccionado.
En caso de que en dicho objeto existan varias referencias a distintos elementos de la página, el dato se almacena en todos los elementos.

19

El método removeData() se emplea para eliminar un dato almacenado previamente en un elemento. Su funcionamiento se basa en el envío por parámetro del dato que se desea eliminar del elemento.



20

```
$(document).ready(function() {
	$(“#button”).click(function(evento) {
		let selectorId = $(“#id”).attr(“value”);
		let selectorTag = $(“span”);
		let selectorClase = $(“.clase”).attr(“value”);
		let selectorMultiTags = $(“.clase1.clase2”).attr(“value2);
		let selectorAll = $(“*”);
	});
});
```