Diseño Web — Tecnologías Involucradas

Requerimientos Participantes del Curso

info@covetel.com.ve [1]

[1] Cooperativa Venezolana de Tecnologías Libres R.S.

Requerimientos Participantes del Curso

Diferencia entre .attr() y .prop()

Diferencia entre .attr y .prop()

En la versión 1.6 de JQuery se han dividido los métodos para el manejo de atributos y propiedades DOM.

- Atributos DOM: *representan el estado de la información DOM una vez recuperados del Documento.*
  Ejemplo

```
1  <input type="text" value="abc">
```

- Propiedades DOM: *representan el estado dinámico del documento. .prop() selecciona u obtiene las propiedades de los elementos DOM.*
  Ejemplo
  Si un usuario hace click en un elemento de entrada y define .prop("value") como abcdef .attr("value") seguirá siendo abc.

Diferencia entre .attr y .prop()

En la mayoría de los casos, el navegador trata el valor del atributo como el valor inicial, pero los atributos booleanos como el checked o disabled tienen una semántica inusual.

Ejemplo

Considere la etiqueta:

```
1  <input type="checkbox" checked>
```

La presencia del atributo checked significa que la propiedad DOM .checked es verdadera, a pesar que el atributo no tiene valor. En el código anterior el valor del atributo checked es una cadena vacía pero el valor de la propiedad checked es verdadero.

Antes de JQuery 1.6 .attr(ẹchecked") devolvía el valor de la propiedad booleana (true) pero a partir de JQuery 1.6 devuelve el valor real del atributo (una cadena vacía), que no cambia cuando el usuario hace click en la casilla de verificación para cambiar su estado.

Método .noConflict()

Muchas bibliotecas JavaScript usan $ función o nombre de variable, como lo hace JQuery. $ es solo un alias para JQuery, por lo que toda la funcionalidad está disponible sin necesidad de utilizar $. Si tenemos que usar otra librería de JavaScript junto con Jquery podemos devolver el control de $ a la biblioteca JavaScript usando el método .noConflict():

Ejemplo

```
<script type="text/javascript" src="other_lib.js"> </ script>
<script type="text/javascript" src="jquery.js"> </ script>
<script type="text/javascript">
  . $ NoConflict ();
  jQuery (document). ready (function ($) {
  //El codigo que utiliza $ jQuery puede seguir aqui.
  });
  //El codigo que utiliza otras bibliotecas $ puede seguir aqui.
</script>
```

```
jQuery.noConflict();
//Algo con jQuery
jQuery("div p").hide();
//Algo con otra libreria $()
$("content").style.display = 'none';
```

Manejo de Objetos, Interacción de Array con JQuery

- .each() *: método de iteración genérico, que puede ser usado para recorrer sin problemas los objetos y las matrices. Las matrices y los objetos de matriz, como una propiedad de longitud.*

Ejemplo

```
<body>
  <div id="one"></div>
  <div id="two"></div>
  <div id="three"></div>
  <div id="four"></div>
  <div id="five"></div>
<script>
    var arr = [ "one", "two", "three", "four", "five" ];
    var obj = { one:1, two:2, three:3, four:4, five:5 };

    jQuery.each(arr, function() {
      $("#" + this).text("Mine is " + this + ".");
       return (this != "three"); // will stop running after "three"
   });

    jQuery.each(obj, function(i, val) {
      $("#" + i).append(document.createTextNode(" - " + val));
    });
</script>
</body>
```

Manejo de Objetos, Interacción de Array con JQuery

- .map() *Traduce todos los elementos de una matriz o un objeto a un nuevo array de items.*

Ejemplo

```
1  <body>
2    <div></div>
3    <p></p>
4    <span></span>
5  <script>
6      var arr = [ "a", "b", "c", "d", "e" ];
7      $("div").text(arr.join(", "));
8      arr = jQuery.map(arr, function(n, i){
9        return (n.toUpperCase() + i);
10     });
11     $("p").text(arr.join(", "));
12     arr = jQuery.map(arr, function (a) {
13       return a + a;
14     });
15     $("span").text(arr.join(", "));
16 </script>
17 </body>
```

Creación de Plugins JQuery

Para crear nuestro plugin, podemos extender de dos objetos:

- JQuery: que maneja la lógica interna de la librería.
- JQuery.fn: que maneja interacciones con los elementos visuales

En Función del tipo de plugin extenderemos de uno u otro objeto. Nuestro plugin debemos incluirlo en una librería JavaScript que deberia llamarse: jquery.myplugin.js, donde myplugin es el nombre de nuestro plugin.
Creamos el objeto principal, sus metodos y propiedades

Ejemplo

```
1    jQuery.comprobarAlert = function(message){
2         alert(message);
3    }
4
5    jQuery.fn.alerter = function(){
6    this.each( function(){
7        alert(this);
8    });
9    }
```

Creación de Plugins JQuery

Creamos un archivo HTML en el que vinculamos la librería JQuery y nuestro plugin y lo usamos.

```
1  $(document).ready( function(){
2      $("#victima").alerter();
3  });
```

Podemos extender la opciones del plugin. Como ejemplo añadimos al plugin de alerta la posibilidad de enviar una cadena adicional al comienzo y al final del mensaje, para esto hay dos opciones:

- La primera usar `JQuery.extend`: 'textit : una función de built-in de JQuery, pensada para proveer variables "default" modificadas por el usuario.

```
1  jQuery.comprobarAlert = function( mensaje, opciones_user ){
2
3      opciones_default = {
4          inicio_str : "Alerta: ",
5          final_str: " /fin "
6      }
7
8
9      opciones = jQuery.extend(opciones_default , opciones_user);
10     alert(opciones.inicio_str + mensaje + opciones.final_str);
11 }
```

Creación de Plugins JQuery

- la segunda es hacerlo sin ayuda de extend que puede ser mas entendible.

```
1  jQuery.comprobarAlert = function( mensaje, opciones_user ){
2      if( opciones_user == undefined ){
3          opciones = ({
4              inicio_str : "Alerta: ",
5              final_str: " /fin "
6          });
7      }else{
8          opciones = opciones_user;
9      }
10     alert(opciones.inicio_str + mensaje + opciones.final_str);
11 }
```

Serialización de Arreglos y Objetos

- .param() *: crea una representación en serie de un array o un objeto, adecuado para usar en una cadena de consulta URL o solicitud Ajax.*

```
1  var myObject = {
2    a: {
3      one: 1,
4      two: 2,
5      three: 3
6    },
7    b: [1,2,3]
8  };
9  var recursiveEncoded = $.param(myObject);
10 var recursiveDecoded = decodeURIComponent($.param(myObject));
11
12 alert(recursiveEncoded);
13 alert(recursiveDecoded);
```

Serialización de Arreglos y Objetos

- .serializeArray() *Codifica un conjunto de elementos de formulario como una matriz de nombres y valores.*

```
<form>
  <div><input type="text" name="a" value="1" id="a" /></div>
  <div><input type="text" name="b" value="2" id="b" /></div>
  <div><input type="hidden" name="c" value="3" id="c" /></div>
  <div>
    <textarea name="d" rows="8" cols="40">4</textarea>
  </div>
  <div><select name="e">
    <option value="5" selected="selected">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
  </select></div>
  <div>
    <input type="checkbox" name="f" value="8" id="f" />
  </div>
  <div>
    <input type="submit" name="g" value="Submit" id="g" />
  </div>
</form>
```

Serialización de Arreglos y Objetos

```
<script>
$('form').submit(function() {
  console.log($(this).serializeArray());
  return false;
});
</script>
```

Esto produce la siguiente estructura de datos (siempre que el navegador es compatible con console.log):

```
[
   {
     nombre: "a",
     valor "1"
   },
   {
     nombre: "b",
     valor: "2"
   }
]
```

Llamadas Asíncronas

- jQuery.ajax() *: Realizar una solicitud asíncrona HTTP (Ajax).*
- jQuery.ajax(url, [settings])
  - Url *: una cadena que contiene la URL a la que se envía la solicitud.*
  - Settings *: un conjunto de pares llave/valor que configuran la petición Ajax. Todas las configuraciones son opcionales. Un defecto se puede configurar para cualquier opción con .ajaxSetup().*
    Ver `http://api.jquery.com/jQuery.ajax/jQuery-ajax-settings` *para ver una lista completa de todos los ajustes.*

Por defecto el método `.ajax()` realiza todas las solicitudes de forma asincrónica (es decir, se establece en true por defecto). Si necesita solicitudes sincrónicas, establezca esta opción false .

Llamadas Asíncronas

Ejemplo Recuperar la versión más reciente de una página HTML.

```
1  $.ajax({
2    url: "test.html",
3    cache: false,
4    success: function(html){
5      $("#results").append(html);
6    }
7  });
```

Callbacks de .ajax()

Estos son los ganchos de devolución de llamada proporcionada por $.ajax() :

- beforeSend *: este recibe el objeto jqXHR y las opciones de mapeo con parámetros.*
- error *: recibe una cadena que indica el tipo de error y un objeto de excepción en su caso.*
- dataFiler *: se invoca inmediatamente después de recibir la respuesta correcta de los datos.*
- success *: se ejecuta si la solicitud se realiza correctamente.*
- complete *: se ejecuta cuando la solicitud termina, ya sea en el fracaso o éxito.*