

Diseño Web — Tecnologías Involucradas

JQuery

info@covetel.com.ve¹

¹Cooperativa Venezolana de Tecnologías Libres R.S.

Anatomía de un Script de JQuery

Tablas Dinámicas de Contenido

Disección del Script

Selector de Expresiones

Selectores CSS

JQuery

JQuery es una biblioteca que contiene una amplia variedad de métodos y funciones de los cuales sera muy útil saber las categorías básicas y como utilizarlas para nuestro beneficio.

Obtención de JQuery

El sitio web oficial de JQuery siempre es el recurso más al día para el código y noticias relacionadas con la biblioteca. Para empezar necesitamos una copia de JQuery, que se puede descargar desde la página principal del sitio. Varias versiones de JQuery pueden estar disponibles en cualquier momento dado, la última versión sin comprimir será la más apropiada para nosotros.

Para usar JQuery no es necesaria su instalación sólo tenemos que colocarlo en nuestro sitio en un lugar público. Cada vez que necesitamos un metodo de JQuery simplemente lo referenciamos desde el archivo del documento HTML.

Configuración del documento HTML

Existen tres componentes en el uso de JQuery:

Componentes

- ▶ El Documento HTML.
- ▶ El Archivo CSS con los estilos.
- ▶ Los Archivos Javascript.

En este ejemplo se muestra como referenciar los componentes de JQuery.

Configuración del documento HTML

```
1 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
4   <title>JQuery</title>
5   <link rel="stylesheet" href="jquery.css" type="text/css" />
6   <script src="jquery.js" type="text/javascript"></script>
7   <script src="appjquery.js" type="text/javascript"></script>
8 </head>
9 <body>
10 </body>
11 </html>
```

Escribiendo el Código JQuery

Nuestro código personalizado lo hemos incluido en el código HTML utilizando `<script src="appjquery.js" type="text/javascript"></script>`. A pesar de lo mucho que realiza tiene pocas líneas de código:

Escribiendo el Código JQuery

```
1 JQuery.fn.toggleNext = function() {  
2   this.toggleClass('arrow-down')  
3   .next().slideToggle('fast');  
4 }  
5  
6 $(document).ready(function () {  
7   $('div id="page-contents"></div>')  
8   .prepend('<h3>Page Contents</h3>')  
9   .append('<div></div>')  
10  .prependTo('body');
```


Escribiendo el Código JQuery

```
11 $('#content h2').each(function(index) {  
12     var $chapterTitle = $(this);  
13     var chapterId = 'chapter~' + (index + 1);  
14     chapterTitle.attr('id', chapterId);  
15     $('<a></a>').text($chapterTitle.text())  
16     .attr({  
17         'title': 'Jump to' + $chapterTitle.text(),  
18         'href': '#' + chapterId  
19     })  
20     .appendTo('#page-contents div');  
21 });
```

Escribiendo el Código JQuery

```
22 $( '#page-contents h3' ).click(function() {  
23     $(this).toggleNext();  
24 });  
25  
26 $( '#introduction > h2 a' ).click(function() {  
27     $( '#introduction' ).load(this.href);  
28     return false;  
29 });  
30 } );
```

Este ejemplo crea una tabla dinámica para los usuarios.

Disección del Script

Este script ha sido elegido específicamente ya que ilustra la capacidad general de la biblioteca JQuery e identifica las categorías de los métodos utilizados.

Selector de Expresiones

Antes de que podamos actuar en un Documento HTML se debe localizar las partes pertinentes, en el script algunas veces utilizamos un método simple para buscar un elemento:

```
1 $('#introduction')
```

Esta expresión se crea un nuevo objeto JQuery que hace referencia al elemento de ID introduction, algunas veces se requiere un selector más complejo:

```
1 $('#introduction > h2 a')
```

Aquí se obtiene un objeto JQuery que se refiere a los elementos descendientes de <h2> y que son hijos del elemento con ID introduction.

Métodos de recorrido DOM

A veces tenemos un objeto JQuery que ya hace referencia a un conjunto de elementos DOM, pero tenemos que realizar una acción en uno diferente, en estos casos los métodos de recorrido DOM son útiles, esto lo podemos ver en parte de nuestro script:

```
1 this.toggleClass('arrow-down')  
2   .next()  
3   .slideToggle('fast');
```

Debido al contexto de esta pieza de código, la palabra reservada `this` se refiere al objeto JQuery en este caso el objeto apunta la etiqueta `<h3>`. La llamada al método `.toggleClass` manipula este elemento de partida, el método `.next()` cambia el funcionamiento del elemento y `.slideToggle` llama las acciones del elemento `<div>`

Métodos de Eventos

Aun cuando podemos modificar la página a nuestra voluntad necesitamos métodos que reaccionen a eventos causados por los usuarios en el momento adecuado:

```
1 $( '#introduction > h2 a' ).click(function() {  
2   $( '#introduction' ).load(this.href);  
3   return false;  
4 });
```

En este fragmento se registra un controlador que se ejecutará cada vez que se hace click en la etiqueta de anclaje seleccionado.

Métodos de Eventos

```
1 $(document).ready(function() {  
2     //..  
3 });
```

Este método nos permite registrar comportamiento inmediatamente después de que la estructura del DOM este disponible en nuestro código

Métodos de Efectos

Los métodos de evento nos permiten reaccionar a las entradas del usuario. En los métodos de efecto se hace con efecto, en lugar de ocultar y mostrar inmediatamente elementos lo hace con animación.

```
1 this.toggleClass('arrow-down')  
2   .next()  
3   .slideToggle('fast');
```

Este método realiza una rápida transición de deslizamiento en el elemento, permitiendo ocultar y mostrar alternativamente con cada invocación.

Métodos AJAX

Muchos sitios web modernos emplean técnicas para carga el contenido cuando no se ha solicitado una actualización de la página. JQuery permite lograr esto con facilidad, los métodos de AJAX inician la solicitud de contenido y nos permite monitorear su progreso.

```
1 $('#introduction > h2 a').click(function() {  
2   $('#introduction').load(this.href);  
3   return false;
```

En la línea 2 el método `.load` nos permite obtener otro documento HTML desde el servidor e insertarlo en el documento actual, todo ello con una línea de código.

Elemento: T

Todos los elementos que tienen un nombre de etiqueta para T. **Ejemplo**

- ▶ `$('div')` : *selecciona todos los elementos con un nombre de etiqueta `div` en el documento.*
- ▶ `$('em')` : *selecciona todos los elementos con un nombre de etiqueta `em` en el documento.*

JQuery utiliza JavaScript `getTagName()` para los selectores de nombre de etiqueta.

ID: #myid

El único elemento con un ID igual a myid.

Ejemplo

- ▶ `$('#myid')` : *selecciona el unico elemento con id = 'myid', independientemente de su nombre de etiqueta.*
- ▶ `$(p#myid)` : *selecciona un solo párrafo con id de 'myid', el único elemento `<p id = 'myid'>`.*

Cada nombre asignado a un id debe ser utilizado sólo una vez en un documento.

Para un selector id JQuery utiliza el `getElementById()`.

Clase: .myclass

Todos los elementos que tienen una class myclass. **Ejemplo**

- ▶ `$('.myclass')` : *selecciona todos los elementos con una clase myclass.*
- ▶ `$(p.myclass)` : *selecciona todos los párrafos con una clase myclass.*
- ▶ `$('.myclass.otherclass')` : *selecciona todos los elementos con una clase myclass y otherclass*

En términos de velocidad, el ejemplo 2 es preferible al 1, ya que el primero utiliza el nativo `getElementsByTagName()` función de JavaScript para filtrar la búsqueda, y las miradas de la clase dentro del subconjunto emparejado de elementos DOM.

Descendiente: E F

Todos los elementos encontrados por F que son descendientes de uno de los elementos coincidentes por E.

Ejemplo

- ▶ `$('#container p')` : *selecciona todos los elementos encontrados por `<p>` que son descendientes de un elemento que tiene como `id` `container`.*
- ▶ `$(a img')` : *selecciona todos los elementos encontrados por `` que son descendientes de un elemento coincidente con `<a>`.*

Un descendiente de un elemento puede ser un hijo, nieto, bisnieto y así sucesivamente. Por ejemplo en el siguiente código HTML, el elemento `` es descendiente de los elementos ``, `<p>`, `<div id="inner">` y `<div id="container">`:

```
1 <div id="container">
2   <div id="inner">
3     <p>
4       <span></span>
5     </p>
6   </div>
7 </div>
```

Hijo: E_iF

Todos los elementos encontrados por F que son hijos de un elemento acompañado por E. **Ejemplo**

- ▶ `$('li ul')` : selecciona todos los elementos que comienzan por `` que son hijo de un elemento acompañado por ``
- ▶ `$('p code')` : selecciona todos los elementos encontrados por `<code>` que son hijos de un elemento acompañado por `<p>`

El combinador de hijo puede ser pensado de una forma mas especifica para obtener los descendientes seleccionando sólo el descendiente de primer nivel, en el siguiente código HTML el elemento `` es un hijo único del elemento ``

```
1 <div id="container">
2   <div id="inner">
3     <p>
4       <span></span>
5     </p>
6   </div>
7 </div>
```

Hermanos adyacentes: E + F

Todos los elementos encontrados por F seguidos inmediatamente por un Elemento E y que tengan un mismo padre. **Ejemplo**

- ▶ `$('ul + p')` : selecciona todos los elementos por `<p>` que están inmediatamente después de un elemento ``.
- ▶ `$('strong + em')` : selecciona todos los elementos encontrados por `` que están después de un elemento ``.

Un punto importante a considerar tanto con el combinar + como con el `es` es que solo seleccionan los hermanos.

```
1 <div id="container">
2   <ul>
3     <li></li>
4     <li></li>
5   </ul>
6   <p>
7     <img />
8   </p>
9 </div>
```

Hermanos adyacentes: E + F

Del ejemplo anterior:

- ▶ `$('ul + p')` : selecciona `<p>` ya que esta inmediatamente después de `` y los dos elementos comparten el mismo padre `<div id="container">`.
- ▶ `$('ul + img')` : no selecciona nada porque `` es un nivel mas alto que `` en el árbol DOM.
- ▶ `$('li + img')` : no selecciona nada porque `` y `` estan en el mismo nivel en el árbol DOM.

Hermanos en General: E F

Todos los elementos encontrados por F que están después de un elemento E y tienen el mismo padre. **Ejemplo**

- ▶ `$('p ul')` : selecciona todos los elementos encontrados por `` que están después de un elemento `<p>`.
- ▶ `$('code code')` : selecciona todos los elementos encontrados por `<code>` después de `<code>`.

La diferencia entre el combinador + y el es el alcance. mientras que el combinador + alcanza los elementos inmediatamente siguientes el combinador se extiende a todos los elementos de un mismo nivel.

```
1  <ul>
2    <li class="first"></li>
3    <li class="second"></li>
4    <li class="third"></li>
5  </ul>
6  <ul>
7    <li class="fourth"></li>
8    <li class="fifth"></li>
9    <li class="sixth"></li>
10 </ul>
```

Hermanos adyacentes: E + F

Del ejemplo anterior:

- ▶ `$('li.first li')` : selecciona `<li class="second">` and `<li class="third">`.
- ▶ `$('li.first + li')` : selecciona `<li class="second">`.

Multiple Elementos: E, F, G

Selecciona todos los elementos encontrados por el selector de expresiones E, F o G.

- ▶ `$('code, em, strong')` : *selecciona todos los elementos encontrados por `<code>` or `` or ``.*
- ▶ `$('p strong, .myclass')` : *selecciona todos los elementos por `` que son descendientes de un elemento encontrado `<p>` y tienen una clase `.myclass`.*

El combinador coma (,) es eficiente para seleccionar elementos dispares.

Enésimo Hijo (:nth-child(n))

Todos los elementos que son enésimos hijos de sus padres. **Ejemplo**

- ▶ `$('li:nth-child(2)')` : *selecciona todos los elementos encontrados por `` que son segundo hijo del padre.*
- ▶ `$('p:nth-child(5)')` : *selecciona todos los elementos encontrados por `<p>` que son quinto hijo de su padre.*

```
1 <div>
2   <h2></h2>
3   <p></p>
4   <h2></h2>
5   <p></p>
6   <p></p>
7 </div>
```

- ▶ `$('p:nth(1)')` : *selecciona el segundo `<p>` porque la numeración para `:nth(n)` comienza en 0.*
- ▶ `$('p:nth-child(1)')` : *no selecciona nada porque el primer elemento hijo del padre no es `<p>`.*
- ▶ `$('p:nth(2)')` : *selecciona el primer `<p>` porque es el segundo hijo del padre.*

Hijos

- ▶ Primer Hijo (:first-child) : *Todos los elementos que son primer hijo de su padres.*

Ejemplo

- ▶ `$('li:first-child')` : *selecciona todos los elementos encontrados por `` que son el primer hijo del padre.*
- ▶ Ultimo Hijo (:last-child) : *Todos los elementos que son el último hijo del padre.*

Ejemplo

- ▶ `$('li:first-last')` : *selecciona todos los elementos encontrados por `` que son el último hijo del padre.*
- ▶ Único Hijo (:only-child) : *Todos los elementos que son único hijo del padre.*

Ejemplo

- ▶ `$('li:first-last')` : *selecciona todos los elementos encontrados por `` que son único hijo del padre.*

Diversos Selectores

- ▶ No (:not(s)) : Todos los elementos que no coinciden con el selector s.

Ejemplo

- ▶ \$('li:not(.myclass)') : Selecciona todos los elemntos encontrados por que no tienen class="myclass".
- ▶ Vacio (:empty) : Todos los elementos que no tienen hijos (incluyendo los nodos de texto).

Ejemplo

- ▶ \$('p:empty') : Selecciona todos los elementos encontrados por <p> que no tienen hijos.
- ▶ Universal (:*): Todos los elementos.

Ejemplo

- ▶ \$('*') : Selecciona todos los elementos en el documento.