Diseño Web — Tecnologías Involucradas

JQuery

info@covetel.com.ve [1]

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

Efectos Pre-empaquetados

- .show([speed][, callback]) *speed= una cadena o un número para determinar cuánto tiempo se ejecutara la animación.*
  callback= una función para saber cuando ha finalizado la animación.

```
<div class="content">
  <div class="trigger button">Trigger</div>
  <div class="target"><img src="hat.gif" width="80" height="54"></div>
  <div class="long"></div>
</div>
```

```
$('.trigger').click(function() {
  $('.target').show('slow',  function() {
    $(this).log('Effect Complete.');
  });
});
```

- .hide().
- .toggle().
- .slideDown() .slideUp() .slideToggle()
- .fadeIn() .fadeOut() .fadeTo()

Efectos Personalizados

- .animate(properties [, speed][, easing][, callback]) *properties= un mapa de propiedades CSS por los que se moverá la animación.*
  speed= una cadena o un número para determinar el tiempo que se ejecutara la animación.
  easing= una cadena facilita el uso de la función en la transición.
  callback= una función para saber cuando ha finalizado la animación.

```
1    <div class="content">
2      <div class="trigger button">Trigger</div>
3      <div class="target"><img src="hat.gif" width="80" height="54"></div>
4      <div class="long"></div>
5    </div>
```

```
1    $('.trigger').click(function () {
2      $('.target').animate({
3      'width': 300,
4      'left': 100,
5      'opacity': 0.25
6      }, 'slow', function() {
7        $(this).log('Effect Complete');
8      });
9    });
```

Interfaz de Bajo Nivel

- .ajax(settings) *settings= un mapa que contiene las siguientes opciones para loas solicitudes:*
  - url.
  - dataType.
  - timeout.
  - error.
  - success.

```
1    $.ajax({
2      url: 'ajax/test.html',
3      success: function(data) {
4        $('.result').html(data);
5        $().log('Load was performed');
6      },
7    });
```

- .ajaxSetup(settings) *settings= un mapa de opciones para solicitudes a futuro.*

```
1    $.ajaxSetup({
2      url: 'ping.php',
3    });
```

Métodos de Taquigrafía

- .get(url[, data][, success]) *url= una cadena que contiene la URL donde la solicitud debe ser enviada.*
  data= un mapa de datos enviados con la solicitud.
  success= una función que se ejecuta cuando la solicitud a sido procesada.

```
1     $.get('ajax/test.html', function (data) {
2       $('.result').html(data);
3       $().log('Load was performed');
4     })
```

- .getModified().
- .loadModified().
- .post().

Métodos de Taquigrafía

- .load(url[, data][, success]) *url= una cadena que contiene la URL donde la solicitud debe ser enviada.*
  data= un mapa de datos enviados con la solicitud.
  success= una función que se ejecuta cuando la solicitud a sido procesada.

```
1    $('.result').load('ajax/test.html');
```

- .getJSON(url[, data][, success]) *url= una cadena que contiene la URL donde la solicitud debe ser enviada.*
  data= un mapa de datos enviados con la solicitud.
  success= una función que se ejecuta cuando la solicitud a sido procesada.

```
1    $.getJSON('ajax/test.json', function(data) {
2      $('.result').html('<p> + data.foo + '</p><p>' + data.baz[1] + '</p>')
           ;
3      $().log('Load was performed');
4    });
```

- .getScript().

Controladores Globales de Eventos

- .ajaxComplete(handler) *handler= la función que se invoca.*

```
1    $('.log').ajaxComplete(function() {
2      $(this).log('Triggered ajaxComplete handler');
3    });
```

- .ajaxError().
- .ajaxStart().
- .ajaxStop().
- .ajaxSuccess().
- .ajaxSend(handler) *handler= la función que se invoca.*

```
1    $('.log').ajaxSend(function() {
2      $(this).log('Triggered ajaxSend handler');
3    });
```

Funciones de Ayuda

- .serialize(param)

```
<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="text" 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>
```

```
$('form').submit(function() {
  $(this).log($('input, textarea, select').serialize());
  return false;
});
```

Métodos de Codificación y decodificación base-64

- .atob() *Método que decodifica una cadena codificada en base-64.*
- .btoa() *Método para codificar una cadena en base-64. Este método utiliza los caracteres A-Z, a-z, 0-9, +, / y = para la codificación de una cadena.*

Ejemplo

```
1  <html>
2  <head>
3      <script type="text/javascript">
4          function EncodeDecode () {
5              input = document.getElementById ("myInput");
6
7              encodedData = window.btoa (input.value);
8              alert ("encoded data: " + encodedData);
9
10             decodedData = window.atob (encodedData);
11             alert ("decoded data: " + decodedData);
12         }
13     </script>
14 </head>
15 <body>
16     <input type="text" id="myInput" value="The text to encode"/>
17     <button onclick="EncodeDecode ();">Encode in base-64!</button>
18 </body>
19 </html>
```

JSON.parse

Método para convertir una cadena en Objeto.

- Usando JavaScript **Ejemplo**

```
1  var JsonString = '[{"name": "Jhon", "apellido":"Smith"}]';
2  var JsonObjects = JSON.parse(JsonString);
3  alert(JsonObjects);
```

- Usando JQuery **Ejemplo**

```
1  var obj = jQuery.parseJSON('{"name": "Jhon", "apellido":"Smith"}');
2  alert(obj.name === "Jhon");
```