1 O plugin Form en jQuery

Ata este momento no tutorial centrámosnos nas capacidades do núcleo jQuery e da biblioteca; pero todo isto é soamente a punto do iceberg!

A colección de plugins para jQuery dispoñible ata estes momentos é inmensa e impresionante. Os creadores do núcleo jQuery coidadosamente escolleron as funcións necesitadas pola maioría de desarrolladores de páxinas web e crearon un marco para que outros puideran construir plugins fácilmente.

Isto mantén o núcleo de jQuery o máis pequeno posible, de xeito que si queremos engadir máis funcionalidade teremos que empregar plugins adicionais.

Para consultar a lista de plugins dispoñibles teremos que ir a http://plugins.jquery.com/

A lista de plugins máis populares en http://plugins.jquery.com/most_popular

Aquí imos a ver algúns dos plugins máis útiles para a nosa programación web. Imos comezar co plugin Form.

1.1 Sumario

1.2 O plugin Form para jQuery

Traballar con formularios en moitos casos soe ser un traballo tedioso e moi repetitivo. Cada tipo de control ten as suas particularidades e o envío de formularios a miúdo pode incluir a validación de campos, etc..

O núcleo de jQuery ten algúns métodos que nos poden axudar, pero isto non é suficiente. O propósito do plugin Form é axudarnos a cubrir estas carencias e proporcionarnos máis poder nos controles dos formularios.

Este plugin pódese atopar en http://jquery.com/plugins/project/form e reside no ficheiro jquery.form.js.

Este plugin proporciona funcionalidades nas seguintes áreas:

? Obtención dos valores dos controles do formulario

? Limpando e reseteando controles do formulario

? Envío de formularios (incluíndo envío de arquivos) via Ajax

1.2.1 Obtendo valores dos controles do formulario

O plugin Form proporciona dous xeitos de obter os valores dos controles do forumario: como un array de valores ou como un string serializado. Dispomos de tres métodos para obter esos valores: **fieldValue(), formSerialize() e fieldSerialize()**.

Dos tres métodos anteriores o máis empregado é **formSerialize()**. Este método accede a tódolos valores do formulario e devolve unha cadea no formato empregado polo método GET para enviar os datos. Por exemplo si temos un formulario con id form1 con tres campos de texto: nome, apelidos e dni, se pomos o seguinte código:

```
$('#form1').formSerialize()

// Devolverá a seguinte cadea: nome=rafa&apelidos=veiga%20cid&dni=12345678T
```

1.2.2 Limpando e reseteando os controles

O plugin Form proporciona dous comandos que afectan ós valores dos controles. O comando **clearForm()** limpa tódolos campos, mentras que o

comando **resetForm()** resetea os campos.

¿Cal é a diferencia entón? O método **clearForm()** realiza o seguinte:

- Os campos de texto, password e textarea son axustados a valores en blanco.

- Os campos de tipo select son deseleccionados.

- Os campos check box e radio son desmarcados.

Cando chamamos o método **resetForm()**, o método reset() é invocado. Isto provocará que os campos tomen os valores que teñen asignados por defecto na definición do formulario (os value que teñamos no HTML)

1.2.3 Enviando formularios con Ajax

O plugin Form introduce dos novos comandos para enviar formularios mediante Ajax: un deles que inicia unha solicitude Ajax baixo o control dun script, enviando os datos ó destino no formato solicitado, e o outro comando que controla o formulario de xeito que calquera envío é reconvertido automáticamente a unha solicitude Ajax.

Para enviar un formulario mediante este plugin faise dun xeito moi sinxelo. Simplemente teremos que empregar o método **ajaxSubmit()**.

O método ajaxSubmit() ten varias opcións, aquí pomos un resumen das mesmas:

Name	Description
`url`	(String) The URL to which the Ajax request will be submitted. If omitted, the URL will be taken from the `action` attribute of the target form.
`type`	(String) The HTTP method to use when submitting the request, such as GET or POST. If omitted, the value specified by the target form's `method` attribute is used. If not specified and the form has no `method` attribute, GET is used.
`dataType`	(String) The expected data type of the response, which determines how the response body will be post-processed. If specified, it must be one of the following: ■ `xml`—Treated as XML data. Any success callback will be passed the `responseXML` document. ■ `json`—Treated as a JSON construct. The JSON is evaluated, and the result is passed to any success callback. ■ `script`—Treated as JavaScript. The script will be evaluated in the global context. If omitted, no post-processing of the data (except as specified by other options such as `target`) takes place.
`target`	(String|Object|Element) Specifies a DOM element or elements to receive the response body as content. This can be a string depicting a jQuery selector, a jQuery wrapper containing the target elements, or a direct element reference. If omitted, no element receives the response body.
`beforeSubmit`	(Function) Specifies a callback function invoked prior to initiating the Ajax request. This callback is useful for performing any pre-processing operations including the validation of form data. If this callback returns the value `false`, the form submission is cancelled. This callback is passed the following three parameters: ■ An array of the data values passed to the request as parameters. This is an array of objects; each contains two properties, `name` and `value`, containing the name and value of a request parameter. ■ The jQuery matched set that the command was applied to. ■ The `options` object that was passed to the command. If omitted, no pre-processing callback is invoked.
`success`	(Function) Specifies a callback invoked after the request has completed and returned as response with successful status. This callback is passed the following three parameters: ■ The response body as interpreted according to the `dataType` option. ■ A string containing *success*. ■ The jQuery matched set that the command was applied to. If omitted, no success callback is invoked. If this is the only option to be specified, this function can be passed directly to the command in place of the `options` hash. Note that no provisions have been made for a callback upon error conditions.
`clearForm`	(Boolean) If specified and `true`, the form is cleared after a successful submission. See `clearForm()` for semantics.
`resetForm`	(Boolean) If specified and `true`, the form is reset after a successful submission. See `resetForm()` for semantics.
`semantic`	(Boolean) If specified and `true`, the form parameters are arranged in semantic order. The only difference this makes is in the location of the parameters submitted for input element of type *image* when the form is submitted by clicking that element. Because there's overhead associated with this processing, this option should be enabled only if parameter order is important to the server-side processing and image input elements are used in the form.
other options	Any options that are available for the core jQuery `$.ajax()` function, as described in table 8.2, can be specified and will pass through to the lower-level call.

Exemplos de envío de datos mediante ajaxSubmit():

```
// Si queremos enviar o formulario sin xestionar ningunha resposta faremos:
   $('#idformulario').ajaxSubmit();

// Si queremos cargar a resposta nun elemento  ou elementos de destino faremos:
   $('#idformulario').ajaxSubmit( { target: 'elementosdestino' } );

// Si queremos xestionar a responsa cunha función
   $('#idformulario').ajaxSubmit(function(resposta){
      /* facemos algo coa resposta  */
   });
```

Sin embargo hai outro xeito de enviar automáticamente (sen usar un script) o formulario mediante Ajax. Para elo empregaremos a función ajaxForm(), que enlazaremos ó noso formulario.

Vexamos un exemplo:

```
<form id="Formulario" action="comentarios.php" method="post">
    Name: <input type="text" name="name" />
    Comentarios: <textarea name="comment"></textarea>
    <input type="submit" value="Enviar Comentario" />
</form>
<head>
    <script src="ruta/jquery.js"></script>
    <script src="ruta/form.js"></script>

    <script>
        // Agardamos a que o DOM sexa cargado
        $(document).ready(function() {
            // enlazamos o noso 'formulario' e proporcionamos unha función de retorno
            $('#Formulario').ajaxForm(function() {
                alert("Grazas polo comentario!");
            });
        });
    </script>
</head>
```

Cando o formulario é enviado o nome e comentarios serán enviados á páxina comentarios.php. Si o servidor devolve un estado de correcto, entón o usuario recibirá unha alerta de "Grazas polo comentario".

Ver máis exemplos de ajaxSubmit() e ajaxForm()

1.2.4 Enviando arquivos

Unha das características do plugin Form é que automáticamente detecta aqueles formularios que necesitan enviar arquivos.

Debido a que non é posible enviar arquivos usando o obxecto XMLHttpRequest (XHR), o plugin Form emprega un iframe oculto creado dinámicamente para axudar nesa tarefa. Esta é unha técnica moi común, pero ten limitacións. O iframe é usado como destino do formulario, o que implica que a resposta do servidor será escrita no iframe. Isto funciona ben si o tipo de resposta é HTML ou XML, pero non funciona correctamente si a resposta é un script ou JSON, xa que ambos conteñen caracteres especiales que necesitan ser representados empregando o seu equivalente HTML.

Para solucionar o caso anterior dos script e JSON, o plugin Form permite que estas respostas sexan representadas nun textarea, e é recomendable que o fagamos así cando fagamos envío de arquivos nos formularios. Algo a ter en conta é que si non enviamos arquivos ó servidor a solicitude empregará unha petición XHR para enviar o formulario (non un iframe). Isto fai que dependendo do envío o código do servidor ten que saber cando usar un textarea ou non. Para evitar isto podemos usar a opcion de iframe no plugin para forzar que sempre empregue un iframe co cal teremos sempre a resposta nun textarea.

Exemplo de file upload.

--Veiga (discusión) 23:10 26 ene 2015 (CET)