EBook Gratis

APRENDIZAJE jquery-plugins

Free unaffiliated eBook created from
Stack Overflow contributors.

#jquery-plugins

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jquery-plugins

It is an unofficial and free jquery-plugins ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jquery-plugins.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con jquery-plugins

Observaciones

Un complemento de jQuery es simplemente un nuevo método que usamos para extender el objeto prototipo de jQuery. Al extender el objeto prototipo, habilita a todos los objetos jQuery para que hereden los métodos que agregue. Según lo establecido, siempre que llame a `jQuery()` está creando un nuevo objeto jQuery, con todos los métodos de jQuery heredados.

La idea de un complemento es hacer algo con una colección de elementos. Podría considerar que cada método que viene con el núcleo de jQuery es un complemento, como `.fadeOut()` o `.addClass()` .

Puede crear sus propios complementos y usarlos de forma privada en su código o puede liberarlos al público. Hay miles de complementos de jQuery disponibles en línea. La barrera para crear un complemento propio es tan baja que puede crear uno fácilmente.

Examples

Instalación o configuración

Los complementos de jQuery generalmente se instalan a través de NPM o Yarn (si están alojados allí), o haciendo referencia a un archivo de script externo que contiene el complemento, ya sea de un directorio relativo o de un CDN.

```
<script type="text/javascript" src="/path/to/plugin.jquery.js"></script>
```

Complemento básico que cambia el color del texto a verde.

```
// plugin initialization
$.fn.greenify = function() {
    // within the function you can use any of the jQuery methods
    // and `this` context refers to jQuery object
    this.css( "color", "green" );
};

// apply plugin
$( "a" ).greenify();
```

Encadenamiento

Esto funciona, pero hay un par de cosas que debemos hacer para que nuestro complemento sobreviva en el mundo real. Una de las características de jQuery es el encadenamiento, cuando vincula cinco o seis acciones en un selector. Esto se logra haciendo que todos los métodos de objeto jQuery devuelvan el objeto jQuery original nuevamente (hay algunas excepciones: `.width()` llamado sin parámetros devuelve el ancho del elemento seleccionado, y no es posible). Hacer que

nuestro método de plugin chainable tome una línea de código:

```
$.fn.greenify = function() {
    this.css( "color", "green" );
    // return the reference for chaining
    return this;
}

$( "a" ).greenify().addClass( "greenified" );
```

Opciones de soporte con valores predeterminados

Puedes personalizar tu plugin aceptando opciones.

```
$.fn.colourize = function(options) {

    // This is one method to support default options
    var style = $.extend({
        color: "green",
        backgroundColor: "white"
    }, options);

    // Set the colours on the current selection based on the option parameters
    return this.css({
        color: style.color,
        backgroundColor: style.backgroundColor
    });
};
```

Ejemplo de uso:

```
$("button").colourize({
    color: "orange"
});
```

El valor predeterminado para la opción de color "verde" se reemplaza con `$.extend()` para que sea "naranja".

Estructura de plugin jQuery típica

Mientras que escribir complementos de jQuery es simple, queremos incluir nuestros complementos en un ámbito local. Esto evitará conflictos de espacio de nombres y contamina el espacio de nombres global, además de garantizar que jQuery se cargue antes de que nuestro complemento lo amplíe.

```
// Encapsulate our plugins in a local scope
(function($) {

    // Plugin definition
    $.fn.colourize = function() {

        // Plugin code
```

```
    };

// Pass the jQuery object into our local scope
}(jQuery));
```

El envoltorio de alcance local se puede omitir en otros ejemplos para mantenerlos simples y concisos.

Usando el método each ()

Su objeto jQuery típico contendrá referencias a cualquier número de elementos DOM, y es por eso que a los objetos jQuery se les suele llamar colecciones. Si desea manipular elementos específicos (por ejemplo, obtener un atributo de datos, calcular posiciones específicas), debe usar .each () para recorrer los elementos.

```
$.fn.myNewPlugin = function() {
    return this.each(function() {
        // Do something to each element here.
    });

    // return the reference for chaining
    return this;
};
```

Lea Empezando con jquery-plugins en línea: https://riptutorial.com/es/jquery-plugins/topic/8277/empezando-con-jquery-plugins

Creditos

S. No	Capítulos	Contributors
1	Empezando con jquery-plugins	Community, Pranav C Balan, Samuel Liew