JQUERY - PLUGINS

 Copyright © tutorialspoint.com

A plug-in is piece of code written in a standard JavaScript file. These files provide useful jQuery methods which can be used along with jQuery library methods.

There are plenty of jQuery plug-in available which you can download from repository link at http://jquery.com/plugins.

How to use Plugins

To make a plug-in's methods available to us, we include plug-in file very similar to jQuery library file in the <head> of the document.

We must ensure that it appears after the main jQuery source file, and before our custom JavaScript code.

Following example shows how to include **jquery.plug-in.js** plugin −

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script src="jquery.plug-in.js" type="text/javascript"></script>

      <script src="custom.js" type="text/javascript"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {
            .......your custom code.....
         });
      </script>

   </head>

   <body>
      ..........................
   </body>

</html>
```

How to develop a Plug-in

This is very simple to write your own plug-in. Following is the syntax to create a a method −

```
jQuery.fn.methodName = methodDefinition;
```

Here *methodNameM* is the name of new method and *methodDefinition* is actual method definition.

The guideline recommended by the jQuery team is as follows −

- Any methods or functions you attach must have a semicolon ; at the end.
- Your method must return the jQuery object, unless explicity noted otherwise.
- You should use this.each to iterate over the current set of matched elements - it produces clean and compatible code that way.
- Prefix the filename with jquery, follow that with the name of the plugin and conclude with .js.
- Always attach the plugin to jQuery directly instead of $, so users can use a custom alias via

noConflict() method.

For example, if we write a plugin that we want to name *debug*, our JavaScript filename for this plugin is −

```
jquery.debug.js
```

The use of the **jquery.** prefix eliminates any possible name collisions with files intended for use with other libraries.

Example

Following is a small plug-in to have warning method for debugging purpose. Keep this code in *jquery.debug.js* file −

```
jQuery.fn.warning = function() {
   return this.each(function() {
      alert('Tag Name:"' + $(this).prop("tagName") + '".');
   });
};
```

Here is the example showing usage of warning method. Assuming we put *jquery.debug.js* file in same directory of html page.

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script src="jquery.debug.js" type="text/javascript"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("div").warning();
            $("p").warning();
         });
      </script>

   </head>

   <body>
      <p>This is paragraph</p>
      <div>This is division</div>
   </body>

</html>
```

This would alert you with following result −

```
Tag Name:"DIV"
Tag Name:"P"
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js