jQuery Plugins

Recap: JavaScript

We started with plain JavaScript and learned the basics

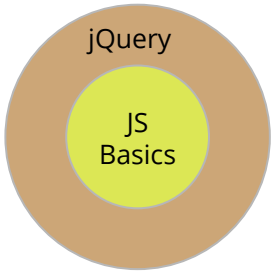
1. Working with variables
2. Flow control structures
3. Creating reusable functions
4. Manipulating the DOM
5. Handling events

With these basics, you can actually do quite a lot, but it might take a long time to write the kinds of scripts you want to write.

Recap: jQuery

With jQuery, you have learned that a lot of helper functions can help you write scripts that are a bit shorter.

Your world has become a little larger.

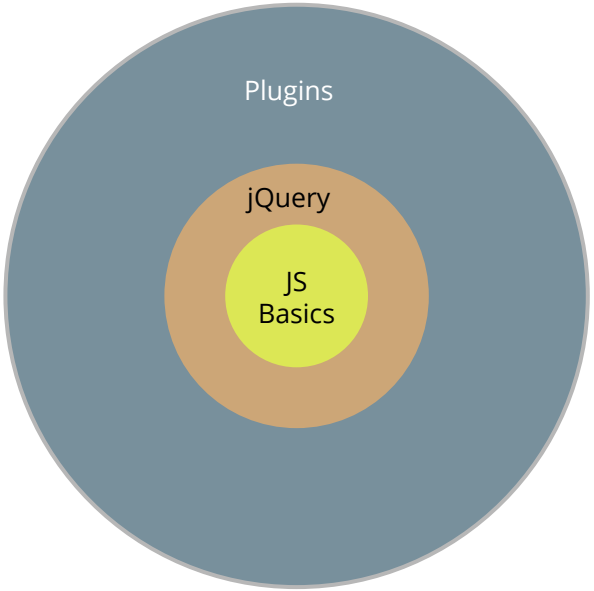


Introducing: Plugins

You have already used the easing plugin, but now it is time to formally introduce the idea of plugins.

The availability of thousands of plugins to jQuery is one of the other reasons learning jQuery is still worth while.

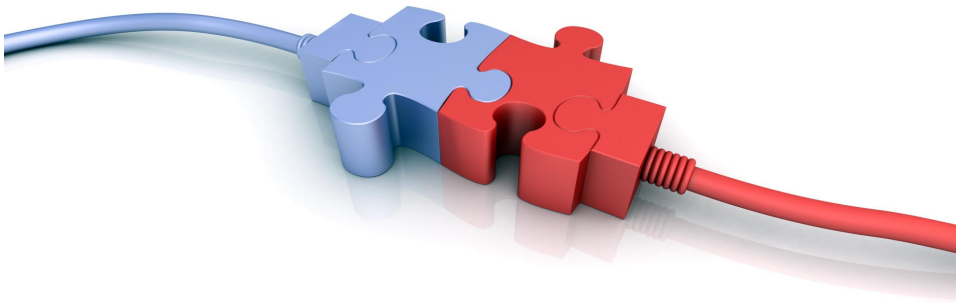
With access to all this code, written by other people, your world is about to become A LOT bigger.



Adding Functionality

Plugins add additional functionality to jQuery. You have already seen this with the easing plugin.

Using plugins can give you some complex functionality without having to write a lot of your own code.



Using Plugins And Essential Steps

Adding plugins to your project, configuring them, and learning how to use them still takes some work and effort.

But not nearly as much as writing all the code yourself!

Many of the larger, more sophisticated plugins require five steps to get them to work.

Step 1: Use markup indicated by the plugin documentation.

Step 2: Make sure jQuery is linked. Decide if you are going to put your scripts at the top or bottom of the page.

Step 3: Link the plugin file. There may be a compressed and uncompressed version. Use the compressed one.

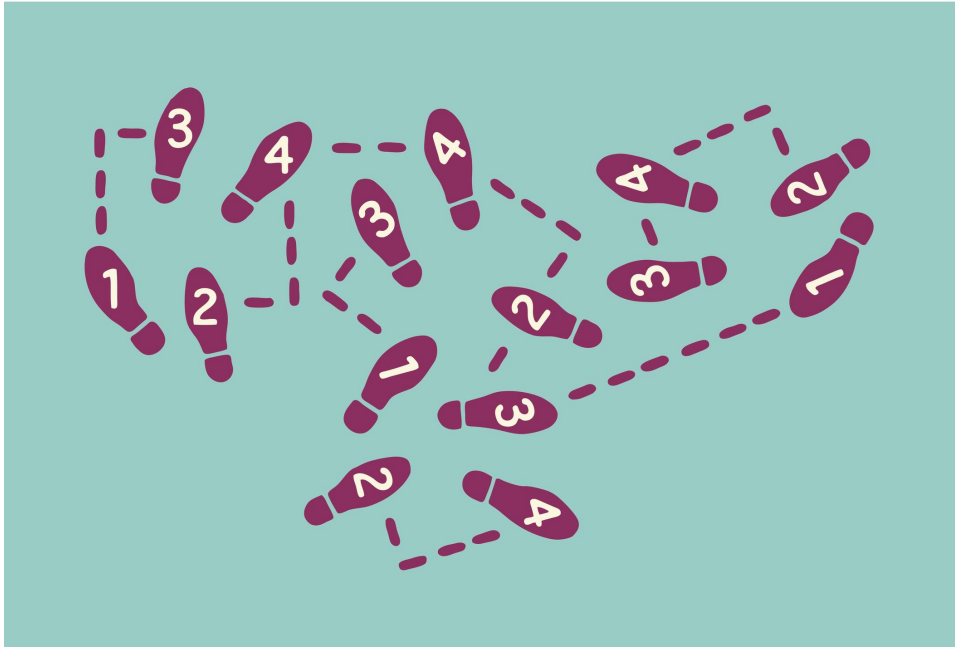
Step 4: Link the plugin's CSS file. This is usually included to provide styles for the elements the plugin uses to make the interface look right and work properly.

Step 5: Initialize the setup of the plugin in your own script tag, or on your own linked JS file. Often you can set additional options when the function is initialized. *Make sure this script loads last*.

Documentation Mish-Mash

Suppose you find a jQuery plugin you want to try on a website. How do you go about adding it to your page? How do you know what to do?

Because these plugins are all made by different developers, or groups of developers, there is no real standard in terms of documentation.



Gotchas and Rookie Mistakes

1. **Multiple copies of jQuery!** Only load the jQuery library once. You may have multiple plugins, but you should only load one copy of the jQuery library, which is used by all plugins.
2. **File Management:** Be sure all files are saved in the project in a place that make sense and that they are linked properly. This gotcha causes lots of frustration.
3. **Top of the page v.s. bottom of the page**: Make sure you are placing the plugin scripts in the correct place. Some may require you to load them at the top of the page.
4. **Document.ready(), Window.load() or window.on('load', function(){})**: Be sure to use these functions properly.
5. **Override CSS on YOUR stylesheet:** If you want to customize the look of the plugin (and you should), put rules that override the styles they have set on their CSS file on your CSS file and make sure your CSS file loads last.

Beware the Plugin

1. Plugins can bloat your project with lots of additional code that you may not need. (All the features are programmed in, but you may only be using one or two.)
2. Without modification, plugins can make your site look generic.
3. Plugins can turn us into lazy designers, looking for a quick fix, rather than the right one.
4. Plugins can be poorly written and have all sorts of bugs built into them.
5. Plugins take time to research and test.

Looking to the Future

Getting used to using other people's code now via jQuery plugins is a good preparation for the future, if you get deeper into JavaScript coding where you might use other repositories of shared code, such as NPM, which is the world's largest software registry.

It is very common in the software development world to be working on modularized blocks of code, and working with plugins can be seen as an introduction to that way of thinking and working.