AQUENT

GYMNASIUM

# JQUERY: BUILDING BLOCKS

Lesson 4

# MATERIALS & ASSIGNMENT

## THIS LESSON'S CORE CONCEPTS

**Plugins**

1. Tabs are in the same family of space-saving widgets as last lesson's carousels – but with clearer user communication and less room for designer error. When your content fits naturally into categories, tabs provide a natural way for the user to toggle between them. Tabs can also replace navigation bars with in-page loading/linking. (The loading and linking pieces are out of scope for this lesson.)
2. You can create tabs in jQuery in two fundamental ways:
    - 1.) Using a plugin or
    - 2.) Creating the tabs form scratch. When deciding whether to use a plugin or to build it yourself, keep in mind that with plugins there's always a cost associated with finding and learning the plugin. With any plugin (not just tabs,) we offer the following rule of thumb: If you can implement it yourself within 30-45 minutes, don't bother learning a new plugin. (Unless you think it will become an integral part of your future workflow.) Additionally, plugins typically come with their own assumptions on how best to implement them, so it's important to make sure that they match with your needs. For example, jQuery UI requires the tabs to be in a separate element from the content elements; the scenario in this lesson does not match that requirement, so we need to be comfortable building our tabs from scratch.
3. Tab sets often have different styling for their first and last tabs. You can conveniently select the first and last elements from a jQuery list using `$().first() and $().last()`.
4. The differences between active and inactive tabs can – and should – be expressed stylistically in CSS; in order to switch a tab's state you simply add and remove classes in your jQuery code. You can accomplish this using `$().addClass() and $().removeClass()`.
5. To retrieve an arbitrary element from a jQuery list, use square brackets like so: `$(selector)[index]`. Note that this returns the raw HTML element, not one wrapped in a jQuery list.
6. To wrap a raw element in jQuery, pass it into the `$` method like so: `$(rawElement)`.Building upon the previous idea, if you want to extract a single, jQuery-wrapped element from a jQuery list, combine them like so: `$( $(selector)[index] )`.
7. Copying large blocks of code, pasting it and then editing it is a BAD IDEA and creates unmaintainable code. Much of the art of writing good code is finding other ways to run similar code in different places. One such: To loop through an entire jQuery set, use `$().each(function(index, element){...})`;

**Creating a plugin**

8. Plugins add methods to the jQuery object. Plugins are fundamentally about abstraction: hiding complex behavior behind simple commands. To create your own plugin, move your code to a new file and add it as a method on the "jQuery.fn" object: `jQuery.fn.myPlugin = function() {...};`
9. As a convenience to your plugin's users, you can add your own stylesheet with: `$(document.head).append('<link rel="stylesheet" href="path/to/stylesheet.css"/>');`

When your plugin method runs, “this” is the jQuery object that called it. For example, if I call `$('#element').myPlugin()`, “this” will refer to `$('#element')`.
Don’t forget that your plugin can accept arguments in the same way a function can: `$('#element').myPlugin('aString', 4);`

10. When writing plugins, there’s a natural urge to pack in as many features as you can. Avoid this. You’ll end up over-designing for features that never get used, and unable to implement features that do. Remember, your tabs plugin isn’t aiming to take over the world – there are plenty of tab plugins out there. Its purpose is to handle your company’s specific needs, and save the people around you some time.
11. Error handling and generation is a crucial part of good plugin development. If someone calls your plugin in a way that doesn’t make sense or will cause problems, it’s appropriate to throw a descriptive error, explaining what has gone wrong and how to fix it. You can generate and throw errors with: `throw new Error(descriptive error message);`

## FURTHER READING:

- Tabs and Accessibility: http://www.nomensa.com/blog/2011/accessible-tabs-part-2-the-solution/
- More on plugins: http://learn.jquery.com/plugins/basic-plugin-creation/
- Even more on plugins: http://learn.jquery.com/plugins/advanced-plugin-concepts/
- Why predicting every one of your plugin’s possible feature needs is bad: http://sebastiansylvan.com/2013/08/16/the-perils-of-future-coding/

## ASSIGNMENTS

1. Quiz
2. Fix the broken tabs. (They’re in the “broken_tabs” folder.)

## EXTRA CREDIT:

Turn the Lesson 3 carousel into a plugin. (It’s in the “shelf” folder.) I’ve moved the code into separate files for you already.