CIS 3344 Tutorial Requirements

For their tutorial, most students will implement a JavaScript framework (in an external JavaScript file) that provides a "web widget" (also called JavaScript plugin or client side API). Your "web widget" should be an HTML element with several children elements inside and provide several methods (things that your widget can do). Creating a widget is not the only option for tutorial, but if you have other ideas, please talk to the instructor ASAP. Everyone's tutorial must include substantial and sophisticated JavaScript coding and meet all the requirements listed on this page.

Your tutorial shall:

1. be contained in a page named **tutorial/index.html**.
2. be referenced by a **blog** in your labs page that describes what your tutorial covers.
3. contain an **introduction** that explains and demonstrates the code you are about to build within your tutorial. Provide a link to published / working code and a downloadable zip file, if necessary.
4. Your final code (that you are writing the tutorial about) shall utilize the "framework style" javaScript design and it shall demonstrate 2 javaScript objects (A.K.A. widgets or plugins or components) on one HTML page. If your widget is very big (like full screen), then create two virtual pages within one HTML page like we did for lab 2 responsive design.
5. After instantiating the (two or more) components on the page, your HTML page shall invoke (responding to a user initiated event) various methods that modify the look or behavior of the components (similar to the lab 3 requirement).
6. After showing an example of your implemented code (and how an HTML page can use it), provide "API specifications" which show the parameters one can use to instantiate your widget and the public methods that are provided by your widget.
7. contain about **3-7 additional sections** of content, where each section explains a concept that was important and necessary to build your solution.
    - Each section shall demonstrate the concept by including "code snippets" and working simple examples of the snippet code (like W3schools "Try It Yourself" pages). You can do this by using a helpful site, like jsfiddle or github, or you can "roll your own" (see next item).
    - To "roll your own" code snippet demos, copy the code into a <pre> tag which you have appropriately styled (do NOT include an image of your code). If the code you want to show within the <pre> tag has a < in it, change that to `&lt;` so that your code is readable and not interpreted as HTML code on your page. You also need to link to a working page that surrounds the snippet code with the minimal code that's needed to make it work.

   At the end of each section, link to a working intermediate stage of your solution (that includes just the concepts that you have you presented so far).
8. contain a **final section** that summarizes the tutorial and gives instructions that an HTML developer would need to be able to use your widget framework. You might remind the reader that the introduction provided a link to the final working code (and downloadable zip file, if necessary).
9. include **references**, the links where you obtained your information and sample code. For each link, explain the information that the link provided to you.

Some tutorials will be selected for **class presentation**. Not all tutorials that are presented will get an A, but you do need to present in order to get an A. Students who present must be prepared, demonstrate that they understand the concepts from their tutorial, present concepts clearly, and be able to answer questions.

If you have already written or found **some complicated JavaScript code** and you want to use that as a basis for your tutorial, follow these quidelines (after getting your topic approved).

1. Consider all the separate elements (or concepts) that you needed to write your demo code. Provide a description of each concept and provide a simple code example that demonstrates just that concept, in isolation - "w3schools style" .
2. If there are too many of these concepts for the scope of the tutorial assignment, select a subset of concepts and then make a simplified version of your demo code so that it just uses the concepts that you presented.
3. Discuss how you debugged your code, e.g., you may point out some commented out console.log code that you used.
4. Make your (possibly simplified) demo code be a show case of best "oo" javascript coding practices (see below). Adding all of these would not be hard - almost a find/replace exercise, but make sure not to break the code as you do so.
   - Really good naming (self-documenting)
   - Good object design using Single Responsibility Principal. Add properties to objects instead of having global "variables". Add methods to objects instead of having global functions.
5. If you simplified your code (to limit it to the concepts you presented), you can also show your original (more complex) code that might not have the "js oo programming style" enhancements.
6. If your code is a modification of code you found on the internet, then (at the end of your tutorial) provide a link to the code you originally found and list the changes you made. Otherwise (you created the program yourself), as you present each concept/section, provide the links that helped you grasp that concept (at the end of the section).

How To Find Ideas For Tutorial Topic

Simpler approach. You can get ideas from W3Schools How To, where they show the simple code that an HTML developer would write to implement a "widget" from scratch. Your job would be to move most of the code into your JavaScript framework so that the HTML developer only has to create an HTML container for the widget, invoke a "make widget" method from your framework, then call any/all the methods that your framework provides for the widget.

Let's say you found this Contact Chip example from "W3Schools How To" and you wanted to create a Contact Chip framework. Click here to see some very initial work that we did on this (you can View Source to see code, then click on the JavaScript referenced by the <script> tag within the View Source). The framework would need to provide a "MakeContactChip" method that

- accepts a parameter object that has the id of the HTML element that is supposed to be turned into the Contact Chip, as well as the desired default values, such as background color behind the contact's name, font-size, image of the person, name of callback function for when the user clicks the "x"
- enhances the contact chip with various public methods (like move or show or hide or grow or shrink, ...)

Alternative approach. Another source for ideas would be for you to google "JavaScript plugins" or "JQuery plugins" so that you can find some interesting web effects. If you go this route, I suggest that you first try to use their plugin (they should provide simple instructions). Once you have used their plugin, try to write your own simplifed version of their plugin. Keep in mind that you have to isolate the concepts you use, provide working sample code, and explain these concepts. So, you cannot just do a "File - Save As" from their framework and call it your own code. Here are some links that I found when I googled for JavaScript/JQuery Plugins:

- jQuery Plugins
- Plain JS Plugins
- JS Effects Plugins

OR you could even look for a tutorial that tells you how to write a plugin, like this one at tells you how to write your own modal window

As you work though all of this, keep notes or a log about you whole experience: finding what you want to implement, documenting what you had to learn in order to accomplish the task. Once you complete your framework, you'll have to backtrack and remember all the things you needed to put it together. Your goal is to write a tutorial that can help others write a web widget framework, but lack the experience to do it without guidance.