CSCE 120: Learning To Code

Manipulating Data I
Hacktivity 7.2

Introduction

Prior to engaging in this hacktivity, you should have completed all of the pre-class activities as outlined in this module. At the start of class, you will be randomly assigned a partner to work with on the entirety of this hacktivity as a *peer programming* activity. Your instructor will inform you of your partner for this class.

One of you will be the driver and the other will take on the navigator role. Recall that a driver is in charge of the keyboard and computer while the navigator is in charge of the handout and directing the activity. However, you are *both* responsible for contributing and discussing solutions. If you were a driver/navigator in the prior activity, switch roles for this activity.

You will use the same project from the previous Hacktivity. If you need to, you can re-download it from GitHub using the URL, https://github.com/cbourke/jQueryProject.

1 Making Dynamic Elements

You will familiarize yourself further with jQuery by adding dynamic elements to a web page. We have provided a starter webpage and some code in the `DynamicElements` folder. For the first few exercises you should place your code inside a `ready()` function in the `main.js` source file:

```
$(document).ready(function() {
  //place your code here
});
```

Recall that the code in this function is executed only after the page has fully loaded.

1.1 Page Load Animations

1. Make the page load more dynamic by sliding in the top two elements. In particular, make the title "jumbotron" element slide down and make the "jQuery Conference" information box slide in from the left (see http://api.jqueryui.com/slide-effect/ for documentation on the slide effect).
2. Make the images more visually dynamic by fading them in *slowly* in when the page is loaded.
3. Click any of the external links. You'll notice that clicking them will open the page in the same browser. Let's change this so that all *external* links open in a new tab/browser. An external link is an anchor element, `<a>` whose `href` attribute *begins* with `http` (your code should not affect any other anchor elements).

   To make a link open in a new tab/browser, you can set the anchor element's `target` attribute to the value `_blank`.

   Hint: think of how you might do this using an advanced CSS selector.

1.2 Teaser Preview

Many webpages give a "teaser" of their content by displaying the first few words or lines of an article and hiding the rest. A link or button is displayed to allow a user to "read more." When clicked, the rest of the content is displayed. We'll implement this feature using jQuery.

In particular, when the page is loaded, we will change the content of the `mainArticle` element to provide a "teaser" and a "Read More" button.

1. Start by saving off the HTML content of the main article's content into a *global* variable so that it can be accessed and "restored" later when the user presses the button. To save to a global variable, simply omit the usual `var` keyword.
2. Use jQuery to get the text content of the `leadParagraph` and truncate it to, say the first 50 characters. You can truncate a string using the `substr` function. For example, `s.substr(0, 50)` would return a new string containing only the first 50 characters of `s`.
3. Change the content of the main article to the truncated string to provide a teaser of the article. In addition, you should include HTML that provides a link or button that the user can click to "Read More".

   In the button, specify an `onclick` attribute that calls the `reveal()` function.
4. Define the `reveal()` function that *restores* the original content of the main article. Give the user a visual cue of the change by fading the main article in slowly.

1.3 Collapsable Comments

Notice that the page has a comments section. Many popular forums have *threaded* comments: users post and reply to comments with each reply being nested (indented) under the original comment. Inspired by Reddit, we have made it so that each comment and sub-thread can be collapsed or expanded for easier reading. Try it out with the "Collapse All" button (which changes to an "Expand All" button).

In this exercise, you will implement this functionality for each individual comment. The "collapse" links, `[-]` and the "expand" links `[+]` are already clickable and invoke a function `commentToggle()`. You will need to implement this function by:

1. Determine if the action is collapsing or expanding the comment.
2. Slide down or up the relevant `commentBody` `<div>` element.
3. Change the toggle "button" identified by `commentToggle` (that is, switch between `[-]` and `[+]`)

To help you, note that the `onClick` attribute invokes the `commentToggle()` function passing it a `commentNumber` that can be used by jQuery to select the relevant `commentBody` and `commentToggle` elements.

2 jQuery UI Widgets

The jQuery UI library provides several standard graphical user interface *widgets* that can be brought into your web apps and pages that provide various dynamic functionality. These widgets are also highly customizable.

For this exercise, you will familiarize yourself with the *accordion* widget that allows multiple sections of text or data to be collapsed so that the user only has to focus on one section at a time.

1. Open the `majors.html` page and note that we've provided several sections describing majors and minors offered by our department.
2. Read the basic information on the jQuery UI accordion widget:
   http://jqueryui.com/accordion/
3. Integrate the accordion widget and apply it to the content in the `majors.html` page
4. By default, the height of the accordion is the maximum height of any of the subsections. Change this by configuring your accordion with the "no auto height" option