jQuery/AJAX exercises

Before starting the exercises, download this tutorial kit. Download a copy of jQuery into the folder (make sure the script import in starterkit.html is consistent with the name of your copy).

Open custom.js and starterkit.html in your favorite text editor. We love vim/emacs, but we find Sublime Text is more convenient for frontend work. Syntax highlighting and linting are easier to set up.

Handy Chrome extensions: Color Generator (pick hex colors), JSON view

The starter kit gives you all the necessary HTML and CSS; you should only have to write Javascript. Do NOT add inline HTML or change the CSS to complete the exercises. Use the jQuery API freely, but try not to Google solutions.

Feel free to skip ahead if the early exercises are too easy.

DOM

Beginners – to help you learn the basics of jQuery DOM manipulation/traversal, try this interactive tutorial. You can skip exercise D.

The stylesheet in the tutorial kit comes with some rules for classes 'red', 'blue', and 'green' on lines 79 to 81. Using only Javascript:

1. Change the background color of the first ordered list in starterkit.html to red.
2. Give all the items in that list blue font.
3. Append the text “ in the list!” to each element in the first ordered list.
4. Select every link that has a “name” attribute, and change its background color to #eee.
5. Insert a button with id “ajax-button” and text “More!”, and then a div with id “ajax” and inner HTML “Hello World!” at the very top of the document. We'll use this later.

Event handling

1. Set up an event handler to show an alert whenever the “Some link” link is clicked. (Add your code to custom.js, instead of putting an inline onclick in the HTML.)
2. Look at the “jQuery Tablekit” table. Give any hovered row in the table green font. (And return it to normal if the user is no longer hovering.)
3. Look at the HTML structure of the “Bird FAQ”. Toggle (hide or show) the FAQ answers whenever “First button” is clicked.

AJAX

Many APIs require you to register a user account and obtain an authentication token before making requests, so that they can place limits on the number of requests you make. For simplicity, we'll only be using APIs without that restriction for these exercises.

This hipster filler text API returns JSON containing filler text. Example request for four paragraphs: http://hipsterjesus.com/api/?type=hipster-centric&paras=4

(Note that the responses are deterministic with regard to their parameters, so a request for one hipster-centric paragraph will always return the same paragraph.)

1. Using HipsterJesus, fetch three paragraphs of filler text and set them as the inner HTML for the #ajax div we created in DOM exercise 5.

2. Every time the #ajax-button button (created in DOM exercise 5) is clicked, append a single new paragraph of filler text (that hasn't already been used) to the #ajax div. There are several ways to do this. Your method doesn't need to be pseudorandom.