6G5Z2107: Web Design and Development

Lab 03 – jQuery Events and Callback Methods

Matthew Crossley

Objectives

- Be able to form jQuery selectors to obtain DOM elements
- Be able to form jQuery statements to manipulate CSS & the DOM itself

The aim of today is to reinforce the concepts we covered in this week's lecture. You can, and should, refer to the slides used in the lecture, and you may even wish to challenge yourself by running through the quiz again. We build on the foundational jQuery work we covered in last week's labs, extending our practice to include writing some event-driven jQuery. If you couldn't handle Exercise 4 of last week's lab, why not try it after this week's lab…?

Exercise 1: A Slide-down Menu

You have been given a website with a pre-prepared menu, and three divs representing content (news, aboutus and contactus).

Your task is to design a series of jQuery statements to create an interactive menu for the site. Your menu should appear when the user puts the mouse cursor over it, and disappear when the mouse moves away. Clicking on any of the elements in the menu should either make that section of the website appear if it is currently hidden, or disappear if it is currently displayed.

You will need to think about a number of things:

1. What jQuery is needed to set the page up? Do we want to hide any elements as soon as the page is loaded?
2. How can we handle the display of the menu? What type of events do we need? What do we want to happen when those events are triggered?
3. How can we handle the menu items themselves? What type of events do we need? What do we want to happen when those events are triggered?
4. The divs and the menu items have the **same id** – a problem we need to solve with our selectors. Think back to *Exercise 3* from last week to think how we might solve this (Hint: A selector such as $("div#news") will only select divs with the id 'news').

There is a short video demonstrating what your final page might look like (**Exercise One - Example**) available on Moodle – please use this for inspiration if you are not sure what you are trying to achieve.

Exercise 2: A Background of Rainbows

For your second exercise, you have been given a website with six divs, each with a different background colour. Your task is to make it so that, upon one of these being clicked, the div expands to fill the width of the page and, when the animation is complete, the page's background changes to match that of the newly expanded div.

Should another div be clicked, all of the boxes should return to their normal width (16%) and the background change back to white, before the most recently clicked box expands, and the background of the page changes again.

You will need to think about a number of things:

1. How do we define a custom animation, and what CSS properties do we need to change, to get the divs to expand to fill the width of the page?
2. How do we write code (i.e. functions) which execute once an animation has finished? Remember callbacks from the lecture? We will need to use these!
3. If you are aiming to do this well, think carefully – each of the divs has a unique ID, but actually, you can solve this entire problem without using the IDs. The best solution writes some jQuery which applies to all the div tags, but somehow treats them individually…

There is a short video demonstrating what your final page might look like (**Exercise Two - Example**) available on Moodle – please use this for inspiration if you are not sure what you are trying to achieve.

Extension Exercise 3: A Simple Browser Word Processor

We met the following code in one of the quiz questions in the lecture:

```
<script>
    $(document).ready(function() {

        $("body").on("keypress", function(event){
            $(this).append(event.key);
        });

    });
</script>
```

The answer to the quiz question suggested that this code will allow us to create a simple word processor – that is, when a key is pressed, it is appended to the page's body, thus appearing as part of the webpage.

Try this chunk of code out, and, try adding

```
console.log(event);
```

within the event code. Use Google Chrome's *Inspect* function to explore what the event variable (actually an object) contains – what other things could we do with a keypress? What things change when different keys (and types of keys) are pressed?

Your challenge here is to add one small element of functionality to your simple word processor: When the 'Enter' key is pressed, the page begins a new paragraph (<p>), into which newly typed keys appear.

This is a challenging piece of work, and you will need to recall much of what you covered last year of JavaScript (e.g. if-else statements) and combine them with what we have covered this year in jQuery.

Help! I don't know what to do!

Don't panic!

There are plenty of resources available, including the following, which are all great starting points:

- The lecture slides (available on Moodle) which recap what we covered in the lecture
- w3school's jQuery tutorial: http://www.w3schools.com/jquery/
- The jQuery Cheat Sheet: https://oscarotero.com/jquery/
- jQuery Essential Training on Lynda.com – covers more than you need to know!

The online resources are the same as last week, but you want to look at different sections of them this time! There's plenty of animation, and callback methods, available at each of the links.

Exercises 1 and 2 can be solved using combinations of everything we talked about in the lecture in some way. You may have to use them creatively, but by combining ideas, you can solve all of the problems. As with all things programming, try to solve things one at a time – e.g. for exercise one, first try hiding some of the items on the page, then turn your attention to getting the menu to re-appear when you mouseover it.

If you are having difficulties, please discuss with your tutor, who will be able to guide you through some of the early steps to solving the puzzles.

I fancy a challenge, what do you recommend?

Have a go at the extension exercise (#3). This requires you to do some research into combining JavaScript with using jQuery in ways we have not yet discussed. You may find some of the following resources useful:

- http://www.w3schools.com/js/ - You will need to use some JavaScript (from last year) for things like if statements, etc.
- **Spoiler alert:**
    - https://api.jquery.com/keypress/ this particular page might come in handy…

Next week we are going to look at AJAX – particularly for file loading – why not start getting some research in early?