# Web Design (COMP 20030)

## JavaScript II

This week, we will take a closer look at JavaScript, including the use of functions in JavaScript and how to add dynamic behaviours to a web page that react to user interactions with the various elements of the page. The following brief notes serve as an introduction to the topics relevant to this practical.

## External scripts

Functions should be written either in <script>…</script> tags in the <body>…</body> of a page, or in an **external file that is included using <script type="text/javascript" src=" name of your file " />**. The latter method is more useful, since a JavaScript function can be defined once and called on every page of the site as long as the function file is included. This not only aids organisation: users will only have to download the code once. External script sheets can also be fine-tuned using either the async or defer attributes, making it suitable for external .js to be included in the <head> of a page, along with external CSS.

## Events

An event is JavaScript is an action performed by either the user (clicking, etc.) or the browser (loading a page, etc.). Examples of user-performed actions are moving the mouse, clicking a button, typing some text, etc. In JavaScript, we write ***event handlers*** to react to these actions. An event handler is simply a piece of code that gets executed when a certain action takes place. Most events are associated with a HTML element, and the events that have handler code for an element can be defined as required.

It is easy to see how JavaScript can be used for such useful things as validation of online forms, web page presentation (image rollovers, dynamic menus, etc), and even the generation of dynamic web page content (you wrote a dynamic page in last week's practical).

# Practical Exercises

All JavaScript functions for this practical should be written in an external JavaScript sheet called 'Practical5.js'. Inline JavaScript should be avoided when answering the questions.

## *Exercise 1*

Create a web page with a form that has three text boxes named **tb1**, **tb2**, **tb3** and a button-type input named b1. The button should be labelled "go".

Create a function called myAdd() in Practical5.js. Practical5.js should be imported in the head of the web page as shown above.

When the button **b1** is clicked, myAdd() should take the contents of **tb1** and **tb2**, add them together and return the result. This result should appear in **tb3**. (Use the DOM to access the text boxes and their values). If a non-numeric value is entered in one of the text boxes, the user should be alerted of an invalid value.

`isNaN()` will be necessary to verify that numerical values are input. Hard-coding the references to the individual text boxes is acceptable. Some means to defer code execution until the DOM is constructed will also be necessary.

Save as q1.html

## *Exercise 2*

Create a web page with a form. This form should have 6 text boxes.

The form in this question can be specified using either document.forms[0] or by using getElementById(*formID*).

Add an event listener to the form that gets triggered by submit. The callback for this listener should check if any field in the form is empty. If any of the form fields is empty an alert should trigger, informing the user that they have not filled out all the values in the form.

Note: in this exercise getElementById should not be used multiple times to obtain the values from each of the 6 text boxes. Instead, the form attribute, **elements**, should be used to obtain an array of the elements of the form.

Save as q2.html

## *Exercise 3*

Create a web page.

Create a css class called tomatoSoup. tomatoSoup should have the following rule background-color: #ff7c7c;

This css can be written in a style element in the head of the page.

Create a button on the web page and add an event listener to it. This event listener should be triggered by click. This event listener should be given the function giveClass

Make the function giveClass. This function should cause the body element on the web page to have the class of tomatoSoup. This can be achieved by using classList.toggle

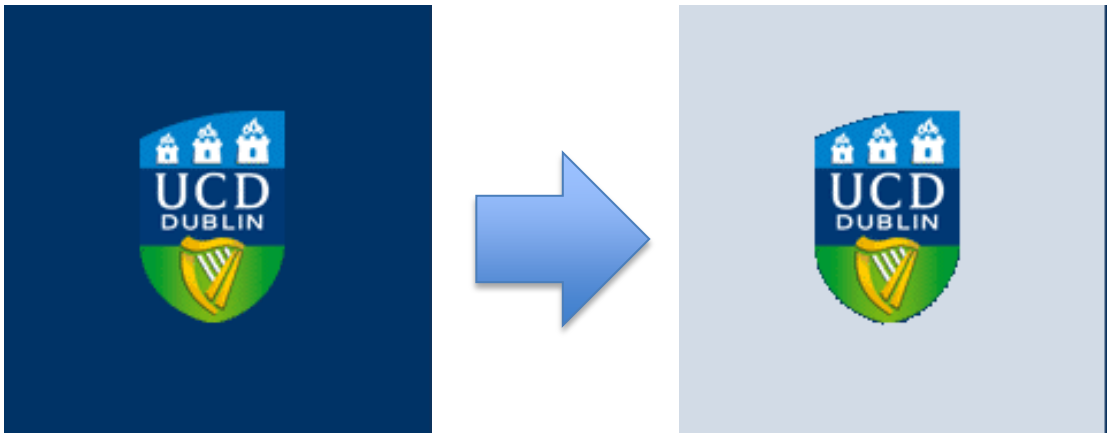
When the button on the web page is clicked, the colour of the web page should change.

Save as q3.html

## *Exercise 4*

In this exercise you will create a rollover using the UCD crest as an image.

1. Download the two sample images to a Practical5/images folder. The sample images are to be found at on the course Moodle and are called image1.gif and image2.gif
2. Prefetch the two images using link rel="prefetch"
3. Write the body of the web page to contain a single image – the preloaded image1.gif. Make the image link to the UCD home page.
4. Add an event listener for mouseover that swaps image1 for image2. On mouseout image1 should show again. This can be achieved by changing the value of the img source.




Store this file as q4.html