

COMP3006 – Full-Stack Development

Workshop 2 – JavaScript

Autumn 2020

This week's workshop will require you to implement some JavaScript programs to test your knowledge of the language following this week's lecture. You will begin by developing a small test harness – a web page with which you can test your programs. **You should complete this ahead of the workshop session you attend.**

You will also extend your work on the mini project by adding some JavaScript functionality to the interface you began developing last week.

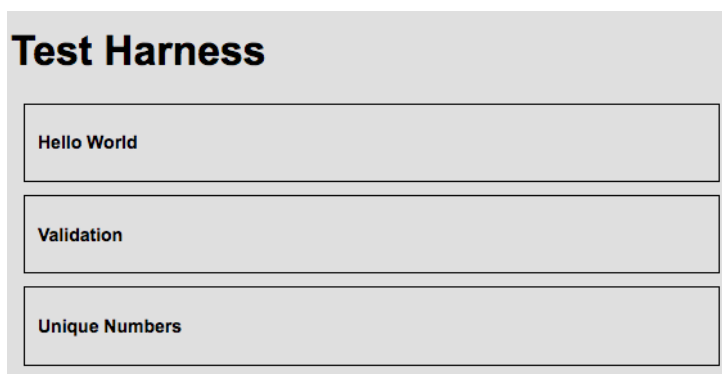
Before the Workshop

Exercise 1

Start a HTML file and name it **testharness.html**. This is the HTML document that will contain the work for the first 4 exercises in this workshop. You should do the following:

1. Give the page a title with a level 1 heading ("Test Harness").
2. Add three `section` elements, and within each add a level 1 heading ("Hello World", "Validation" and "Unique Numbers").
3. Apply some basic styles – maybe add a background colour and change the font size of the section headings.
4. Include two scripts. One should link to jQuery and the other should link to a local file called **javascript.js**

When you have finished you should have something like this (depending on the styling you've chosen):



You will need to use the doc-type html to access HTML5 elements

You can link to a CDN version – have a look on W3Schools

Your JS file doesn't have to be called **javascript.js** – that's what I've called mine.

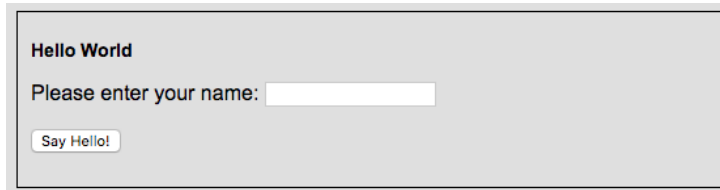
Exercise 2

This exercise will test that you've set up the test harness page properly. You will create a JavaScript program that extracts a person's name from a text input and displays it in the page with a cheerful message.

Within the "Hello World" section, add the following beneath the level 1 heading you added previously:

1. A paragraph containing a message “Please enter your name” and an input of type text with the ID `name`.
2. A button with the ID `sayHello` and the text “Say hello!”.
3. An empty paragraph with the ID `sayHello_result` below the button.

That should give you something like this:



Having set up your HTML, you should now modify the [javascript.js](#) program. Add following to serve as an onload event for the window:

```
$(function() {  
});
```

You need to set up an event handler for your button. Do this by adding modifying the program as follows:

```
$(function() {  
    $("#sayHello").click(function() {  
    });  
});
```

Within the event handler you need to do two things:

1. Extract the current value of the `userName` input.
2. Create the message and put it into the `sayHello_result` paragraph.

To do this, modify your code as follows:

```
$(function() {  
    $("#sayHello").click(function() {  
        // Extract the current value of the userName input.  
        let userName = $("#name").val();  
  
        // Create the message and put it into the result paragraph.  
        let message = "Hello " + userName + "!";  
        $("#sayHello_result").html(message);  
    });  
});
```

That completes your program and you can run it. Refresh your browser page, enter a name into the input and click the button. Your program should behave like this:



I'm using jQuery for these exercises. You could do it all in vanilla JS but you will end up writing much more code.

The **val** method extracts the current value from the specified form element and the **html** method puts the specified string into the specified element as its content

If it doesn't, have a look at the Dev Tools (**Ctrl-F12** in Chrome) and see what's in the console. There may be an error message.

During the workshop

Exercise 3

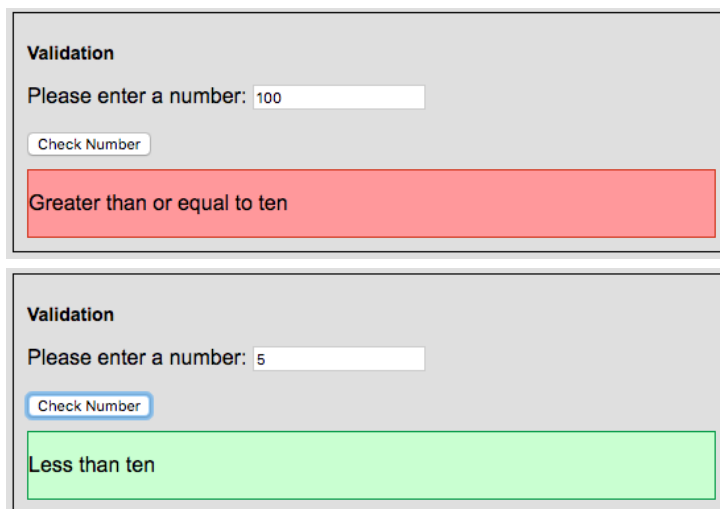
Add a form to the “Validation” section of your test harness. You should an input of type number and use a button to submit the information entered in the field for validation.



The form is titled "Validation". It contains a text input field with the placeholder text "Please enter a number:". Below the input field is a button labeled "Check Number".

You should also add a `div` element with the ID `numberMessage` which has two classes – `message` and `hidden`. Within your `numberMessage` `div` you should add an empty paragraph with ID `numberMessageText`.

When the button is clicked, if the number entered is less than ten you should display a message in the `numberMessage` that tells the user “less than ten”, with a green background. Otherwise the message should say “greater than or equal to ten” with a red background. Your message should be entered in a `div` with the class `message` to define some of the basic styling, and then have the class `greater` or `less` to refine the look and feel depending on the value. The result should look something like this:



The first screenshot shows the form with the input field containing the number "100". The "Check Number" button is clicked, and a red message box appears below it with the text "Greater than or equal to ten".

The second screenshot shows the form with the input field containing the number "5". The "Check Number" button is clicked, and a green message box appears below it with the text "Less than ten".

You may need to look up the [number](#) element.

If you can't remember how to trigger a JavaScript function using a button, look at the model answer for last week's lab session in which you did this.

Using the syntax

`<elem class="a b">` assigns both class `a` and class `b` to the element.

In order to view the message you will have to remove the `hidden` class from the `numberMessage` `div`. You can use the jQuery method [removeClass](#). You can use the corresponding [addClass](#) to add the `greater` or `less` class to the message. Look up their documentation online to see how they work

Exercise 4

Write a function that, when given an array of numbers, finds the number of *unique* elements in that array. The function should return the number of unique elements as an integer.

Having written the function, create a website that provides the user with a form. The user should type in numbers separated by commas as shown:

1,4,3,4,4,7,0,9,3

You should then write a JavaScript program that takes this input, creates an array from the numbers, and displays the number of unique elements in the webpage when a button is clicked.

Add the HTML for this section to the “Unique Numbers” section of your test harness. The numbers should be entered in a HTML form (use an input of

This exercise requires you to construct the formal [set](#) of elements in the array – JavaScript will do this for you, but the intention is for you to practice using loops and conditionals. Please only use the inbuilt set operations if you are experienced with JavaScript.

type “text” so that you can include the commas) and display the list of unique numbers in the page beneath the form.

Mini Project

This part of the workshop extends the mini project you began last week. In the previous session you began implementing a user interface for your online chat, though it didn't do anything. In this session you will start to add behaviour to the interface. A vital component of any web-based system is validation, ensuring that the users cannot misuse the system. Your task here is to add some simple validation to your interface.

As with last week you must use GitHub to commit any changes you make to the code.

Exercise 5

You should consider the requirement that for a message to be sent there must first be a message in the form. You must use JavaScript to prevent the user from sending an empty message. To do this, you must **disable** the button so that it is only enabled when there is text in the text box.

There are different ways you could do this; you could look at attaching an event listener to the text box so that you check if pressing a key causes text to go into the box. If so, you can enable the send button. You will find resources online that will help with this.

Extension task

Exercise 6

Add a section to your test harness and implement a program that uses a **promise** to put the program to sleep for a random number of milliseconds before waking it again. The sleep should be triggered by a button. You should add two paragraphs to the section so that you can print out the current system time when the program goes to sleep, and the current system time when the program wakes up again.

You will need the **Date** to get the current system time, and the **setTimeout** method to put the program to sleep.