CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1

Due Date: Check Moodle Class Instance
Time: Check Moodle Class Instance
Weight: 5%

Introduction

In this lab you are going to use the skills developed in the last couple of weeks to add a submit event to a form and validate user input. The html and css are completed but you can make alterations to both the css and the html as you require.

Check The Script Tag

Check the HTML page and make sure the script tag is in the head element and that the tag contains a valid src attribute and is marked with the defer attribute.

Open the web page and scroll to the bottom. Notice that the script tag is at the bottom of the page.

This is no longer considered to be a best practice in JavaScript. so we need to update the page markup.

Task

```
</body>
  <script src="js/index.js"></script>
</html>
```

You need to update the page markup. Move the script tag to the head.

Task

```
    <head>
      <title>Form Validation</title>
      <link rel="stylesheet" href="css/reboot.css" />
      <link rel="stylesheet" href="css/styles.css" />
      <script src="js/index.js" defer></script>
    </head>
```

Tell Me More About Async and Defer

When the browser loads HTML and comes across a script tag <script src="..."></script> the browser must wait for the script to download, then execute the downloaded script, and only then can it process the rest of the page.

The defer attribute tells the browser not to wait for the script. Instead, the browser will continue to process the HTML and build the DOM. The script loads "in the background", and then runs when the DOM is fully built.

The async attribute is somewhat like defer in that it makes the script non-blocking. But it has important behavioural differences. Async scripts load in the background and run when ready. The DOM and other scripts don't wait for them, and async scripts don't wait for the DOM Load or the DOMContentLoaded events, they are fully independent and run when loaded.

Start Coding

Open the index.js file and enter a console.log command just to check that you script tag is correctly linked

Task

```
console.log("script running")
```

CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1

Create Your Reference Variables

For the example we need to create reference variables to the form submit button, the input field and the warning element.

Task

```
const formToCheck = document.querySelector('#check-types')
const checkInput = document.querySelector('#field-to-check')
const warning = document.querySelector('.warning')
```

Add Submit Event to the Form

When starting out writing javascript we often make the error of adding the submit event to the button and not to the form. Make sure when you add the submit event correctly. If you click on the button you will see the console.log statement appear then disappear rapidly.

Task

```
formToCheck.addEventListener('submit', function(e){
        console.log("submit event")
})
```

Stop The Submit Event Default Behaviour

The role of the form is to send the name value pairs of the input to the script specified by the action attribute using the http method found in the method attribute. We need to stop the form from submitting in order to validate the input value on the client.

Task

```
formToCheck.addEventListener('submit', function(e){
        console.log("submit event");
        e.preventDefault();
})
```

Trim the Incoming String

We need to remove any white space from the input string. To do this we use the javascript string.trim() method. This method will remove white space from the start and the end of a string.

Task

```
formToCheck.addEventListener('submit', function(e){
        e.preventDefault();
        let fieldValue = checkInput.value.trim()
})
```

Convert Input Data To A Number

To validate that the string is in fact a number we can convert the string to a Number by wrapping the check input value in the Number() wrapper object. If the string is a number we will get a numeric value and if the string has any characters other than a decimal point the return value will be NaN which stands for not a number.

Task

```
formToCheck.addEventListener('submit', function(e){
        e.preventDefault();
        let fieldValue = Number(checkInput.value.trim());
})
```

Check The Field Value against isNaN()

Using conditional statements is a good way to test that the field value is in fact a number. If it is, we can make the calculation and update the display box. JavaScript has a way to test a value to see if the value is in fact NaN (not a number). The isNaN() takes a single parameter the value you want to check.

Test the field value and add some input. Try it with text, text+numbers, numbers. Make sure that you log out to the console and test your conditional logic.

Task

```
formToCheck.addEventListener('submit', function(e){
        e.preventDefault()
        let fieldValue = Number(checkInput.value.trim())

        if(isNaN(fieldValue)){
          console.log("issue a warning message");
        }else{
        console.log("update the display box with the value of the calculation");
        }
})
```

Display The Warning

If the return value from the isNaN(fieldValue) is true. Then the data is not a number and we need to offer a helpful hint to the user to order that they can enter the correct data.

To do this we need to access the paragraph element with the class warning (look at the markup). Currently the element is hidden. If you look at the css of the class called hide the display property is set to none. Doing this hides the element in the DOM.

Using the JavaScript element property **classList** will give you a list of the classes attached to a given element.

To add a class to an element use ***element.classList.add('classname")***
To remove a class use ***element.classList.remove('classname')***.

Task

```
formToCheck.addEventListener('submit', function(e){
    e.preventDefault()
    let fieldValue = Number(checkInput.value.trim())
    if(isNaN(fieldValue)){
            warning.classList.remove('hide')
            warning.querySelector('span').textContent = "helpful hint"
    }else{
            // make calculation and update the display
    }
})
```

Make The Calculation And Update The Display Box

If the return value from the isNaN(fieldValue) is false then the field value is in fact a number and we can use it to make our calculation.

Task

```
formToCheck.addEventListener('submit', function(e){
    e.preventDefault()
     let fieldValue = Number(checkInput.value.trim())

       if(isNaN(fieldValue)){
          warning.classList.remove('hide')
          warning.querySelector('span').textContent = "helpful hint"
        }else{
          displayField.textContent = fieldValue * 10
      }

})
```

Removing The Hint When THe User Clicks Back In The Input.

If you test the form at this point it would seem that everything is working. However when the user clicks back into the field the warning message is still visible. We need to hide that element again by re-attaching the ‘hide’ class.

To do this we need to add a focus event on the input element. The focus event is triggered when the input element is selected by the user either by touch or mouse action or by selecting the input using the tab key.

Note below that the checkInput.value = '' is two single quotes with no space between them.

Task

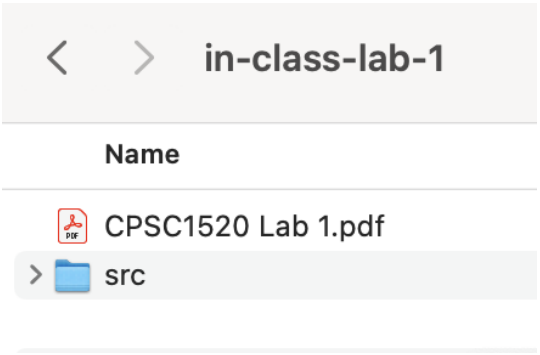
```
checkInput.addEventListener('focus', function () {
    warning.classList.add('hide')
    checkInput.value = ''
})
```

CPSC1520 Client Side Scripting With JavaScript

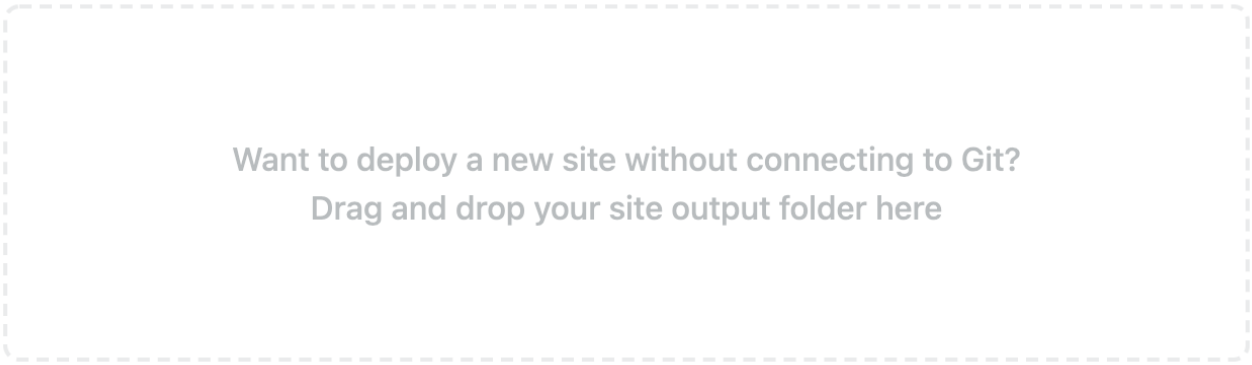
In Class Lab 1

Submission

When deploying to netlify you are going to add the src folder.
Do not add the whole lab-1 folder or the deployment will not work.



Login to your Netlify Account
Go to the sites tab.
Scroll to the bottom of the page.
Located the deployment drop box



Drag your lab folder on the drop site.
Netlify will build and deploy your site.
Test the site and make sure your form works.
Copy the link and submit it via moodle.

No Late Submissions.

CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1

Tasks	Grade	Marks
Form Event Listener		1
Input focus listener		1
Display warning		1
Update the display with calculation		1
Deployment to Netlify		1

Sub-Total		/5

CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1

CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1

Marking Rubric

Marks	5 Marks Criteria
5	Task was completed with the highest of proficiency adhering to best practices and followed subject matter guidelines all tasks were completed to a professional standard.
4	Task was completed well some minor mistakes. Well above average work shows good understanding of the task and high degree of competence
3	Satisfactory work some features missing or incorrectly implemented. Show a moderate level of understanding in the task with room for improvement.
2	Below average work. Task was poorly complete. Show understanding of the task and the requirements to implement but implementation was poorly executed.
1	Some of the task was completed. Showed a lack of understanding in the subject matter and very poorly executed
0	Not completed.

Marks	3 Marks Criteria
3	Proficient shows a high degree of competence in completing task.
2	Capable above average degree of competence in completing task
1	Satisfactory shows a satisfactory degree of competence in completing task.
0	Shows a limited degree of competence in completing task.

Marks	1 Marks Criteria
1	Task Completed satisfactorily
0	Task was not

CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1