

The background features a decorative pattern of hexagons. The hexagons on the right side contain blurred images of code snippets, likely from a web development framework like Bootstrap, showing terms like 'col-xs', 'has_post', and 'post'. The hexagons on the left are solid light grey.

CSY1063

Web Development

Week 8

Chris.Rafferty@northampton.ac.uk



Learning Objectives

- This week we will be covering
 - Linking JavaScript to HTML
 - alert()
 - Variables
 - let
 - const
 - Functions
 - Console.log
 - Selecting HTML elements
 - Click events



JavaScript

- HTML and CSS can be used to display information and present it in a specific way
- However, CSS and HTML have limitations:
 - You cannot change the contents of the page after it has been drawn on the screen
- HTML is not very interactive, you can display a page but not control what happens when the user interacts with it

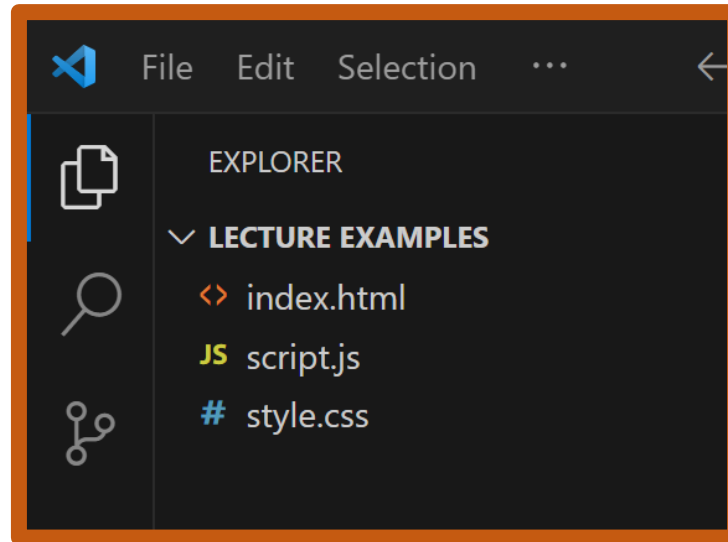


JavaScript pt2

- HTML is a Markup Language this means it describes how data is structured.
- When the HTML code is run, it is interpreted by the browser to generate a visual output and run in order, line by line top to bottom
- JavaScript is a programming language. This means you as the developer have control over how the program is executed, it's not usually executed in linear fashion.

JavaScript file

- Like CSS, JavaScript code should be placed in its own file
- JavaScript files have a .js extension
- To run the .js file in a HTML page you must reference the JavaScript file using a `<script>` HTML tag



<script>

- The script tag has an src attribute (like the tag) that points to the JavaScript file
- <script> tags should go inside the pages <head> tag
 - There has been some debate about whether to place the <script> tag in the head or body tag.
 - For modern browsers the <head> tag is preferred

<script> example

- The script tag does not need anything between <script> and </script> but both start and end tags are required

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="script.js"></script>
</head>
```

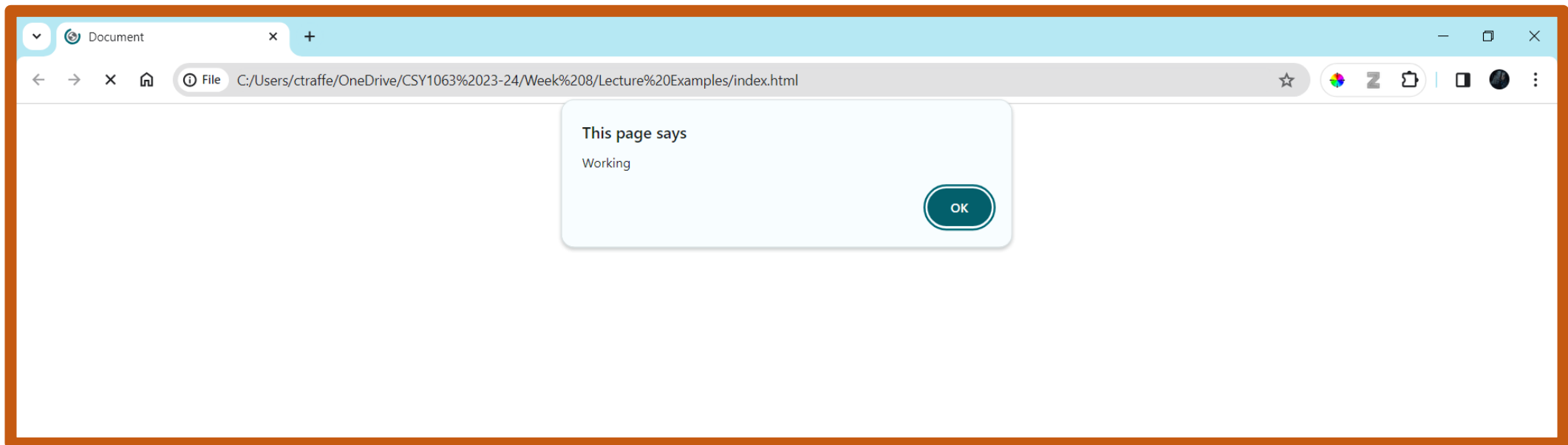
alert()

- To check your script.js is loading correctly you can add some code to it
- JavaScript includes the function alert
- The alert function lets you create a pop-up alert box to display some text
- Using this code in script.js:

```
alert('Working');
```


alert() example

- This will look different in each browser as the browser controls what happens when alert() is called





alert() pt2

- You'll notice that when the alert popup appears that the contents of the page are not visible behind it
- Once you click 'OK' the contents of the page appear
- This is because the JavaScript runs before the page has been drawn on the screen



JavaScript uses

- JavaScript can be used to control HTML elements on the page
- JavaScript can be used to:
 - Assign CSS to the element
 - Add or remove HTML attributes
 - Read the contents of form elements
 - Detect when an element is interacted with (moused over, clicked, typed into, etc)
 - Control what happens when a key is pressed



Variables

- JavaScript allows you to give values labels
- A label is called a variable and can store a single value
- You can give the variable any name you like, this is chosen by you, not JavaScript
- The value of the variable is also chosen by you
- In modern JavaScript there are two different types of variables
 - Let
 - Const

Variables - let

- The first type of variable is called a let
- To declare a let variable use the code

```
let variableName = 'value';
```

- You can give the variable any name you like, this is chosen by you, not JavaScript.
- The value of the variable is also chosen by you
- The only parts that are defined by the language are
 - The let keyword
 - “=” sign

Variables - let (reassign)

- By declaring a variable with let we can reassign the value of that variable at any time
- The code below shows the value of variableName when used with an alert() the value is 'and again!'

```
let variableName = 'value';  
  
variableName = 'this now';  
variableName = 'changed again';  
variableName = 'and again!';
```

This page says
and again!

OK

Variables - const

- The second type of variable is called const (constant)
- To declare a const variable use the code

```
const variableName = 'value';
```

- Const variables cannot be reassigned, they cannot be changed once they have been declared
- If you try changing the value of a const you will get an error

var (old)

- The var keyword was used before let and const
- It functions very similarly to let
- You can reassign the value
- If you see an example using

```
var variableName = 'value';
```

- Use let instead, var is mostly used to be comparable with some older versions of JavaScript.

Which to use

- In general you should not be using var
- Let and const are the two variables you should be using

#	var	let	const
Uses	Legacy code, backward compatibility	Preferable for most use cases	Preferable for variables that wont be reassigned
Reassignment	NA	Can be reassigned	Cannot be reassigned
Why use	Avoid it in modern JavaScript	Default choice for variables	Use for constants and variables that wont be reassigned

Data types

- The two simplest types of variable in JavaScript are:
 - Numbers
 - Strings (text)
- To assign a number variable you can use

```
let number = 10;
```

- To assign a string variable you must surround the string with quotes

```
let string = 'text';
```



Variables pt2

- The code can be translated into English
- let can be read as 'Create a variable called'
- And the = symbol can be read as 'and set it to' The code on the previous slide can be read as:
 - Create a variable called number and set it to 10
 - Create a variable called string and set it to text

Using variables

- Once you have stored a value inside a variable (label) you can reference it later, for example in the alert function
- This will create an alert box with the text 'Script loaded'

```
let alertText = 'Script loaded';  
alert(alertText);
```



Strings

- A string is a piece of data that will be used in the program
- Anything in quotes is treated as data
- Anything that's not inside quotes is treated as a JavaScript command
- JavaScript sees the opening quote as 'Start of string' and the next quote as end of string

```
let text = 'This is a string';
```



String pt2

- The first quote designates the start of the string
- The second quote designates the end of the string
- Anything in quotes is data
- Anything outside of the quotes is treated as a command
- Your editor will colour code strings from commands
 - Orange for strings
 - White for commands
 - Yellow for functions
 - Ect.



Command vs data

- Remember:
 - Anything in quotes is data to be processed
 - Anything not in quotes is a command
- It is important that you understand the difference as this is a very common problem beginners face

Semicolon

- Note that each statement must be ended with a semicolon ;
- The semicolon means 'end of statement' and can be thought of like a full stop in an English sentence.

```
let text = 'This is a string';  
let number = 10;  
  
alert(text);  
alert(number);
```




JavaScript syntax

- Spelling matters!
- Even a slight typo will stop the entire script from running
- Javascript is cAsE SenSItive.
- If you create a variable with uppercase letters you must reference it with uppercase letters throughout your program!
- Missing a bracket, semicolon, dot, or other seemingly trivial mistake will prevent your code from working.
- Get used to looking for the tiniest of mistakes!

Variables pt3

- You can create as many variables as you like
- Once a variable is created you can perform operations on it
- For numeric variables you can perform mathematical operations e.g.

```
let num1 = 5;  
let num2 = 10;  
  
let total = num1 + num2;  
  
alert(total);
```

This page says

15

OK



Exercise

- Create a basic web page with a html, body and head tag.
- Inside the body tag, place a <h1> tag and a <p> tag that contain text of your choice
- Add a script tag that references a file called script.js
- Create script.js and add an alert() command.
- Change the text to say whatever you like.
- Make sure your html file and JavaScript file are in the same directory.
- Open your html file in a browser and verify that the message is shown

Exercise 2

- Create two number variables one that equals 10 and the other 15
- Try the following mathematical operations and use alert() to show the output
 - +
 - -
 - *
 - /
 - % (<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Remainder>)
- Try using the + operation on two strings

Functions

- You can label a block of code using a function
- This will store the code for later use where it can be referenced and used
- This allows you to write code out of sequence

```
function addition() {  
    let num1 = 5;  
    let num2 = 10;  
  
    let total = num1 + num2;  
  
    alert(total);  
}
```

Function breakdown

- The syntax for a function looks like this

Function name (chosen by you, can be anything)

Function keyword

```
function addition() {  
  let num1 = 5;  
  let num2 = 10;  
  
  let total = num1 + num2;  
  
  alert(total);  
}
```

Code to run, as many lines as you like, needs to be between braces { and }



Functions pt2

- When code is stored inside a function it is not executed as it's defined
- Once a function has been defined it has to be called
- A function is called using the name followed by brackets
- Normally code gets run in the order it is written
- Functions allow you to run code in a different order

Calling functions

- To run the code in the addition function, it must be called using the code
- `addition();`

```
function addition() {  
    let num1 = 5;  
    let num2 = 10;  
  
    let total = num1 + num2;  
  
    alert(total);  
}  
  
addition();
```




Functions pt3

- You can think of a function like a program on your computer
- Creating a function is like installing the program
- It is there on the computer but doesn't do anything until you run it
- The JavaScript interpreter will ignore code inside function blocks until it is called
- If code is inside a function but the function is never called the code inside the function will never run



Reducing duplicated code

- This allows you to repeat code by calling the function more than once
- Putting repeated code inside a function is quicker and easier than typing it out twice but will produce the same result

Reducing duplicated code example

- Both these scripts will produce the same output

```
function addition() {  
    let num1 = 5;  
    let num2 = 10;  
  
    let total = num1 + num2;  
  
    alert(total);  
}  
  
addition();  
addition();  
addition();  
addition();
```

```
let num1 = 5;  
let num2 = 10;  
let total = num1 + num2;  
  
alert(total);  
  
let num3 = 5;  
let num4 = 10;  
let total2 = num3 + num4;  
  
alert(total2);  
  
let num5 = 5;  
let num6 = 10;  
let total3 = num5 + num6;  
  
alert(total3);  
  
let num7 = 5;  
let num8 = 10;  
let total4 = num7 + num8;  
  
alert(total4);
```

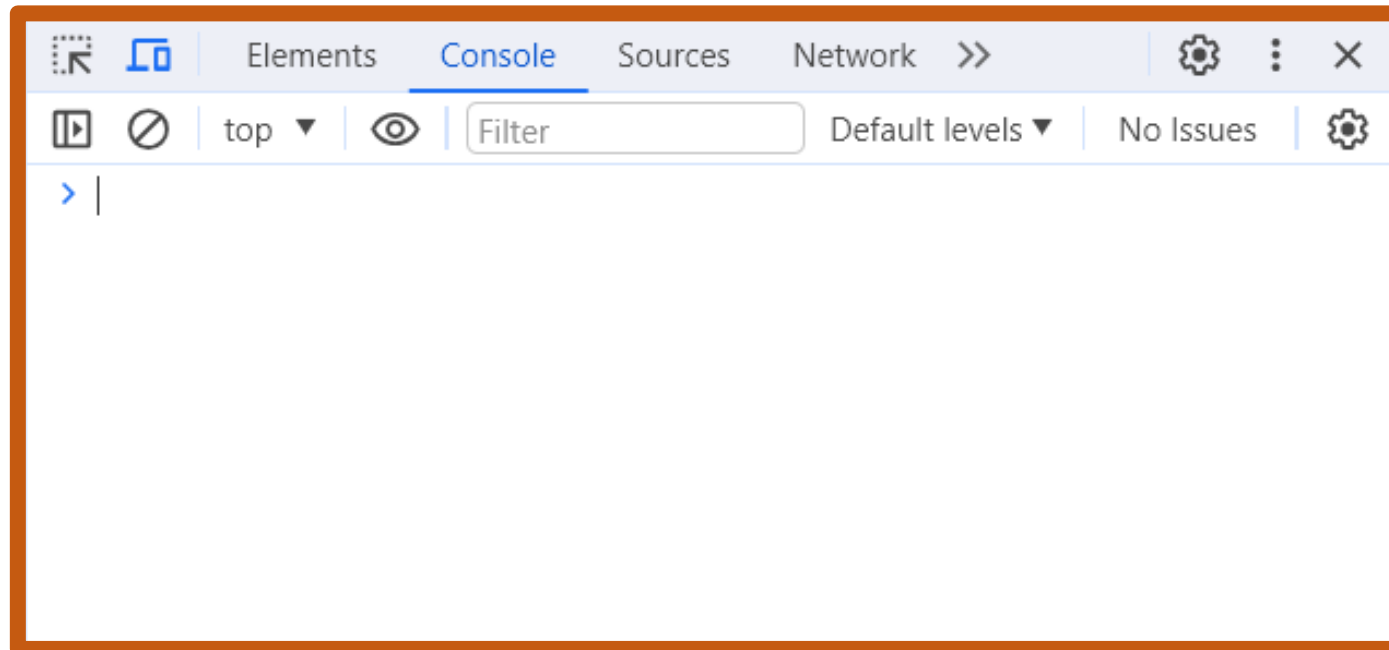


Alert is annoying!

- Clicking through all those alerts can be annoying
- You can print to the JavaScript console
- This is a tool available in browser's developer tools
- In most browsers you can open it by pressing F12 on the keyboard

console

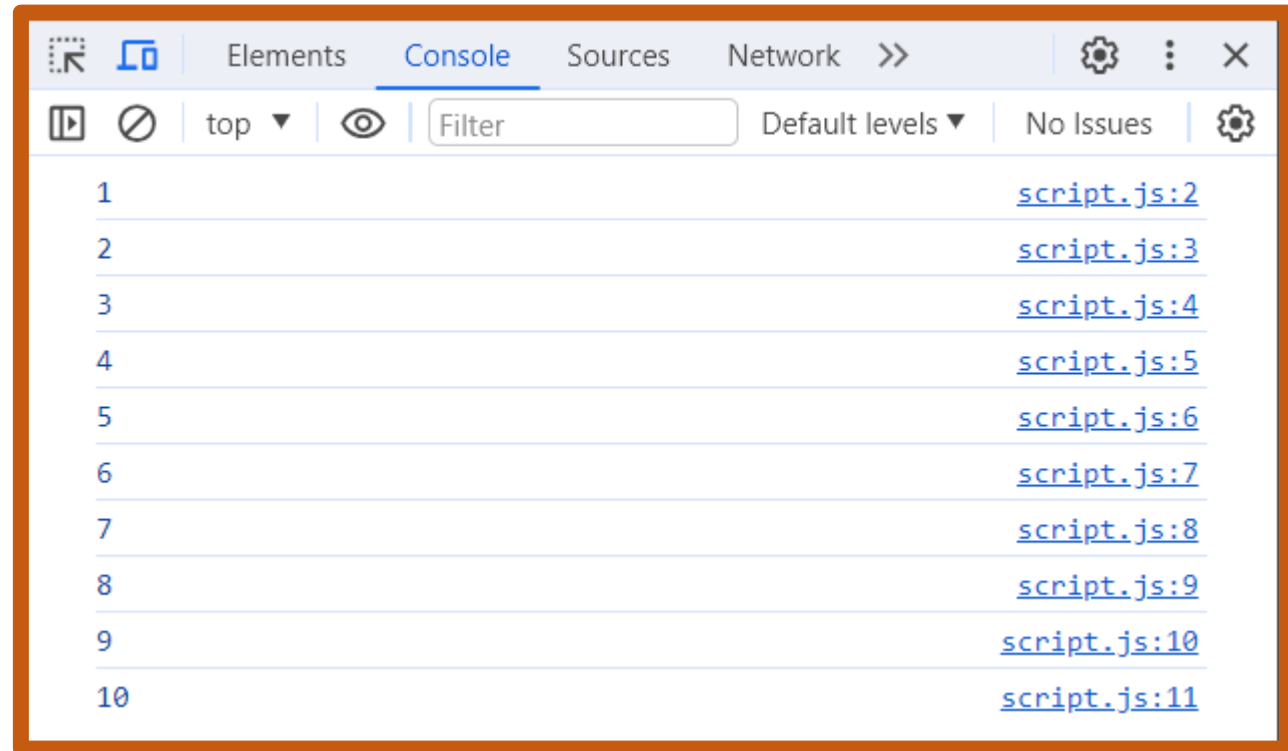
- We are going to be using the console for JavaScript



console.log()

- We can use console.log() instead of alert()

```
function printNumber() {  
    console.log(1);  
    console.log(2);  
    console.log(3);  
    console.log(4);  
    console.log(5);  
    console.log(6);  
    console.log(7);  
    console.log(8);  
    console.log(9);  
    console.log(10);  
}  
  
printNumber();
```



Selecting elements in JavaScript

- JavaScript contains inbuilt functions for selecting HTML elements
- We can then change the properties using CSS in JavaScript
- The simplest way of getting an element is by using an ID

```
<body>  
  <h1 id="heading">Heading</h1>  
  <p id="content">Content</p>  
</body>
```

querySelector()

- Once an element on the page has an ID, we can use the JavaScript function `document.querySelector()` to select it and store the element in a variable
- Just like CSS when we select an element by its ID we need to use #

```
let heading = document.querySelector('#heading');
```

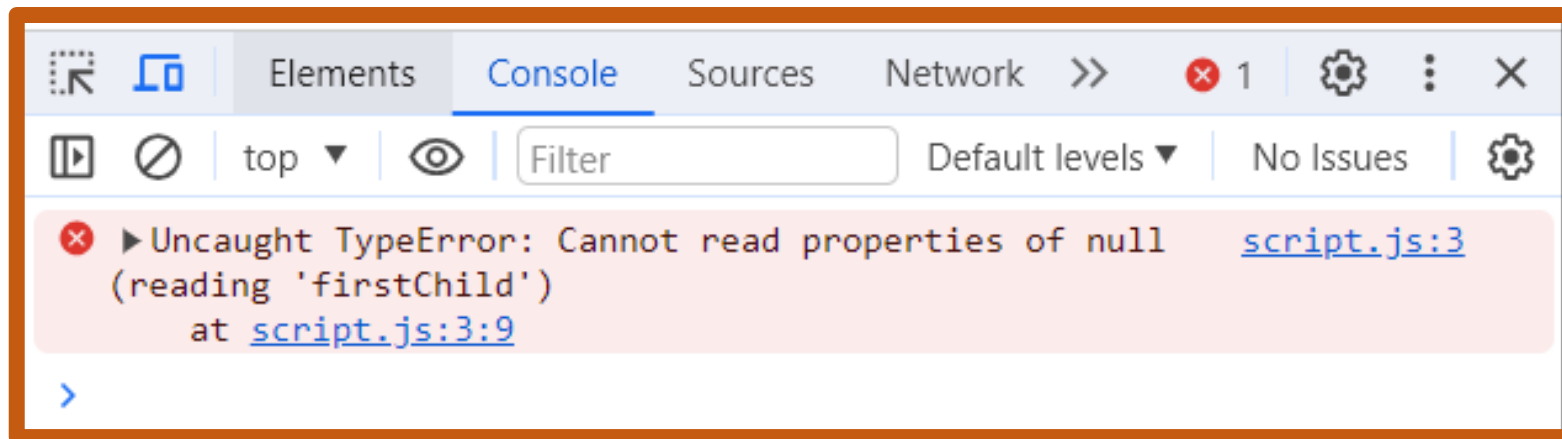

Change the text

- Once you have the element selected you can then make changes to it
- To update the text you can use `variable.firstChild.nodeValue`

```
let heading = document.querySelector('#heading');  
heading.firstChild.nodeValue = 'This has been changed!';
```

Error

- If we actually run the code it wont quite have the desired effect!
- Hint: Always keep the console open as it will tell you if there are any errors in your code





Fixing the error

- Remember the first alert box()
- The JavaScript code is run before any elements exist on the page, which is why the code is failing
- Rather than having the code run before the page has loaded, we can use an attribute to notify the browser to load our HTML first and then run our JavaScript

defer

- You can use defer in the script tag
- Defer can be used to specify that the code in our script should be executed after the HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script src="script.js" defer></script>
</head>
```

Error resolved

- When we run the code now the text in the <h1> now changes to what was specified in the JavaScript code

```
let heading = document.querySelector('#heading');  
heading.firstChild.nodeValue = 'This has been changed!';
```

This has been changed!

Content

Event Listeners

- We can use event listeners to run a function when a specific event occurs
- This could be
 - A click on an element
 - When the page loads
 - Specific key has been pressed or released
 - The mouse hover over an element
 - Etc.

```
element.addEventListener('EVENT', functionName);
```

Event Listeners pt2

- We can use DOMContentLoaded in a similar way to how defer was used
 - It will only run the changeHeading function once the HTML has loaded
 - 'document' refers to the whole web page

```
function changeHeading() {  
    let heading = document.querySelector('#heading');  
    heading.firstChild.nodeValue = 'This has been changed!';  
}  
  
document.addEventListener('DOMContentLoaded', changeHeading);
```

When this happens

Run this function

Click events

- There is also a 'click' event which triggers whenever an element is clicked on
 - The heading will only change when the user clicks on the page

```
function changeHeading() {  
    let heading = document.querySelector('#heading');  
    heading.firstChild.nodeValue = 'This has been changed!';  
}  
  
document.addEventListener('click', changeHeading);
```




document

- If you click **anywhere** on the document, the contents of both elements will be updated
- It's possible to assign a click event to a particular element
- You can call `element.addEventListener` to add an event to a specific element
- This works the same way as `document.addEventListener` however it will only call your function when that element is clicked on

Click event on an element

- We can run the changeHeading function only when the heading has been clicked on

```
let heading = document.querySelector('#heading');  
  
function changeHeading() {  
    heading.firstChild.nodeValue = 'This has been changed!';  
}  
  
heading.addEventListener('click', changeHeading);
```

Only when heading is clicked

Forms

- When we use `<input>` (week 4) you can read the contents of the box (the value the user has typed in) using the `.value` property

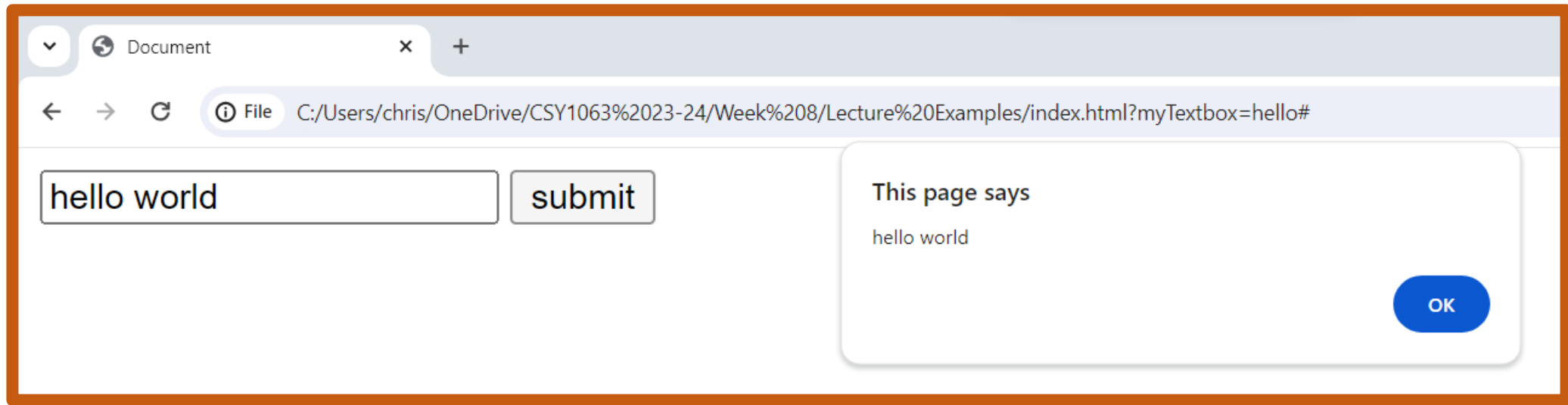
```
<form action="#" method="get">
  <input type="text" name="myTextbox" id="myTextbox">
  <input type="submit" value="submit" id="submit">
</form>
```

```
const submit = document.querySelector('#submit');

function readContent() {
  const textBox = document.querySelector('#myTextbox');
  alert(textBox.value);
}

submit.addEventListener('click', readContent);
```

Forms example



The screenshot shows a web browser window with a single tab titled "Document". The address bar displays the file path: `C:/Users/chris/OneDrive/CSY1063%2023-24/Week%208/Lecture%20Examples/index.html?myTextbox=hello#`. The main content area contains a text input field with the text "hello world" and a "submit" button. To the right of the form, a message box is displayed with the text "This page says hello world" and an "OK" button.



Exercise 3

- Write a function called `print5` that has 5 alerts, the numbers 1 – 5 in their own alert box
- Call the function twice
- When the page loads you should get 10 alert popups in total

- Replace `alert()` with `console.log`
- Open the console and refresh the page
- Make sure you can see where the numbers have been printed to!



Exercise 4

- Add a `<h1>` element to your page
- Give the `<h1>` element an ID
- Using `addEventListener` update the content of the `<h1>` element on the page when it loads
- Make sure it works by testing it in your browser

- Add another element to the page such as a `<p>` and update the contents the same way you did with the `<h1>`
- You should only need one function and one line with `addEventListener`



Exercise 5

- Use a click event instead
- Refresh the page – the `<h1>` and `<p>` elements should start with one value and change when you click somewhere on the page!
- Make it so that when `<h1>` is clicked the contents of the `<h1>` element is updated (and not the contents of `<p>`)
- Make it so that when the `<p>` element is clicked the contents of the `<p>` is updated (not the contents of `<h1>`)
- **Hint:** You will need two event listeners and two functions

Exercise 6

- Create a HTML form with the following:
 - 1 text box (`<input type="text" />`)
 - 1 button (`<input type="button" />`)
- When the button is pressed display the value in the text box in an alert
- Add a `<div>` element to the page
- When the button is pressed instead of displaying the contents of the input in the alert display it in the created `<div>`

links

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps
- <https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector>
- <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script>
- <https://developer.mozilla.org/en-US/docs/Web/API/Window/alert>
- https://developer.mozilla.org/en-US/docs/Web/API/console/log_static
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions>
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Variables
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events
- <https://www.linkedin.com/learning/javascript-essential-training/scope?u=69908402>
(good LinkedIn learning course)