ROBERT GORDON UNIVERSITY ABERDEEN

Scripting for User Engagement

CM1114

Kyle Martin

What we will learn

- What is scripting?
- What is JavaScript?
- JavaScript Engines
  - Embedded Engines
  - Server Engines
- Using JavaScript to adapt HTML Content

What is Scripting?

- Scripting is different from coding
  - Coding is an umbrella term for writing in any technical language
  - Scripting is a specific type of coding

- Script is mostly used to make interfaces dynamic
  - Code to automate processes which might normally have to be executed in sequence

Scripting
VS
Programming Languages

Comparison Chart

Programming Language	Scripting Language
A programming language is an organized way of communicating with a computer.	A scripting language is a programming language that supports scripts.
Traditional programming is based on low level languages.	Scripting prefers high level languages.
The traditional programming languages such as C, C++, and Java are compiled.	Perl, Python, JavaScript, and other languages used for scripting are interpreted and do not require the compilation step.
General programming leads to closed software applications.	Scripting promotes open projects and is used for web applications.

What is JavaScript?

- JavaScript is a scripting language
- Invented to ‘make webpages come alive’.
- It is a means to interact with HTML/CSS using code
- NOT JAVA – they just used the name to sell the language

Why Use JavaScript?

- JavaScript is the most widely used tool to create browser interfaces for a reason – three reasons, in fact:
  - Full integration with HTML/CSS
  - Simple things done simply
  - Supported by all major browsers

- Combined, these three things exist in JavaScript and **no** other browser technology.

Where to Execute JavaScript?

- JavaScript can only execute in a JavaScript engine
  - Can exist on a server, or embedded within a browser
  - Sometimes called JavaScript virtual machine
  - Usually has a cool code name

- Different JavaScript engines support different features
  - Another reason to try your code in as many browsers as possible!

How Does a JavaScript Engine Work?

- Engines are complicated – so we will only talk about them in an extremely simplified manner

- When presented with a webpage, a JavaScript engine basically does 3 things:
  1. Parses (reads) the script
  2. Compiles (converts) the script in machine language
  3. Runs the code

- At each step the engine optimises the code so that it runs fast

Where Can One Find a JavaScript Engine?

- Basically – almost everywhere
- Client-side and server-side scripting
- Two important ones (or at least, for our purposes):
  - JavaScript Engine that exists on a server
  - JavaScript Engine embedded in a browser
- Each has different permissions – embedded engines are not allowed to do nearly as much as server engines

What Can an Embedded Engine Do?

- Capabilities depend on the JavaScript environment

- Add new HTML to a webpage
- React to user actions
- Send requests over the network
- Get and set cookies
- Remember the data on the client-side

What Can’t an Embedded Engine Do?

- Embedded engine functions are limited for user safety
- This prevents nasty website stealing user’s information
- Or harming user’s data

Can't – Access OS

- No direct access to OS
- Not allowed to read/write files
- Modern browsers allow JavaScript to work with files, but this is controlled by the user
  - Drag and drop boxes
  - Webcam/microphone requires permission from user

Can’t – Have Knowledge of Other Tabs

- Different tabs/windows should not know about one another

- Same Origin Policy – both webpages need to contain JavaScript code to facilitate and code data exchange
  - This should only exist in webpages from the same origin, hence the name

Can’t – Communicate with Other Websites

- Though JavaScript **can** technically achieve this, it’s really difficult
  - Requires express user permission
  - Requires protocol and port permission and access

Alternatives

- CoffeeScript – Shorter syntax
- TypeScript – Simplifies development and support of complex systems
- Dart – Has its own engine that runs in non-browser environments (like on mobile). However, it now must be translated into JavaScript to function on most phones (as do the others).

JavaScript in 60 Seconds

- The next few slides are intended to be a very brief introduction to how to write JavaScript
- I know some of you have already seen this in CM1100
  - But not everyone takes that class
- We will be quick!

JavaScript in 60 Seconds

- JavaScript stores information in variables
  - Unlike other languages, variable types are undeclared

```
var info = 'hello!'
```

var – states that we are declaring a variable

info – states the name of that variable

= 'hello!' – sets the value of that variable to be 'hello' (note the use of quotations as it is a String)

JavaScript in 60 Seconds

- We can manipulate the information in variables by using functions

- These can be simple mathematical or logical functions:

```
var answer = 5 + 6
```

- These are built in

JavaScript in 60 Seconds

- We can manipulate the information in variables by using functions

- Or they can be more complex

```
function performAddition (num1, num2) {
answer = num1 + num2
}
```

- These must be written by the developer.

JavaScript in 60 Seconds

- We can manipulate the information in variables by using functions
- In some situations, functions exist as part of an Object
- An object is a special type of variable, which collects together several different sub-variables and functions
- The HTML document itself can be accessed as a variable in this way

Using JavaScript to Change the HTML Document

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>

</body>
</html>
```

The Structure

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>

</body>
</html>
```

Using JavaScript to Change the HTML Document

```
<p id="demo">JavaScript can change HTML content.</p>
```

- id attribute – really important
- Using this attribute allows JavaScript to identify the specific HTML component it will perform an action on

Using JavaScript to Change the HTML Document

```
<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>
```

- So first we create a button element using the <button></button> tag
- We set the type attribute to button with type="button"
- We set an action with the onclick attribute
  - Onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"
- We set the text of the button between the button tags – 'Click Me!'

onclick action

```
onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'
```

- We have different types of actions associated with a button, so first we declare this action occurs when the button is clicked using the onclick attribute

- Then we identify which HTML to perform an action on by calling the getElementById() function
  - document is an object – in this case, the current webpage
  - getElementById() is a function to retrieve the identified HTML element
  - “demo” is the ID of the HTML element we are retrieving
  - innerHTML is the variable we are altering of this element
    - i.e. we are setting the content equal to “Hello JavaScript”

onclick action

- InnerHTML corresponds to the HTML content – i.e. the actual thing your element displays

- Attributes of HTML can be changed too – instead of .innerHTML, one simply calls the attribute to be changed

- For example:

```
onclick="document.getElementById('myImage').src='pic_bulboff.gif'"
```

Creating JavaScript Functions

- Sometimes you will want to create longer JavaScript functions
- Or reuse similar functionality often throughout the site
- In these scenarios it would be inconvenient to code (potentially repeated) functions within every button
- Instead you can specify a function like any other coding language

JavaScript Functions in HTML

```
<button type='button' onclick='replaceContent("demo", "Many times!")'>Or Me!</button>

<script>
  function replaceContent(id,content) {
    var container = document.getElementById(id);
    container.innerHTML = content;
  }
</script>
```

JavaScript Functions in HTML

```
<button type='button' onclick='replaceContent("demo", "Many times!")'>Or Me!</button>
```

```
<script>
  function replaceContent(id,content) {
    var container = document.getElementById(id);
    container.innerHTML = content;
  }
</script>
```

- Notice that in the button we are now calling only the function name and variables
  - Which is much shorter

JavaScript Functions in HTML

```
<button type='button' onclick='replaceContent("demo", "Many times!")'>Or Me!</button>
```

```
<script>
  function replaceContent(id,content) {
    var container = document.getElementById(id);
    container.innerHTML = content;
  }
</script>
```

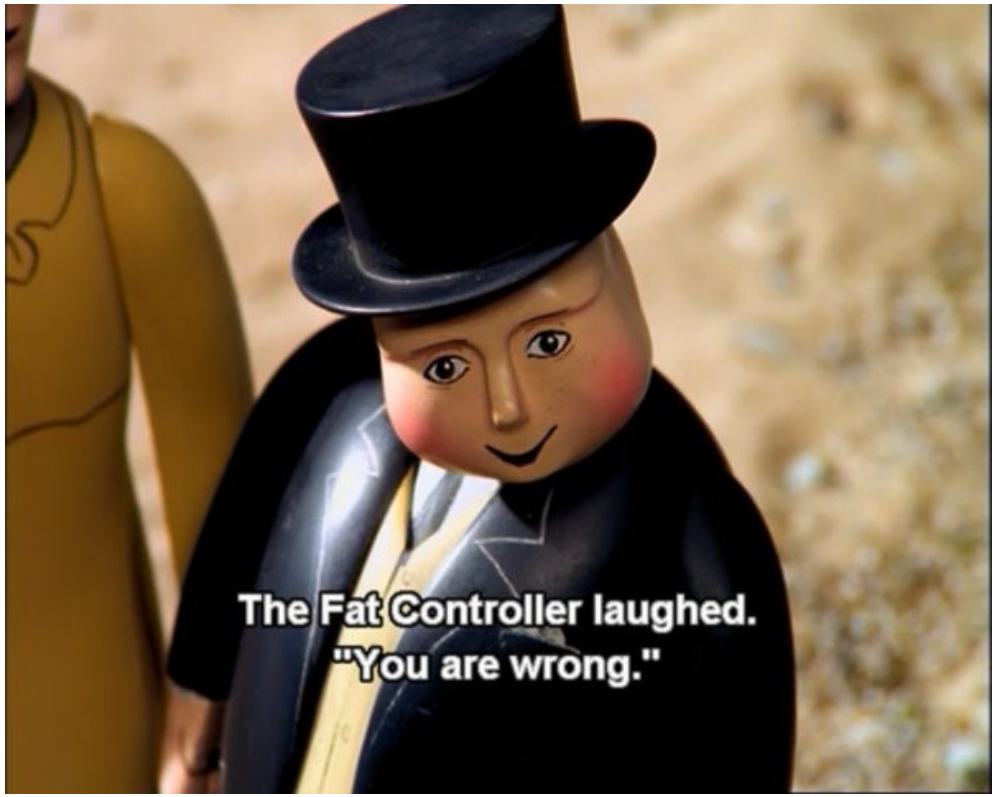
- To define the function:
  - First we open a <script> tag (within the <body> tag)
  - Then we state that we are defining a function, its name and the required variables [function replaceContent (id, content)
  - Then we create our function within curly brackets { }

JavaScript for Alerting the User

- JavaScript can also be used to send alerts which force user interaction before they continue

JavaScript Without a Button

- So JavaScript relies on a button to be pressed?



JavaScript Without a Button

- JavaScript (and many other scripts for that matter) can be instigated in quite a few different ways
- The most common one is making use of EventListeners
- EventListeners… Well they listen for a specified event…

JavaScript with EventListeners

```
<html>
<body>

<h2>JavaScript addEventListener()</h2>

<p>Don't touch my vase!</p>

<img id="myVase" src="vase.jpg"> (Just download any image for this part...)

<script>
document.getElementById("myVase").addEventListener("mouseover", brokeTheLaw);

function brokeTheLaw() {
  alert ("You going to jail!");
}
</script>

</body>
</html>
```

JavaScript with EventListeners

```
<img id="myVase" src="vase.jpg">

<script>
document.getElementById("myVase").addEventListener("mouseover", brokeTheLaw);

function brokeTheLaw() {
  alert ("You going to jail!");
}
</script>
```

- We have an image, with a specific ID
- We have a script tag containing:
  - An EventListener
  - And a function

JavaScript with EventListeners

```
<img id="myVase" src="vase.jpg">

<script>
document.getElementById("myVase").addEventListener("mouseover", brokeTheLaw);

function brokeTheLaw() {
  alert ("You going to jail!");
}
</script>
```

- Firstly, we have specified the document and element we are retrieving
  - document.getElementByID(“myVase”)
- Then we have added an EventListener
  - addEventListener
- Which activates the ‘brokeTheLaw’ function when the pointer goes over the identified element

What to Listen for?

- EventListeners look for a wide range of different stimuli to take action

- https://www.w3schools.com/jsref/dom_obj_event.asp

In Summary

- We have discussed the difference between scripting and coding
- We have introduced JavaScript and JavaScript Engines
  - Embedded Engines
  - Server Engines
- We have demonstrated how to use JavaScript to adapt HTML Content
  - With or without a button