Edinburgh Napier
UNIVERSITY

JAVASCRIPT: CORE LANGUAGE

Dr Simon Wells
s.wells@napier.ac.uk
http://www.simonwells.org

Edinburgh Napier UNIVERSITY

TL/DR

JavaScript is how we add dynamism to our web pages…

Edinburgh Napier UNIVERSITY

AIMS

- At the end of this (sub-section) of the topic you will be able to:
  - Understand the role that JavaScript plays amongst the other core web technologies
  - How JavaScript interacts with HTML, CSS, & the Browser
  - Gain an understanding of the core JavaScript syntax

Edinburgh Napier UNIVERSITY

OVERVIEW

- At the end of this (sub-section) of the topic you will be able to:
  - Understand the role that JavaScript plays amongst the other core web technologies
  - Gain an understanding of the core JavaScript syntax
  - How JavaScript interacts with HTML, CSS, & the Browser

Edinburgh Napier
UNIVERSITY

JAVASCRIPT

- The third part of our triumvirate of web technologies (alongside HTML & CSS)
- Separates functionality (or behaviour) from a web page's content & presentation
- A high-level, dynamic, weakly typed, prototype-based, multi-paradigm, interpreted programming language.
- Originally a project at Netscape which lead to the first Javascript implementation
- This was then formalised to produce the ECMAScript specification which anyone can implement to produce their own variation. NB. The name "Javascript" itself is a trademark of Oracle
- No single Javascript implementation - different engines in most web browsers - each implements ECMAScript specification to varying degrees (some feature might be missing & some engines have additional features)

Edinburgh Napier
UNIVERSITY

THE LANGUAGE

- Multi-paradigm - so can support different programming styles, e.g. event-driven, functional, imperative, OO,
- API for working with:
  - Text, arrays, dates, regular expressions, DOM manipulation
- No support for I/O,
  - i.e. networking, storage, graphics - facilities which are generally provided by the host environment (browser API, server side libraries)
- Designed as a "*glue language*" - easy for web designers & part-time programmers to assemble components of web pages

Edinburgh Napier UNIVERSITY

ASIDE: JAVA(SCRIPT)?

- Javascript & Java are two wholly separate languages
- For historical reasons there are some superficial similarities between the two:
  - Some syntax, standard libraries, & the name
- Brendan Eich was originally employed by Netscape to write a version of Scheme to run in the browser. Management made the decision that Javascript should "look like Java" (which was the *hot new thing* in the mid/late-90s)
- That all said, syntactically speaking, Java isn't hugely different from its predecessors - *why might this be?*

Edinburgh Napier UNIVERSITY

JS ENGINES

- Interpreted & VM based so requires an environment to live in: **JS Engine**
- Many engines, but some are more important than others
- V8 - the google chrome engine (also used in Node.js)
- Spidermonkey - Firefox engine
- JavaScriptCore - marketed as Nitro & used in Safari
- You can work with JS in the browser fairly easily.
- Can also install a standalone tool to run JS outside the browser (e.g. Node.js) - this enables Javascript to run server side

Edinburgh Napier UNIVERSITY

WHAT'S JAVASCRIPT FOR?

- (slightly weird) General purpose programming language
- Web programming (it's raison d'être)
  - Primarily: Interacting with the DOM (for a given web page) and the user agent (via various Browser APIs)
  - Secondarily: Providing a server-side environment so that our core tools can be (reasonably) cohesive - (Advanced Web Tech considers the server side but for now *we are staying client side*)

Edinburgh Napier
UNIVERSITY

WHAT DOES JS LOOK LIKE?

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <button id="hellobutton">Hello</button>
    <script>
        document.getElementById('hellobutton').onclick = function() {
            alert('Hello world!');
            var myTextNode = document.createTextNode('Some new words.');
            document.body.appendChild(myTextNode);
        };
    </script>
  </body>
</html>
```

Edinburgh Napier UNIVERSITY

USING JAVASCRIPT

- Similar to CSS, we have choices for how to integrate HTML with Javascript
- **Inline**, e.g.
  - <button id="hellobutton" onclick="alert('Hello World')">Click Me</button>
- Within a **script block** <script></script>, e.g.

```
<script>
    document.getElementById('hellobutton').onclick = function() {
        alert('Hello world');
    };
</script>
```

- Using an **external script** (placed within either <head> or <body>)
  - HTML 4:

```
<script type="text/javascript" src=‘‘javascript.js"></script>
```

  - HTML 5:

```
<script src="javascript.js"></script>
```

  - NB. Can use any number of scripts

Edinburgh Napier
UNIVERSITY

BROWSER CONSOLE JS

- Useful for learning JS & playing around without too much HTML & CSS (we'll quickly grow beyond this though)

- Gives us a little programming environment almost anywhere there is a (reasonably modern & not-locked-down) web browser.

- Let's look at some simple examples of how we can write JS directly in the browser console:

#1

HELLO NAPIER

New Tab

Search with DuckDuckGo or enter address

Inspector | Console | Debugger | Style Editor | Performance | Memory | Network

Filter output

Persist Logs

```
>> console.log('hello napier!!!')
   hello napier!!!                                   debugger eval code:1:1
<- undefined
```

#2

INTERACT WITH THE WEB PAGE/SCREEN

New Tab

Search with DuckDuckGo or enter address

Inspector | Console | Debugger | Style Editor | Performance | Memory | Network

Filter output

Persist Logs

```
document.body.style.backgroundColor = "lightgreen";
"lightgreen"
```

#3

USE STANDARD JAVASCRIPT FUNCTIONS

New Tab

Search with DuckDuckGo or enter address

Today's date is Thu Sep 13 2018 16:10:26 GMT+0100 (BST)

Inspector | Console | Debugger | Style Editor | Performance | Memory | Network

Filter output

Persist Logs

```
let d = new Date();
undefined
document.body.innerHTML = "<h1>Today's date is " + d + "</h1>"
"<h1>Today's date is Thu Sep 13 2018 16:10:26 GMT+0100 (BST)</h1>"
```

#4

CONSTRUCT A WEB PAGE

New Tab

Search with DuckDuckGo or enter address

Hello Napier!!!

Inspector | Console | Debugger | Style Editor | Performance | Memory | Network

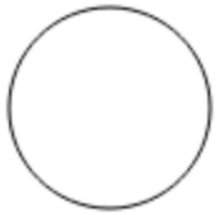
Filter output | Persist Logs

```
let p = document.createElement("P");
let t = document.createTextNode("Hello Napier!!!");
p.appendChild(t);
document.body.replaceWith(p);
```

undefined

#5

GRAPHICS



```
let c = document.createElement("canvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
document.body.replaceWith(c);
```

```
undefined
```

#6

SOUND - BEEPS

New Tab

Search with DuckDuckGo or enter address

Inspector | Console | Debugger | Style Editor | Performance | Memory | Network

Filter output

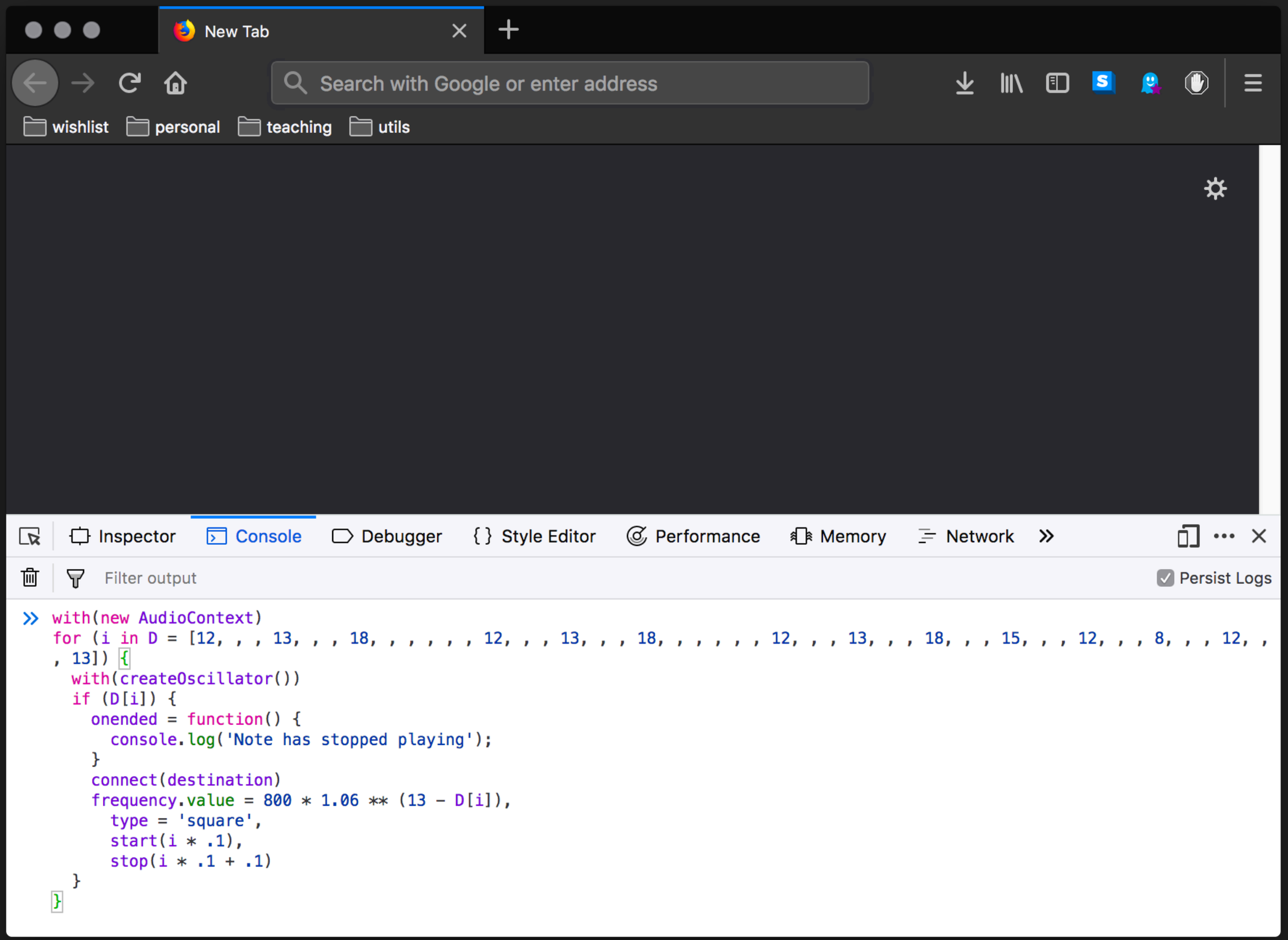
Persist Logs

```
var context = new (window.AudioContext || window.webkitAudioContext)();
var oscillator = context.createOscillator();
oscillator.type = 'sine';
oscillator.frequency.value = 440;
oscillator.connect(context.destination);
oscillator.start();
```

undefined

#7

SOUND - BIT TUNES


New Tab
Search with Google or enter address
wishlist
personal
teaching
utils
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Filter output
Persist Logs
with(new AudioContext)
for (i in D = [12, , , 13, , , 18, , , , , , 12, , , 13, , , 18, , , , , , 12, , , 13, , , 18, , , 15, , , 12, , , 8, , , 12, ,
, 13]) {
with(createOscillator())
if (D[i]) {
onended = function() {
console.log('Note has stopped playing');
}
connect(destination)
frequency.value = 800 * 1.06 ** (13 - D[i]),
type = 'square',
start(i * .1),
stop(i * .1 + .1)
}
}

#8

SOUND - THEREMIN

New Tab

Search with DuckDuckGo or enter address

Inspector | Console | Debugger | Style Editor | Performance | Memory | Network

Filter output

Persist Logs

```
var context = new (window.AudioContext || window.webkitAudioContext)();
var oscillator = context.createOscillator();
h = window.innerHeight;
oscillator.connect(context.destination);
oscillator.start(0);
document.addEventListener("mousemove", function(e) {
    oscillator.frequency.value = e.clientY / h * 1000 + 300;
});
```

undefined

Edinburgh Napier UNIVERSITY

JS AS A LANGUAGE

- We are going to:
    - Look at JavaScript as a standalone language
- This means:
    - To consider it's keywords, commenting, variables, primitive datatypes, native objects, & operators
    - Examine control structures (if-else, conditional, switch) & looping
    - Functions, Objects, & Exception handling
    - JavaScript object serialisation using JSON
    - We also want some idea of some best practice when applying JS to real problems

Edinburgh Napier
UNIVERSITY

LEARNING JS

- Having tried out some JS we can, with the benefit of hindsight and some experience, now let's look at the language in a little more detail.

- **IMPORTANT: If you want to become a good programmer in *any* language then you need to practise. Reading code or documentation is much less useful than writing code & solving problems.**

Edinburgh Napier UNIVERSITY

JS LANGUAGE FEATURES

- Case sensitive - Like all programming we have to be precise. Develop a habit for using case then stick to it
    - NB. I nearly always use lowercase (& snake_case rather than camelCase)
- Whitespace - spaces, tabs, newlines outside of strings are whitespaces. Generally not a problem. Choose a way to use whitespace then be consistent in laying out your code
- Semicolons are automatically inserted if you leave them out
    - This can interfere with the whitespace and cause valid code to be incorrectly parsed
    - So good practise to end statements with a semicolon

Edinburgh Napier UNIVERSITY

KEYWORDS

- Avoid the following when naming variables, function names, or labels

- **JavaScript keywords:** abstract, arguments, await*, boolean, break, byte, case, catch, char, class*, const, continue, debugger, default, delete, do, double, else, enum*, eval, export*, extends*, false, final, finally, float, for, function, goto, if, implements, import*, in, instanceof, int, interface, let*, long, native, new, null, package, private, protected, public, return, short, static, super*, switch, synchronized, this, throw, throws, transient, true, try, typeof, var, void, volatile, while, with, yield

- **Built-in Objects, Properties, & Methods:** Array, Date, eval, function, hasOwnProperty, Infinity, isFinite, isNaN, isPrototypeOf, length, Math, NaN, name, Number, Object, prototype, String, toString, undefined, valueOf

- **HTML & Window objects & properties:** alert, all, anchor, anchors, area, assign, blur, button, checkbox, clearInterval, clearTimeout, clientInformation, close, closed, confirm, constructor, crypto, decodeURI, decodeURIComponent, defaultStatus, document, element, elements, embed, embeds, encodeURI, encodeURIComponent, escape, event, fileUpload, focus, form, forms, frame, innerHeight, innerWidth, layer, layers, link, location, mimeTypes, navigate, navigator, frames, frameRate, hidden, history, image, images, offscreenBuffering, open, opener, option, outerHeight, outerWidth, packages, pageXOffset, pageYOffset, parent, parseFloat, parseInt, password, pkcs11, plugin, prompt, propertyIsEnum, radio, reset, screenX, screenY, scroll, secure, select, self, setInterval, setTimeout, status, submit, taint, text, textarea, top, unescape, untaint, window

- **HTML Event Handlers:** onblur, onclick, onerror, onfocus, onkeydown, onkeypress, onkeyup, onmouseover, onload, onmouseup, onmousedown, onsubmit

Edinburgh Napier
UNIVERSITY

COMMENTS

```
// A short, one line comment

/* a longer, multi-line
comment about something

that must be important */


/* Comments /* cannot be nested */ as this is a syntax error */
```

Edinburgh Napier UNIVERSITY

VARIABLES

- Variables can be declared in three ways:
  - Block level using **let**
  - function level using **var** or **const**
- Don't have a type attached - any value can be stored in any variable
- Variable declared anywhere inside a function will resolve to that variable when it's named is used (within the function)
- Variable *value* is undefined until initialised, e.g. var a = 203
- Variables declared outside a function are *global*
- If a variable isn't found then a ReferenceError exception will happen

Edinburgh Napier UNIVERSITY

PRIMITIVE DATATYPES

- Undefined, Null, Number, String, Boolean, Symbol
- Undefined - assigned to all uninitialised variables & returned where checking for object properties that don't exist
- Null - When something has been declared but the value is empty
- Numbers - IEEE754 doubles (floating point, use toFixed() to round) - accuracy to 16 significant digits
- Strings - sequence of characters enclosed in double quotes. Access individual letters using charAt(). Compare using ==
- Boolean - true & false. Useful for comparisons
- Symbol - New (ECMAScript 6). A unique & immutable identifier.

Edinburgh Napier UNIVERSITY

NATIVE OBJECTS

- Arrays, Dates, Errors, Math, Regular Expressions, Functions
- Arrays - Objects representing lists of values indexed by keys. Keys are numeric & use zero based indexing. Can be multi-dimensional.
- Date - Object that stores a signed millisecond count from zero (representing 1970-01-01 00:00:00 UT). Various methods to access fields of data object
- Error - Object that can be used to create custom error messages
- Math - Object storing math-related constants, e.g. E, Natural Log, Pi, and functions, e.g. max, min, random
- Regular Expressions - Object used to store text patterns
- Function - Object constructed using the Function constructor & used to collect statements into reusable groups

Edinburgh Napier UNIVERSITY

OPERATORS

- Arithmetic: +, -, *, /, %
- Unary: + (string to number), - (reverse sign), ++, —
- Assignment: =, +=, -+, *=, /=, %=
- Destructuring assignment: [a, b, c] = [3, 4, 5];
- Logical: !, ||, &&
- String: =, +, +=
- NB. Bitwise (operators & assignments), Ternary conditional

Edinburgh Napier UNIVERSITY

CONTROL STRUCTURES

If...else

```
if (expr) {
  //statements;
} else if (expr2) {
  //statements;
} else {
  //statements;
}
```

Switch Statement

```
rswitch (expr) {
 case SOMEVALUE:
  // statements;
  break;
 case ANOTHERVALUE:
  // statements;
  break;
 default:
  // statements;
  break;
}
```

Conditional Operator

```
result = condition ? expression :
alternative;
```

LOOPING

For Loop

```
for (initial; condition; loop statement) {
  /*
   statements will be executed every time
   the for{} loop cycles, while the
   condition is satisfied
  */
 }
```

For In Loop

```
for (var property_name in some_object) {
  // statements using some_object[property_name];
}
```

While Loop

```
while (condition) {
  statement1;
  statement2;
  statement3;
  ...
}
```

Do… While Loop

```
do {
  statement1;
  statement2;
  statement3;
  ...
} while (condition);
```

Edinburgh Napier UNIVERSITY

FUNCTIONS

Various ways to declare functions:

```
function add(x,y) { return x + y; };

var add = function(x,y) { return x + y; };

var add = new Function('x','y','return x + y');
```

Edinburgh Napier UNIVERSITY

OBJECTS

- JS supports object orientation using *prototype-based inheritance*

- Can create objects & assign data and method to them in various ways, e.g.

```
var cow = new Object();
cow.colour = 'brown';
cow.commonQuestion = 'What now?';
cow.moo = function(){
  console.log('moo');
}
cow.feet = 4;
cow.accordingToLarson = 'will take over the world';
```

```
var cow = {
  colour:'brown',
  commonQuestion:'What now?',
  moo:function(){
    console.log('moo);
  },
  feet:4,
  accordingToLarson:'will take over the world'
};
```

EXCEPTION HANDLING

- Handle run time errors by:
  - Trying to execute the statements
  - Catching any thrown exceptions
  - & Finally cleaning things up.

```
try {
  // Statements in which exceptions might be thrown
} catch(errorValue) {
  // Statements that execute in the event of an exception
} finally {
  // Statements that execute afterward either way
}
```

Edinburgh Napier UNIVERSITY

JAVASCRIPT BEST PRACTICES

- W3C: https://www.w3.org/wiki/JavaScript_best_practices
- Some are debatable but thinking about them **will** make you into a better web developer, e.g.
  - Call things by their name — easy, short and readable variable and function names
  - Avoid globals
  - Stick to a strict coding style
  - Comment as much as needed but not more
  - Avoid mixing with other technologies
  - Use shortcut notation when it makes sense
  - Modularise — one function per task
  - Enhance progressively
  - Allow for configuration and translation
  - Avoid heavy nesting
  - Optimize loops
  - Keep DOM access to a minimum
  - Don't yield to browser whims
  - Don't trust any data
  - Add functionality with JavaScript, don't create too much content
  - Build on the shoulders of giants
  - Development code is not live code

Edinburgh Napier UNIVERSITY

SUMMARY

- We've considered:
  - The role that JavaScript plays amongst the other core web technologies
  - Gained an understanding of the core JavaScript syntax and considered JavaScript as a standalone language

Edinburgh Napier UNIVERSITY

SUMMARY

- In this topic we have :
  - Looked at the role that JavaScript plays amongst other web technologies
  - Gained an understanding of the core JavaScript syntax
  - Discovered how JavaScript interacts with HTML, CSS, & the Browser

Edinburgh Napier
UNIVERSITY

QUESTIONS ???

Edinburgh Napier
UNIVERSITY

COMING UP…

- We'll take a look at Designing for the Web.

- Specifically the tools & techniques (& other miscellaneous stuff) the help us to put HTML, CSS, & JS to good use in producing robust, reliable, & well engineered web interfaces.

Edinburgh Napier UNIVERSITY

RESOURCES

- MDN JS Reference:

  https://developer.mozilla.org/bm/docs/Web/JavaScript

- W3Schools JS & DOM reference:

  https://www.w3schools.com/jsref/