GENERAL ASSEMBLY

FRONT END WEB DEVELOPMENT

CLASS 07: JS BASICS

James Willock
Software Engineer, General Assembly

OBJECTIVES

- JavaScript in a nutshell
- Learn the basics of JavaScript
- The importance of syntax
- Variables
- Arithmetic
- Conditionals

WHAT IS JAVASCRIPT?

- An object-orientated scripting language, designed to complement HTML and CSS
- Released by Brendan Eich in 1995 as part of Netscape Navigator
- It’s not Java, no sir-ee!
- It’s the behaviour in the separation of concerns diagram

SEPARATION OF CONCERNS

HTML
Content

CSS
Presentation

JS
Behaviour

COMMON USES

- Modifying the DOM in real time – adding or removing elements, modifying CSS, etc
- Making requests to external services after page render (Facebook chat, etc)
- Provides the interface to Browser APIs (LocalStorage, Geolocation, etc)
- Hipsters are now using it on the server, too! (NodeJS)

JAVASCRIPT SYNTAX

- Syntax matters! HTML will let you off when you’re being sloppy, JavaScript won’t.

Semicolon	`2 + 1;`
Brackets	`fruits[2];`
Parenthesis	`playVideo();`
Quotes	`"My name is James";`

CODEALONG:

COLOUR SWITCHER

DATA TYPES

- Our programs will often need to store different types of data
- In the same way we mark up a headline with a different tag to a paragraph in HTML, JavaScript exposes different types for different kinds of data

TYPES

Name	Implementation	Description
Number	`48, 4.5607`	An integer or floating point number
Boolean	`true, false`	A type with one of two values – true or false – representing the truth values of boolean algebra
String	`“James”, “Bob”`	A collection of characters that could make up a name or paragraph of text
null	`null`	An explicitly set null value – we use this when we want to set that something is explicitly empty
undefined	`undefined`	The value returned when something has not been set

CODEALONG:
PLAYGROUND

BUT… WHY?

- It’s important that our script knows what data types are values are, because we can run different functions – or “methods” – on different data types
  - Consider `32 - 10` and `32 - “10"`
  - Consider `parseInt("James")`

VARIABLES

- We often need to store values for usage later on in our program
- We can store any value we need in a variable, access it later and even modify it if required

SYNTAX

Name	Implementation	Description
Declaration	`var age;`	Creates a space in memory to save a value
Assignment	`age = 25;`	Puts a value inside the variable
Access	`age;`	Returns the value stored inside the variable
Declaration and assignment	`var age = 25;`	Declares a variable and assigns its value at the same time

CONVENTION

- “Camel case” is the preferred convention for naming variables in Javascript, so use `numberOfBeers` over `number_of_beers`

JS REVIEW

STRING REVIEW

- Strings store text data
- Can be implemented using either double quotes or single quotes
  - `'They "purchased" it'`
  - `"It's a beautiful day"`
- Can be escaped using the `\` character
  - `"It was a \"beautiful day\" indeed"`

NUMBERS REVIEW

- Numbers can be integer or floating point values
  - `562`
  - `0.466`
- Can be signed if required
  - `+4`
  - `-0.245`
- Arithmetic can be performed on number values
  - `42 + 32 - 12`

BOOLEANS REVIEW

- `true` or `false`. That’s it.

ARITHMETIC

Name	Implementation	Description
Addition	`2 + 3`	
Subtraction	`5 - 3`	
Multiplication	`4 * 5`	
Division	`6 / 2`	
Remainder	`12 % 5`	Returns the integer remainder of the division
Increment	`2++`	Adds one to the value
Decrement	`3--`	Subtracts one from the value

COMPARISON OPERATORS

Name	Implementation	Description
Equal	`2 == "2"`	Returns `true` if values are the same, and will attempt to coerce both values to the same data type
Not equal	`2 != "2"`	Returns `true` if values are not the same, and will attempt to coerce both values to the same data type
Strict equal	`2 === 2`	Returns `true` if values are the same, but does not negotiate data type
Strict not equal	`2 !== 2`	Returns `true` if values are not the same, but does not negotiate data type

COMPARISON OPERATORS PT. 2

Name	Implementtion	Description
Greater than	`2 > 3`	Returns `true` if first value is more than second value
Less than	`2 < 3`	Returns `true` if first value is less than second value
Greaten than or equal to	`2 >= 3`	Returns `true` if first value is more than or equal to the second value
Less than or equal to	`2 <= 3`	Returns `true` if first value is less than or equal to the second value

CONDITIONALS

- We need to control the flow of our programs – conditionals can help us to do this
- The simplest conditional is the modest `if` statement

  -
    ```
    if (passwordValid) {
       document.write("Come in!");
    }
    ```
- The contents of the brackets is an expression that evaluates to `true` or `false`

ELSE

- An if statement can have an accompanying `else` statement, with a block of code that will run should the expression evaluate to false

  -
    ```
    if (age >= 18) {
       document.write("Over 18");
    } else {
       document.write("Under 18");
    }
    ```
- You can create a decision tree by chaining `if..else if..else if..else if..else`

CHAINING EXPRESSIONS

- Sometimes you need to check more than one value in your expression is true
- You can use logical operators to evaluate the truthiness of multiple values
- Using the `and` + `or` logical operators
  - `if (2 > 1 && 3 < 2) // false`
  - `if (2 > 1 || 3 < 3) // true`

CODEALONG:

NUMBER COMPARISON

LAB:

CELSIUS CONVERTER