JavaScript Notes

JavaScript must be written in order - read from top to bottom

Loops - repeating tasks
Functions - small amount of program/code that is called when needed via page interaction
Variable - storing data

Debug

- Console.log, prompts and alerts should be removed before going live, only used as a developer tool
- Use console.log alot! After every calculation
- NaN = not a number

Inserting into HTML

- `onclick="functionName();"`
- `.innerHTML` use inside the `<div>` tags
- In HTML when writing out a string add `"+"` at the front and end. i.e: `'<h5 class="card-title">'+movie.title+'</h5>';`
- `document.write` cannot be used to add new data onto page once page has loaded
- `inner.text` only renders text so no HTML tags will work
- `appendChild` - append means add to the end

Strings

- Quotes only for text (a string)
- A string is a block of text
- String will be black, number will be blue
- Add in space manually to strings

Variables

- No spacing should be used for variable names

- Numbers can't be at the start of a variable name
- Follow a naming convention - camelcase etc.
- Can't use dashes, can use underscores
- `var` only needs to be declared once
- You can declare var on one line - seperate each var with a comma;

Operators

- = will always declare
- = after will override
- += adds to what is already there
- ++ adds one to current number
- -- subtracts from current number
- == comparing 2 values together
- === comparing 2 values together and making sure they are the same type
- != does not equal
- !== does not equal or does not equal the same type

Notes:

- Concatenating - including the `"="` or `"+"` between 2 different data types. Joining things together.
- `"="` or `"+"` to the exact same data type
- For multiplication use `*` instead of `x`
- `+` and `-` are calculated after `*(x)` and `/`. Example of using `*(x)` or `/` first; `total = (2+4) * 10`

Booleans

- Boolean only has 2 values; `true` or `false`. Does not need quotes

Array

- Array is a list of other data types / stored values. `[ ]` means an array.
- Seperate arrays with a `","`
- The first entry is 0

- `splice` - doesn't remove but takes a copy of the array

Loop

- When nesting within a `for` loop, make sure to change the var `'[i]'` is given a new name like `'[a]'` - they id needs to be different from one another
- `break;` breaking the loop early. Only used in a loop or switch
- `continue;` does not `break` but stops here and starts the next iteration
- for (initial value; final value; increment{ }
- `for (i = 0; i< fruits.length, i++)`
- `i++`
- `i= i+1`
- `i=0 +1=1`
- In a loop, do not use greater than `(>)` unless you have a `break` or `else`

Function

- A function is a block of reusable code
- If a `var` is declared in a function it is only locally registered
- **Write/declare all global variables at the top of the page, even if they don't have a value yet.** Do this if the `var` is reusable. If not, keep `var` within the function (local). If you move local to global - remove `var` from the function.
- `id`s must have a unique identifier even if writing the same functions to see which one is more effective.

Notes:

- Javascript is very hackable. If it is global - it can be editable

Return

- `return` finishes/ends/leaves the function then and there. `return` always means `true`. `return` is used in place of a `if/else`. It doesn't need a value.

If and else

- Can write more than 2 `if/else` statements in a function - must `return` after each `if/else` or could make loading time longer. Just avoid infinite loops.

Objects

- Objects is a collection of data types that hold other data types. Objects have `{}`. Arrays have `[]`
- Books - remember to include ISBN in a book object

Dataset

`data - x` you can name it what you want in place of `x`
`data-id="'+movie.id+'"`
`dataset` gets data from value
dataset is a object