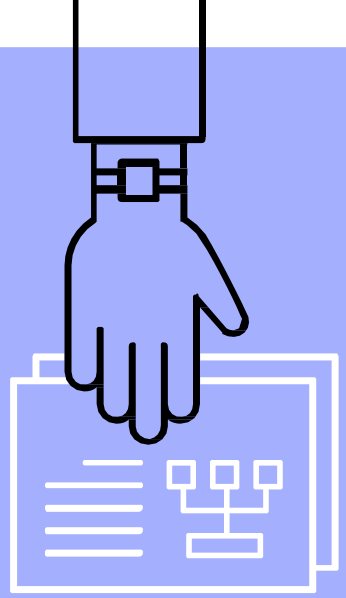
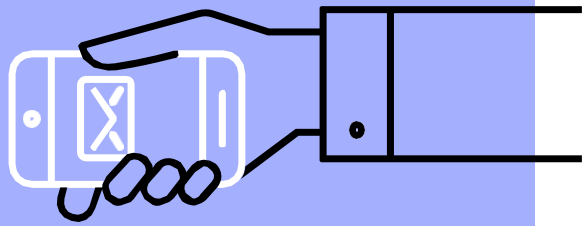
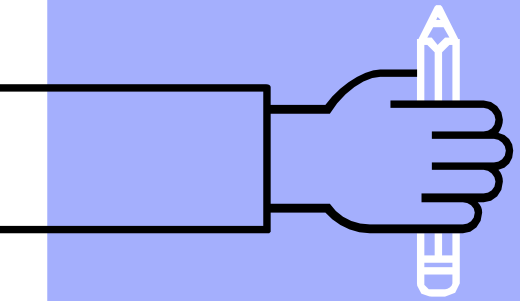
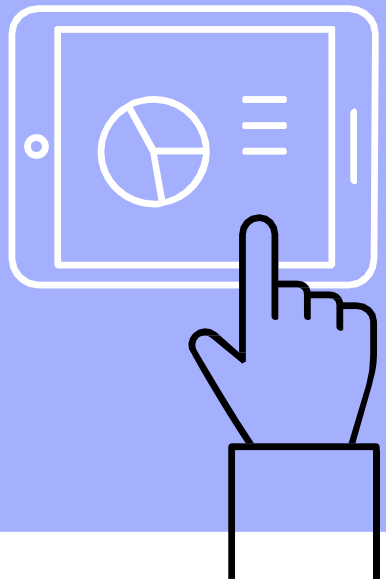


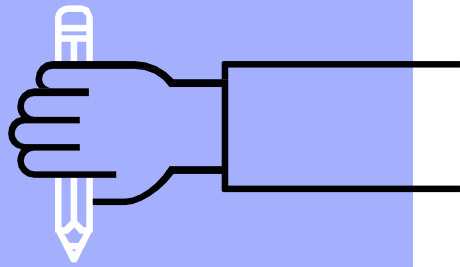


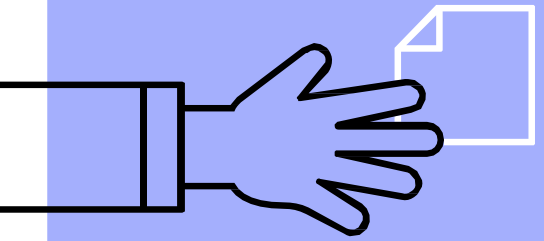
CMPT 281: INTRODUCTION TO JAVASCRIPT





WHAT IS JAVASCRIPT?





JavaScript

It **is** a programming language.

- Allows us to add **functionality** to websites
- On the frontend, it is used for many **interactive features**
- Not related to Java!
- Web browsers execute scripts on the **client side**

JavaScript Use Cases

- Client side functionality (e.g., buttons)
- Internet of Things
- Used with node.js for server side development
- Developing browser extensions

WHERE DOES JAVASCRIPT RUN?

JavaScript typically runs on the clientside, within a user's web browser.

- Every browser has a JavaScript Engine that executes JavaScript code
- These engines have different names (e.g. Chrome's is called V8, FireFox's is called SpiderMonkey).

In this course you don't need to understand the JavaScript engines, but useful to understand what is calling the scripts

MODERN JAVASCRIPT

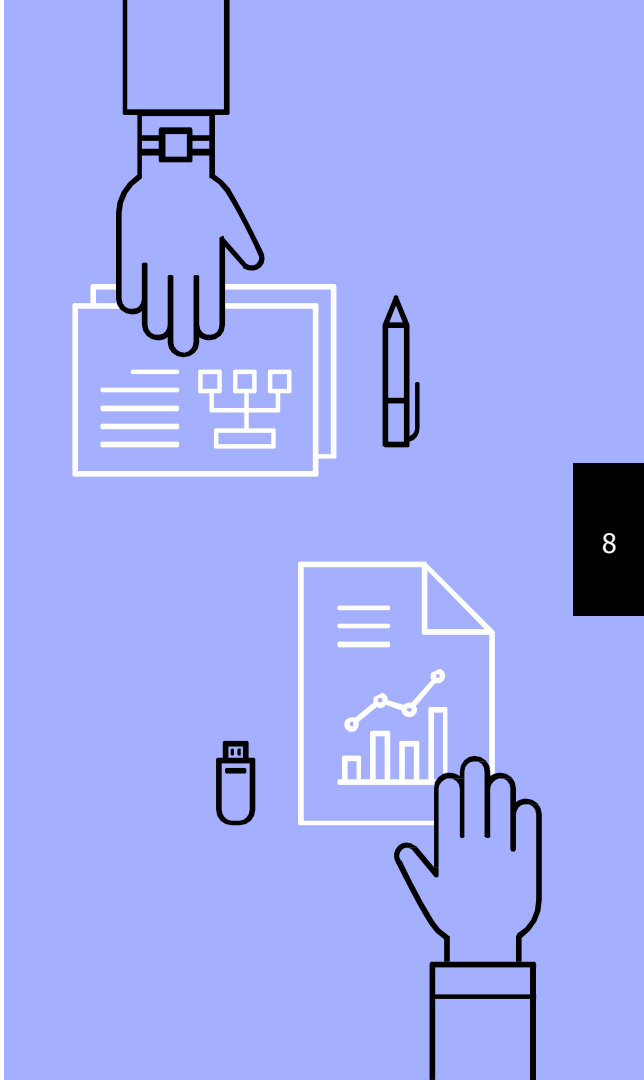
- JavaScript was originally intended as a frontend language for client side execution.
- In 2009, JavaScript was embedded in a C++ program called Node
- Node uses the V8 JavaScript Engine.
- JavaScript code can now be passed to Node for execution

Effectively, JavaScript developers can now be full-stack developers (i.e. front end [client side develop] and back end [server side stuff])

OUR CONTEXT

In CMPT 281, we are focused on website design.

- Only doing front end development with JavaScript
- All of our JavaScript will be run client side



JS DEVELOPMENT ENVIRONMENT

Need to use a code editor (e.g., VSCode, Sublime Text, Atom, Notepad++)

- Whatever works for CSS and HTML likely works for JavaScript

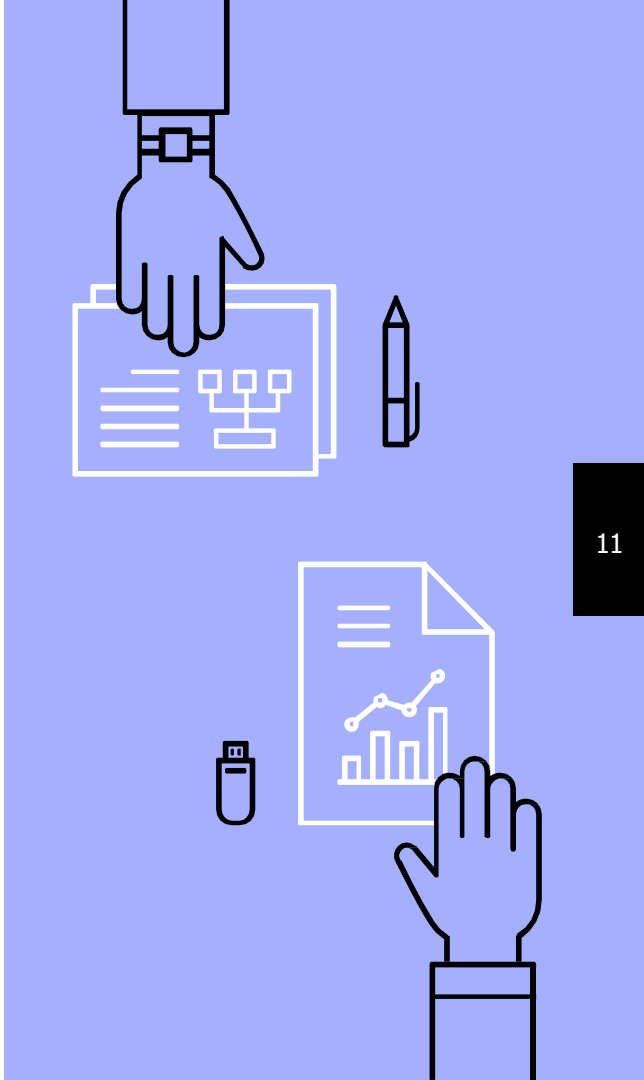
JavaScript files are saved with the .js extension (e.g., helloWorld.js).

USING JAVASCRIPT

REFERENCING JAVASCRIPT

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

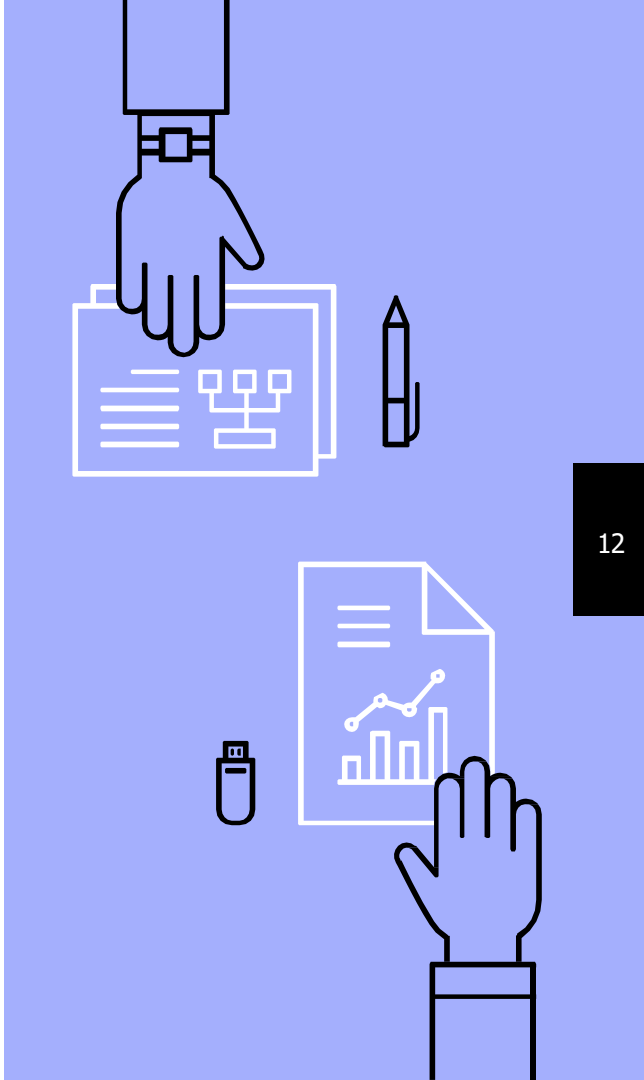
- JavaScript can be used using <script> tags
- Uses ‘src’, similar to other sourced files
- Follows the same directory structure as images, CSS files, etc.



INLINE JAVASCRIPT

- Much like CSS, you can write inline JavaScript.
- For the same reasons as CSS, inline JavaScript is not considered good practice.

```
<script> alert(“Hello World!”);
</script>
```



VARIABLES AND DATA TYPES

Variable names in JavaScript are **dynamically typed –** i.e., we **do not need to** specify data type identifiers.

We use the var keyword or the let keyword when defining variables.

- var is used to define global variables
- let is used to define local variables

VARIABLES AND DATA TYPES

Good programming practice is to avoid global variables and use locally scoped variables **wherever possible.**

So – most of the time, we want to use let.

EXAMPLE DECLARATION

```
var x = 10;
var firstName = "Madison";
```

- Both variables at global scope
- Value of ‘x’ variable set to numeric value 10
- Value of ‘firstName’ variable set to string value “Madison”

SCOPE EXAMPLE

```
let  scopeExample = 1;

{
  let scopeExample = 2;
}

// What is the final value of scopeExample?
```

- Inside of {}the value of scopeExample is 2
- Outside of {}the value of scopeExample is 1

DATA ASSIGNMENT EXAMPLES

```
let  x = 10;
x = 11;
```

Variable assignment statements are simple!

Just variable =value

DATA ASSIGNMENT FOR CONSTANTS

Like other programming languages, it's good practice to avoid using numeric and string literals in your code.

Instead, we should create **constant** variables with the const keyword.

Scope can be either global or local.

```
const pi = 3.14;
const pie = "pie";
const doIWantPie = true;
```

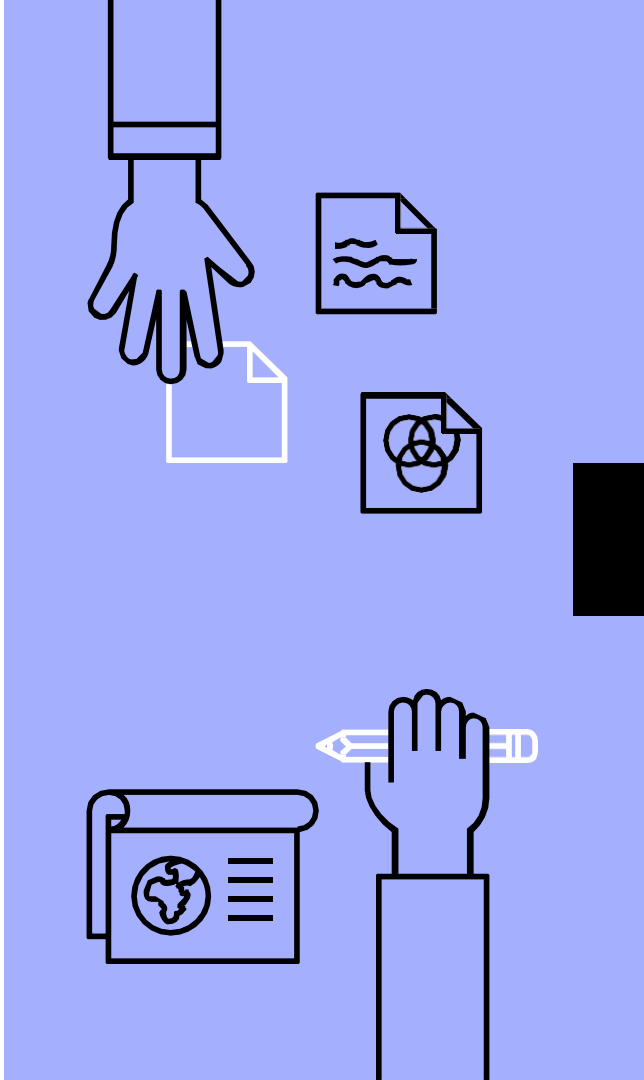
Using literals is hard to maintain - but changing values of constant variables is easy!

JAVASCRIPT SYNTAX

Like CSS, JavaScript uses semi-colons to end statements.

For example:

```
const helloText = “Hello World!”;
document.getElementById(“demo”).innerHTML = helloText;
```



WORKING WITH STRING DATA

When working with string data, you often want to determine the length of the string (e.g., password use case).

In JavaScript, that can be done with the .length property.

```
let text = "Hello World!";
let numberOfCharacters = text.length;
```

STRING METHODS

Method	Description
toLowerCase()	Makes a string lower case
toUpperCase()	Makes a string upper case
concat()	Joins two or more strings
trim()	Removes whitespace from either side of a string
trimStart()	Removes whitespace from the start of a string
trimEnd()	Removes whitespace from the end of a string
charAt(index)	Gets the character at a specified index
Number("string")	Converts a string value to a number

NUMBER METHODS

Function	Description
toString()	Converts a numeric value into a string
toFixed(num)	Returns a value with a fixed number of decimal places
parseInt()	Coverts an argument to an integer
parseFloat()	Converts an argument to a floating point

COMMENTING IN JAVASCRIPT

Use double slashes to add inline comments – for example:

```
const shortPi = 3.14; //The value of pi with two decimal
places
const pi = 3.1415926535; //Pi to the tenth decimal place
```

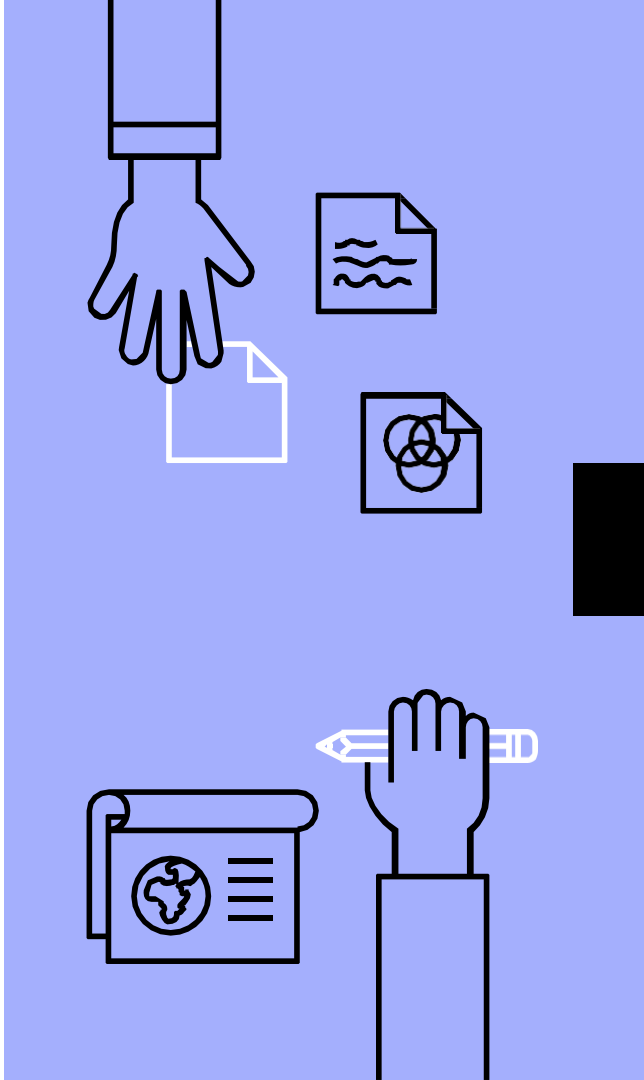
Comments will appear as a different colour in your script editor/IDE:

```
//Comment example
```

JAVASCRIPT CONDITIONALS

Very similar to Python and other languages.

- If
- Else If
- Else



JAVASCRIPT CONDITIONAL EXAMPLE

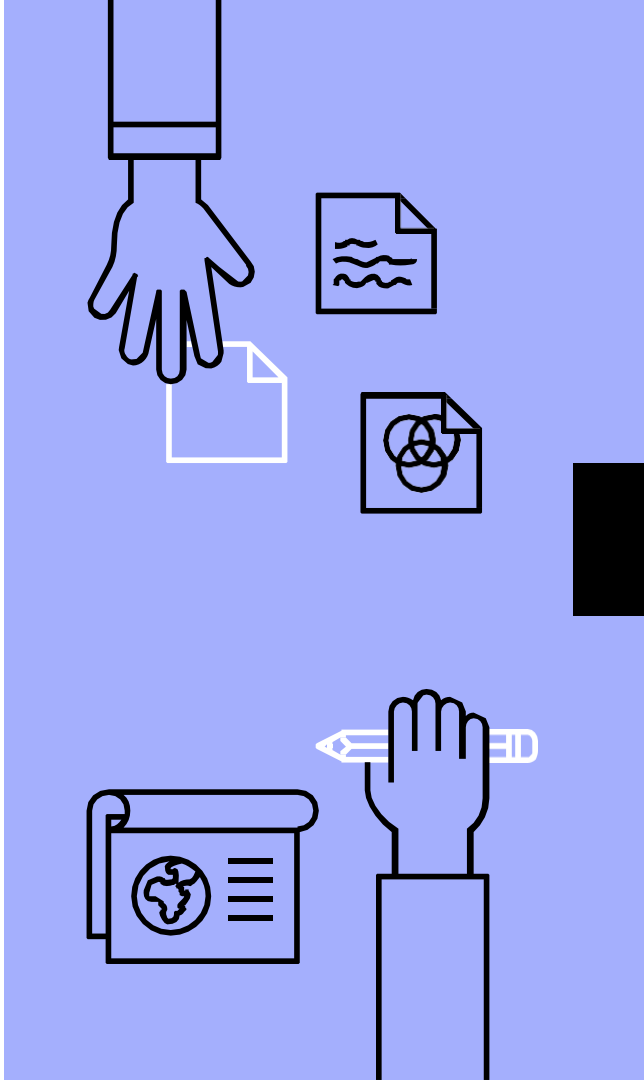
```
if (hourOfDay > 4 && hourOfDay < 12) {
    greeting = "Good Morning";
}

else if (hourOfDay >= 12 && hourOfDay < 18) {
    greeting = "Good Afternoon";
}

else {
    greeting = "Good evening";
}
```

CONDITIONAL ASSIGNMENT

```
if (y==4)
```

The code above checks if the value of y is equal to 4.

- If the value of y is 4, it evaluates as true.
- If the value of y is not 4, it evaluates as false.



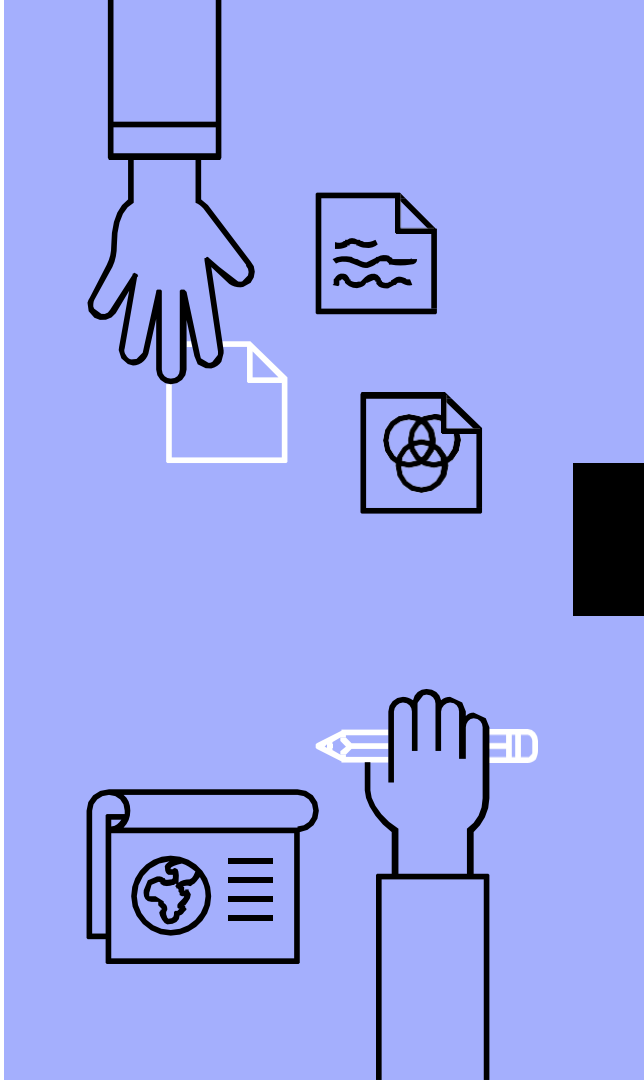
TRUTHY AND FALSY

Truthy and Falsy sound made up, but it's actually what it's called.

(I promise.)

- Almost all values in JavaScript are truthy.
- Empty string (e.g., ""), and the keywords **false** and **NaN,** are falsy.
- Falsys return as false when working with conditionals.

You need to be mindful of this when working with empty string ("" is a falsy!).



FUNCTIONS

JAVASCRIPT FUNCTIONS

Like other programming languages, JavaScript allows you to define your own functions.

Syntax is similar to most other languages.

Doesn't require a 'def' to define the function, as in Python – 'function' is the keyword to define a function.

```
function myFunction(p1, p2) {
  return p1 * p2;
}
```

BUTTONS

The HTML tag <button> can be used to create buttons that call JavaScript functions.

EXAMPLE (<script> within an HTML file)

```
<button onclick="myFunction()">Try it</button>
<p id="demo">Hello World!</p>
<script>
function myFunction() {
  let text = document.getElementById("demo").innerHTML;

    if(text=="Hello World!") {
        document.getElementById("demo").innerHTML =
        text.toUpperCase();
    }
    else {
        document.getElementById("demo").innerHTML =
        text.toLowerCase();
    }
}
</script>
```

LOOPS

LOOPS

The while keyword can be used to initialise a loop based on a condition.

```
let counter = 0;

while (counter < 10) {
    counter++;
}
```

LOOPS WITH THE DO KEYWORD

do is generally frowned upon in most contexts, but exists.

You may encounter this if working with JS made by other developers – so useful to know.

Generally considered more difficult to read.

```
do {
    counter ++;
} while (counter < 10)
```

FOR LOOPS

for loops are similar to other languages you have likely been exposed to, but the syntax may be different. Uses a very similar syntax to C#.

```
for (var i = 0; i < 10; i++) {
    //Do something
}
```

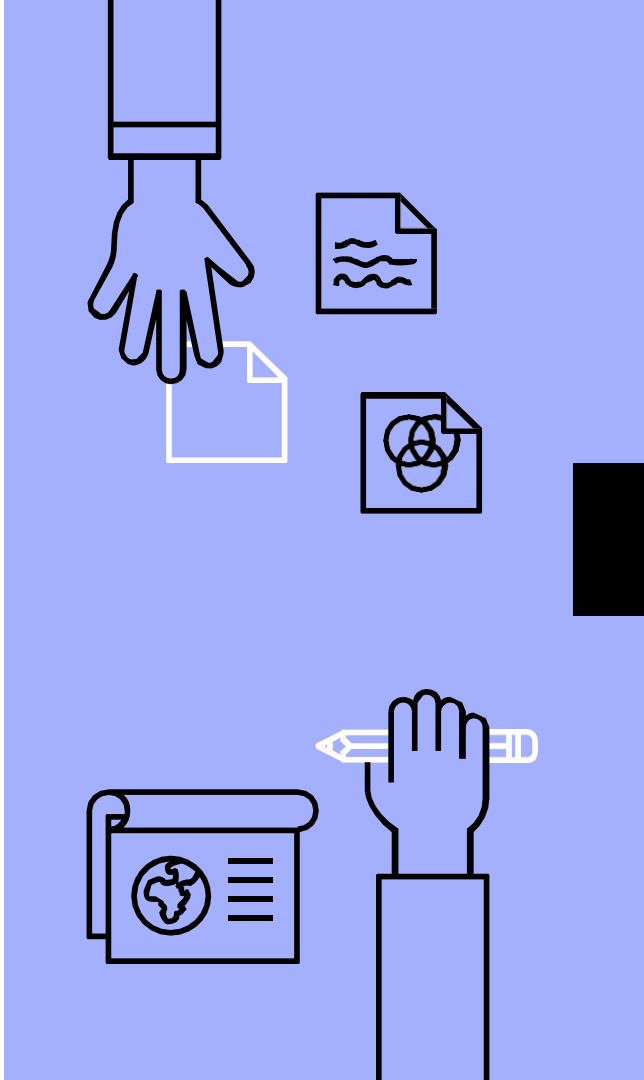
- First statement declares and sets the value of the variable
- Second statement sets up a condition that must evaluate as true to continue
- Third statement is a post-loop operation (typically sets value increment)

MORE ON FOR LOOPS

for loops can also use let instead of var.

Functionally equivalent to a while loop, just a different way of doing it.

- More often used when you know the exact number of times you want the loop to run



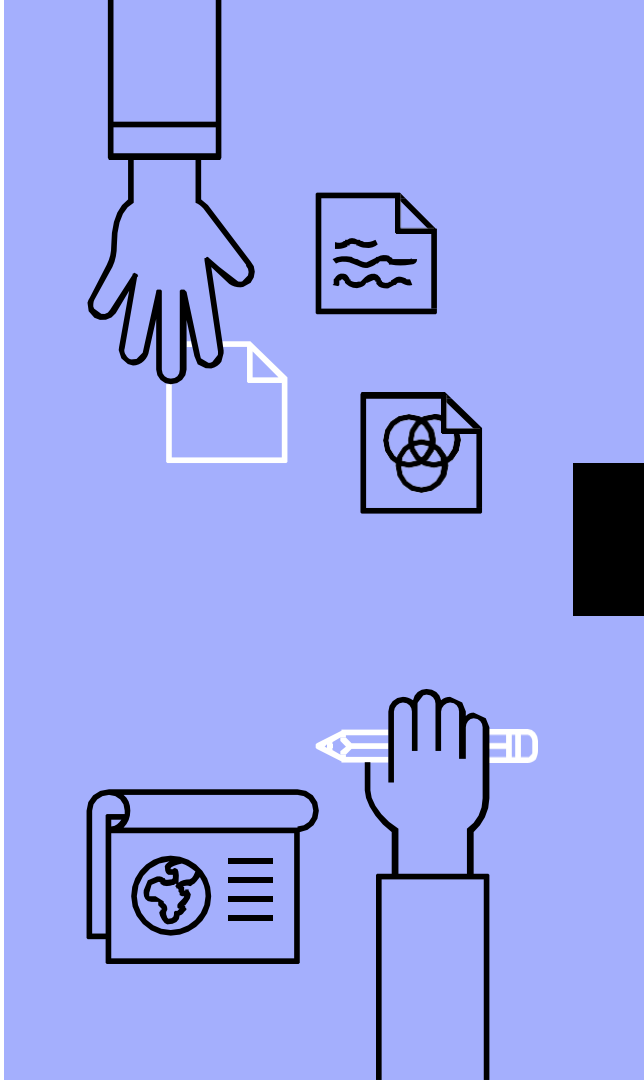
ARRAYS

ARRAYS IN JAVASCRIPT

Arrays are zero indexed.

[] notation for access.

Some examples…



ARRAY EXAMPLE 1

Uses let, but could also use var.

years[0] has a value of 1855

```
let years = [1855, 1648, 1420];
```

ARRAY EXAMPLE 2

Array declaration broken over multiple lines.

```
let placesWithWeirdNames =
["Saskatoon"],
["Saskatchewan"],
["Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch"];
```

ARRAY EXAMPLE 3

Array with multiple data types:

- hotMess[0] has a numeric value of 53
- hotMess[1] has a string value of “Canada”
- hotMess[2] has a Boolean value of true
- hotMess[3] has a numeric value of 1420

```
let hotMess = [53, "Canada", true, 1420];
```

ARRAYS: BEST PRACTICE

JavaScript lets you do some things that are bad practice – and arrays with multiple data types is probably one of them.

In general, it's probably better to use an **array of arrays** rather than an array with multiple different types of data.

WORKING WITH ARRAYS

Common Predefined Functions	Description
.push()	Adds items to an array, and returns the new length
.pop()	Removes the last element from an array, and returns the removed element
.concat()	Concatenates (joins) two arrays
.slice()	Create a new array by setting a start and stop point within an array
.join()	Used to create joining words between array elements
.reverse()	Reverses the order of elements in an array
.shift()	Removes the first item in an array and returns the shifted element
.sort()	Sorts an array alphabetically and ascending order

WHAT IS THE VALUE OF fruits[2]?

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();
fruits.reverse();
document.getElementById("demo").innerHTML = fruits;
```