JavaScript Programming Cheat Sheet

Programs

A program takes **input** and performs a **calculation** to get to an **output.** It uses variables to hold the input values and other useful **state** will the calculation is ongoing. The calculation is just a **series of steps** that are completed in a **certain order** to get from the **input** to the **output**.

Variables

var sets a variable in memory

```
var aNumber = 10;
var aString = 'I am a sentence';
var aBoolean = true                    (or false)
```

```
var anArray = [3,2,4,5,6];
```
all elements in array have to be the same TYPE e.g. a number or a string. Each element is **separated by a comma**.

```
anArray[0] gives 3 – the first element
anArray[2] gives 4 – the third element
```

```
var anObject = {
        name: value,
        name: value
};
```
an object has name-value pairs, with a 'property' a name and a value where the value can be anything, a simple number, an array, or even another object. Each pair is **separated by a comma**.

```
e.g. var person = {
        name: 'Bob Smith',
        age: 27,
        childrensAges: [5, 7, 11],
        address: {
                street: '10 Madeup Ave',
                city: 'Nowhere',
                postcode: 'NO1 7WH'
        }
};
```
use the **dot** (.) syntax to access properties inside the object. Keep in mind what you are given back as you dive into the layer of the object.

```
person.name gives 'Bob Smith'
person.childrensAges [1] gives 7
person.address.city gives 'Nowhere'
```

Operations

+	add numbers or join (concatenate) strings	2 + 3 'Say Hello ' + ' Wave Goodbye'
(-)	subtract numbers	10 - 7
(*)	multiply numbers	20 * 4
/	divide number	100 / 10
>	greater than	6 > 3 gives true, 3 > 6 gives false
<	less than	2 < 4 gives true, 4 < 2 gives false

===	equal	3 === 3 or 'John' === 'John' gives true, 4 === 3 or 'Amy' === 'John' gives false
>=	greater than or equal to	4 >= 1 gives true, 5 >= 10 gives false
<=	less than or equal to	5 <= 9 gives true, 5 <= 2 gives false
||	or	combines any of the four above together e.g. 5 < 10 || 9 < 5
&&	and	e.g. 11 >= 10 && 4 < 12

Shortcuts

+=	add a variable to itself or concatentate a variable with itself
-=	subtract a variable from itself
*=	multiple a variable with itself
/=	divide a variable by itself

Repeating steps

for blocks are used to repeat any set of steps more than once

for (**repeater start value** ; **continue condition** ; **how to change the value of the repeater**)
{
    set of steps we want to repeat
}

e.g.

for (**repeat = 1** ; **repeat <= 10** ; **repeat += 1**)
{
    counter = counter + repeat;
}

here we use the repeat variable value to do something useful.

Choosing steps

if else blocks let you choose a path for the calculation. Can take different forms depending on number of conditions. We can have any number of conditions we like.

if (**condition**)
{
    a set of steps to perform if condition is true
}

if (**condition**)
{
    a set of steps to perform if condition is true
}
else
{
    a set of steps to perform if condition is false
}

if (**condition1**)
{
	a set of steps to perform if condition1 is false
}
else if (**condition2**)
{
	a set of steps to perform if condition2 is false
}
else
{
	a set of steps to perform if condition1 is false and condition2 is false (a catch all)
}