JavaScript Syntax Cheatsheet

Basics

```
var a = 5;
var name = "Alex";
var isReadyToLearn = true;
name = "Pat";
```

Assignment: Put data into variables. New variables are "declared" with *var* keyword.

```
var c = 10;
var d = 4;
c = c + (d * 3);
```

Expressions: Like arithmatic formulas, found everywhere in JavaScript. "Order of operations" applies.

```
var isPrepared = true;
var timeSpent = 5;
var readyToStart = isPrepared && timeSpent > 3;
```

Booleans: Expressions can also compute conditions, resulting in *true* or *false*.

```
var array = ['sam', 900, false];
console.log('Name is ', array[0]);
console.log('Age is ', array[1]);
```

Array: Sequential collection of data, each item is numbered or *indexed* starting with 0.

```
var myObj = {
    name: 'Sam',
    age: 900,
};
console.log('Name is ', myObj.name);
console.log('Age is ', myObj['age']);
```

Object: Like a "dictionary" list of definitions, consists of associated key and value pairs.

Properties can be accessed with either . or [].

Debugging

```
console.log('Hello there!');
var a = 0;
console.log('The value of a is ', a);
```

console.log: Outputs back to the console. Variables can outputed by separating with commas.

Conditionals

```
if (a > 3) {
    console.log('A is greater than 3');
}
```

IF statements: Conditionally execute the code in the curly-braces.

```
if (name === "Alex") {
    console.log('Hi Alex');
} else {
    console.log('Hey there stranger');
}
```

IF ELSE statement: Presents two code paths, conditionally executing one block of code or the other.

Loops

```
var a = 0;
while (a < 5) {
    console.log('Increasing');
    a = a + 1;
}
```

WHILE loop: Like if, except it repeats, possibly forever, until the condition no longer is true.

```
var array = ['pat', 'alex', 'max', 'sam'];
for (var i = 0; i < array.length; i++) {
    var item = array[i];
    console.log('The name is ', item);
}
```

FOR loop: Like while, but with an odd syntax. Useful to repeat code for each item in an array.

Functions

```
var myFunction = function () {
    console.log('This code can be reused...');
};
myFunction();
```

Function: Stores code for re-use.

Function call: Commences the execution of the code between the curly-braces { }

```
function myFunction () {
    console.log('This code can be reused...');
}
myFunction();
```

Named function: Shortcut for giving a function a name, same behavior.

```
function addAndShow (a, b) {
    var sum = a + b;
    console.log('The sum of the numbers: ', sum);
}
addAndShow(10, 5);
addAndShow(-30, 1000);
```

Parameters: A type of variable used by functions, provides "input" in order to be more re-usable. "*Arguments*" when used.

```
function add (a, b) {
    return a + b + (a * b);
}
var total = add(10, 5);
var total2 = add(total, 100);
console.log('Final calculation: ', total2);
```

Return statement: Use to send data back to the caller, represents the "output" of the function.

Advanced shortcuts

```
a = a + 1;
a += 1;
a++;
```

Three ways to increment ("add one to") a variable, all identical.

```
var beverage = age > 21 ? 'beer' : 'soda';
```

Ternary: shortcut to IF

```
name = name || 'default name';
```

OR: shortcut to set default value

```
var a = 0;
while (true) {
    a++;
    if (a > 10) {
        break;
    }
    if (a === 3) {
        continue;
    }
}
```

Flow control: Loops can be prematurely exited with *break*, or skipped to the next iteration with *continue*.
This stops at 11, but "skips" 3.

```
switch (name) {
    case "pat":
        console.log("hi Pat!");
        break;
    case "alex":
        console.log("hey there Alex!");
        break;
    default:
        consoe.log("Howdy stranger");
}
```

SWITCH statement: Many IF-statements can be replaced with one switch statement, but it has strange syntax.