**JavaScript Syntax Sheet**

.length Identifies length of var.

Set variables: var email = ”sam.comber@xaxis.com”;

Console.log(email): Will take whatever is inside the parentheses and log itto the console below the code (otherwise known as printing out).

Confirm(\_\_\_\_) Boxes used to confirm things for users – e.g. unsaved changes.

Prompt(\_\_\_) – e.g. “Where are you from?” user inputs value: var feedback = prompt(“Rate the game from 1 to 10);

Strings – e.g. “DOGS GO WOOF”

Numbers – 123456789

Booleans: true or false values – e.g. 23 > 10 is true.

If statements: evaluates a condition – e.g. if (10 > 5) {console.log(“Greater Than”)}

Else statements: Evaluates whether first condition is true, if so then passes a second condition – e.g. else {console.log(“Nope.”)}

Modulo (%): When % is placed between two numbers, the computer will divide the first number by the second, and then return the remainder of that division – e.g. use modulos in comparison where 10 % 2 === 0 evaluates to true whereas 7 % 3 === 0 evaluates to false as there is 1 left over.

Substrings: For displaying just part of a string – subsetting begins at 0. Example = console.log(“wonderful”.substring(0,3));

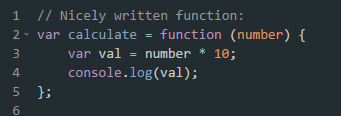
===: Check whether two values are equal.

Confirm: Requests user to confirm.

Functions: take an input, does something, and produces an output.

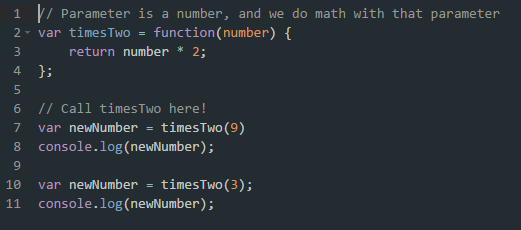
1. Declare a function using var, giving it the name sayHello.
2. Use function keyword to tell the computer you’re making a function.
3. Code in the parentheses is called a parameter, a placeholder word we give a specific value when we call the function.
4. Write reusable code between { }, with every line of code in this block ending with a ;.

To use the function, we call the function by just typing the function’s name and putting a parameter value inside parentheses after it. The computer will run the reusable code with the specific parameter value substituted into the code. Add period at end of each line in {} so computer knows where there are stopping points in the code.



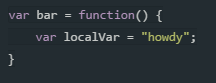
Concatenation: Joining of strings – e.g. (“HEY” + “ “ + “YOU”)

Return: Gives the programmer back the value that comes out of the function. So the function runs, and when the return keyword is used, the function will immediately stop running and return the value. When we call a function, its return value is just the result from running the function.



Functions with two parameters: var areaBox = function(length, width){return length \* width;};

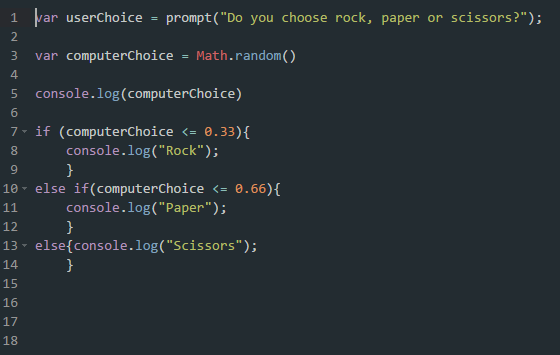
Scope: Variables defined outside a function are accessible anywhere once they have been declared. They are called global variables and their scope is global. Variables inside a function are local variables, they cannot be accessed outside of that function.



Math.random: If we declare a variable and make it equal to Math.random, that variable will equal a number between 0 and 1.

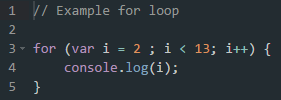
Else if: additional argument for else/if.

&&: and clause.



For loops: Every for loop makes use of a counting variable – e.g. first part of a loop tells the computer to start with a value of 1 for i. For loops only run when the condition is true. So, loop keeps repeating while i is less than the total length of that array. Every loop, we increase value of index by 1 until second condition is false.

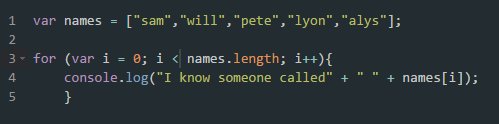
1. i++ increments by 1, i— decrements by 1. I += means incrementing by any value e.g. i += 3 increments by 3s and i -= decrements by 3s.



For IN loops: e.g. for (var index in continents){ response=confirm(“Have you been to “ + continetns[index] + “?”) if (sresponse) count++}.

For OF loops: e.g. for (var continent of continents) {response = confirm (“Have you been ”)}

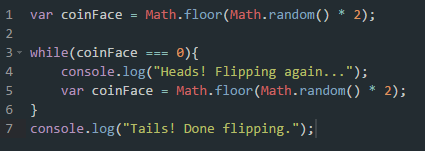
Arrays: Can store lists of data, store different data types at the same time and are ordered so the position of each piece of data is fixed = delineated by [ ] – e.g. console.log(starWars[3]).



Zero-based indexing: where we index the position of elements in an array starting from 0.

Counter variable i for For loops: The counter variable i starts at “start” and stops looping when it reaches “end”.

While loops: will re-run…



Loop control: We introduce break and continue totally stop the loop and continue to stop the current iteration.

E.G.

While (true){this\_amount=prompt(“How much is in this account?”)

This\_amount=parseFloat(this\_amount); if (this\_amount>0) total\_amount+=this\_amount;

Else break;} alert(“Your total savings: “ + total+amount)

**EXAMPLE CODE:**

var age = prompt("What's your age?");

if(age >= 13){

console.log("You're old");

}

else{

console.log("You're young");

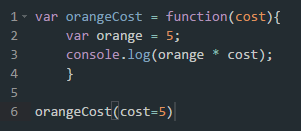
}

**FUNCTIONS:**

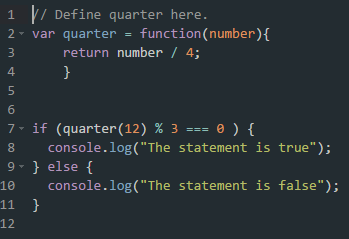
Var functionName = function(){

## CODE CODE CODE ##

}



Functions and IF/ELSE conditions:



**Coursera HTML/CSS/JavaScript:**

Events in Javascript: e.g. onload=””, triggered when the object has loaded then run line of Javascript. <body onload=”alert(‘Hello!’); prompt(‘Excited?’)”>. You write code, and code handles this event.

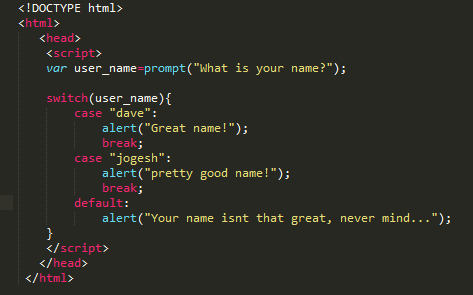
Functions – as opposed to squeezing code into code – puts code into a collection. First we define the function with any kind of name e.g. function do\_something() {} then do\_something();

Onload will execute a JavaScript immediately after a page has been loaded.

Recursive function – a function that calls itself. So, code calls same function.

JavaScript has three kind of popup boxes: alert box, confirm box and prompt box.

Use switch clause to check condition, and say if case is \_\_\_\_, then do \_\_\_\_. Else if break then case is \_\_\_\_ then do \_\_\_\_\_\_. Break stops any more test being made. Use default to catch all items not captured by the case clauses.



**While** loops in JavaScript. Each time the loop content is executed we call it an iteration.

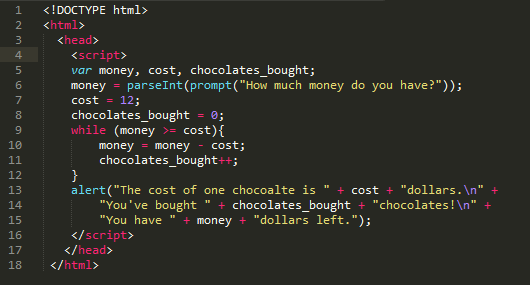
IndexOf() in JavaScript: string.indexOf(“text”) gives you the location of the first “text” in the string. E.g. var text = “The cat’s aht was wet”; THEN result = text.indexOf(“at”); result will be 5 because at is at position 5.

Parseint: parses a string and returns an integer

Do while loops: Perform task whilst condition is true.

Local variables: variables declared within a function can only be accessed within the function – they are local to the function so are called local variables. What if you have local and global variables sharing the same name? Well, JavaScript will give priority to the local variable inside the function. If same variable has not been created inside a function, it will become a global variable.

<body onload=”<FUNCTION>”> loads function when webpage loads.



Logical operators: Work with Boolean values. Manipulating/checking true/false values using logical operators including: Logical And (&&), Logical Or, Logical Not

&& (logical and) - the result is true if both inputs are true, otherwise the result is false.

|| (logical or) – the result is false if both inputs are false, otherwise the result is true.

! (logical not) – does the opposite – i.e. a = false !a = true.

Arrays: an array is a linear continuous storage which can be thought as a group of boxes based on zero-based indexing. E.g. var pets = new Array(10) (ten empty boxes)

Use array.join to convert array into strings.

E.g. var pets = [“rabbit”,”cat”,”dog”]

Alert(pets.join(“ and “)). You can retrieve an item like this: alert(pets[2]) to show “cat”.

Add new element using the .push clause e.g. pets.push(“hamster”);

Add new element to the front with array.unshift e.g. pets.unshift(“hamster”);

Delete element from the end using array.pop – e.g. var result = pets.pop();

Delete element from the front using. Array.shift)\_ - e.g. var results = pets.shift();

Combine two arrays using array1.concat(array2) – e.g. var result = pets.concat(primes);

Generating random numbers (RNG): e.g. var random\_number – Math.random(). The resulting range is [0,1] from 0 up to 1 but not including 1.

Setting up a range: so far the random number is in the range 0 up to 1, multiply in order to get the range you want – i.e. random\_number = Math.random() \* max\_value; So, random\_number = Math.random() \* 8 …. To 7.99999

Throw away the decimal place, we want an integer: use Math.floor() to dump decimal place. So, math.floor(2.7534) = 2. So random\_number = Math.random() \* 50; random\_number = Math.floor( random\_number); alert(random\_number)