**Table of Contents**

**While loop**

**How to write a while loop . . . . . . . . 2**

**Do / while loop . . . . . . . . . 3**

**JavaScript tip . . . . . . . . . 3**

**Random using Math.floor . . . . . . . . 3**

**Control flow**

**IsNaN . . . . . . . . . 4**

**Switch statement**

**What is Switch statement . . . . . . . . 4**

**How to write a Switch statement . . . . . . . 4**

**While loop**

The while loop is ideal when you want to use a loop, but you don't know how many times you'll have to execute that loop.

If you give a while loop a condition that is *true*and you don't build in a way for that condition to possibly become *false*, the loop will go on forever and your program will crash. To prevent this from happening, you *always* need a way to ensure the condition between your while parentheses can change.

**\*\*\*\*\*\* Example below**

var bool = true;

while(bool){

console.log("Less is more!");

bool = false;

}

**Above code displays:**

**Less is more**

**False;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**How to create a while loop that prints “I’m a nigga!” three times? Look below.**

var loop = function(myNigger){

var i = 0;

while(i < myNigger){

console.log("I'm looping!")

i++;

}

};

loop(3);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**DO \ WHILE loop**

Sometimes you want to make sure your loop runs *at least one time* no matter what. When this is the case, you want a modified *while* loop called a *do/while*loop.

Example below:

var loopCondition = false;

do {

console.log("I'm gonna stop looping 'cause my condition is " + loopCondition + "!");

} while (loopCondition);

The code above will print: *“I’m gonna stop looping ‘cause my condition is false!”*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\*\*\*\*\*\* TIP: JavaScript shortcut:**

nigga = nigga + niggers;

//can be written easily by typing:

nigga += niggers;

you're telling JavaScript to *nigga* and *niggers*, then assign that new value *nigga*.

**How to random using Math.floor**

var youHit = Math.floor(Math.random() \* 2);

This sets *youHit*to a random number that's either 0 (which JavaScript reads as false) or 1 (which JavaScript reads as true).

var damageThisRound = Math.floor(Math.random() \* 5 + 1);

This sets *damageThisRound* to a random number that's between 1 and 5 (up to and including 5).

**Control FLOW**

**isNaN** – if you call isNaN on something it checks to see if that thing is not a number.

isNaN('berry'); // => true

isNaN(NaN); // => true

isNaN(undefined); // => true

isNaN(42); // => false

**Switch statement**

if you have a *lot* of choices you want to cover in a program, it might be annoying to type *else if () ten times*. That's why *JavaScript* has the *switch* statement!

*switch* allows you to preset a number of options (called cases), then check an expression to see if it matches any of them. If there's a match, the program will perform the action for the matching case; if there's no match, it can execute a default option.

Example below:

var color = prompt("What's your favorite primary color?","Type your favorite color here");

switch(color) {

case 'red':

console.log("Red's a good color!");

break;

case 'blue':

console.log("That's my favorite color, too!");

break;

case 'yellow':

console.log ("That's Marian's favourite color!");

break;

default:

console.log ("I don't think that's a primary color!");

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Logical operators**

**And operator**

The logical operator **and** is written in JavaScript like this: &&. It evaluates to true when both expressions are true; if they're not, it evaluates to false.

true && true; // => true

true && false; // => false

false && true; // => false

false && false; // => false

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Or operator**

The logical operator **or** is written in JavaScript like this: ||. It evaluates to true when one or the other or both expressions are true; if they're not, it evaluates to false.

true || true; // => true

true || false; // => true

false || true; // => true

false || false; // => false

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Not operator**

The logical operator **not** is written in JavaScript like this: !. It makes true expressions false, and vice-versa.

!true; // => false

!false; // => true

**\*\*\*\*\*\*\*\*\*\*\*\*\* Upper case and Lower case \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

.toUpperCase(); 🡨 This convert’s a string to an uppercase

.toLowerCase(); 🡨 This converts a string to lowercase

Below example converts the users answer to Uppercase to eliminate problems

var answer = prompt("Question to the user").toUpperCase();

**Heterogeneous Arrays**

You can have a heterogeneous array, which means a mixture of data types, like so:

var mix = [42, true, "towel"];

**Arrays of Arrays / Two dimensional array**

you put a mixture of types in an array, you can even put *other arrays* inside arrays. You can make a **two-dimensional array** by nesting arrays one layer deep, like so:

var twoDimensional = [[1, 1], [1, 1]]

This array is **two-dimensional** because it has two rows that each contain two items. If you were to put a new line between the two rows, you could log a 2D object—a square—to the console, like so:

[1, 1]

[1, 1]

**Jagged Arrays**

You may have three elements in the first row, one element in the second row, and two elements in the third row. JavaScript allows those, and they're called **jagged arrays**.

var jagged = [[1,2,"pussy"], ["Midashim", true], [5,6,7, false]];

**Objects**

Using ***objects***, we can put our information and the functions that use that information in the same place. You can also think of objects as combinations of key-value pairs (like arrays), only their keys don't have to be numbers like 0, 1, or 2: they can be strings and variables.

There are two ways to create an object: using **object literal notation** and using the **object constructor.**

Literal notation is just creating an object with curly braces, like this:

var myObj = {

type: 'fancy',

disposition: 'sunny'

};

var emptyObj = {};

When you use the constructor, the syntax looks like this:

var myObj = new Object();

This tells JavaScript: "I want you to make me a ***new***thing, and I want that thing to be an ***Object.***

myObj["name"] = "Charlie";

myObj.name = "Charlie";

Both are correct, and the second is shorthand for the first. See how this is sort of similar to arrays?