Extras

# Set Timeout(function, timeInMilliSeconds):

# Set Interval(function,timeInMilliSeconds):

# Uglify:

­You can minify and now also uglify your code online. What uglifying really does is that changes the variable names along with minifying the code (making it unreadable – uglyyy!!).

# False values:

‘ ’ (an empty string), null, undefined, 0, false all are considered as not true by JS

# add()

This is a method that works with arrays. The best example for now would be that if you call a list of classes of any element and add to it.

var styles = document.getElementById(‘myEle’).classList;

styles.add(‘hide’);

# remove()

You guessed it right! It is just like add except that its function is to remove the element.

It Removes all child elements with selected element.

You could also use empty() or delete but they have different functions.

# empty()

This removes all the content within the selected element.

# delete:

This has the power to completely delete an object’s property and value together. Although some believe that it is slow when working in a loop for instance.

# Self Executing Functions:

(function(){

//Bunch of code...

})();

The above function defines itself and calls itself. These are also known as anonymous functions.

Any thing that is inside the function will be private unless you yourself want it to be public (global in scope).

# const keyword:

The const keyword is now interchangeably used with var. the key difference is that the value of any var can be changed later on.

However const is an immutable variable whose value will not change like getting an element by its Id.

# addEventListener(,):

Normally when we discuss in JS we say that it is not a good practice to place an onclick or onMouseover attribute on the HTML tag.. Well what was the alternate? This is the alternate!

We get the element by any method..

myEle.addEventListener(event\_name,handler);

Event\_name can be anything such as click, mouseover, mouseout and many more which you can google.

Where as handler is a callback function.

Some people use “e => {}” instead of a function/handler. Why?

Here e is a parameter that contains the whole function enclosed in curled brackets. The output of the function will be contained in e.

# CallBack functions:

Call back functions are functions that are passed within other functions as parameters and are executed after another function has been executed.

In other words, suppose you want to execute a function but before its execution you require some additional data from elsewhere which will require some milliseconds delay.. so how do you tell JS to execute function 2 after function 1 has completed execution..

CallBacks.

function myFunc (a, callback) {

console.log(a);

callback();

}

function bye() {

console.log(“good bye”);

}

myFunc(“hey”, bye);

# Promise:

# Promise.catch:

A promise has 2 conditions resolved, rejected.. In case of rejected it ‘throw’s something.

So we ‘catch’ that something and then we console.log() it to view the error messages that it gives.

# Promise.then: