Notes

Callback functions

A callback function is a function that is passed as an argument to another function, to be “called back” at a later time.

Lets take a look at an example to see what this means in more detail.

Example:

```
function sum(x, y, callBack){
    var total = x + y
    callBack(total)
}

var print = function(toPrint){
    console.log(toPrint)
}

sum(4, 5, print)
```

Output:

```
9
```

As you can see we passed the print function as an argument to the sum function. First the sum function calculated the sum of x and y and then the print function was called after the sum was calculated to print the sum to the console.

Higher Order Functions

Functions that operate on other functions, either by taking them as arguments or by returning them, are called higher-order functions.

Here are some commonly used inbuilt methods for arrays.

- array.forEach()
- array.map()
- array.filter()
- array.reduce()

each of these takes an argument.

forEach:

The forEach() method executes a provided function once for each array element.

Syntax

```
array.forEach(function(element){
    console.log(element)
})
```

```
array.forEach(function(currentElement, currentIndex, array){...})
// It takes a function as an argument
// currentElement
//     The current element being processed in the array.
// currentIndex || optional
//     The index of the current element being processed in the array.
// array || optional
// The original array forEach() was called upon.

var names = ['Nrupul', 'Albert', 'Sid', 'Aman']

names.forEach(function(element){
      console.log(element)
})
// Nrupul
// Albert
// Sid
// Aman
```

map:

The map() method creates a new array with the results of calling a provided function on every element in the calling array.

Syntax

```
array.map(function(element){
      console.log(element)
})
array.map(function(currentElement, currentIndex, array){...})
// It takes a function as an argument
// currentElement
//     The current element being processed in the array.
// currentIndex || optional
//     The index of the current element being processed in the array.
// array || optional
// The original array map() was called upon.
```

Arguments:

```
var names = ['Nrupul', 'Albert', 'Sid', 'Aman']

newNames = names.map(function(element, index){
      return [element ,index]
```

```
})
console.log(newNames)
// [ ["Nrupul", 0], ["Albert", 1], ["Sid", 2], ["Aman", 3] ]

The main difference between .forEach and .map() is that .map() returns a new array. If you need the result, but do not wish to mutate the original array, .map() is the clear choice. If you simply need to iterate over an array, forEach is a fine choice.
```

filter:

The filter() method creates a new array with all elements that pass the test implemented by the provided function.

Syntax

```
array.filter(function(element){
      return condition to check
})
var newArray = array.filter(function(currentElement, currentIndex, array){})
// It takes a function as an argument
// currentElement
//     The current element being processed in the array.
// currentIndex || optional
//     The index of the current element being processed in the array.
// array || optional
// The original array forEach() was called upon.
```

A new array with the elements that pass the test. If no elements pass the test, an empty array will be returned.

Arguments:

```
var names = ['Nrupul', 'Albert', 'Sid', 'Aman']

newNames = names.filter(function(element, index){
      return element.length>4
})
console.log(newNames)
// [ "Nrupul", "Albert" ]
```

reduce:

The reduce() method executes a reducer function (that you provide) on each element of the array, resulting in a single output value.

Syntax

```
array.reduce(function(accumulator,element){
```

```
    return condition for accumulator
})
array.reduce(function(accumulator, currentValue, currentIndex, array){},0)
// It takes a function as an argument
// accumulator
//    The accumulating value which will be accumulated over time to be the final output
// currentValue || optional
//    The current element being processed in the array.
// currentIndex || optional
//    The index of the current Index being processed in the array.
// array || optional
// The original array forEach() was called upon.
// the value 0 after the function, is the initial value of the accumulator.

// What is accumulation?
// the acquisition or gradual gathering of something.
```

Arguments:

```
array = [1, 2, 3, 4, 5]

array.reduce(function(a,c){
    return a + c
})
// 15
```