1. **Write script for following methods of Array object:** 
   1. **Find()**

Ans.-> const arr = [3, 8 ,19, 23, 44 ];

const found = arr.find(element => element > 15);

console.log(found);

* 1. **Filter()**

Ans.-> const flowers = ['rose','lily','Lotus','Tulip','sunflower','Orchid'];

const result = flowers.filter(flowers => flowers.length > 5);

console.log(result);

* 1. **Map()**

Ans -> const array = [2,6,8,12];

const map1 = array.map(x => x \* 4);

console.log(map1);

* 1. **Reduce()**

Ans -> const array1 = [4,5,6,7,8];

const initialvalue = 0;

const sumWithInitiaal = array1.reduce((previousvalue,currentValue) => previousvalue + currentValue,initialvalue);

console.log(sumWithInitiaal);

* 1. **Slice()**

Ans-> const fruits = ['apple','banana','Gava','Jackfruit','Mango'];

console.log(fruits.slice(2));

console.log(fruits.slice(2, 4));

console.log(fruits.slice(1, 5));

console.log(fruits.slice(-2));

console.log(fruits.slice(2, -1));

console.log(fruits.slice());

* 1. **Splice()**

Ans:- const months = ['Jan', 'March', 'April', 'June'];

months.splice(1, 0, 'Feb');

console.log(months);

months.splice(4, 1, 'May');

console.log(months);

1. **Study closure in JavaScript and write script for the same.**

Ans:-

A closure is a function having access to the parent scope, even after the parent function has closed. OR

closure is created when a child function keep the environment of the parent scope even after the parent function has already executed.

It makes it possible for a function to have "private" variables.

The counter is protected by the scope of the anonymous function,

and can only be changed using the add function.

Example of closure:-

function foo()

{

var b = 1;

function inner(){

return b;

}

return inner;

}

var get\_func\_inner = foo();

console.log(get\_func\_inner());

console.log(get\_func\_inner());

console.log(get\_func\_inner());

Ex:- 2 // Outer function

function outer()

{

var arr = [];

var i;

for (i = 0; i < 4; i++)

{

// storing anonymous function

arr[i] = function () { return i; }

}

// returning the array.

return arr;

}

var get\_arr = outer();

console.log(get\_arr[0]());

console.log(get\_arr[1]());

console.log(get\_arr[2]());

console.log(get\_arr[3]());

1. **Write a JavaScript function to merge two arrays and removes all duplicates elements**

**Test data :**

**var array1 = [1, 2, 3]; var array2 = [2, 30, 1];**

**console.log(merge\_array(array1, array2));**

[3, 2, 30, 1]

Ans :-

Ex.1 var array1 = [1,3,6,8];

var array2 = [2,6,9,1];

var merge\_array =[...array1,...array2];

console.log(merge\_array);

//remove all duplicates

const unique = [...new Set(merge\_array)];

console.log(unique);

Ex.2 let array1 = ['Jack','Nick','John'];

let array2 = ['Nick','lily','Jack'];

let arr = array1.concat(array2);

console.log(arr);

//remove all duplicates

let unique = arr.filter(function(item,index) {

return arr.indexOf(item) === index;

});

console.log(unique);

1. **Write a pattern that matches e-mail addresses.**

**The personal information part contains the following ASCII characters.**

* + **Uppercase (A-Z) and lowercase (a-z) English letters.**
  + **Digits (0-9).**
  + **Characters ! # $ % & ' \* + - / = ? ^ \_ ` { | } ~**
  + **Character . ( period, dot or full stop) provided that it is not the first or last character and it will not come one after the other.**

Ans:-

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>JavaScript form validation - checking email</title>

<link rel='stylesheet' href='form-style.css' type='text/css' />

</head>

<body onload='document.form1.text1.focus()'>

<div class="mail">

<h2>Input an email and Submit</h2>

<form name="form1" action="#">

<ul>

<li><input type='text' name='text1'/></li>

<li>&nbsp;</li>

<li class="submit"><input type="submit" name="submit" value="Submit" onclick="ValidateEmail(document.form1.text1)"/></li>

<li>&nbsp;</li>

</ul>

</form>

</div>

<script>

function ValidateEmail(inputText)

{

var mailformat = /^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/;

if(inputText.value.match(mailformat))

{

alert("Valid email address!");

document.form1.text1.focus();

return true;

}

else

{

alert("You have entered an invalid email address!");

document.form1.text1.focus();

return false;

} }

</script>

</body></html>

1. **Write a JavaScript function to get the values of First and Last name of the following form: Sample HTML file :**

**<!DOCTYPE html>**

**<html><head>**

**<meta charset=utf-8 />**

**<title>Return first and last name from a form - w3resource</title>**

**</head><body>**

**<form id="form1" onsubmit="getFormvalue()">**

**First name: <input type="text" name="fname" value="David"><br>**

**Last name: <input type="text" name="lname" value="Beckham"><br>**

**<input type="submit" value="Submit">**

**</form>**

**</body>**

**</html>**

Ans:-

<!DOCTYPE html>

<html><head>

<meta charset=utf-8 />

<title>first and last name from a form </title>

<style type="text/css">

body {margin: 25px;}

</style>

</head><body>

<form id="form" onsubmit="getFormvalue()">

First name: <input type="text" name="fname" value="David"><br>

Last name: <input type="text" name="lname" value="Beckham"><br>

<input type="submit" value="Submit">

</form>

<script>

function getFormvalue()

{

var x=document.getElementById("form");

for (var i=0;i<x.length;i++){

if (x.elements[i].value!='Submit') {

console.log(x.elements[i].value);

}

}

}

</script>

</body></html>

1. **Write a JavaScript program to remove items from a dropdown list.**

**Sample HTML file:**

|  |
| --- |
| **<!DOCTYPE html>**  **<html><head>** |
| **<meta charset=utf-8 />**  **<title>Remove items from a dropdown list</title> </head><body><form>** |
| **<select id="colorSelect">** |
| **<option>Red</option>**  **<option>Green</option>**  **<option>White</option>**  **<option>Black</option>**  **</select>**  **<input type="button" onclick="removecolor()" value="Select and**  **Remove">**  **</form></body></html>** |

Ans:-

<!DOCTYPE html>

<html> <head>

<style type="text/css">

body {margin: 30px;}

</style>

<meta charset=utf-8 />

<title>Remove items from a dropdown list</title>

</head><body><form>

<select id="colorSelect">

<option>Red</option>

<option>Green</option>

<option>White</option>

<option>Black</option>

</select>

<input type="button" onclick="removecolor()" value="Select and Remove">

</form>

<script>

function removecolor()

{

var x=document.getElementById("colorSelect");

x.remove(x.selectedIndex);

}

</script>

</body></html>

1. **Write a JavaScript program to highlight the bold words of the paragraph, on mouse over a certain link.**

**Test data**:

**Link text is – Click here to highlight bold words.**

**Ans:-**

<!doctype html>

<html>

<head>

<meta charset="UTF-8">

<title>Get And Style All Tags</title>

</head>

<body>

<p>[<a href="#" onMouseOver="highlight()" onMouseOut="return\_normal()">Link text is – Click here to highlight bold words. </a>]</p>

<p><strong>Used</strong> in those texts,<strong>which</strong>are the <strong>important</strong>and need to <strong>highlight.</strong></p>

</body>

<script>

var bold\_Items;

window.onload = getBold\_items();

function getBold\_items()

{

bold\_Items = document.getElementsByTagName('strong');

}

function highlight()

{

for (var i=0; i<bold\_Items.length; i++)

{

bold\_Items[i].style.color = "red";

}

}

function return\_normal()

{

for (var i=0; i<bold\_Items.length; i++)

{

bold\_Items[i].style.color = "black";

}

}

</script>

</html>

1. **Write a JavaScript function to get the month name from a particular date.**

**Test Data :**

**console.log(month\_name(new Date("10/11/2009"))); console.log(month\_name(new Date("11/13/2014")));**

**Output :**

**"October"**

**"November"**

**Ans**:-

var month\_name = function(dt){

month = [ "January", "February", "March", "April", "May", "June", "July", "August",

"September", "October", "November", "December" ];

return month[dt.getMonth()];

};

console.log(month\_name(new Date("10/11/2009")));

console.log(month\_name(new Date("11/13/2014")));

**9.Write a JavaScript function to test whether a date is a weekend. Go to the editor**

**Note : Use standard Saturday/Sunday definition of a weekend.**

**Test Data :**

**console.log(is\_weekend('Nov 15, 2014')); console.log(is\_weekend('Nov 16, 2014')); console.log(is\_weekend('Nov 17, 2014'));**

**Output :**

**"weekend"**

**"weekend"**

**Undefined**

**Ans:-**

var is\_weekend = function(date){

var dt = new Date(date);

if(dt.getDay() == 6 || dt.getDay() == 0)

{

return "weekend";

}

}

console.log(is\_weekend('Nov 15, 2014'));

console.log(is\_weekend('Nov 16, 2014'));

console.log(is\_weekend('Nov 17, 2014'));

**10.Write a JavaScript program to display the reading status (i.e. display book name, author name and reading status) of the following books.**

|  |
| --- |
| **var library = [** |
| **{**  **author: 'Bill Gates', title: 'The Road Ahead', readingStatus: true**  **},** |
| **{** |
| **author: 'Steve Jobs', title: 'Walter Isaacson', readingStatus: true**  **}, {**  **author: 'Suzanne Collins',**  **title: 'Mockingjay: The Final Book of The Hunger Games', readingStatus: false**  **}];** |
|  |

**Ans:-**var library = [

{

title: 'Bill Gates',

author: 'The Road Ahead',

readingStatus: true

},

{

title: 'Steve Jobs',

author: 'Walter Isaacson',

readingStatus: true

},

{

title: 'Mockingjay: The Final Book of The Hunger Games',

author: 'Suzanne Collins',

readingStatus: false

}];

for (var i = 0; i < library.length; i++)

{

var book = "'" + library[i].title + "'" + ' by ' + library[i].author + ".";

if (library[i].readingStatus) {

console.log("Already read " + book);

} else

{

console.log("You still need to read " + book);

} }

**11.Write a JavaScript program to create a Clock.**

**Note: The output will come every second.**

***Expected Console Output:***

**"14:37:42" "14:37:43" "14:37:44" "14:37:45" "14:37:46"**

**"14:37:47"**

**Ans:-**

function my\_Clock()

{

this.current\_date = new Date();

this.hours = this.current\_date.getHours();

this.minutes = this.current\_date.getMinutes();

this.seconds = this.current\_date.getSeconds();

}

my\_Clock.prototype.run = function ()

{

setInterval(this.update.bind(this), 1000);

};

my\_Clock.prototype.update = function ()

{

this.updateTime(1);

console.log(this.hours + ":" + this.minutes + ":" + this.seconds);

};

my\_Clock.prototype.updateTime = function (secs)

{

this.seconds+= secs;

if (this.seconds >= 60)

{

this.minutes++;

this.seconds= 0;

}

if (this.minutes >= 60)

{

this.hours++;

this.minutes=0;

}

if (this.hours >= 24)

{

this.hours = 0;

}

};

var clock = new my\_Clock();

clock.run();

**12.Write a JavaScript program to sort an array of JavaScript objects.**

**Sample Object:**

**var library = [**

**{**

**title: 'The Road Ahead', author: 'Bill Gates', libraryID: 1254**

**}, {**

**title: 'Walter Isaacson', author: 'Steve Jobs', libraryID: 4264**

**}, {**

**title: 'Mockingjay: The Final Book of The Hunger Games', author: 'Suzanne Collins', libraryID: 3245**

**}];**

**Ans:-**

var library = [

{

title: 'Bill Gates',

author: 'The Road Ahead',

libraryID: 1254

},

{

title: 'Steve Jobs',

author: 'Walter Isaacson',

libraryID: 4264

},

{

title: 'Mockingjay: The Final Book of The Hunger Games',

author: 'Suzanne Collins',

libraryID: 3245

}];

var sort\_by = function(field\_name, reverse, initial){

var key = initial ?

function(x)

{

return: initial(x[field\_name]);

} :

function(x)

{

return x[field\_name];

};

reverse = !reverse ? 1 : -1;

return function (x, y) {

return x = key(x), y = key(y), reverse \* ((x > y) - (y > x));

} ;

};

var newobject = library.sort(sort\_by('libraryID', true, parseInt));

console.log(newobject);

**13. Study Promises in JavaScript and provide script for the same.**

**Ans:-**

A Promise is a JavaScript object that links producing code and consuming code.

Producing code" is code that can take some time.

Consuming code" is code that must wait for the result.

Promise Syntax:-

let myPromise = new Promise(function(myResolve, myReject) {

// "Producing Code" (May take some time)

myResolve(); // when successful

myReject(); // when error

});

// "Consuming Code" (Must wait for a fulfilled Promise)

myPromise.then(

function(value) { /\* code if successful \*/ },

function(error) { /\* code if some error \*/ }

);

Example Using Callback:-

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript SetTimeout()</h2>

<p>Wait 3 seconds for this page to change.</p>

<h1 id="demo"></h1>

<script>

setTimeout(function() { myFunction("I Am Rutuja.."); }, 3000);

function myFunction(value) {

document.getElementById("demo").innerHTML = value;

}

</script>

</body>

</html>

**Example Using Promise;-**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Promise</h2>

<p>Wait 3 seconds for this page to change.</p>

<h1 id="demo"></h1>

<script>

const myPromise = new Promise(function(myResolve, myReject) {

setTimeout(function(){ myResolve("I Am Rutuja.."); }, 3000);

});

myPromise.then(function(value) {

document.getElementById("demo").innerHTML = value;

});

</script>

</body>

</html>