JAVASCRIPT

**Array functions:**

The split() method

Is use to convert a text into an array in the way you want. eg.

// below is saying, convert each single text into a single item in an array. OR simply replace all white space with comma "," and return an array.

var str = "How are you doing today?";

var res = str.split(" ");  // replace all white space “ ”  with comma

// result == [ “How”, “are”, “you”, “doing”, “today?”]

or

// below is saying, counting from left, add comma to the first 3 spaces and return that as an array.

var str = "How are you doing today?";

  var res = str.split(" ", 3);

// result == [“How”, “ are”, “ you”, “ doing”]

The pop() method

// The pop() method removes the last element of an array and returns the new length eg.

var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.pop();

//result == ["Banana", "Orange", "Apple"];

The push() method

//The push() method adds new items to the end of an array, and returns the new length.

var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.push("Kiwi");

//result == ["Banana", "Orange", "Apple", "Mango", "kiwi"];

The shift() method

// The shift method removes the first element of an array and returns the new length.

var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.shift();

//result == [ "Orange", "Apple", "Mango"];

The unshift() method

The unshit method Add element at the beginning of an array

var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.unshift("Lemon","Pineapple");

// result = [Lemon,Pineapple,Banana,Orange,Apple,Mango]

The splice() method

It's used to add/remove item to/from an array.

It usually takes in 3 parameters splice(at\_what\_index, how\_many\_items\_to\_remove, items\_to\_add);

"at\_what\_index" ==required an integer that specifies what position of index in the array do you want to add item or remove item.

"how\_many\_items\_to\_remove" == Optional. an integer that specifies how many items you want to remove from the array. Specifies 0 if you don’t want to remove anything

"items\_to\_add" == Optional. The new item(s) to be added to the array

//Below is saying: at index position 2 of fruits array, remove zero items and add 'lemon' and 'kiwi'

var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.splice(2, 0, "Lemon", "Kiwi");

//result = [Banana,Orange,Lemon,Kiwi,Apple,Mango]

The indexOf() method

The indexOf() method is used to find the index of a specific element in an array. If the element is not find in the array, it will return -1

It usually takes in 1 or 2 parameters:

var fruits = ["Banana", "Orange", "Apple", "Mango", “Orange”];

// Means what is the index position of “Orange” in the fruits array?

fruits.indexOf(“Orange”)

//result = 1

// Means, Hey, skip from position 0 to 2 of the array and only look for the index Of “Orange” from position 2 going and tell me.

// So this will skip the first occurrence of “orange” it occurs before you started counting from position 2

var fruits = ["Banana", "Orange", "Apple", "Mango", “Orange”];

b =fruits.indexOf(“Orange”,2)

console.log(b);

//result = 4

**The Map() fuction**

The map method is method that is used to perform certain operation on all element in a specific array. It retunes a new array. Let say, you want to add 2 to all the element in an array, or you want a certain function to run on all element of an array. Then use the map function.

The syntext:

Below is the syntax to follow in both using arrow function, inline callback function and callback function

const colors = ["red", "yellow", "green", "blue"];

/\*

items => represent the items in the colors array that: "red", "yellow", "green", "blue"

index => represnt the index of each item in the colors array that's: 0, 1, 2,3

array\_obj => represent the object of the colors array thats: colors = ["red", "yellow", "green", "blue"];

\*/

// Arrow function

map((items) => {... })

map((items, index) => {... })

map((items, index, array\_obj) => {... })

// Inline callback function

map(function callbackFn(items) {... })

map(function callbackFn(items, index) {... })

map(function callbackFn(items, index, array\_obj) {... })

map(function callbackFn(items, index, array\_obj) {... }, thisArg)

// Callback function

map(callbackFn)

map(callbackFn, thisArg)

Example:

const colors = ["red", "yellow", "green", "blue"];

// pass a function to map so that it will run all element inside the colors array.

/\*

colors\_items => represent the items in the colors array that: "red", "yellow", "green", "blue"

colors\_index => represnt the index of each item in the colors array that's: 0, 1, 2,3

colors\_obj => represent the object of the colors array thats: colors = ["red", "yellow", "green", "blue"];

\*/

const getIndex = colors.map(

    function(colors\_items, colors\_index, colors\_obj) {

        return colors.index; // print the index of  the colors arrary.

    }

);

console.log(getIndex);

// results: [0, 1, 2,3]

OR

// add "justice" to all items in the colors array

const addJustice = colors.map(

    // n => represent the items in the array thats: "red", "yellow", "green", "blue"

    function(n) {

        return n + "Justice";

    }

);

// Or you can easily write the above method in arrow function as

const addJustice = colors.map(

 n => n + "Justice"

);

console.log(addJustice);

// results: [redJustice, yellowJustice, greenJustice, blueJustuce]

**==============================================================================**

**Event**

In real life, event is an upcoming occasions like party, wedding and birthday. it simply an occasion that people are waiting to attend it at a particular day.

In programming too, an event is something that your program wait for, and when the event happens the program execute a callback function.

The event could happen on the whole page or a specific element in the dom. And JavaScript allows you do execute some code when an event happens.

NOTE: Event are something a browser or a user does for Example:

* The browser has finish loading a page
* The user clicks on a particular Dom element

**Concept:**

The program Wait for an event to occur on the whole page or specific Dom element & execute a callback function

**Formats:**

object.addeventListener(event, callbackFunction);

object = the element the event should occur on

event = the type of event the program is waiting for (click, change, mousedown, mouseout)

callbackFunction = the call-back function the program should call when the event happened

How to add event to dom element

To add an event to a dome element, us the element.addEventListener(“event\_type”, callback\_fuck) method.

  <h1>HELLO HOW ARE YOU DOING?</h1>

  <button id="mybtn">Click Me</button>

<script>

    var myBtn = document.getElementById("mybtn");

    // add a click event to the button.

    myBtn.addEventListener("click", doSomething);

    // this is the callback function that will be executed when the event happened

    // "ev" below represent the event object. it contains information about the event ant the Dom element the event occurred on

    function doSomething(ev) {

        // get the event type

        console.log(ev.type);

        // get the element the event occurred on

        console.log(ev.target);

    }

    /\* RESULT:::

    click

    <button id="mybtn">

    \*/

</script>

The element.preventDefault() method helps you to stop the default operation of a dom element and do what in the callback event function rather.

For example, we know that the default job of the <a> tag is to open a link. Well, you can use the preventDefault() method to disable that feature and have the <a> tag do something else when it clicked.

   <a href="google.com" id="mylink">go to google</a>

<script>

    var mylink = document.getElementById("mylink");

    // add a click event to the button.

    mylink.addEventListener("click", doSomething);

    function doSomething(ev) {

        // Stop the dom element default behavior and do what is in

        // this function instead

        ev.preventDefault();

        console.log("hi");

    }

    /\* RESULT:::

    so instead of opening google.com it will rather output:

    hi

    \*/

</script>

**==============================================================================**

**How To Validate Email With Regular expression**

You can easily validate email in javaScript by using the test() function. This test function takes in a string and compare it to a regular expression pattern. Below is how to do it.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <!-- CLICK BUTON THAT CONTAINS USER EMAIL -->

  <button onclick="validatEmail('java@gmail.com')">Validate Email</button>

    <!-- SCRIPT TO VALIDATE THE USER EMAIL  -->

    <script>

        // CREATE A FUNCTION WITH PARAMETER

function validatEmail(email){

    //ASING THE REGEX EMAIL VALIDATION PARTERN TO A CONSTANT

    const pattern  = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

    // CHECK IF THE USER-EMAIL FOLLOWS THE ABOVE REGEX-EMAIL-VALIDATION-PARTTERN

    //The test() method executes a search for a match between a regular expression and a specified string (it return yes or no)

    if(pattern.test(email.toLowerCase())){

        // if is correct pring out "valid email"

    console.log("valid email");

   }

   else{

       //else pring "invalid email"

    console.log("invalid email");

}

}

    </script>

</body>

</html>