**Array methods**

As anything in JS consider an object with properties and methods, arrays also consider objects with properties and methods. Most of these methods **cause change** in the main array.

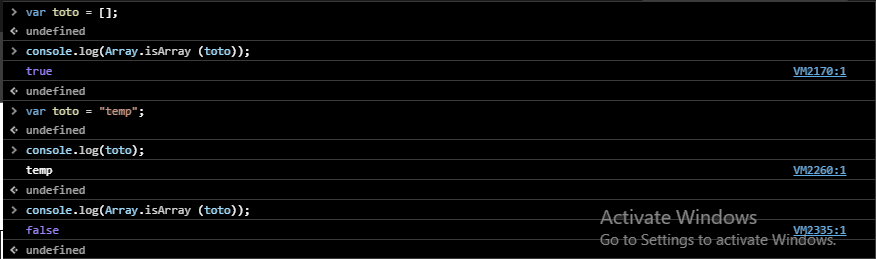
Methods are:

1. **Array.isArray(value);**

This method is used to check any value (primitive | special | composite) if it is an array value or not. If it array value then “**true**” will return, if not “**false**” is return.

**Note:** this method is used instead of typeof as the last one return object data type not array. (As you know that array is a special object)

**Example:**

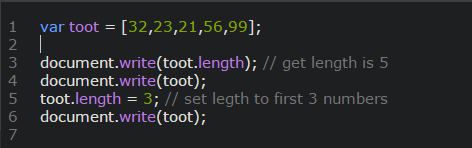


1. **Arrayname.length**

This method is used to **get** & **set** number of elements present in a specific array.

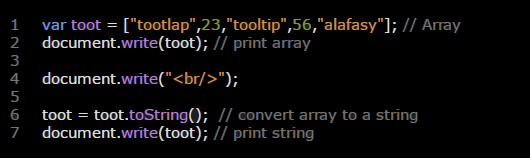
* 1. **Get** **syntax**: Arrayname.length
  2. **Set** **syntax**: Arrayname.length = number;

**Example:**



1. **Arrayname = Arrayname,toString();**

The toString() method is used for converting an array to string. It returns the string containing the specified array elements. By default commas separate these elements, and the string does not affect the original array.



1. **identifier.toLocaleString(“language code- country code”);**

This method is used to converts a Date object to a string.

**Note:**

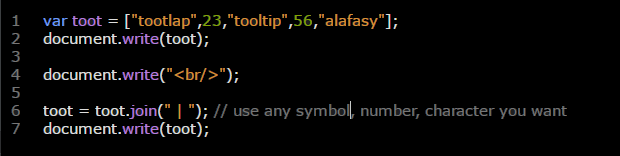
* 1. The value of identifier should be new Date();
  2. Language code should be written with lowercases while country code should be written with capital cases.
  3. Language and country code” is an optional so if you don’t determine it, then local location setting will be determined.



1. **Arrayname.join();**

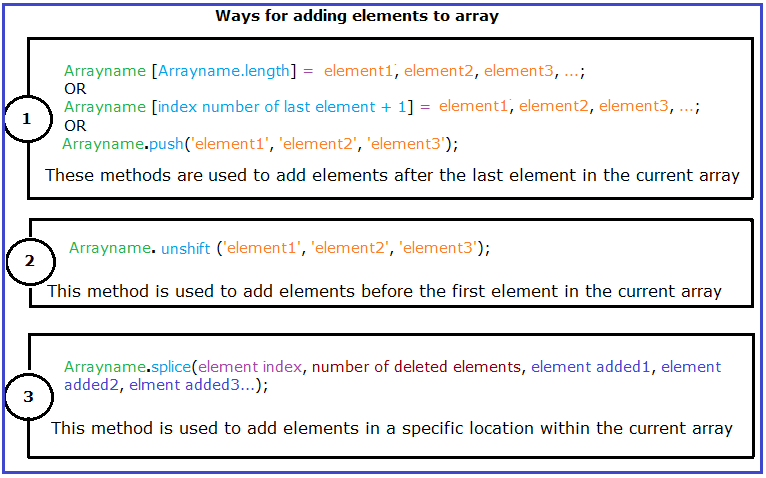
This method is used to return a new string by concatenating all of the elements in an array separated by commas or a specified **separator string**. If the array has only one item, then that item will be returned without using the separator.

**Note:** by default elements array were separated by commas. To change these commas add any symbol, character, number you want between quotes inside the curvy brackets (“”);

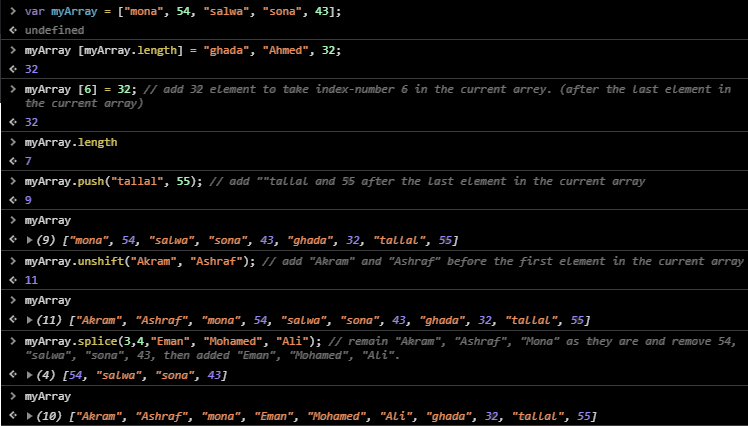


1. **Adding elements to array**

There are 3 ways to add elements to array. These ways are:

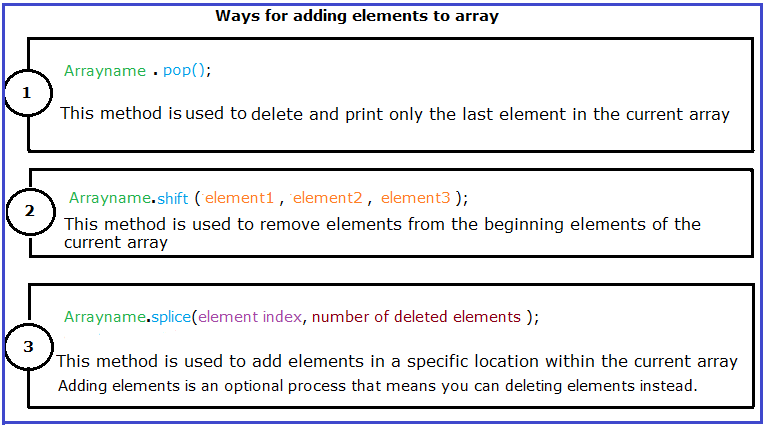


**Example**:

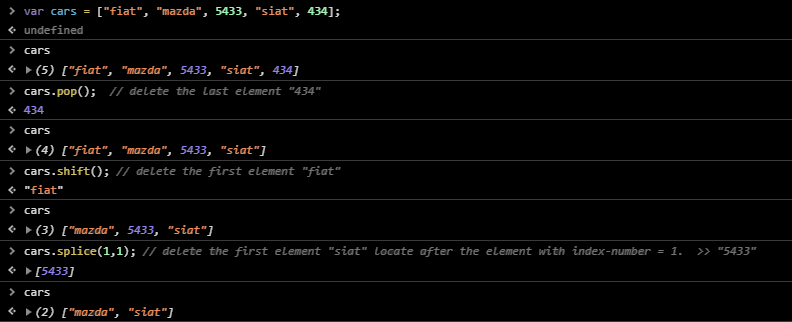


1. **Removing elements from array**

There are 3 ways to remove elements from array. These ways are:



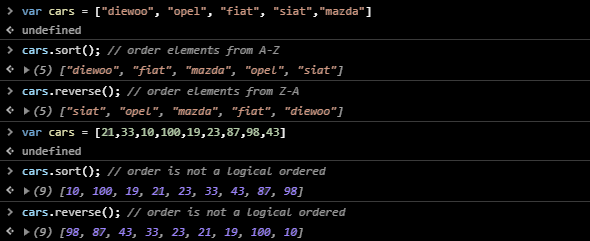
**Example:**



1. **Arrayname.sort(compareFn); and Arrayname.reverse();**

These methods are used to order elements alphabetically (Unicode of characters) in the current array, where Arrayname.sort(); ordered elements from A-Z while Arrayname.reverse(); ordered them from Z-A.

**Note:** These methods are not a good chose to ordered numbers; as the ordering depending on Unicode of numbers.



To know ordering numbers and know how this method works visit[..\..\..\Compare function.docx](../../../Compare%20function.docx)

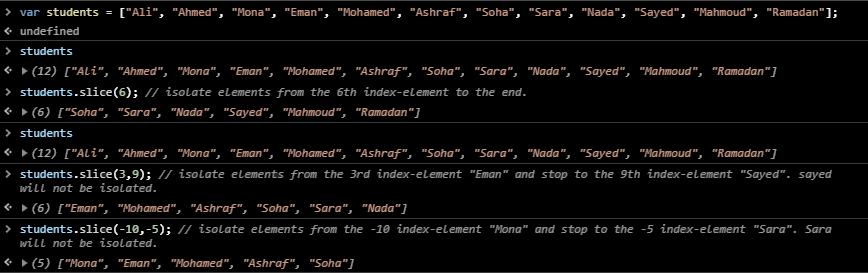
1. **Arrayname.slice(start index (required) – end index (optional));**

This method is used to separate | isolate some elements in a specific array.

**Note:**

* Always start index-number should be **smaller** than End index-number.
* This method doesn’t change the main array.

**Example:**



1. **Arrayname1.concat(Arrayname2, Arrayname3,….);**

This method is used to concatenate Arrays with each other. Where Arrayname1 will print first then Arrayname2, then Arrayname3…

This method doesn’t change the main array.

**Example:**



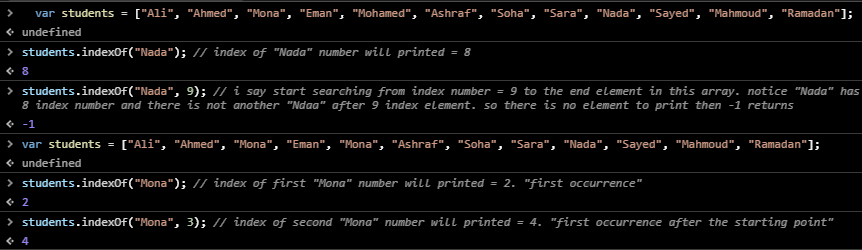
1. **Arrayname.indexOf(element, starting point);**

This method is used to search for an element in the array, and returns its position or returns -1 if the element is not found.

* Element in this syntax referee to the element that we need to search for.
* Starting point is an optional setting and it used to start the searching from a specific point. If you don’t determine starting point then searching will be from the **beginning** of the array to the **end** of it.

If the element is present more than once, the indexOf method returns the position of the **first** occurrence.

**Example:**



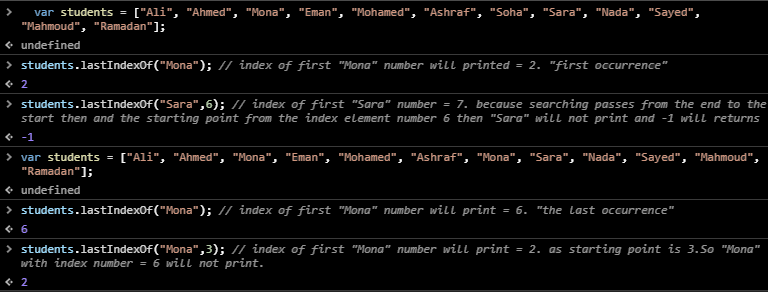
1. **Arrayname.lastIndexOf(element, starting point);**

This method is also used to search for an element in a specific array, and returns its position or returns -1 if the element is not found.

* Element referee to the element that we need to search for.
* Starting point is an optional setting and it used to start the searching from a specific point. If you don’t determine starting point then searching will be from the **ending** of the array to the **beginning** of it.

If the element is present more than once, the lastIndexOf method returns the position of the **last** occurrence.

**Example:**

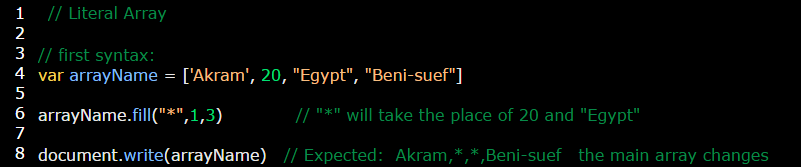


1. **Arrayname.fill(value, S.T, E.D);**

This method is used to replace current elements in the array with new elements. Where:

* Value referee to the new element.
* S.P referee to from which index-number should filling process start.
* E.D referee to where should I stop filling process. The E.D index-number will not be filled with the new element.

**Example:**

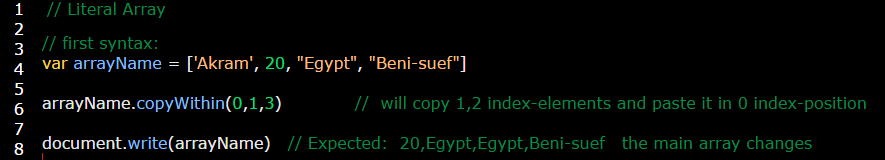


1. **Arrayname.copyWithin(target, S.P, E.P);**

This method is used to take a copy of the current array and paste it in a specific position inside the same array. Where:

* Target: referee to pasting position.
* S.P: referee to where I will start coping process.
* E.P: referee to where I will end coping process.

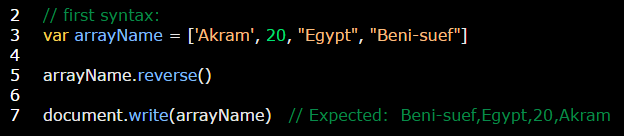
**Example:**

****

1. **Arrayname.reverse();**

This method is used to reverse elements of the array. This method changes the main array.

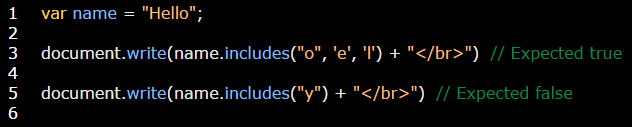
**Example:**



1. **Arrayname.includes(element1,element2,element3);**

This method is used to check the presence of a specific element in the current array. If this element present ‘true’ will return; else ‘false’ returns.

**Example:**

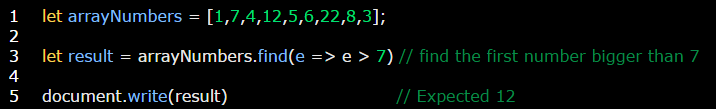


1. **Arrayname.find(testing function);**

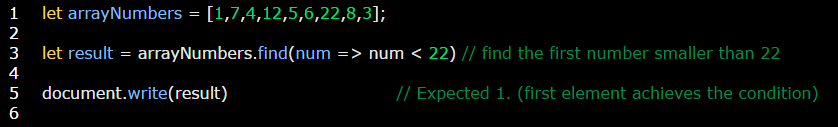
This method is used to find the presence of specific element in the array. Find process done by [testing function](../../../testing%20function.docx).

**Note:** the return of find method is one element only (the first element achieves the condition). If there is no element achieves the condition **undefined** returns.

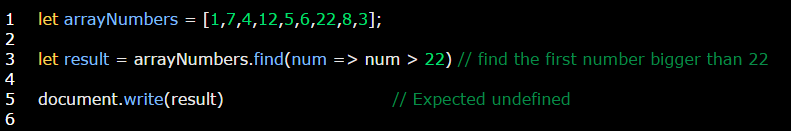
**Example:**



**Example:**



**Example:**



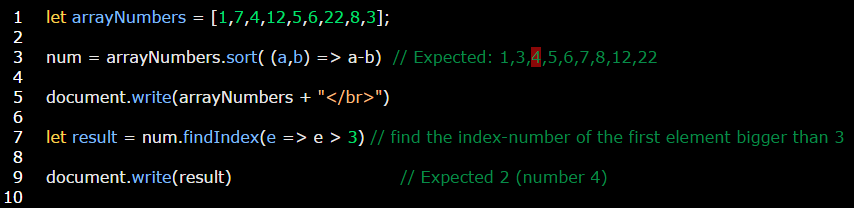
1. **Arrayname.findIndex(testing function);**

This method is used to find the index-number of a specific element in the array. FindIndex process done by [testing function](../../../testing%20function.docx).

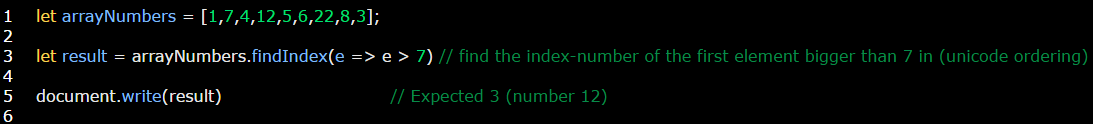
**Note:**

* The return of findIndex method is index-number of one element only (the first element achieves the condition). If there is no element achieves the condition **-1** returns.
* If array elements are numbers then it’s better to ordered them, to get a correct result. As this methods depend on Unicode of elements.

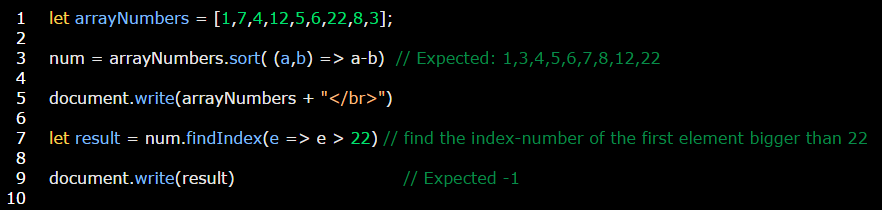
**Example:**



**Example:**



**Example:**



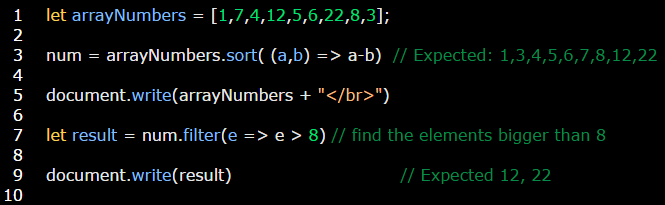
1. **Arrayname.filter(testing function);**

This method is used to find specific element**s** in the array. Filtering process done by [testing function](../../../testing%20function.docx).

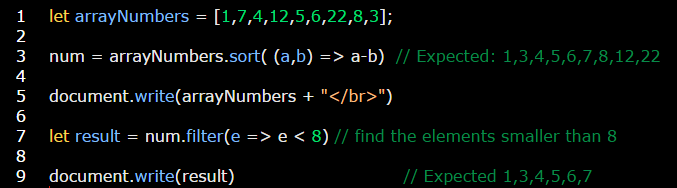
**Note:**

* The return of filter method is **all** elements that achieve the condition. If there is no element achieves the condition **[]** returns.

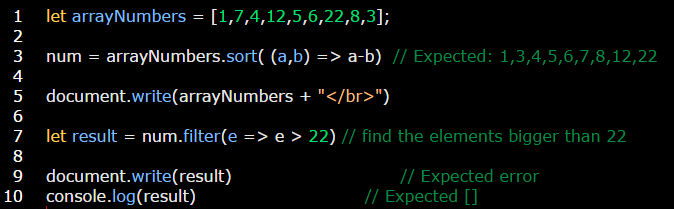
**Example:**



**Example:**



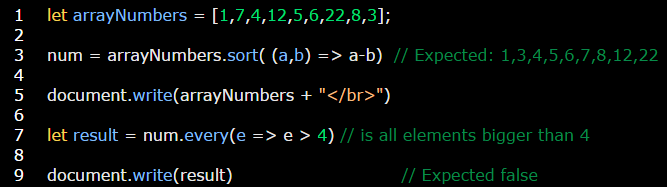
**Example:**



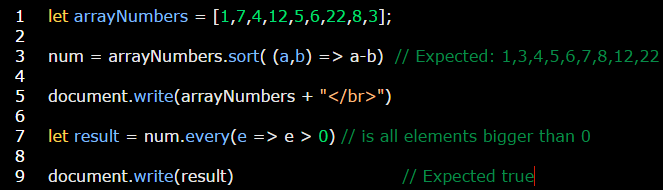
1. **Arrayname.every(testing function);**

This method is used to check are **all** elements in the array achieving the condition or not? If they are return **true** if not return **false**.

**Example:**



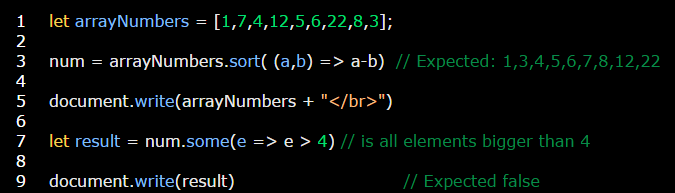
**Example:**



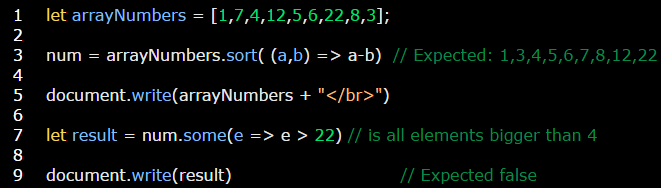
1. **Arrayname.some(testing function);**

This method is used to check is at least **one** element in the array achieving the condition or not? If they are return **true** if not return **false**.

**Example:**



**Example:**



**Interview question: how to get “olleH” from “Hello” without changing string type?**

Answer:

