Array Methods

JavaScript Array length

The length property returns the length (size) of an array:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let size = fruits.length;
```

JavaScript Array toString()

The JavaScript method toString() converts an array to a string of (comma separated) array values.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
```

Banana,Orange,Apple,Mango

Join() Method

The join() method also joins all array elements into a string.

It behaves just like toString(), but in addition you can specify the separator:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
```

Banana * Orange * Apple * Mango

Popping and Pushing

When you work with arrays, it is easy to remove elements and add new elements.

This is what popping and pushing is:

Popping items out of an array, or pushing items into an array.

JavaScript Array pop()

The pop() method removes the last element from an array:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();
```

The pop() method returns the value that was "popped out":

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fruit = fruits.pop();
```

JavaScript Array push()

The push() method adds a new element to an array (at the end):

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Kiwi");
```

The push() method returns the new array length:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let length = fruits.push("Kiwi");
```

Shifting Elements

Shifting is equivalent to popping, but it works on the first element instead of the last.

JavaScript Array shift()

The shift() method removes the first array element and "shifts" all other elements to a lower index.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();
```

The shift() method returns the value that was "shifted out":

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fruit = fruits.shift();
```

JavaScript Array unshift()

The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");
```

The unshift() method returns the new array length:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");
```

Comparing and Contrasting: push() and unshift()

Both push() and unshift() are used to add elements to the array but push() appends the element to the array while unshift() prepends the element to the array.

On a side note
One more way to append an element to an array is using the length property:

The length property provides an easy way to append a new element to an array:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Kiwi";
```

Changing Elements

Array elements are accessed using their index number:

Array indexes start with 0:

[0] is the first array element
[1] is the second
[2] is the third ...

const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[0] = "Kiwi";

JavaScript Array length

The length property provides an easy way to append a new element to an array:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Kiwi";
```

JavaScript Array delete()

Warning !

Array elements can be deleted using the JavaScript operator delete.

Using delete leaves undefined holes in the array.

Use pop() or shift() instead.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
delete fruits[0];
```

Merging (Concatenating) Arrays

The concat() method creates a new array by merging (concatenating) existing arrays:

```
const myGirls = ["Cecilie", "Lone"];
const myBoys = ["Emil", "Tobias", "Linus"];

const myChildren = myGirls.concat(myBoys);
```

The concat() method does not change the existing arrays. It always returns a new array.

The concat() method can take any number of array arguments:

Example (Merging Three Arrays)
```
const arr1 = ["Cecilie", "Lone"];
const arr2 = ["Emil", "Tobias", "Linus"];
const arr3 = ["Robin", "Morgan"];
const myChildren = arr1.concat(arr2, arr3);
```

The concat() method can also take strings as arguments:
Example (Merging an Array with Values)
```
const arr1 = ["Emil", "Tobias", "Linus"];
const myChildren = arr1.concat("Peter");
```

Flattening an Array

Flattening an array is the process of reducing the dimensionality of an array.

The flat() method creates a new array with sub-array elements concatenated to a specified depth.

```
const myArr = [[1,2],[3,4],[5,6]];
const newArr = myArr.flat();
```

Splicing and Slicing Arrays

The splice() method adds new items to an array.

The slice() method slices out a piece of an array.

JavaScript Array splice()

The splice() method can be used to add new items to an array:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 0, "Lemon", "Kiwi");
// Fruits now becomes: [ "Banana", "Orange", "Lemon", "Kiwi", "Apple", "Mango" ]
```

The first parameter (2) defines the position where new elements should be added (spliced in).

The second parameter (0) defines how many elements should be removed.

The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be added.

The splice() method returns an array with the deleted items:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 2, "Lemon", "Kiwi");
```

Original Array:
Banana,Orange,Apple,Mango

New Array:
Banana,Orange,Lemon,Kiwi

Removed Items:
Apple,Mango

Using splice() to Remove Elements

With clever parameter setting, you can use splice() to remove elements without leaving "holes" in the array:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(0, 1);
```

The first parameter (0) defines the position where new elements should be added (spliced in).

The second parameter (1) defines how many elements should be removed.

The rest of the parameters are omitted. No new elements will be added.

JavaScript Array slice()

The slice() method slices out a piece of an array into a new array.

This example slices out a part of an array starting from array element 1 ("Orange"):
Example

```
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const citrus = fruits.slice(1);
```

The slice() method creates a new array.

The slice() method does not remove any elements from the source array.

This example slices out a part of an array starting from array element 3 ("Apple"):
Example

```
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const citrus = fruits.slice(3);
```

The slice() method can take two arguments like slice(1, 3).

The method then selects elements from the start argument, and up to (but not including) the end argument.

```
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const citrus = fruits.slice(1, 3);
```

If the end argument is omitted, like in the first examples, the slice() method slices out the rest of the array.