JavaScript

Arrays

MDN Web Docs says:

Arrays are list-like objects whose prototype has methods to perform traversal and mutation operations. Neither the length of a JavaScript array nor the types of its elements are fixed. Since an array's length can change at any time, and data can be stored at non-contiguous locations in the array, JavaScript arrays are not guaranteed to be dense; this depends on how the programmer chooses to use them. In general, these are convenient characteristics; but if these features are not desirable for your particular use, you might consider using typed arrays.

Arrays cannot use strings as element indexes (as in an associative array) but must use integers. Setting or accessing via non-integers using bracket notation (or dot notation) will not set or retrieve an element from the array list itself, but will set or access a variable associated with that array's object property collection. The array's object properties and list of array elements are separate, and the array's traversal and mutation operations cannot be applied to these named properties.

Array

```
const arr = ['🍎', '🍊', '🍇', '🍓', '🍒'];
console.log(arr);

> [ '🍎', '🍊', '🍇', '🍓', '🍒' ]
```

Get the length of an array

Returns the length of an array, starting from 0

```
const arr = ['🍎', '🍊', '🍇', '🍓', '🍒'];
console.log(arr.length);

> 5
```

Accessing array items

Using bracket syntax, you can select items from an array by index.

```
const arr = ['🍎', '🍊', '🍇', '🍓', '🍒'];
console.log(arr[2]);

> '🍇'
```

Add item to an array

Use the `push()` function to add an item to the end of an array. It returns the new length of the array.

```
const arr = ['🍎', '🍊', '🍇', '🍓', '🍒'];
console.log(arr.push('🥑'));

> 6

console.log(arr);

> [ '🍎', '🍊', '🍇', '🍓', '🍒', '🥑' ]
```

Use the `unshift()` function to add an item to the beginning of an array. It returns the new length of the array.

```
const arr = ['🍎', '🍊', '🍇', '🍓', '🍒'];
console.log(arr.unshift('🥑'));

> 6

console.log(arr);

> [ '🥑', '🍎', '🍊', '🍇', '🍓', '🍒' ]
```

Remove item from an array

Use the `pop()` function to remove an item from the end of an array. It returns the item it removed.

```
const arr = ['🍎', '🍊', '🍇', '🍓', '🍒'];
const removed = arr.pop();
console.log(removed)

> '🍒'

console.log(arr);

> [ '🍎', '🍊', '🍇', '🍓' ]
```

Use the `shift()` function to remove an item from the front of an array. It returns the item it removed.

```
const arr = ['🍎', '🍊', '🍇', '🍓', '🍒'];
const removed = arr.shift();
console.log(removed)

> '🍎'

console.log(arr);

> [ '🍊', '🍇', '🍓', '🍒' ]
```

Both the shift and pop functions mutate the array. If you are simply trying to select an item from an array, refer to 'accessing array items'

Find the location of an item in an array

Use the `indexOf()` function to find the index of an item in an array. If the item is not in the array, the function will return `-1`.

```
const arr = ['🍎', '🍊', '🍇', '🍓', '🍒'];
console.log(arr.indexOf('🍓'))

> 3
```

Iterate through an array

Use the `forEach()` or `map()` functions to iterate through an array.

```
const arr = ['🍎', '🍊', '🍇', '🍓', '🍒'];
arr.forEach((element, index) => {
  console.log(element, index);
})

> '🍎' 0
> '🍊' 1
> '🍇' 2
> '🍓' 3
> '🍒' 4
```

The map function us used regularly in UI libraries like React, but forEach is more common in most javascript apps.