ARRAYS

IN THIS LESSON

What are arrays?

Creating an array

Accessing array items

Modifying array items

Array property: length

Array methods:
pop(), push(), shift(), unshift()
join(), includes(), indexOf()

WHAT ARE ARRAYS?

Arrays are numerically indexed collections of values

Arrays can contain strings, numbers,
Booleans, even other arrays

Each item in an array is associated with a number based on its order of appearance in the array, starting with 0

CREATING AN ARRAY

To create an array, use the array literal syntax:

```
[item1, item2, item3, ...]
```

To use an array more than once, store the array literal in a variable, ex.:

```
const groceryList = ['eggs', 'coffee beans', 'salad'];
```

```
let luckyNumbers = [7, 23, 99, 11, 777];
```

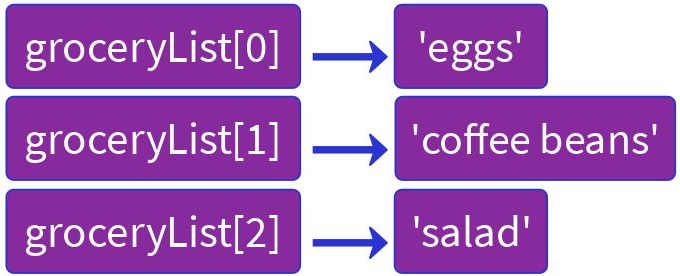
```
let anArray = [7, 'eggs', 99, 'salad', true];
```

ACCESSING ARRAY ITEMS

Arrays are zero-indexed: an array's index starts with 0

```
const groceryList = ['eggs', 'coffee beans', 'salad'];
```

array item	'eggs'	'coffee beans'	'salad'
array index	0	1	2



MODIFYING ARRAY ITEMS

Use the index with bracket notation & assignment operator

```
const groceryList = ['eggs', 'coffee beans', 'salad'];
```

array item	'eggs'	'coffee beans'	'salad'
array index	0	1	2

```
groceryList[0] = 'bananas';
groceryList[2] = 'soap';
```

```
const groceryList = ['bananas', 'coffee beans', 'soap'];
```

ARRAY PROPERTY: LENGTH

All arrays have a length property: a count of the total number of items inside the array - arrayName**.length**

```
const groceryList = ['bananas', 'coffee beans', 'soap'];
```

```
groceryList.length
```
3

This is the count of items, which starts at 1 - different from the array index, which starts at 0!

Though the **length** of this array is 3, the index is 0 through 2.

To access the last item in an array:

```
groceryList[groceryList.length-1]
```
'soap'

ARRAY METHODS

All arrays can access a large number of built-in JavaScript array functions, called array methods: - arrayName**.methodName()**

Some array methods have parameters, some don't

Most array methods return a value

Some array methods are mutators - they mutate the array; others are not, and it's important to know which are which

ARRAY METHODS: POP() & PUSH()

```
const groceryList = ['bananas', 'coffee beans', 'soap'];
```

pop() removes an item from the end of an array

```
groceryList.pop()
```

```
['bananas', 'coffee beans'];
```

push() adds one or more items to the end of an array

```
groceryList.push('milk')
```

```
['bananas', 'coffee beans', 'milk'];
```

Both are mutator methods, and they both return a value

The return value from **pop()** is the item that was removed

The return value from **push()** is the new length of the array

ARRAY METHODS: SHIFT() & UNSHIFT()

```
const groceryList = ['bananas', 'coffee beans', 'soap'];
```

shift() removes an item from the beginning of an array

```
groceryList.shift()
```

```
['coffee beans', 'soap'];
```

unshift() adds one or more items to the beginning of an array

```
groceryList.unshift('milk')
```

```
['milk', 'coffee beans', 'soap'];
```

Both are mutator methods, and they both return a value

The return value from **shift()** is the item that was removed

The return value from **unshift()** is the new length of the array

shift() and **unshift()** change the index of all other items in the array

ARRAY METHODS: JOIN()

```
const groceryList = ['bananas', 'coffee beans', 'soap'];
```

join() takes all items in an array and returns a string containing those items

It takes an optional string argument that will be used as the separator - if no argument is provided, a comma is used as the default separator

```
const groceries = groceryList.join();
```

```
groceries 'bananas,coffee beans,soap'
```

```
const groceries = groceryList.join('--');
```

```
groceries 'bananas--coffee beans--soap'
```

The **join()** method does not mutate the original array

ARRAY METHODS: INCLUDES()

```
const groceryList = ['bananas', 'coffee beans', 'soap'];
```

includes() is used to check if an item exists inside an array

```
const itemInArray = groceryList.includes('soap');
```

```
itemInArray  true
```

```
const itemInArray = groceryList.includes('tea');
```

```
itemInArray  false
```

It will return a Boolean value of **true** or **false**

includes() does not mutate the original array

ARRAY METHODS: INDEXOF()

```
const groceryList = ['bananas', 'coffee beans', 'soap'];
```

indexOf() also checks if an item is in an array, but instead of true or false, it returns the **index** of the item if it exists in the array

```
const itemIdx = groceryList.indexOf('soap');
```

itemIdx 2

If the item does not exist in the array, it will return the number **-1**

```
const itemIdx = groceryList.indexOf('tea');
```

itemIdx -1

Consider: Why does it return **-1** instead of 0?

indexOf() is also a non-mutating array method