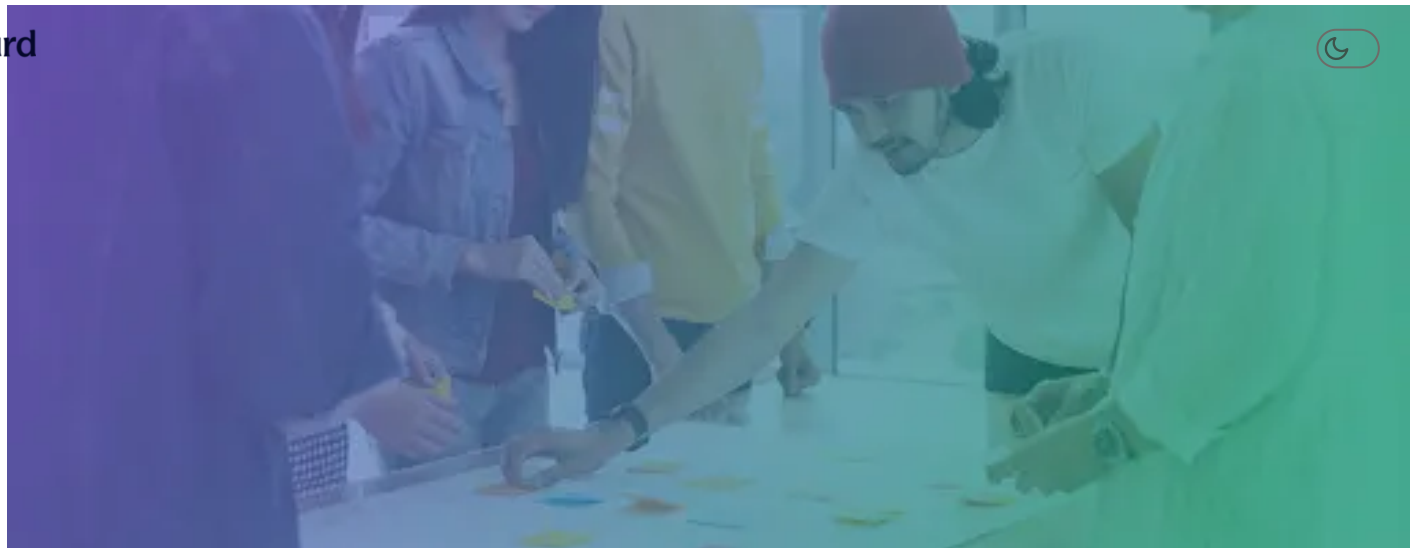
Springboard



Arrays

Goals

- Describe what arrays are, and why they're useful
- Create new arrays
- Access elements inside of an array
- Store array elements in variables
- Use common array methods

What Is An Array?

An array is an ordered collection of values.

Put another way, it's a list of stuff!

Why Should I Care?

Arrays are one of the most commonly-used data structures in JS

Arrays are found in most other programming languages, too

We very often want to store our data in an orderly way

A World Without Arrays

```
const evenNum = 2; const anotherEvenNum = 4; const
yetAnotherEvenNum = 6; const okayIGetItYouLikeEvenNums = 8;
```

- Lots of code
- No clear ordering
- Hard to modify the collection without adding more variables

A World With Arrays

```
const evenNums = [2, 4, 6, 8];
```

Alternatively, an array of values:

```
const evenNum = 2; const anotherEvenNum = 4; const
yetAnotherEvenNum = 6; const okayIGetItYouLikeEvenNums = 8;
```

```
const evenNums = [ evenNum, anotherEvenNum, yetAnotherEvenNum,
okayIGetItYouLikeEvenNums ];
```

Accessing Values In An Array

You can access elements in an array using brackets. The first element in the array is at index 0, the second element is at index 1, and so on.

```
const children = ["Blue", "Rumi", "Sir"]; children[0]; //
"Blue" children[1]; // "Rumi" children[2]; // "Sir"
children[3]; // undefined children[-1]; // undefined
```

Updating Array Elements

Elements in an array can be changed with a simple assignment.

```
const catToys = ["ball", "string", "xbox", "catnip"];
catToys[2] = "laser pointer"; catToys; // ["ball", "string",
"laser pointer", "catnip"]
```

Storing Values From An Array

Storing values by accessing them

```
const patterns = ["stripes", "polka dots", "plaid", "leopard
print"]; const stripes = patterns[0]; const polkaDots =
patterns[1]; const plaid = patterns[2]; const leopardPrint =
patterns[3];
```

Common Array Methods

push

- Adds values to the end of the array
- Returns the new array length

```
const planets = ["Venus", "Earth", "Mars"];
planets.push("Jupiter"); // 4 planets; // ["Venus", "Earth",
"Mars", "Jupiter"]
```

pop

- Removes values from the end of the array
- Returns the removed value
- If the array is empty, it returns undefined

```
const planets = ["Venus", "Earth", "Mars"]; planets.pop(); //
"Mars" planets; // ["Venus", "Earth"] planets.pop(); // "Earth"
planets; // ["Venus"] planets.pop(); // "Venus" planets; // []
planets.pop(); // undefined
```

unshift

- Adds values to the start of the array
- Returns the new array length

```
const planets = ["Venus", "Earth", "Mars"];
planets.unshift("Mercury"); // 4 planets; // ["Mercury",
"Venus", "Earth", "Mars"]
```

shift

- Removes values from the start of the array
- Returns the removed value
- If the array is empty, it returns undefined

```
const planets = ["Venus", "Earth", "Mars"]; planets.shift(); //
"Venus" planets; // ["Earth", "Mars"] planets.shift(); //
"Earth" planets; // ["Mars"] planets.shift(); // "Mars"
planets; // [] planets.shift(); // undefined
```

More array methods

concat

- Combines two arrays together
- Accepts arrays or comma separated values
- Creates a new array and returns it

```
[1, 2, 3].concat([4, 5, 6]); // [1, 2, 3, 4, 5, 6] [1, 2, 3].concat(4, 5, 6); // [1, 2, 3, 4, 5, 6] [1, 2, 3].concat([4, 5], 6); // [1, 2, 3, 4, 5, 6]
```

join

- Combines all array elements into a string
- Accepts a separator to put in between values
- Returns the created string

```
const commands = ["eat", "drink", "be merry"]; const doEverything = commands.join(" and "); // "eat and drink and be merry" const doSomething = commands.join(" or "); // "eat or drink or be merry" const excited = commands.join("! ") // "eat! drink! be merry"
```

indexOf

- Searches for values in an array
- Returns the index position of the first matching value
- If the value is not found, returns -1

```
const words = [ "it", "was", "the", "best", "of", "times", "it", "was", "the", "worst", "of", "times" ];
words.indexOf("best"); // 3 words.indexOf("times"); // 5
words.indexOf("hamburger"); // -1
```

includes

- Searches for values in an array
- If the value is found, returns true
- If the value is not found, returns false

```
const words = [ "it", "was", "the", "best", "of", "times",
"it", "was", "the", "worst", "of", "times" ];
words.includes("best"); // true words.includes("times"); //
true words.includes("hamburger"); // false
```

Which One Should I Use???

- Don't care about the position of the element you're looking for? - Use includes!
- Need the index of the left-most match? - Use indexOf!

reverse

- Reverses an array
- Returns the reversed array

```
const letters = ["a", "b", "c", "d"]; letters.reverse(); //
["d", "c", "b", "a"]
```

splice

- Super powerful method for adding and removing from an array
- Can add elements anywhere in the array
- Can remove elements from anywhere in the array
- Returns an array of the deleted elements

splice structure

```
array.splice(start, deleteCount, item1, item2, ...)
```

- Where to start removing items
- Number of items to remove

- First item to add at start index
- Second item to add and on...

splice

```
const instruments = ["guitar", "piano", "bass", "tuba",
"triangle", "drums"]; instruments.splice(1, 2); // ["piano",
"bass"] instruments; // ["guitar", "tuba", "triangle", "drums"]
instruments.splice(2, 0, "trombone", "didgeridoo"); // []
instruments; // ["guitar", "tuba", "trombone", "didgeridoo",
"triangle", "drums"] instruments.splice(3, 3, "saxophone"); //
["didgeridoo", "triangle", "drums"] instruments; // ["guitar",
"tuba", "trombone", "saxophone"]
```

sort

- Sorts elements in an array
- By default, sorts values alphabetically
- Returns the sorted array

```
const junkFoods = ["pizza", "ice cream", "potato chips",
"cookies"]; junkFoods.sort(); // ["cookies", "ice cream",
"pizza", "potato chips"]
```

Beware the default sort behavior!

```
const smallNums = [5, 8, 2, 1, 6, 3]; smallNums.sort(); // [1,
2, 3, 5, 6, 8]
```

This doesn't work...

```
const biggerNums = [9, 57, 3100, 84, 101, 12];
biggerNums.sort(); // [101, 12, 3100, 57, 84, 9]
```

slice

- Creates copies of subsets of an array
- Accepts an optional starting index (to begin the slice)
- Accepts an optional ending index (to end the slice)
- Returns the copied (sub)array

```
const days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"
weekend = days.slice(5); // ["Saturday", "Sunday"] const week = days
"Tuesday", "Wednesday", "Thursday", "Friday"] const midWeek = days.s
"Wednesday", "Thursday"] const weekCopy = days.slice(); // ["Monday"
"Thursday", "Friday", "Saturday", "Sunday"];
```

Using *const* with arrays

- values can change as long as the reference remains the same
- ***const*** ensures that the variable **can not** be re-assigned or redeclared

Nested Arrays

- Arrays can be nested inside other arrays
- Access values using another set of ***[]***

```
const gameBoard = [ ["O", null, "X"], ["O", null, "X"], ["X",
null, "O"] ] gameBoard[0][0] // "O" gameBoard[1][1] // null
gameBoard[2][0] // "X"
```