JavaScript

Arrays

Arrays

An array is a list of values. These values can be anything, including other arrays. They can be any length, including 0.

```
let people = ["Sam", "Zara", "Autumn", "Cadence", "Gale"]
let grades = [91, 83, 100, 87]
```

Array indexes

An index is a number that points to a position in an array. Indexes start at 0 and go to (length - 1), where length is the length of the array.

```
let people = ["Sam", "Zara", "Autumn", "Cadence", "Gale"]

people[0] // => "Sam"
people[1] // => "Zara"
people[4] // => "Gale"
people[people.length - 1] // => "Gale"
```

Array properties and methods

Arrays have a `.length` property that gives us the length of the array.

They also have many methods[1] to let us manipulate and interrogate them.

[1] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Looping over an array with `for`

We can use a `for` loop to get each index in an array and then use that index to get each member.

```
for (let i = 0; i < names.length; i++) {
  console.log("Hello, " + names[i] + "!")
}
```

Using `while` to loop over an array

```
let i = 0
while (i < names.length) {
  console.log(i, names[i])
  i = i + 1
}
```

for-of loops

For a simpler way to loop over an array and get each member, we can use a `for-of` loop.

```
for (let name of names) {
  console.log("Hello, " + name + "!")
}
```

As the loop runs, each member of `names` is assigned to `name` in order. We do not get the index in this case.

Adding to/removing from the ends of arrays

```
let students = ["Sam", "Val", "Landry"]
students.push("Charlie")
students // => ["Sam", "Val", "Landry", "Charlie"]

students.pop() // => "Charlie"
students // => ["Sam", "Val", "Landry"]

students.unshift("Logan")
students // => ["Logan", "Sam", "Val", "Landry"]

students.shift() // => "Logan"
students // => ["Sam", "Val", "Landry"]
```

Finding things in arrays

```
let students = ["Sam", "Val", "Landry"]

students.indexOf("Val") // => 1
students.indexOf("Landry") // => 2
students.indexOf("Logan") // => -1
```

Removing things from arrays

```
let students = ["Sam", "Val", "Landry"]
let idx = students.indexOf("Val")
students.splice(idx, 1) // => ["Val"]
students // => ["Sam", "Landry"]
```

Copying arrays

```
students.slice() // returns a new array
```

Common array actions

Three things we often want to do are:

- transform an array (create a new array of the same length with derived values)
- filter an array
- get one value from an array (sum, min, max, etc)

Let's see two techniques for each of these.

Transforming an array

1. Create a new array
2. Loop over the original array
3. For each element of the original array, transform it
4. Push the new transformed element into the new array

Transforming an array

Get word lengths

```
let words = ["tapeworm", "gnarly", "armoire"]
let wordLengths = []

for (let word of words) {
  wordLengths.push(word.length)
}

// wordLengths => [8, 6, 7]
```

Transforming an array

Is the score a passing grade?

```
let scores = [91, 54, 78, 39, 81]
let passingGrades = []

for (let score of scores) {
  passingGrades.push(score >= 60)
}

// passingGrades => [true, false, true, false, true]
```

Filtering an array

1. Create a new array
2. Loop over the original array
3. For each element of the original array, test to see if you want to keep it
4. If you want to keep it, push the element into the new array

Filtering an array

Get only words with length > 6

```
let words = ["tapeworm", "gnarly", "armoire"]
let filteredWords = []

for (let word of words) {
  if (word.length > 6) {
    filteredWords.push(word)
  }
}

// filteredWords => ["tapeworm", "armoire"]
```

Filtering an array

Keep only passing scores

```
let scores = [91, 54, 78, 39, 81]
let passingScores = []

for (let score of scores) {
  if (score >= 60) {
    passingScores.push(score)
  }
}

// passingScores => [91, 78, 81]
```

Getting one value (reducing) an array

1. Find a starting value. This depends on the problem. If you want a sum, start with 0.
2. Loop over your array
3. For each element of the array, compare to the current value. If you need to update the value, do that.

This is not very clear!

Reducing an array

Find the sum

```
let scores = [91, 54, 78, 39, 81]
let sum = 0

for (let score of scores) {
  sum += score
}

// sum => 343
```

Reducing an array

Find the shortest word

```
let words = ["tapeworm", "gnarly", "armoire"]
let shortestWord = null

for (let word of words) {
  if (shortestWord === null || word.length < shortestWord.length)
    shortestWord = word
}

// shortestWord = "gnarly"
```

Another technique for the above

Transforming, filtering, and reducing all can be done with array methods.

- `.map()`
- `.filter()`
- `.reduce()`

These methods take functions as arguments.

Pass in a function as an argument

In JavaScript, functions are another type of value. They can have names (via `function` or `let/const`) or they can be anonymous.

Anonymous functions

Use `function`, but leave the name out.

```
function (score) {
  return score > 60
}
```

Transforming an array

Get word lengths

```
let words = ["tapeworm", "gnarly", "armoire"]
let wordLengths = words.map(function(word) {
  return word.length
})
```

Note that `.map()` runs the loop for us! The function it takes as an argument (the callback function) takes the individual elements one at a time as its argument (`word`).

Filtering an array

```
let words = ["tapeworm", "gnarly", "armoire"]
let filteredWords = words.filter(function(word) {
  return word.length > 6
})

// filteredWords => ["tapeworm", "armoire"]
```

The filtering function should return true or false for each element. Elements which return true are kept.

Reducing an array

```
let scores = [91, 54, 78, 39, 81]
let sum = scores.reduce(function(total, score) {
  return total + score
}, 0)

// score => 343
```

Reducing an array

Note that `.reduce()` takes two arguments:

- a function that takes the current reduced value (also called the "accumulator") and the next array element as arguments
- the starting value (this is optional -- if you don't include it, the first array element is used as the accumulator, and the starting value will be the next item in the array)

Here is a demo that might help you visualize what is happening: http://reduce.surge.sh/

Reducing an array

```
let words = ["tapeworm", "gnarly", "armoire"]
let shortestWord = words.reduce(function(current, word) {
  if (word.length < current.length) {
    return word
  } else {
    return current
  }
})

// shortestWord = "gnarly"
```

We are not providing a starting value here, because we don't need to. It's simpler to use the first word as the value of "current" than to include 'null' as a starting value and then have to handle that case in our function.

Arrow functions

For simple anonymous functions, the arrow syntax is sometimes used. Curly braces are not needed and the return is implicit.

```
function (score) {
  return score > 60
}

// vs
(score) => score > 60

// or even
score => score > 60
```

Arrow function examples

```
let words = ["tapeworm", "gnarly", "armoire"]
let wordLengths = words.map(word => word.length)
let filteredWords = words.filter(word => word.length > 6)

let scores = [91, 54, 78, 39, 81]
let sum = scores.reduce((total, score) => total + score)
```