ARRAY ITERATION AND REDUCTION

Ultimate JavaScript arrays

INTRODUCTION

- Performing the same action on some or every element of the array is called iteration
- Taking an array and turning it into a single value is called reduction
- We will cover every built in way to iterate over arrays including…
  - Maps
  - Filters
  - For Each
- We will cover many ways to reduce arrays including…
- Reduce
- Every

WHY ITERATE OVER OR REDUCE ARRAYS

- Arrays often contain large amounts of information, from which it can be difficult to discern meaning
- Arrays can contain a variety of elements, some of which are not useful for what you need to do
- The elements in an array can be in incorrect, or inconsistent, form
- Array elements can be in the wrong order (next chapter)

ITERATING OVER ARRAY ELEMENTS WITH A FOR LOOP

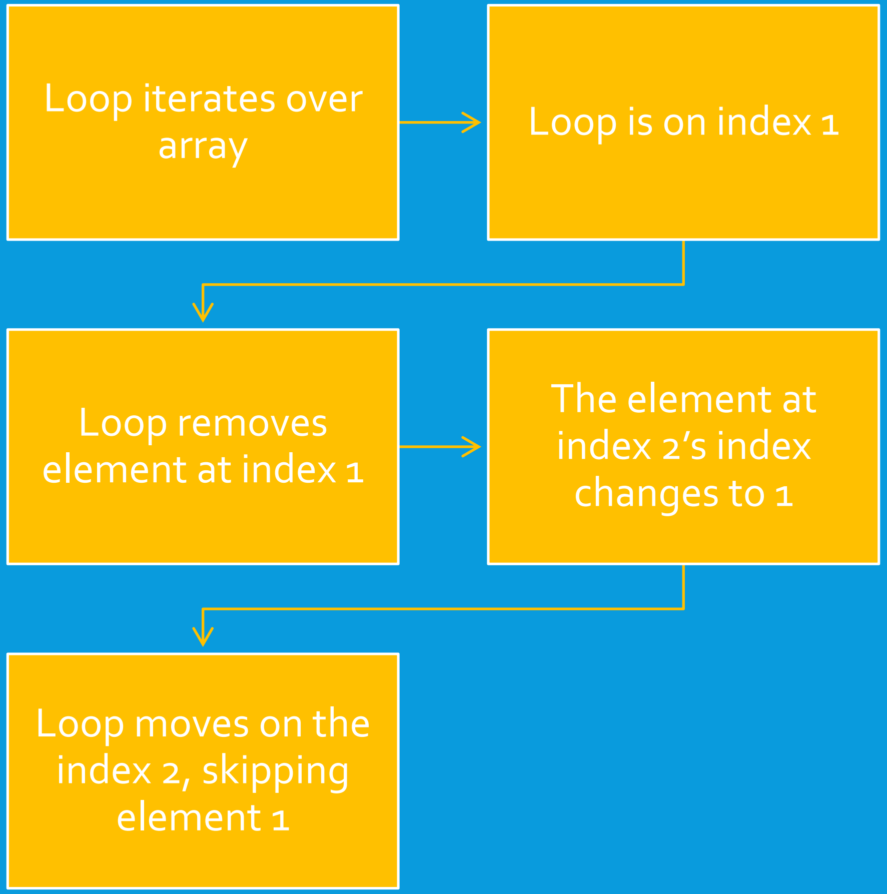
- For Loops are basic loops that are flexible but can be difficult to understand and verbose to write
- Called for loop because it uses the “for” keyword, and loops through all the elements of an array
- For loops can be used for any operation that needs to occur a set number of times, not just arrays

FOR EACH LOOPS

- Very similar to For Loops but ony work for arrays
- Less flexible and less useful for other purposes, but perfect for arrays
- A function will run once for each element of the array, and will be passed arguments corresponding to each element and its index

WHAT HAPPENS WHEN YOU CHANGE AN ARRAY WHILE LOOPING THROUGH IT?

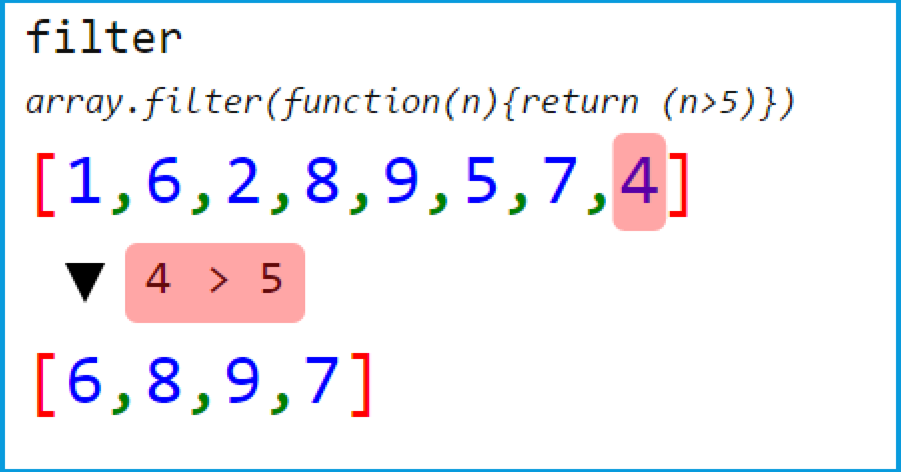
- Removing array elements while iterating over them can cause very confusing results
- This operation throws an error in many other languages but not JavaScript
- As a general rule, do not remove elements from arrays while iterating over them, use filters instead (see later video)
- Possible (but confusing) solution is to use backwards for loop


Loop iterates over array
Loop is on index 1
Loop removes element at index 1
The element at index 2's index changes to 1
Loop moves on the index 2, skipping element 1

LOOPING OVER ARRAYS WITH WHILE LOOPS

- Most brief loop to write
- **While loops can easily crash Chrome or a Node.js server**
- Can be difficult to understand or “grok”
- Not recommended for use outside of recreational coding challenges

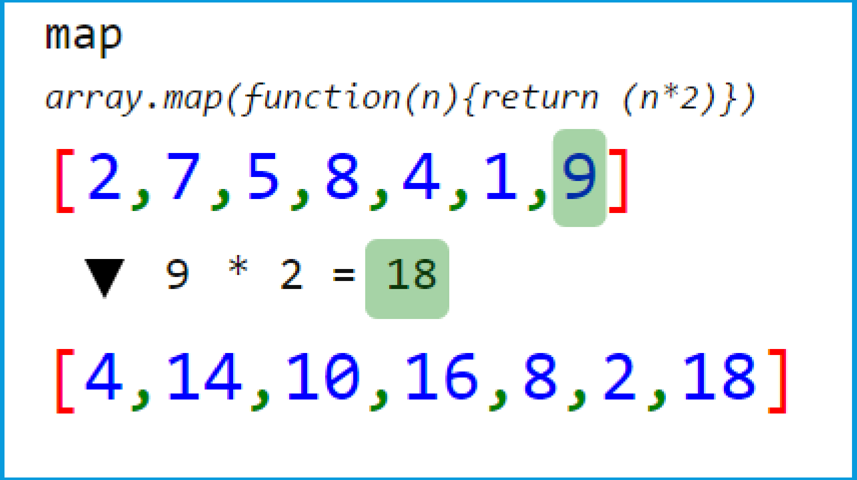
ARRAY FILTERS



```
filter
array.filter(function(n){return (n>5)})
[1,6,2,8,9,5,7,4]
▼ 4 > 5
[6,8,9,7]
```

- Used to weed out unwanted array elements
- An array filter is a safe way to remove multiple elements from an array at once
- Creates a copy of the original array with equal or fewer elements (original array is not changed)
- JavaScript have a built in filter method for this exact purpose
- Think of it like an air filter

ARRAY MAPS



```
map
array.map(function(n){return (n*2)})
[2,7,5,8,4,1,9]
 ▼ 9 * 2 = 18
[4,14,10,16,8,2,18]
```

- Used to transform each element of an array in the same way
- Can turn an array of objects in to an array of strings
- Like filter, creates a copy
- Maps always have the same number of elements in them as the original
- Think of it as an assembly line

ARRAY REDUCTION

- Reducing an array means taking all the values of an array to a single value
- A simple example is to reduce an array of numbers to the sum of all those numbers
- Reductions are very useful as they crystalize (sometimes thousands) of pieces of data into just one number or Boolean or string, etc.
- Think of it as taking a large pot of soup and boiling the excess away until only one serving remains at the bottom

REDUCTIONS: EVERY AND SOME

- Turns an entire array into a single true or false value
- The function passed to every or some which determines if an array element passes is called a *predicate*
- *.every()* only returns true if the predicate is true for every element
- *.some()* returns true if the predicate is true for one or more element

ARRAY INCLUDES

- Returns true if any element of the array matches the value that is passed
- Similar to *.some()*
- New to ES6
- Would have been called “contains” except for historical reasons

CHAINING ARRAY METHODS

```
[1,2,3,4]
   .map(a=>a*a)
   .reduce((a,b)=>a+b)
   //30
```

- Chaining array methods together can create complex operations like reductions or map-filters
- JavaScript is one of the best existing langauges for chaining array methods
- Recommended method for processing large amounts of data

CONCLUSION

- It is often necessary to reduce arrays to just one value – use a reduction for that
- While loops are dangerous to use,and removing elements from an array while looping through it is also not recommended
- JavaScript has a rich variety of built in map, filter and reduction functions
- Map, filter, includes, some, every and reduce are most useful built in features
- Knowledge can be applied directly to front-end code or database code on a Node.js platform