reduce

Download Demo Code <../js-array-methods-reduce-demo.zip>

Goals

- Understand what reduce does
- Use reduce to create new data structures

reduce

Whatever is returned from the callback function, becomes the new value of the accumulator!

- Accepts a callback function and an optional second parameter
- Iterates through an array
- Runs a callback on each value in the array
- The first parameter to the callback is either the first value in the array or the optional second parameter
- The first parameter to the callback is often called "accumulator"
- The returned value from the callback becomes the new value of accumulator

Let's Break It Down

```
let evens = [2,4,6,8,10];

evens.reduce(function(accumulator, nextValue){
  return accumulator + nextValue;
});

/*
  2
  6
  12
  20
  30
*/
```

Adding A Second Parameter

```
let evens = [2,4,6,8,10];

evens.reduce(function(accumulator, nextValue){
  return accumulator + nextValue;
},10);
```

```
/*
  12
  16
  22
  30
  40
*/
```

Let’s Try Something Else

```
let names = ['Maya', 'Tammy', 'Angela', 'Alexis'];

names.reduce(function(accumulator, nextValue){
  if(nextValue !== "Colt"){
    return accumulator += ' ' + nextValue;
  }
  return accumulator;
},'My friends are');

/*
  Here is what reduce will build up:

  'My friends are Maya'
  'My friends are Maya Tammy'
  'My friends are Maya Tammy Angela'

  With a final output of:

  'My friends are Maya Tammy Angela Alexis'
*/
```

When You Would Use Reduce

- It works for almost everything, but is sometimes overkill
- When you want to transform an array into another data structure

Recap

- reduce returns an accumulated value which is determined by the result of what is returned to each callback
- reduce begins with the first value in the array or with an optional second argument for the starting value