Understanding .filter, .map, & .reduce

Instructor: Marc Fisher

Arrays in Javascript

Let's briefly review arrays ...

- Built-in data type in Javascript to provide an ordered way of storing a list of things
- Let's review a few examples...

```
const classGrades = [99.5, 98, 83.5, 83, 82, 79];
const christmasWishList = [
  "PS5", "Macbook Pro", "iPhone12+"
];
const coinFlipAsHeads = [true, true, false];
const userAccounts = [
  { name: "Fred Davies", email: "fred.davies@gmail.com" },
  { name: "Josh Samuels", email: "josh.samuels@gmail.com" },
  { name: "Katy Barret", email: "katy.barret@gmail.com" },
];
const bucketedGrades = [
  [99.5, 98],
  [83.5, 83, 82],
  [79]
];
const badProgrammingExample = [null, undefined];
const mixedBag = [1, "random", null, false, [1, 2, 3]];
```

Arrays in Javascript

Let's review using arrays...

Declaring an array

```
let myArray = [];
let ourArray = new Array();
```

Accessing an ”element”, ie. ”thing” in an array

```
const classGrades = [99.5, 98, 83.5, 83, 82, 79];
const firstStudentsGrade = classGrades[0];
console.log(firstStudentsGrade); // 99.5
let lastStudentsGrade = classGrades[5];
console.log(lastStudentsGrade); // 79
lastStudentsGrade = classGrades[-1];
console.log(lastStudentsGrade); // 79
```

Knowing how many elements are in an array

```
console.log(classGrades.length); // 6
```

- Remember the first element in a Javascript array is at index (i.e. position) 0 not 1
- Negative indices are allowed in Javascript ... yes Javascript lets you do some crazy things!

Check out https://jsisweird.com/ makes even me feel like a n00b

What arrays actually are in Javascript?

You may be asking yourself ... `.length` looks a lot like how I learned to access a field in an object right?

```
const classGrades = [99.5, 98, 83.5, 83, 82, 79];
console.log(typeof classGrades);
```

```
"object"
```

(*lightbulb*) Because arrays are actually special objects in javascript, this means they can have fields pointing to values and functions

```
console.log(classGrades.length); // 6
console.log(classGrades["length"]); // 6
console.log(classGrades[0]); // 99.5
```

So lets fully console.log an array and find out what we see...

Quick Demo

Array’s Built-in Methods

- at
- concat
- filter
- map
- reduce
- ...

NOTE: If you are asking what is Prototype exactly, even though it is a fairly deep topic in Javascript, all you really need to know is it is basically a special way Javascript provides as built-in methods.

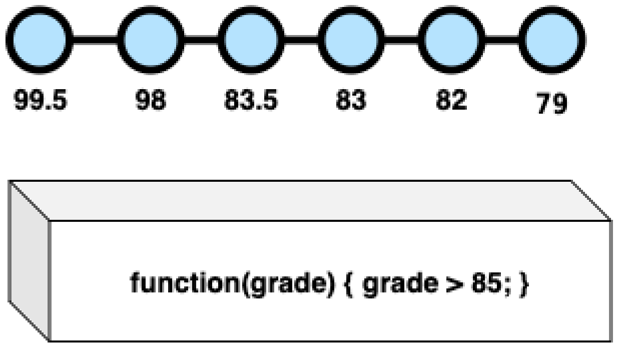
.filter

- It lets us only get the things in the array that meet our criteria
- Say we want only grades above 85% ...

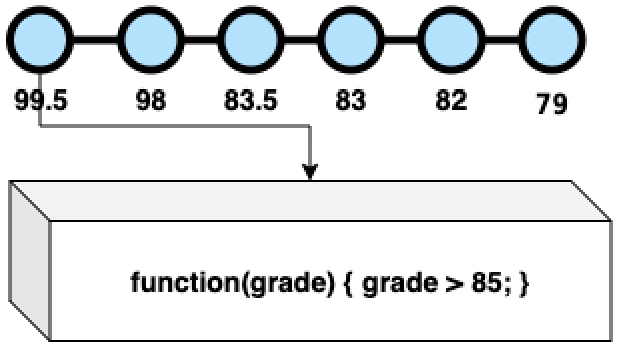
```
const classGrades = [99.5, 98, 83.5, 83, 82, 79];
const gradesAbove85 = classGrades.filter(function(grade) {
  return grade > 85;
});
console.log(gradesAbove85); // [99.5, 98]
```

.filter

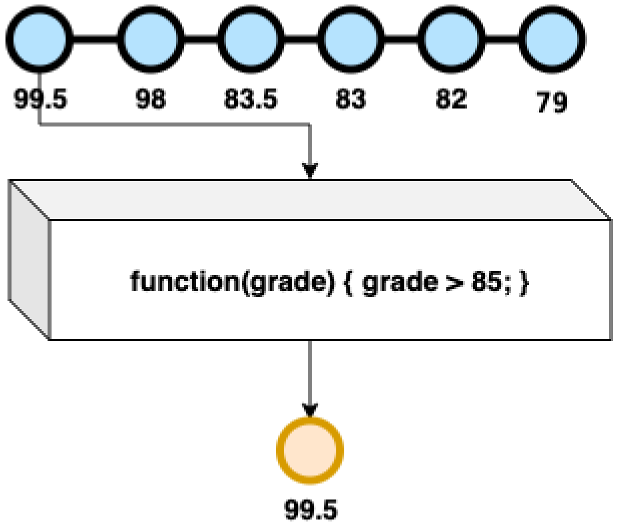
Lets understand what is happening in our example.



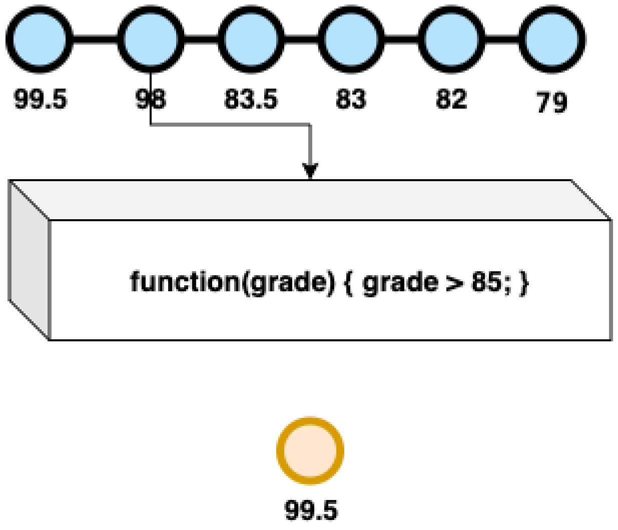
```
function(grade) { grade > 85; }
```

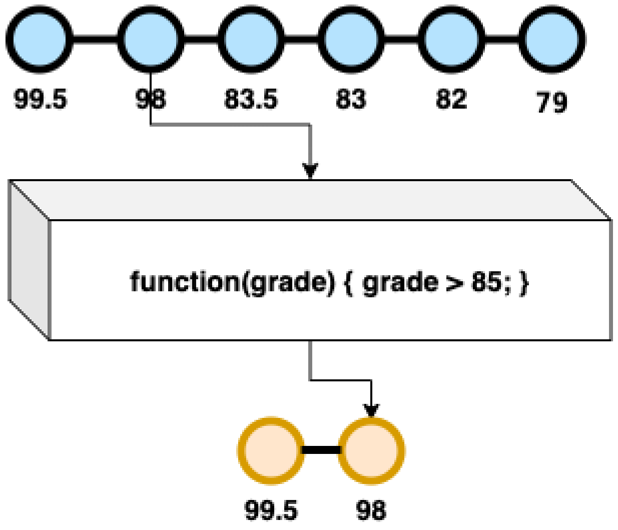


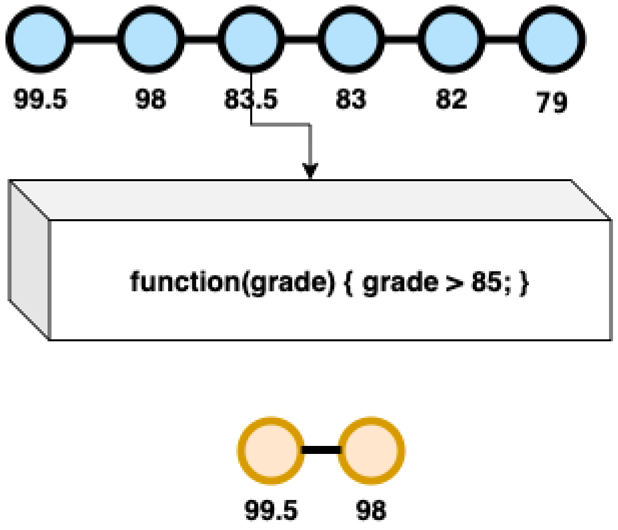
```
function(grade) { grade > 85; }
```

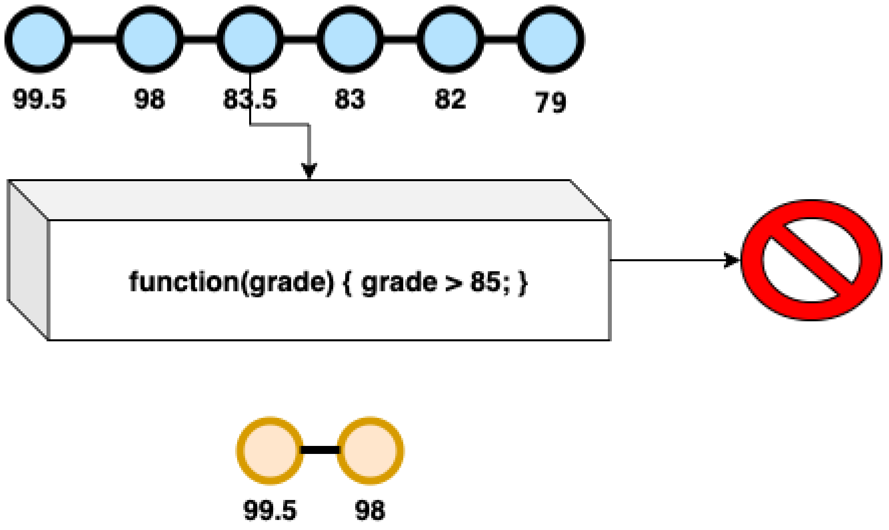


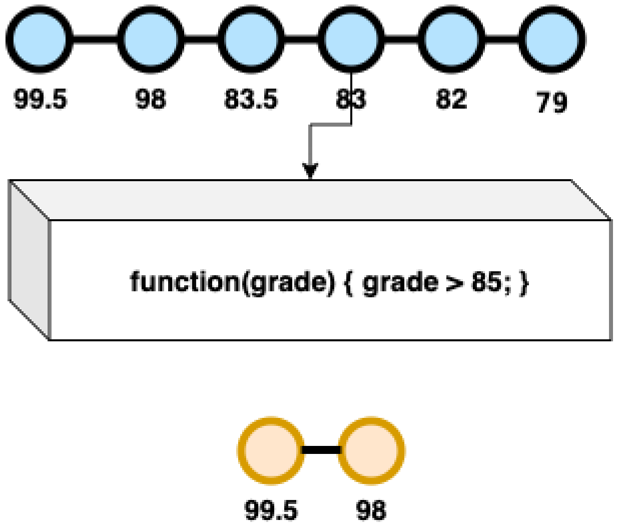
99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5

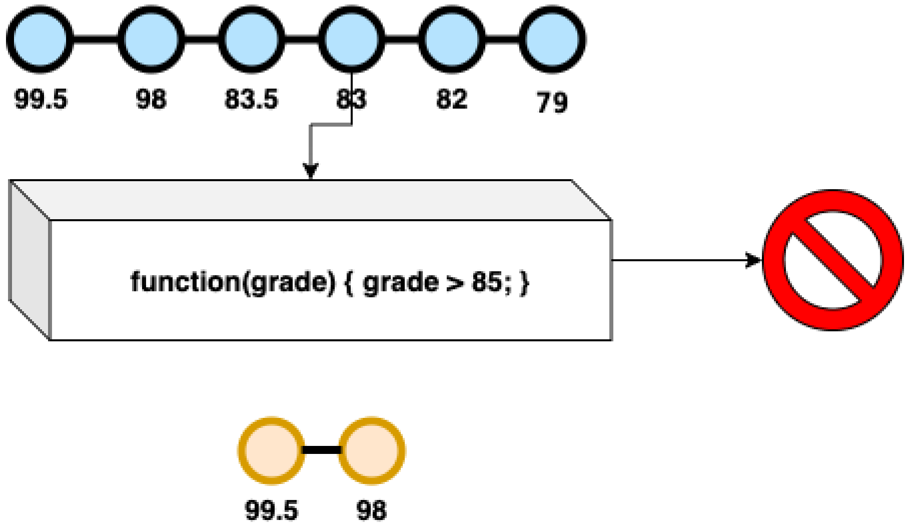

99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5

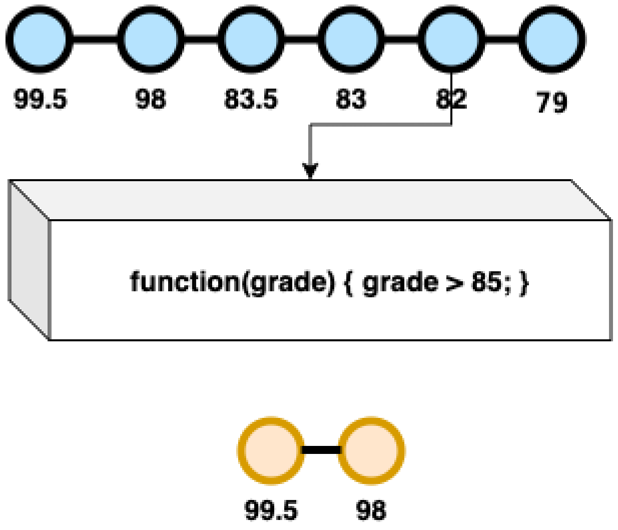

99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5
98

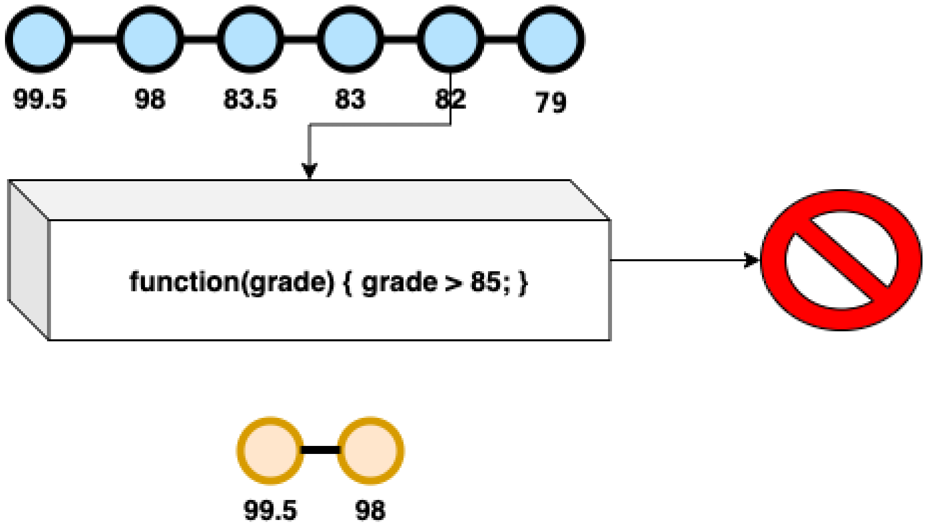

99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5
98

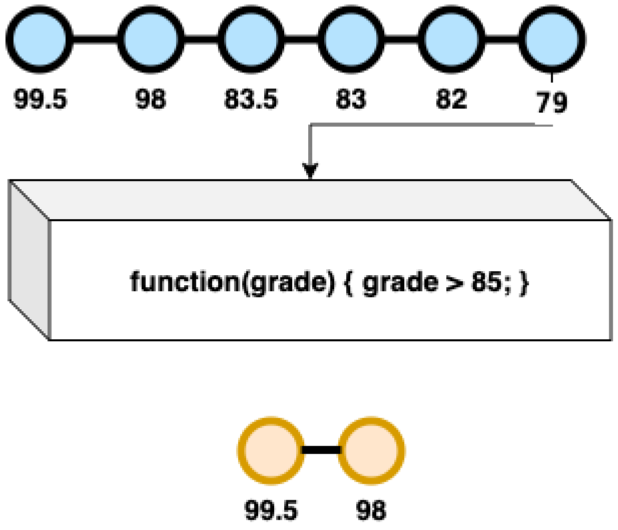

99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5
98


99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5
98

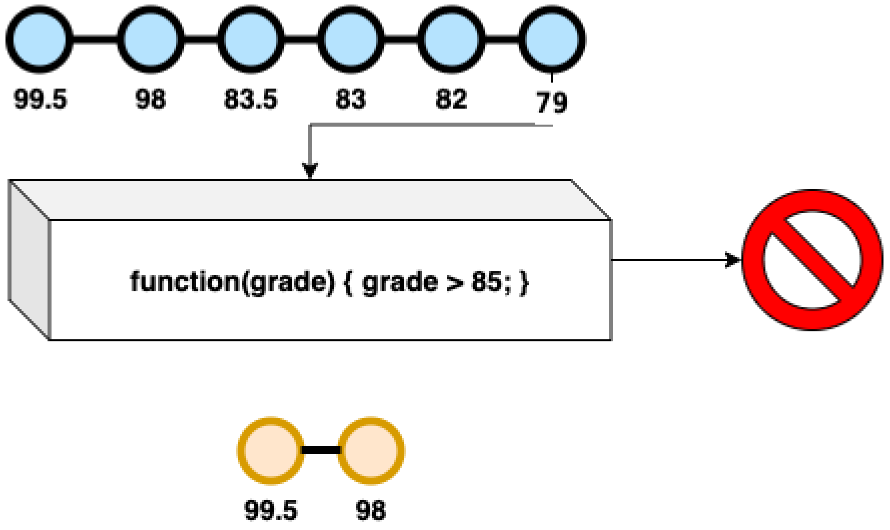

99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5
98

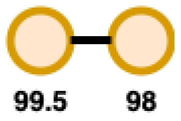

99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5
98


99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5
98



```
function(grade) { grade > 85; }
```


99.5
98
83.5
83
82
79
function(grade) { grade > 85; }
99.5
98


99.5
98

.filter

Re-writing without filter helps us see what is happening...

```
function isGradeGreaterThan85(grade) {
  return grade > 85;
}

const filteredItems = [];
if (isGradeGreaterThan85(99.5)) { filteredItems.push(99.5); }
if (isGradeGreaterThan85(98)) { filteredItems.push(98); }
if (isGradeGreaterThan85(83.5)) { filteredItems.push(83.5); }
if (isGradeGreaterThan85(83)) { filteredItems.push(83); }
if (isGradeGreaterThan85(82)) { filteredItems.push(82); }
if (isGradeGreaterThan85(79)) { filteredItems.push(79); }
return filteredItems;
```

.filter

```
function(value, currentIndex, originalArray)
```

```
The function/callback provides additional parameters if needed.
```

```
const classGrades = [99.5, 98, 83.5, 83, 82, 79];
const firstGradeAndGradesAbove85 = classGrades
  .filter((value, currentIndex, originalArray) => {
    return currentIndex > 0 && value > 85;
  });
```

```
When the function keyword is missing and you see => instead, it
is a fat array function. It is basically the same but with
"scope" differences.
```

.map

- Lets us return a new array based on the existing arrays data
- Say we an gave extra credit assignment and everyone in the class completed it so everyone gets 5 additional points ...

.map

```
const classGrades = [99.5, 98, 83.5, 83, 82, 79];
const gradesWithExtraCredit = classGrades
  .map((value, currentIndex, originalArray) => {
    return value + 5;
  });
// [104.5, 103, 88.5, 88, 87, 84]
```

.map

Once again we are **not** modifying the original array, rather returning a new array of same length.

```
function(value, currentIndex, originalArray)
```

The function/callback provides additional parameters if needed again.

.map

Lets explore one more example which is more common to the use of .map

```
//  We need to get the emails of everyone who opened our
// marketing campaign email to send them a follow-up
const successfulMarketingCampaignData = [
  { email: "john.smith@gmail.com", openedOn: "2021-11-05" },
  { email: "carry.johnson@gmail.com", openedOn: "2021-11-08" },
  { email: "sam.axe@gmail.com", openedOn: "2021-11-08" }
];
const emailAddresses = classGrades.map(function(user) {
  return user.email;
});
console.log(emailAddresses);
// ["john.smith@gmail.com", "carry.johnson@gmail.com", "sam.axe@g
```

BONUS: What if the function doesn't return anything?

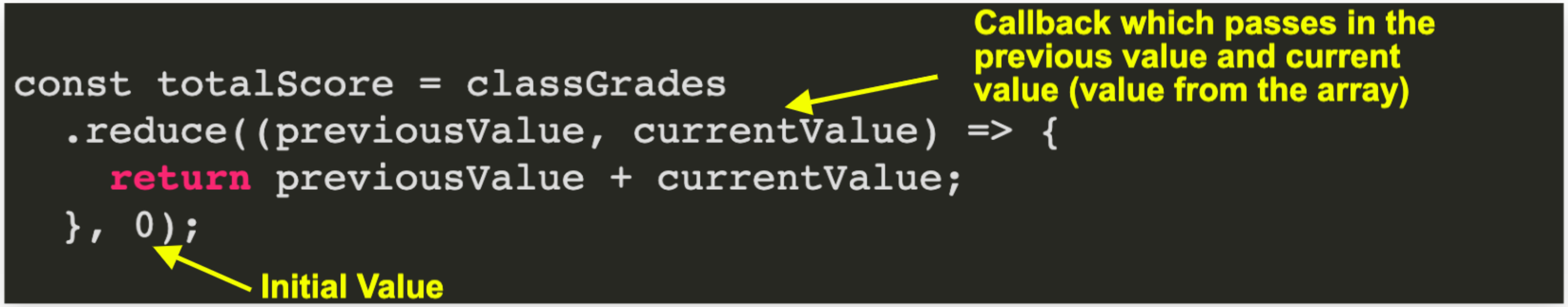
.reduce

- Reduce is different in that it will return only one "thing"
- Can return a number, string, object, array, literally any type in Javascript

```
const classGrades = [99.5, 98, 83.5, 83, 82, 79];
const totalScore = classGrades
  .reduce((previousValue, currentValue) => {
    return previousValue + currentValue;
  }, 0);
console.log("Class Average", totalScore / classGrades.length);
```

.reduce

Breaking this down into its parts...

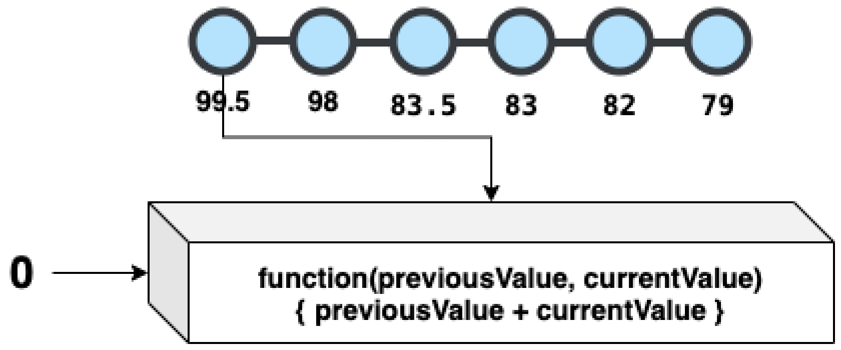


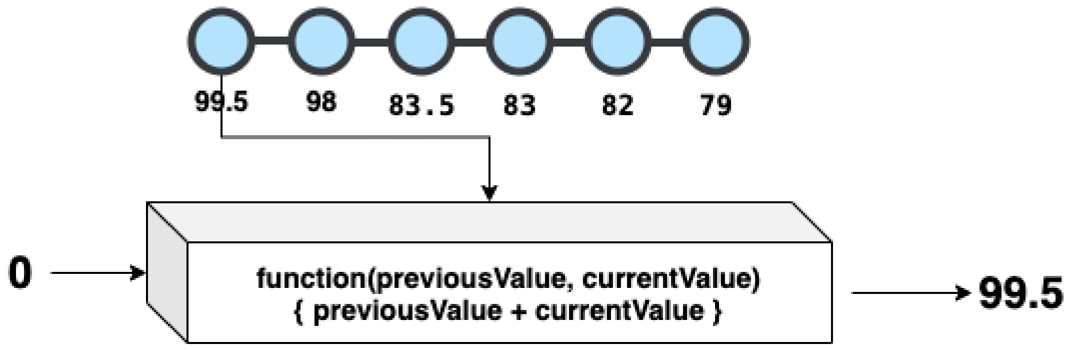
```
const totalScore = classGrades
  .reduce((previousValue, currentValue) => {
    return previousValue + currentValue;
  }, 0);
```

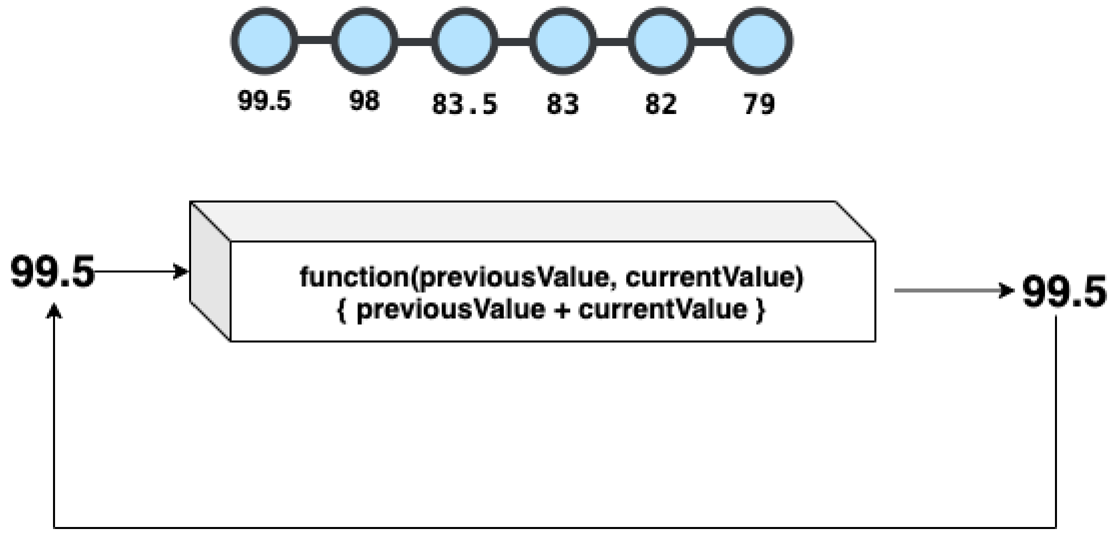
Callback which passes in the previous value and current value (value from the array)
Initial Value

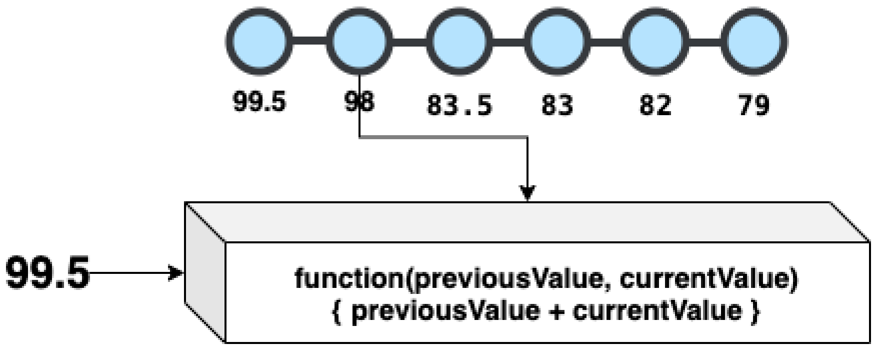
.reduce

Lets understand what is happening visually...

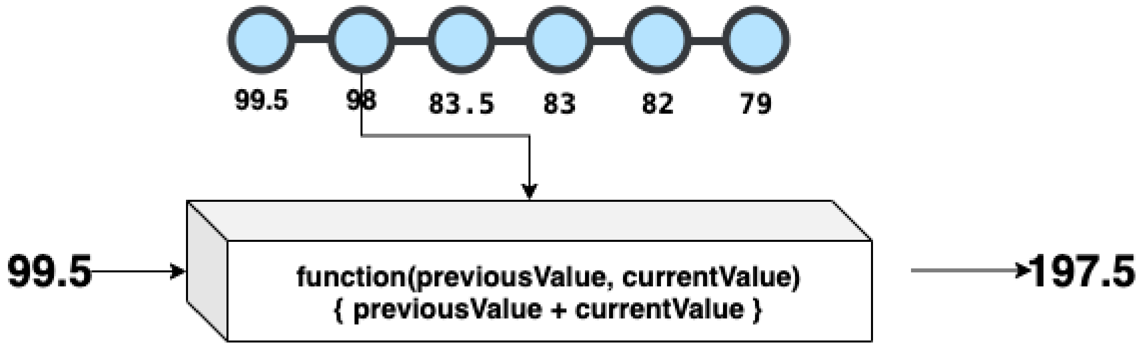

99.5
98
83.5
83
82
79
0
function(previousValue, currentValue)
{ previousValue + currentValue }


99.5
98
83.5
83
82
79
0
function(previousValue, currentValue)
{ previousValue + currentValue }
99.5

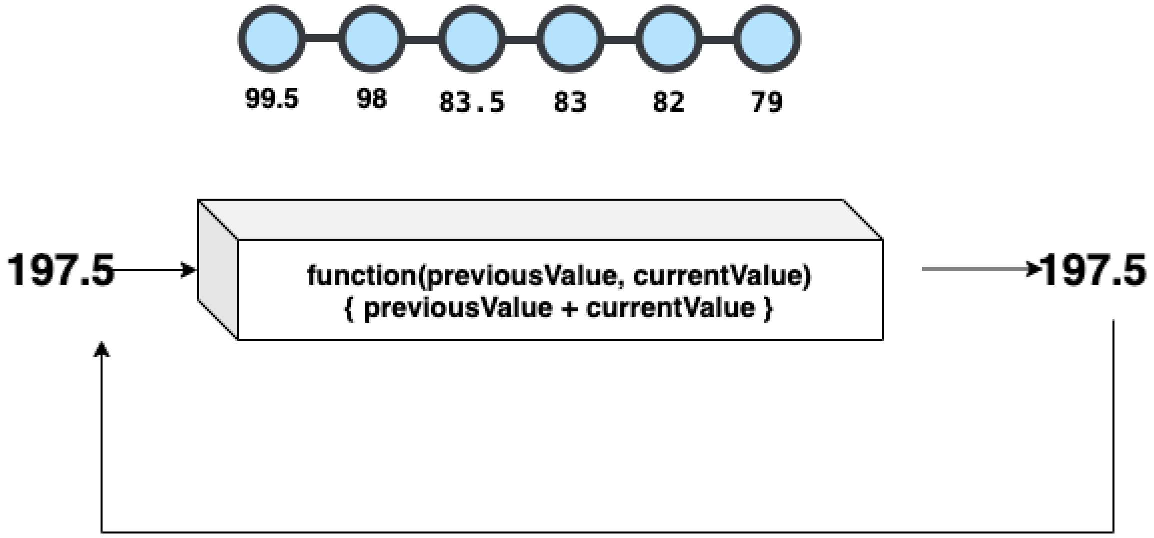

99.5
98
83.5
83
82
79
99.5
function(previousValue, currentValue)
{ previousValue + currentValue }
99.5

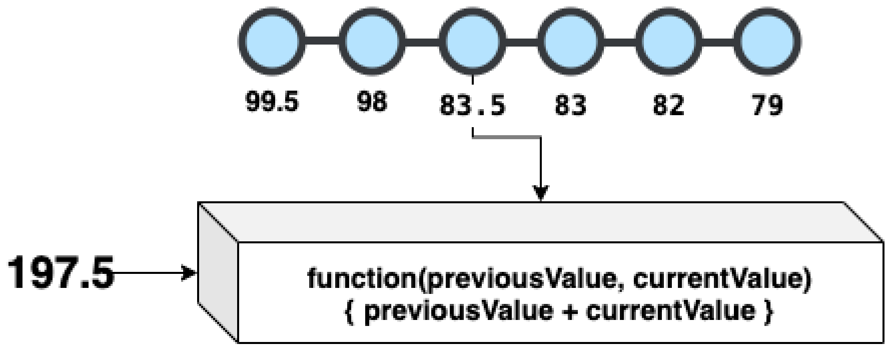


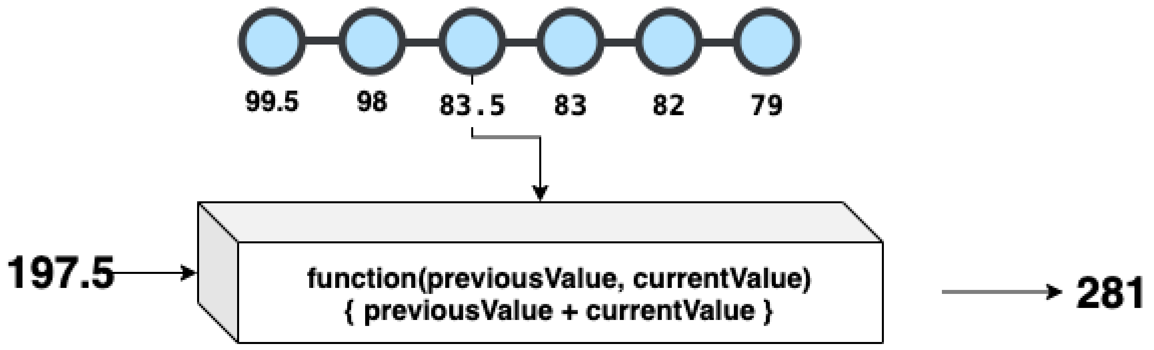
function(previousValue, currentValue)
{ previousValue + currentValue }

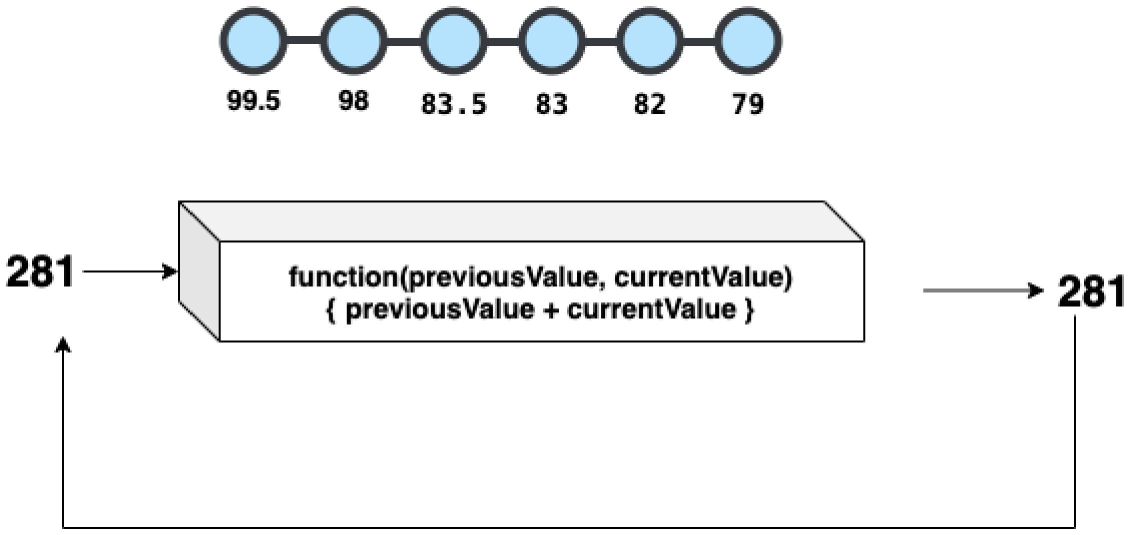


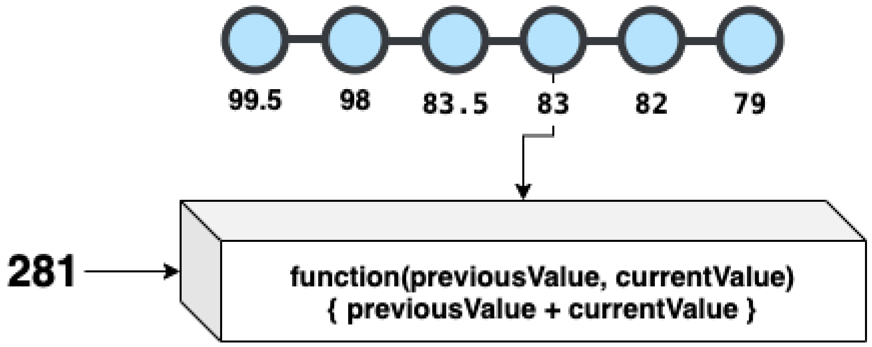
function(previousValue, currentValue)
{ previousValue + currentValue }

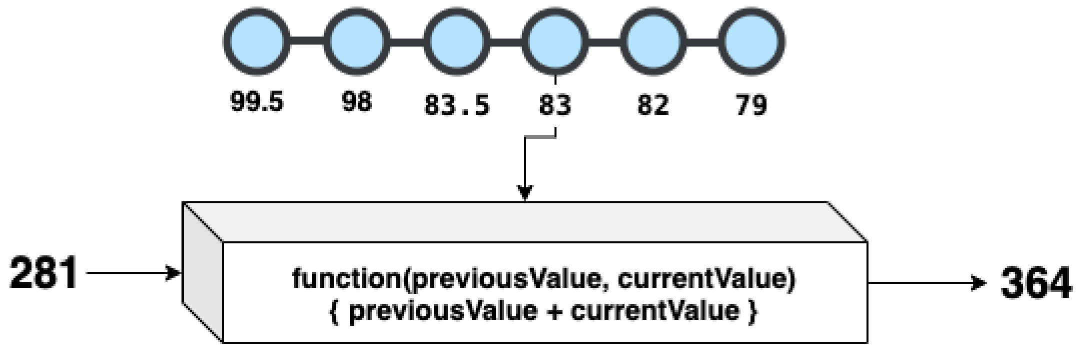

99.5
98
83.5
83
82
79
197.5
function(previousValue, currentValue)
{ previousValue + currentValue }
197.5

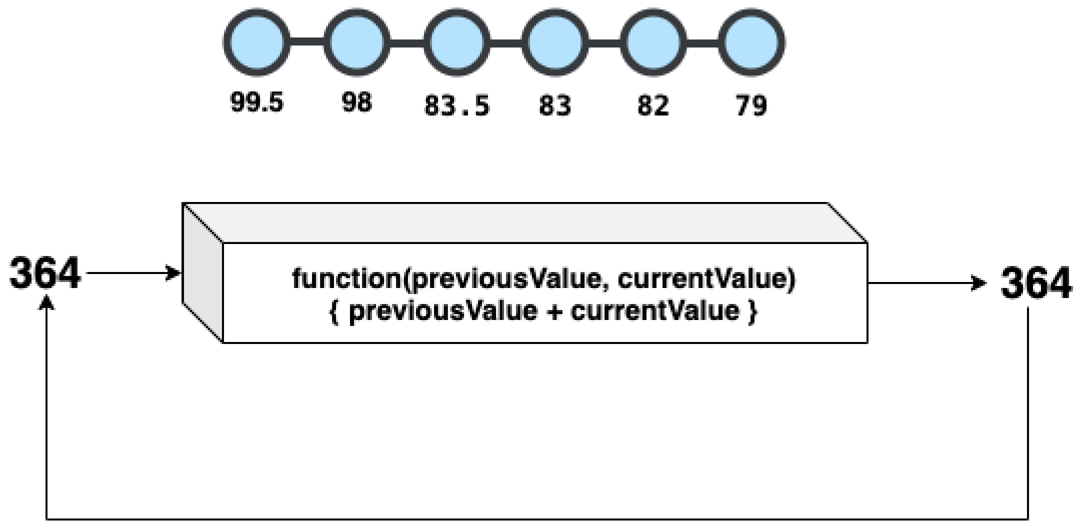

99.5
98
83.5
83
82
79
197.5
function(previousValue, currentValue)
{ previousValue + currentValue }

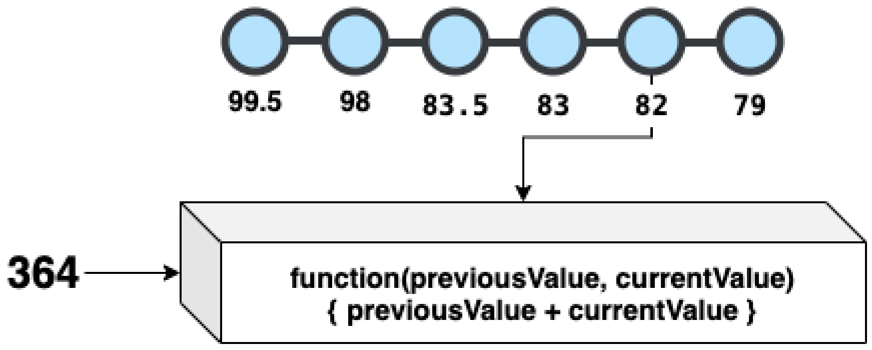

99.5
98
83.5
83
82
79
197.5
function(previousValue, currentValue)
{ previousValue + currentValue }
281

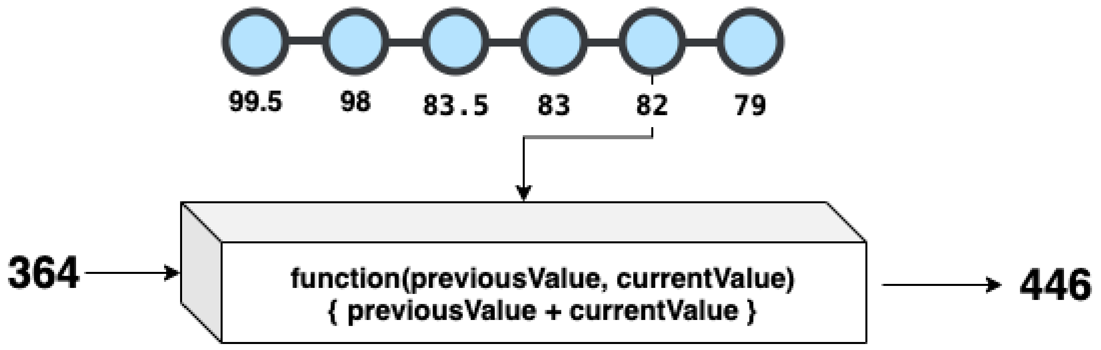

99.5
98
83.5
83
82
79
281
function(previousValue, currentValue)
{ previousValue + currentValue }
281

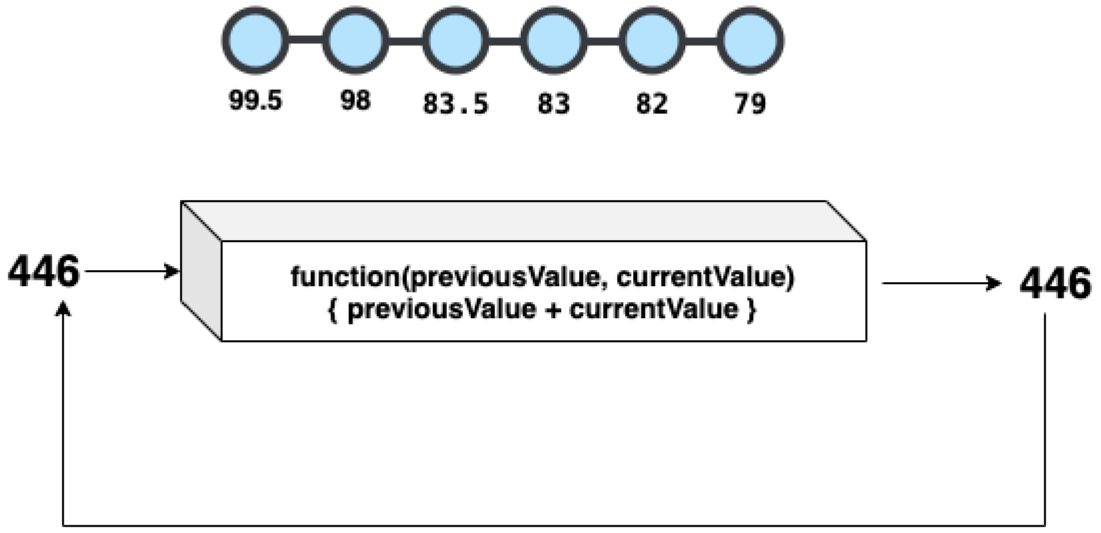

99.5
98
83.5
83
82
79
281
function(previousValue, currentValue)
{ previousValue + currentValue }

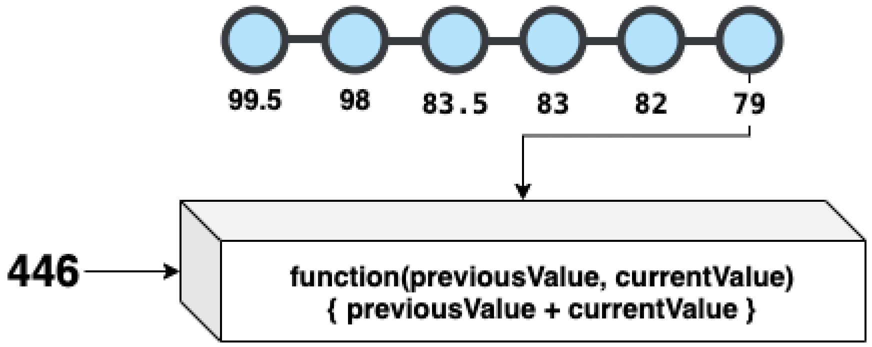

99.5
98
83.5
83
82
79
281
function(previousValue, currentValue)
{ previousValue + currentValue }
364

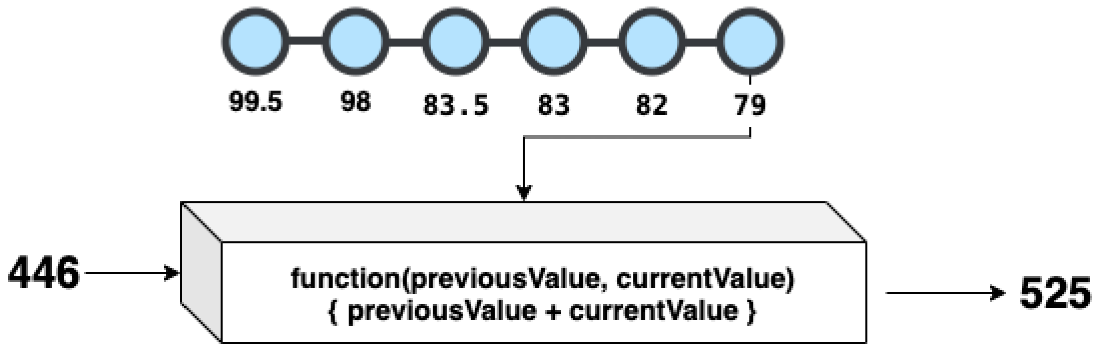

99.5
98
83.5
83
82
79
364
function(previousValue, currentValue)
{ previousValue + currentValue }
364


99.5
98
83.5
83
82
79
364
function(previousValue, currentValue)
{ previousValue + currentValue }


99.5
98
83.5
83
82
79
364
function(previousValue, currentValue)
{ previousValue + currentValue }
446


99.5
98
83.5
83
82
79
446
function(previousValue, currentValue)
{ previousValue + currentValue }
446


99.5
98
83.5
83
82
79
446
function(previousValue, currentValue)
{ previousValue + currentValue }


99.5
98
83.5
83
82
79
446
function(previousValue, currentValue)
{ previousValue + currentValue }
525

525

.reduce

Working with reduce to return an array...Say we want to add 5 points of extra credit to everyone and get back the equivalent letter grade.

```
const classGrades = [99.5, 98, 83.5, 83, 82, 79];
const scoresAsLetterGrades = classGrades
  .reduce((previousValue, currentValue) => {
    const updatedScore = currentValue + 5;
    if (updatedScore >= 90) { previousValue.push("A"); }
    else if (updatedScore >= 80) { previousValue.push("B"); }
    else if (updatedScore >= 70) { previousValue.push("C"); }
    else if (updatedScore >= 60) { previousValue.push("D"); }
    else { previousValue.push("F"); }
  }, []);
console.log("Class Scores", scoresAsLetterGrades);
// ["A","A","B","B","B","B"]
```

Lets have some fun and practice with a real example