reduce

- folds an array from left to right
- Takes a function to fold the accumulated value with the current value
- Returns a single value

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

reduce

```
var numbers = [1, 2, 3];


var sumOfNumbers = numbers.reduce(function(accu, current) {
  return accu + current;
}, 0);


// sumOfNumbers -> 6


// numbers -> [1, 2, 3]
```

reduce

```
var numbers = [1, 2, 3];


var sum = function(a, b) {
  return a + b;
}


var sumOfNumbers = numbers.reduce(sum, 0);

// sumOfNumbers -> 6

// numbers -> [1, 2, 3]


[...].reduce(function(accumulated, current, index, wholeArray) {
  // return the next accumulated value
}, initialValue);
```

reduce

```
var actions = [
  { type: 'increment' },
  { type: 'increment' },
  { type: 'decrement' }
];

var reducer = function(state, action) {
  switch (action.type) {
    case 'increment':
      return state + 1;
    case 'decrement':
      return state - 1;
    default:
      return state;
  }
}

var initialState = 0;

var newState = actions.reduce(reducer, initialState);

// newState -> 1
```

Behind the "magic"

forEach

```
Array.prototype.forEach = function(fn) {
  for (var i = 0; i < this.length; i++) {
    fn(this[i], i, this);
  }
}
```

map

```
Array.prototype.map = function(mapFn) {
  var mapped = [];

  for (var i = 0; i < this.length; i++) {
    mapped.push(mapFn(this[i], i, this));
  }

  return mapped;
}
```

filter

```
Array.prototype.filter = function(filterFn) {
  var filtered = [];

  for (var i = 0; i < this.length; i++) {
    if (filterFn(this[i], i, this)) {
      filtered.push(this[i]);
    }
  }

  return filtered;
}
```

reduce

```
Array.prototype.reduce = function(reducer, initial) {
  var result = initial;

  for (var i = 0; i < this.length; i++) {
    result = reducer(result, this[i], i, this);
  }

  return result;
}
```

Array Reducing Exercises

Canvas: exercises-array-reducing.zip