Map, reduce, and filter - JavaScript functional programming

Three functions that are commonly used when applying functional programming techniques in Javascript are the map, reduce, and filter functions. We'll go over each below and explain how they work.

Map

The map function creates a new array by calling a specific function on each element in an initial array. For example, if you have an array of strings in the form "MM-DD" that represent birthdays and you want to convert each element to be in a different format, you could use the map function to create a new array with new elements.

```
var bdays = ['08-14', '10-04', '04-21'];

// we want a new array where the birthdays will be in the format: MM/DD
// the elem parameter will be each element from the original array
var bdays2 = bdays.map(function(elem) {
  return elem.replace('-', '/');
});

console.log(bdays2); // => ['08/14', '10/04', '04/21']
```

Another simple example using the map function to round an array of numbers up in JavaScript:

```
var arr = [1.5, 2.56, 5.1, 12.33];

// round each number up in an array
var rounded = arr.map(Math.ceil);

console.log(rounded); // => [2, 3, 6, 13]
```

Reduce

The reduce function applies a specific function to all the elements in an array and reduces it to a single value. The reduce function has actually been used in several of the challenge solutions, one example being Mean Mode. We can use the reduce function to add up all the numbers in an array for example. The four arguments the reduce function takes are:

1) previous value
2) current value
3) current index
4) original array

```
var nums = [1, 2, 3, 4];

var sum = nums.reduce(function(prevVal, curVal, curIndex, origArr) {
  return prevVal + curVal;
});

console.log(sum); // => 10
```

Filter

The filter function creates a new array with all elements from an original array that pass a certain functions test. For example, you can use the filter function to create a new array of only positive values, like below. The function being called takes in an argument which is the value of the current element in the array.

```
var nums = [-4, 3, 2, -21, 1];

var pos = nums.filter(function(el) {
  return el > 0;
});

console.log(pos); // => [3, 2, 1]
```

You can also, for example, filter out all objects in a data file that have incorrect or undefined values. In the example below, we filter out all elements that have an incorrect age value.

```
var data = [
  {name: 'daniel', age: 45},
  {name: 'john', age: 34},
  {name: 'robert', age: null},
  {name: 'jen', age: undefined},
  {name: null, age: undefined}
];

// dataMod will now contain only the first two objects in the data array
var dataMod = data.filter(function(el) {
  if (el.name != undefined && el.age != undefined) {
    return true;
  }
  else {
    return false;
  }
});
```