Advanced Amazing arrays –5 (18 coding exercises)

"Strive for continuous improvement, instead of perfection" – K. Collins

5.1 Destructuring Arrays

Exercises

1. Destructure the following array:

```
let pokemonArray = [{name:'Bulbasaur'},
{name:'Squirtle'}, {name:'Pikachu'}, {name: 'Pokemon'},
{name:'Eevee'}];
```

Note: you may use whatever variable names that you like

- Log the fourth variable to the console. What do you get?

2. Destructure the following string so that only the last letter `'s'` is returned:

```
let myName = 'SleepySaurus';
```

3. Destructure the following nested array:

```
let array1 = ['hello', 'world', [100,200]];
```

Note: you may use whatever variable names that you like

4. Swap the values of these 2 arrays. What is the value of `array2` ?

```
let array1 = [{role: 'I.T'}, {role: 'game dev'}];

let array2 = ['thin crust', 'medium crust', 'pan crust'];
```

Answers

1.

```
let [one, two, three, four, five] = pokemonArray;
console.log(four);
```

```
Object {
  name: "Pokemon"
}
```

2.

```
let [,,,,,,,,,,, last] = myName;
console.log(last);
```

```
"s"
```

3.

```
let [greeting1, greeting2, [num1, num2]] = array1;
console.log(greeting1, greeting2,num1, num2);
```

```
"hello" "world" 100 200
```

4.

```
[array1, array2] = [array2, array1];
console.log(array2);
```

```
[[object Object] {
  role: "I.T"
}, [object Object] {
  role: "game dev"
}]
```

5.2 Spread operator

Exercises

1. What will be the output of the following code?:

```
let a =[1,2,3];

let b =[4,5,6, ...a];

let c =[...a,...b];

console.log(c);
```

2. Join these two arrays using the spread operator and reference the result via let `resultArray`

```
let arrayA = [{num: 1, max: 10}, {num:2, max:100}, {num3: 3, max:450}];

let arrayB = [{fruit: 'apple'}, {fruit: 'mango'},{fruit: 'blueberry'}];
```

   - What other array method can be used to join arrays? What is the main difference between this method and using the spread operator?
   - Loop over `resultArray` and return true if the value `'mango'` is within the array

Answers

1.

```
[1, 2, 3, 4, 5, 6, 1, 2, 3]
```

2.

```
let resultArray = [...arrayA, ...arrayB];
```

```
resultArray.forEach(function (arrayItem) {
  if(arrayItem.fruit === 'mango')
    //console.log('true');
    return(true);
```

```
});
```

Some methods used to join arrays:

`concat()` can be used to merge two or more arrays. Concat will return a new array. `concat()` is preferred when working with a larger array of items.

5.3 Map, Reduce and Filter methods

5.3.1 Map

Exercises

1. Consider the following code and use the `map()` method to achieve the output of `array2` logged to the console:

```
let array1 = [10,20,30,40];

let array2 = [];

//Your awesome code here!

console.log(array2); // [80, 60, 40, 20];
```

2. Using the `map()` method, transform the values of the following array by tripling them. Make sure that the return values are rounded off to 2 decimal places:

```
let numbersArray = [1.01, 89.99, 4562.79, 120];
```

Answers

1.

```
let array2 = array1.map(function(item){
  return(item * 2);
}).reverse();
```

2.

```
let tripledArray = numbersArray.map(function(item){
  return (item * 3).toFixed(2);
});
console.log(tripledArray);
```

```
["3.03", "269.97", "13688.37", "360.00"]
```

5.3.2 Reduce

Exercises

1. Multiply all the values in the following array with each other and then double the result

```
let arr = [1,10,10, 2];
```

2. For the following array, calculate the total population in all the cities:

```
let cities= [
  {
    name: 'Albuquerque',
    population: 2304004
  },
  {
    name:'Toronto',
    population: 547569284
  },
  {
    name:'Kuwait city',
    population: 39302848
  },
  {
    name:'Vancouver',
    population: 7834751
  }
];
```

Answers

1.

```
let multiplierArr = arr.reduce(function(acc, value){
  return acc * value;
},2);
console.log(multiplierArr);
```

```
400
```

2.

```
let total_population = cities.reduce(function(acc,
value){
  return acc + value.population;
},0);
console.log(total_population);
```

```
597010887
```

5.3.3 filter()

Exercises

1. Filter the following data and return users that are of the admin type

```
let userData = [{
    email: 'user1@hello.com',
    name: 'user 1',
    type: 'regular'
  },
  {
    email: 'user2@hello.com',
```

```
    name: 'user 2',
    type: 'admin'
  },
  {
    email: 'user3@hello.com',
    name: 'user 3',
    type: 'admin'
  },
  {
    email: 'user4@hello.com',
    name: 'user 4',
    type: 'regular'
  },
];
let keyword = 'admin';
```

2. For the following array, filter the words which have the characters `'er'` in them:

```
let wordArray= ['dog', 'pineapple', 'letter',
'technology', 'chatter', 'donut'];
```

3. In the following array, filter the prime numbers. *A prime number is not divisible by any number other than 1 and by itself:*

```
let numbers = [2, 4, 5, 7, 12, 13, 17, 19, 24, 29, 31,
33, 41, 43, 47, 53];
```

Answers

1.

```
let adminUsers = userData.filter(function(item){
  return(item.type === keyword);
});
console.log(adminUsers);
```

```
[[object Object] {
  email: "user2@hello.com",
  name: "user 2",
  type: "admin"
}, [object Object] {
  email: "user3@hello.com",
  name: "user 3",
  type: "admin"
}]
```

2.

```
function contains_char(wordArray) {
   return wordArray.indexOf('er') !== -1;
}
let filteredArray = wordArray.filter(contains_char);

console.log(filteredWords);
```

```
["letter", "chatter"]
```

Note: *-1 indicates that no words were found. This is how you test for something NOT being in an array in JavaScript:*

3. Code a function to check for non- prime numbers:

```
function isPrime(num) {
  for (let i = 2; num > i; i++) {
    if (num % i === 0) {
      return num ;
    }
  }
  return false;
}
```

Pass in the `isPrime()` function as an argument to the `filter()` method called upon the `numbers` array:

```
let nonPrimeNumbers = numbers.filter(isPrime);
```

Log the `nonPrimeNumbers` array to the console:

```
console.log(nonPrimeNumbers);
```

```
[4, 12, 24, 33]
```

5.4 Chaining Map, Reduce and Filter methods

Exercises

1. Consider the following array:

```
let words = ['ae', 'baed', 'led', 'ce', 'kaede'];
```

   - Filter the words that have the characters `'ae'`
   - Map the filtered words to a length > 3

2. Let's make our own planner for switching over into freelancing using the preceding array. The goal is to find out how much money freelancing makes per week, after all we would want to know that before quitting our day job! :

```
let tasks = [
{
 day: 'Monday',
 minutes: 480,
 tasks: 'client work, coding'
},
{
 day: 'Tuesday',
 minutes: 80,
 tasks: 'reading, coding'
},
 {
 day: 'Wednesday',
 minutes:  300,
```

```
  tasks: 'writing, working out'
 },
     {
  day: 'Thursday',
  minutes:  280,
  tasks: 'client work, coding'
 },
 {
  day: 'Friday',
  minutes:  380,
  tasks: 'client work'
 },
 {
  day: 'Saturday',
  minutes: 180,
  tasks: 'coding'
 },
 {
  day: 'Sunday',
  minutes: 40,
  tasks: 'reading, working out'
 },
];
```

- Filter out all the days that have a `'client work'` task
- Calculate the number of hours worked on these days
- Filter out the days in which you've worked more than 5 hours

- For days that you worked more than 5 hours calculate the total $ amount earned by multiplying the number of hours worked by your hourly rate which is $35
- Return the accumulated $ sum of the hours worked in total

Answers

1.

```
let words = ['ae', 'baed', 'led', 'ce', 'kaede'];
let words1 = words.filter(function(item){
  return(item.indexOf('ae') !== -1)
}).map(function(item){
  return(item.length > 3)
});
console.log(words1);
```

```
[false, true, true]
```

2.

```
let clientBilling = tasks.filter(function(item){
  return(item.tasks.indexOf('client work') != -1);
}).map(function(item){
  return(item.minutes / 60);
}).filter(function(item){
  return(item > 5);
}).map(function(item){
  return(item * 35);
}).reduce(function(accumalator, item){
  return(accumalator + item);
},0);
```

`console.log` the `clientBilling` variable:

```
console.log(clientBilling.toFixed(2));
```

```
"501.67"
```

5.5 Multi-dimensional arrays

Exercises

1. Flatten the following array down to 1 level using the `flat()` method:

```
let nestedArray = [[0,[1, 2]],3, [9], [10, 12]];
```

2. Using the data below, declare a two dimensional array called `weather`, which will contain 4 nested arrays representing the weeks of a month. Each array contains the temperature from Monday to Friday.

```
  [35, 30, 32, 31, 33]
  [30, 32, 33, 31, 30]
  [29, 30, 31, 29, 30]
  [29, 28, 30, 29, 30]
```

3. Consider the following two dimensional array called `weather`. It contains 4 nested arrays with 5 values each representing the temperatures from Monday to Friday:

```
let weather = [
  [35, 30, 32, 31, 33],
  [30, 32, 34, 31, 30],
  [30, 30, 31, 29, 30],
  [29, 30, 31, 29, 30],
  [29, 28, 30, 29, 30]
];
```

   - Filter the number of days that had a temperature equal to or greater than 33
   - Return the sum of the filtered values

Answers

1.

```
let flattenedArray= nestedArray.flat(1);
```

```
console.log(flattenedArray);//[0, [1, 2], 3, 9, 10, 12]
```

2.

```
let weather = [
  [35, 30, 32, 31, 33],
  [30, 32, 33, 31, 30],
  [30, 30, 31, 29, 30],
  [29, 30, 31, 29, 30],
  [29, 28, 30, 29, 30]
];
```

```
[[35, 30, 32, 31, 33], [30, 32, 33, 31, 30], [30, 30, 31,
29, 30], [29, 30, 31, 29, 30], [29, 28, 30, 29, 30]]
```

3. Flatten the array

```
let weatherFlat = weather.flat();
console.log(weatherFlat);
```

```
[35, 30, 32, 31, 33, 30, 32, 34, 31, 30, 30, 30, 31, 29,
30, 29, 30, 31, 29, 30, 29, 28, 30, 29, 30]
```

Use the `filter()` method on the flattened array:

```
let weatherHigh = weatherFlat.filter(function(item){
  return(item >= 33);
});
console.log(weatherHigh);
```

```
[35, 33, 34]
```

Use the `reduce()` method to find the sum of the filtered values:

```
let weatherSum = weatherHigh.reduce(function(acc, value){
  return(acc + value);
},0);
console.log(weatherSum);
```

```
102
```