Rest / Spread Operator Exercises

Author: Mahad Osman

Reference: SpringBoard solution, referenced slice and dice function

In this exercise, you'll refactor some ES5 code into ES2015.

Given this function:

```
function  filterOutOdds() {
  var  nums = Array.prototype.slice.call(arguments);
  return  nums.filter( function (num) {
    return  num % 2 === 0
  });
}
```

Refactor it to use the rest operator & an arrow function:

`/* Write an ES2015 Version */`

```
const filterOutOutdds = (...arr)=> arr.filter((v)=> v % 2 === 0 );
```

findMin

Write a function called findMin that accepts a variable number of arguments and returns the smallest argument.

Make sure to do this using the rest and spread operator.

`findMin(1,4,12,-3)  // -3 findMin(1,-1)  // -1 findMin(3,1)  // 1`

```
//Using spread without rest
const findMin = (...arr) => arr.reduce((num,nextValue)=>{
    return nextValue > num ? num : nextValue;
});

//Wuth spread const findMin = (...arr) => Math.min(...arr);
```

mergeObjects

Write a function called ***mergeObjects*** that accepts two objects and returns a new object which contains all the keys and values of the first object and second object.

`mergeObjects({a:1, b:2}, {c:3, d:4})` `// {a:1, b:2, c:3, d:4}`

```
// By using spread we merge in the two into one object
const mergeObjects = (obj1, obj2) => ({...obj1, ...obj2});
```

doubleAndReturnArgs

Write a function called ***doubleAndReturnArgs*** which accepts an array and a variable number of arguments. The function should return a new array with the original array values and all of additional arguments doubled.

`doubleAndReturnArgs([1,2,3],4,4)` `// [1,2,3,8,8]` `doubleAndReturnArgs([2],10,4)` `// [2, 20, 8]`

```
const doubleAndReturnArgs = (array, ...arrs) => [...array, ...arrs.map(v => v * 2) ]
```

Slice and Dice!

For this section, write the following functions using rest, spread and refactor these functions to be arrow functions!

Make sure that you are always returning a new array or object and not modifying the existing inputs.

`/** remove a random element in the items arrayand return a new array without that item. */` `function` `removeRandom(items) {}`

```
const removeRandomItem = items => {
    let randomIndex = Math.floor(Math.random() * items.length);
    return [...items.slice(0,randomIndex), ...items.slice(randomIndex + 1)];
}
```

`/** Return a new array with every item in array1 and array2. */` `function` `extend(array1, array2) {}`

```
const brandNewArray = (array1, array2) => ([...array1, ...array2]);
```

```
/** Return a new object with all the keys and valuesfrom obj and a new key/value pair
*/ function addKeyVal(obj, key, val) {}
```

```
const brandNewObject = (obj, key, value) => {
    let newObject = {...obj}
    newObject[key] = value
    return newObject;
}
```

```
/** Return a new object with a key removed. */ function removeKey(obj, key) {}
```

```
const removeKey = (obj, key, value) => {
    let newObject = {...obj}
    delete newObject[key];
    return newObject;
}
```

```
/** Combine two objects and return a new object. */ function combine(obj1, obj2) {}
```

```
const combine = (obj1, obj2) => ({...obj1, ...obj2});
```

```
/** Return a new object with a modified key and value. */ function update(obj, key, val) {}
```

```
const update = (obj, key, value) => {
    let newObject = {...obj}
    newObject[key] = value
    return newObject;
}
```