1. **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 filterOutOdds = (...args.filter(num => num % 2 === 0);

1. **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**

const findMin = (...nums) => Math.min(...nums);

1. **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}**

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

1. **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 = (arr, ...args) => [...arr, ...args.map(num => num \* 2)]

1. **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 array**

**and return a new array without that item. \*/**

const removeRandom = items => {

let idx = [Math.floor(Math.random() \* items.length)];

return [...items.slice(0, idx), ...items.slice(idx + 1)];

};

***/\*\* Return a new array with every item in array1 and array2. \*/***

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

***/\*\* Return a new object with all the keys and values***

***from obj and a new key/value pair \*/***

const addKeyVal = (obj, key, val) => {

let newObj = {...obj}

newObj[key] = val;

Return newObj;

};

***/\*\* Return a new object with a key removed. \*/***

const removeKey = (obj, key) => {

let newObj = {...obj};

delete newObj[key];

return newObj;

};

***/\*\* Combine two objects and return a new object. \*/***

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

***/\*\* Return a new object with a modified key and value. \*/***

const update = (obj, key, val) => {

let newObj = {...obj}

newObj[key] = val;

Return newObj;

};