


# ES2015 - Exercises

## *letConst:*



```
var PI = 3.14;  
PI = 42; // stop me from doing this!  
  
/* Write an ES2015 Version */  
const PI = 3.14;
```

- What is the difference between var and let?
  - var is function scope while let is block scope
  - var can be re-declared while let cannot
  - var will be hoisted while let will not
- What is the difference between var and const?
  - var is function scope while const is block scope
  - var can be re-declared while const cannot
  - var can be re-assigned while const cannot
  - var will be hoisted while const will not
- What is the difference between let and const?
  - let can be re-assigned while const cannot
- What is hoisting?
  - Hoisting is the result of JavaScript looking at the value of variables at initialization. The values become immediately available as opposed to returning an error message of trying to access a variable before it is declared.

## *arrow*Functions



```
// Arrow Functions
```

```
const double = arr.map( val => val * 2);
```

```
const squareAndFindEvens = numbers => numbers.map(num => num ** 2)  
  .filter(square => square % 2 === 0);
```

# restSpread

```
// rest/spread

const filteroutOdds = (...nums) => nums.filter(num => num % 2 !== 0);

const findMin = (...nums) => nums.reduce((min, val) => val < min ? min = val : min);

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

const doubleAndReturnArgs = (arr, ...args) => [...arr, ...args.map((val) => val * 2)];

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

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

const addKeyVal = (obj, key, val) => {
  let newObj = { ...obj };
  newObj[key] = val;
  return newObj;
};

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

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

const update = (obj, key, value) => {
  let newObj = { ...obj, key: value };
  return newObj;
};
```

# Object Enhancements

```
// Object Enhancements

function createInstructor(firstName, lastName) {
  return {
    firstName,
    lastName
  }
};

const favoriteNumber = 42;

const instructor = {
  firstName: "Colt",
  [favoriteNumber]: "That is my favorite!"
};

const instructor1 = {
  firstName: "Colt",
  sayHi() {
    return "Hi!";
  },
  sayBye() {
    return [firstName] + " says bye!";
  }
};

const d = createAnimal("dog", "bark", "Wooooof!")
// {species: "dog", bark: f}
console.log(d.bark()); // "Wooooof!"

const s = createAnimal("sheep", "bleet", "BAAAAaaaa")
// {species: "sheep", bleet: f}
console.log(s.bleet()); // "BAAAAaaaa"

function createAnimal(species, verb, noise) {
  return {
    species,
    [verb] () {
      return noise;
    }
  }
};
```

# Destructuring

```
// Destructuring

// obj 1
let facts = {numPlanets1: 8, yearNeptuneDiscovered: 1846};
let {numPlanets1, yearNeptuneDiscovered} = facts;

console.log(numPlanets1); // 8
console.log(yearNeptuneDiscovered); // 1846

// obj 2
let planetFacts = {
  numPlanets: 8,
  yearNeptuneDiscovered: 1846,
  yearMarsDiscovered: 1659
};

let {numPlanets, ...discoveryYears} = planetFacts;

console.log(discoveryYears); // {yearNeptuneDiscovered: 1846, yearMarsDiscovered: 1659}

// obj 3
function getUserData({firstName, favoriteColor="green"}){
  return `Your name is ${firstName} and you like ${favoriteColor}`;
}

getUserData({firstName: "Alejandro", favoriteColor: "purple"}) // "Your name is Alejandro and you like purple"
getUserData({firstName: "Melissa"}) // "Your name is Melissa and you like green"
getUserData({}) // "Your name is undefined and you like green"

// array 1
let [first, second, third] = ["Maya", "Marisa", "Chi"];

console.log(first); // "Maya"
console.log(second); // "Marissa"
console.log(third); // "Chi"

// array 2
let [raindrops, whiskers, ...aFewOfMyFavoriteThings] = [
  "Raindrops on roses",
  "whiskers on kittens",
  "Bright copper kettles",
  "warm woolen mittens",
  "Brown paper packages tied up with strings"
]

console.log(raindrops); // "Raindrops on roses"
console.log(whiskers); // "whiskers on kittens",
console.log(aFewOfMyFavoriteThings);
// [ "Bright copper kettles", "warm woolen mittens","Brown paper packages tied up with strings"]

// array 3

let numbers = [10, 20, 30];
[numbers[1], numbers[2]] = [numbers[2], numbers[1]]

console.log(numbers) // [10,30,20]
```

# Maps and Sets

```
// Maps and Sets

// quick question # 1
new Set([1, 1, 2, 2, 3, 4]); //{1,2,3,4}

// quick question # 2
[...new Set("referee")].join(""); // "ref"

// quick question # 3

let m = new Map();
m.set([1,2,3], true);
m.set([1, 2, 3], false);

// returns: Map(2) {Array(3) => true, Array(3) => false}

// hasDuplicates

const hasDuplicate = arr => new Set(arr).size !== arr.length;

hasDuplicate([1,3,2,1]) // true
hasDuplicate([1,5,-1,4]) // false

//vowelCount

function isVowel(char) {
  return 'aeiou'.includes(char);
}

const vowelCount = (str) => {
  const vowelMap = new Map();
  for (let char of str) {
    let lowerCasechar = char.toLowerCase();
    if (isVowel(lowerCasechar)) {
      if (vowelMap.has(lowerCasechar)) {
        vowelMap.set(lowerCasechar, vowelMap.get(lowerCasechar) + 1);
      } else {
        vowelMap.set(lowerCasechar, 1);
      }
    }
  }
  return vowelMap;
}
```