**Introduction**

In the exercises that follow, you will practice and reinforce your understanding of JavaScript [scope](https://www.codecademy.com/courses/introduction-to-javascript/lessons/scope/exercises/scope), [arrays](https://www.codecademy.com/courses/introduction-to-javascript/lessons/arrays/exercises/arrays), [loops](https://www.codecademy.com/courses/introduction-to-javascript/lessons/loops/exercises/loops), [iterators](https://www.codecademy.com/courses/introduction-to-javascript/lessons/javascript-iterators/exercises/intro), and [objects](https://www.codecademy.com/courses/introduction-to-javascript/lessons/objects/exercises/intro). We encourage you to review the relevant lessons, look things up in the [mdn documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference), check out the hints and solution code if you get stuck, and, most of all, have fun!

**Instructions**

Continue on to the first problem!

**reverseArray()**

**Instructions**

**1.**

Write a function, reverseArray(), that takes in an array as an argument and returns a new array with the elements in the reverse order.

There’s a [built-in method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reverse) that does a lot of this work for you, but here you’re not allowed to use it. Don’t worry you’ll have plenty of opportunities to use built-in methods later on!

const sentence = ['sense.','make', 'all', 'will', 'This'];

reverseArray(sentence);

// Should return ['This', 'will', 'all', 'make', 'sense.'];

Hint

It might be fun to [loop backwards](https://www.codecademy.com/courses/learn-javascript-loops/lessons/loops/exercises/for-loop-ii?action=resume_content_item&course_redirect=introduction-to-javascript):

for (let i = arr.length-1; i >= 0; i--)

or you could use [the .unshift() method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift).

const reverseArray = arr => {

    let reversed = [];

    for (let i = arr.length - 1; i >= 0; i--) {

        reversed.push(arr[i]);

    }

    return reversed

}

/\*

// Alternate solutions:

// Using the .unshift() method

const reverseArray = arr => {

    let reversed = [];

    for (let i = 0; i < arr.length; i++) {

        reversed.unshift(arr[i]);

    }

    return reversed

}

// As a function declaration:

function reverseArray(arr) {

    let reversed = [];

    for (let i = arr.length - 1; i >= 0; i--) {

        reversed.push(arr[i]);

    }

    return reversed

}

\*/

const sentence = ['sense.','make', 'all', 'will', 'This'];

console.log(reverseArray(sentence));

**greetAliens()**

**Instructions**

**1.**

Write a function, greetAliens(), that takes in an array of strings and uses a for loop to print a greeting with each string in the array.

The greeting should take the following format:  
“Oh powerful [stringElement], we humans offer our unconditional surrender!”

const aliens = ["Blorgous", "Glamyx", "Wegord", "SpaceKing"];

greetAliens(aliens);

// Should print:

// Oh powerful Blorgous, we humans offer our unconditional surrender!

// Oh powerful Glamyx, we humans offer our unconditional surrender!

// Oh powerful Wegord, we humans offer our unconditional surrender!

// Oh powerful SpaceKing, we humans offer our unconditional surrender!

Note: You may have noticed how convenient it would be to use [.forEach()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/foreach), but you’re not allowed to use it here. Don’t worry you’ll have plenty of opportunities to use built-in methods later on!

Hint

for (let i = 0; i < arr.length; i++)

// Write your code here:

function greetAliens(arr){

  for (let i = 0; i < arr.length; i++)

  {

    console.log(`Oh powerful ${arr}, we humans offer our unconditional surrender!`);

  }

}

// When you're ready to test your code, uncomment the below and run:

const aliens = ["Blorgous", "Glamyx", "Wegord", "SpaceKing"];

greetAliens(aliens);

**convertToBaby()**

**Instructions**

**1.**

Write a function, convertToBaby(), that takes in an array as an argument and, using a loop, returns a new array with each string in the array prepended with 'baby '.

const animals = ['panda', 'turtle', 'giraffe', 'hippo', 'sloth', 'human'];

convertToBaby(animals);

// Should return ['baby panda', 'baby turtle', 'baby giraffe', 'baby hippo', 'baby sloth', 'baby human'];

Note: You may have noticed how convenient it would be to use [map](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map), but you’re not allowed to use it here. Don’t worry you’ll have plenty of opportunities to use built-in methods later on!

Hint

for (let i = 0; i < arr.length; i++)

const convertToBaby = arr => {

      const babyArray = [];

      for (let i = 0; i < arr.length; i++) {

            babyArray.push('baby ' + arr[i]);

      }

      return babyArray

}

/\*

// As a function declaration:

function convertToBaby(arr) {

      const babyArray = [];

      for (let i = 0; i < arr.length; i++) {

            babyArray.push('baby ' + arr[i]);

      }

      return babyArray

}

\*/

const animals = ['panda', 'turtle', 'giraffe', 'hippo', 'sloth', 'human'];

console.log(convertToBaby(animals))

**Fix The Broken Code!**

**Instructions**

**1.**

We wrote a function, smallestPowerOfTwo(), which takes in an array.

Within our function, we create a new array results. We loop through the argument array and push in the smallest power of two which is greater than the current element.

It’s not doing what we want. Can you fix our code, please?

Hint

Hmmm, we have nested loops here. The problem with our code has to do with scope. Within the scope of our function two variables are stepping on each other’s toes and changing our expected outcome.

Any idea where we might have gone wrong? Check our loops carefully…

There’s something up with that i. Try to console.log(i) at different points in the code, and see if it matches up to what you expect.

Refer back to our [JavaScript exercise](https://www.codecademy.com/courses/introduction-to-javascript/lessons/loops/exercises/for-loops-iii?action=resume_content_item) for an example on how to use nested loops in your code.

const numbers = [5, 3, 9, 30];

const smallestPowerOfTwo = arr => {

      let results = [];

      // The 'outer' for loop - loops through each element in the array

      for (let i = 0; i < arr.length; i++) {

            number = arr[i];

            // The 'inner' while loop - searches for smallest power of 2 greater than the given number

            j = 1;

            while (j < number) {

                  j = j \* 2;

            }

            results.push(j);

      }

      return results

}

console.log(smallestPowerOfTwo(numbers))

// Should print the returned array [ 8, 4, 16, 32 ] instead prints the returned array [8]

**declineEverything() and acceptEverything()**

**Instructions**

**1.**

Write a function declineEverything() that takes in an array of strings and, using .forEach(), loops through each element in the array and calls politelyDecline() with each of them.

The .forEach() function should apply politelyDecline() directly; it should NOT merely receive argument function that *uses* politelyDecline().

You can test your function when you’re ready by passing in the veggies array or by making your own array!

Hint

If you need a refresher on .forEach(), check out [this exercise](https://www.codecademy.com/courses/introduction-to-javascript/lessons/javascript-iterators/exercises/for-each) and the [MDN documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach).

Your function should have an array parameter. You’ll invoke forEach() on the array passed in. What should your argument for the forEach() function be?

**2.**

Now we need to get healthy! Write a function acceptEverything() that takes in an array of strings and loops through each element in the array and grudgingly accepts each of them, by logging to the console in the following format: 'Ok, I guess I will eat some [element].'

You can use any technique you want to accomplish this task. You can test your function when you’re ready by passing in the veggies array or by making your own array!

Hint

.forEach() is quite convenient… If you wanted to use a named function, it might look like this:

const grudginglyAccept = (veg) => {

console.log('Ok, I guess I will eat some ' + veg + '.');

}

const veggies = ['broccoli', 'spinach', 'cauliflower', 'broccoflower'];

const politelyDecline = (veg) => {

      console.log('No ' + veg + ' please. I will have pizza with extra cheese.');

}

// Checkpoint 1 solutions:

const declineEverything = arr => {

  arr.forEach(politelyDecline)

}

// Checkpoint 2 solutions:

// Using an anonymous function and string interpolation:

const acceptEverything = arr => {

  arr.forEach(e => {

    console.log(`Ok, I guess I will eat some ${e}.`)

  })

}

/\*

// Using a named function and string concatenation:

const grudginglyAccept = veg => {

      console.log('Ok, I guess I will eat some ' + veg + '.');

}

const acceptEverything = arr => {

      arr.forEach(grudginglyAccept)

}

// Using a loop:

const acceptEverything = arr => {

 for(let i = 0; i<arr.length; i++){

    console.log(`Ok, I guess I will eat some ${arr[i]}.`)

 }

}

\*/

**squareNums()**

**Instructions**

**1.**

Write a function, squareNums(), that takes in an array of numbers and, using .map(), returns an array with the square of each of the elements of that array.

You can test your function when you’re ready by passing in the numbers array or by making your own array!

Hint

If you need a refresher on .map(), check out [our .map() exercise](https://www.codecademy.com/courses/introduction-to-javascript/lessons/javascript-iterators/exercises/map) and the [MDN documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map)

Your function should have an array parameter. You’ll invoke map() on the array passed in. What should you pass as an argument to .map()?

We wrote a function toSquare() that you might find useful…

const numbers = [2, 7, 9, 171, 52, 33, 14];

const toSquare = num => num \* num

const squareNums = arr => arr.map(toSquare)

**shoutGreetings()**

**Instructions**

**1.**

Write a function shoutGreetings() that takes in an array of strings and returns a new array. This new array should contain all the strings from the argument array but with capitalized letters and an exclamation point appended to the end: 'heya' will become 'HEYA!'

You can use any technique you want to accomplish this task.

You can test your function when you’re ready by passing in the greetings array or by making your own array!

Hint

.map() could prove handy here, but a for loop would work just fine!

You might also be interested in a certain string built-in method:

console.log('ilovecoding'.toUpperCase());

// Prints 'ILOVECODING'

const shoutGreetings = arr => arr.map(word => word.toUpperCase() + '!');

const greetings = ['hello', 'hi', 'heya', 'oi', 'hey', 'yo']

console.log(shoutGreetings(greetings))

**sortYears()**

**Instructions**

**1.**

Write a function sortYears() that takes in an array of years, and, using [the built-in .sort() method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort), returns that array with the years sorted in descending order.

You can test your function when you’re ready by passing in the years array or by making your own array of years!

Hint

The [.sort() method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort) is pretty useful! Just running years.sort() will sort the years in ascending order.

You’ll have to pass in your own function to get the functionality you want. If you were to use a named function, it might look something like:

const checkYears = (year1, year2) => year1 < year2

const sortYears = arr => arr.sort((x, y) => y - x);

/\*

// As a function declaration AND using a named function:

function sortYears(arr) {

      const checkYears = (year1, year2) => year2 - year1

      return arr.sort(checkYears)

}

\*/

const years = [1970, 1999, 1951, 1982, 1963, 2011, 2018, 1922]

console.log(sortYears(years))

**justCoolStuff()**

**Instructions**

**1.**

Write a function justCoolStuff() that takes in two arrays of strings, and, using [the built-in .filter() method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter), returns an array with the items that are present in both arrays.

let arr1 = ['this', 'not this', 'nor this'];

let arr2 = ['thing 1', 'thing 2', 'this'];

justCoolStuff(arr1, arr2); // Should return ['this']

You can test your function when you’re ready by passing in the myStuff and coolStuff arrays or by making arrays of your own!

Hint

The built-in Array method, .includes(), might be helpful here, but you’re welcome to use a loop! Check out [this Mozilla Developer Network documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/includes) for more information on the .includes() method.

If you need a refresher on .filter(), check out our [JavaScript exercise](https://www.codecademy.com/courses/introduction-to-javascript/lessons/javascript-iterators/exercises/filter?action=resume_content_item) in addition to [this Mozilla Developer Network documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter).

const justCoolStuff = (firstArray, secondArray) => firstArray.filter(item => secondArray.includes(item))

/\*

// As a function declaration AND using named function w/ a loop:

function justCoolStuff(firstArray, secondArray) {

      function isInSecondArray(item){

            for(let i = 0; i<secondArray.length; i++){

                  if (secondArray[i] === item){

                        return true

                  }

            }

            return false

      }

      return firstArray.filter(isInSecondArray)

}

\*/

const coolStuff = ['gameboys', 'skateboards', 'backwards hats', 'fruit-by-the-foot', 'pogs', 'my room', 'temporary tattoos'];

const myStuff = [ 'rules', 'fruit-by-the-foot', 'wedgies', 'sweaters', 'skateboards', 'family-night', 'my room', 'braces', 'the information superhighway'];

console.log(justCoolStuff(myStuff, coolStuff))

**isTheDinnerVegan()**

**Instructions**

**1.**

Write a function isTheDinnerVegan() that takes in an array of food objects in the format:

{name: 'cabbage', source: 'plant' }

and returns a boolean value based on whether or not every item in the array has entirely plant-based origins.

const meal = [{name: 'arugula', source: 'plant'}, {name: 'tomatoes', source: 'plant'}, {name: 'lemon', source:'plant'}, {name: 'olive oil', source: 'plant'}];

isTheDinnerVegan(meal); // Should return true

You can test your function when you’re ready by passing in the dinner array or by making your own!

Hint

You might consider using [the built-in .every() method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/every).

You may find the following function useful:

const isVegan = (food){

if(food.source === 'plant'){

return true;

}

return false;

}

const isTheDinnerVegan = arr => arr.every(food => food.source === 'plant');

const dinner = [{name: 'hamburger', source: 'meat'}, {name: 'cheese', source: 'dairy'}, {name: 'ketchup', source:'plant'}, {name: 'bun', source: 'plant'}, {name: 'dessert twinkies', source:'unknown'}];

console.log(isTheDinnerVegan(dinner));

**sortSpeciesByTeeth()**

**Instructions**

**1.**

Write a function sortSpeciesByTeeth() that takes in an array of species objects in the format:

{speciesName: 'shark', numTeeth: 50 }

and sorts the array in ascending order based on the average number of teeth that species possesses (numTeeth) .

You’ll need to access each object’s .numTeeth property. Feel free to either write a named comparison function, or use an anonymous function for your argument to .sort().

You can test your function when you’re ready by passing in the speciesArray array or by making your own!

Hint

Using an arrow function expression, we might write a comparison function that looks like:

const compareTeeth = (speciesObj1, speciesObj2) => speciesObj1.numTeeth > speciesObj2.numTeeth

const speciesArray = [ {speciesName:'shark', numTeeth:50}, {speciesName:'dog', numTeeth:42}, {speciesName:'alligator', numTeeth:80}, {speciesName:'human', numTeeth:32}];

const sortSpeciesByTeeth = arr => arr.sort((speciesObj1, speciesObj2) => speciesObj1.numTeeth > speciesObj2.numTeeth)

console.log(sortSpeciesByTeeth(speciesArray))

**findMyKeys()**

**Instructions**

**1.**

Write a function, findMyKeys(), that takes in an array of strings which may or may not contain 'keys'. If the keys are in the array, your function should return the index at which they can be found. If they’re not in the array, your function should return -1.

const drawer = ['rubber bands', 'tape', 'old menus', 'batteries'];

findMyKeys(drawer);

// Should return -1

You can use any technique you want to accomplish this task. Though, [if you look](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#Iteration_methods), there’s a built-in method that will make pretty quick work of it.

You can test your function when you’re ready by passing in the randomStuff array or by making your own array!

Hint

.findIndex() looks very useful! Check out [the docs](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex) to learn more about it!

const findMyKeys = arr => arr.findIndex(item => item === 'keys')

const randomStuff = ['credit card', 'screwdriver', 'receipt', 'gum', 'keys', 'used gum', 'plastic spoon'];

console.log(findMyKeys(randomStuff))

# dogFactory()

**Instructions**

**1.**

Write a function, dogFactory(). It should:

* have 3 parameters: name, breed, and weight (in that order)
* expect name and breed to be strings
* expect weight to be a number
* return an object
* have each of those parameters as keys on the returned object returned with the values of the arguments that were passed in

dogFactory('Joe', 'Pug', 27)

// Should return { name: 'Joe', breed: 'Pug', weight: 27 }

Hint

For a refresher on objects, refer to [this lesson](https://www.codecademy.com/courses/learn-javascript-objects/lessons/objects/exercises/intro?course_redirect=introduction-to-javascript)

const objFactory = param => {

return {

param: param

}

}

exampleObj = objFactory('example')

exampleObj.param // Returns 'example'

**2.**

Add getters and setters for each of the three properties and change the property names to have an underscore prepended.

Hint

const objFactory = param => {

return {

\_param: param,

get param() {

return this.\_param;

},

}

}

exampleObj = objFactory("example")

exampleObj.param //returns "example"

Check out the [getters](https://www.codecademy.com/courses/introduction-to-javascript/lessons/advanced-objects/exercises/getters) and [setters](https://www.codecademy.com/courses/introduction-to-javascript/lessons/advanced-objects/exercises/setters) exercises of the lesson to review in more depth.

**3.**

Add two methods to your object: .bark() which returns ‘ruff! ruff!’ and .eatTooManyTreats() which should increment the weight property by 1.

Hint

const objFactory = param => {

return {

\_param: param,

get param() {

return this.\_param;

},

aMethod() {

return "This method ran!"

}

}

}

exampleObj = objFactory("example");

exampleObj.aMethod(); //returns "This method ran!"

Check out the [methods section of the lesson](https://www.codecademy.com/courses/learn-javascript-objects/lessons/objects/exercises/methods?course_redirect=introduction-to-javascript) to review in more depth.

// Final solution:

const dogFactory = (name, breed, weight) => {

    return {

        \_name: name,

        \_breed: breed,

        \_weight: weight,

        get name() {

            return this.\_name;

        },

        set name(newName) {

            this.\_name = newName;

        },

        get breed() {

            return this.\_breed;

        },

        set breed(newBreed) {

            this.\_breed = newBreed;

        },

        get weight() {

            return this.\_weight;

        },

        set weight(newWeight) {

            this.\_weight = newWeight;

        },

        bark() {

            return 'ruff! ruff!'

        },

        eatTooManyTreats() {

            this.\_weight++

        }

    }

}

/\*

// Solution to checkpoint 1:

const dogFactory = (name, breed, weight) => {

      return {

            name: name,

            breed: breed,

            weight: weight

      }

}

// Solution to checkpoint 2:

const dogFactory = (name, breed, weight) => {

      return {

            \_name: name,

            \_breed: breed,

            \_weight: weight,

            get name() {

                  return this.\_name;

            },

            set name(newName) {

                  this.\_name = newName;

            },

            get breed() {

                  return this.\_breed;

            },

            set breed(newBreed) {

                  this.\_breed = newBreed;

            },

            get weight() {

                  return this.\_weight;

            },

            set weight(newWeight) {

                  this.\_weight = newWeight;

            }

      }

}

\*/