**Introduction to the Functional Programming Challenges**

Functional programming is all about creating and using non-mutating functions.

**1. Learn About Functional Programming**

/\*\*

\* A long process to prepare tea.

\* @return {string} A cup of tea.

\*\*/

const prepareTea = () => 'greenTea';

/\*\*

\* Get given number of cups of tea.

\* @param {number} numOfCups Number of required cups of tea.

\* @return {Array<string>} Given amount of tea cups.

\*\*/

const getTea = (numOfCups) => {

const teaCups = [];

for(let cups = 1; cups <= numOfCups; cups += 1) {

const teaCup = prepareTea();

teaCups.push(teaCup);

}

return teaCups;

};

// Add your code below this line

//Change: const tea4TeamFCC = null; // :(

const tea4TeamFCC = getTea(40);

// Add your code above this line

console.log(tea4TeamFCC);

**2. Understand Functional Programming Terminology**

/\*\*

\* A long process to prepare green tea.

\* @return {string} A cup of green tea.

\*\*/

const prepareGreenTea = () => 'greenTea';

/\*\*

\* A long process to prepare black tea.

\* @return {string} A cup of black tea.

\*\*/

const prepareBlackTea = () => 'blackTea';

/\*\*

\* Get given number of cups of tea.

\* @param {function():string} prepareTea The type of tea preparing function.

\* @param {number} numOfCups Number of required cups of tea.

\* @return {Array<string>} Given amount of tea cups.

\*\*/

const getTea = (prepareTea, numOfCups) => {

const teaCups = [];

for(let cups = 1; cups <= numOfCups; cups += 1) {

const teaCup = prepareTea();

teaCups.push(teaCup);

}

return teaCups;

};

// Add your code below this line

const tea4GreenTeamFCC = getTea(prepareGreenTea,27); // :(

const tea4BlackTeamFCC = getTea (prepareBlackTea,13); // :(

// Add your code above this line

console.log(

tea4GreenTeamFCC,

tea4BlackTeamFCC

);

**3. Understand the Hazards of Using Imperative Code**

\*\*\*Nothing to change

// tabs is an array of titles of each site open within the window

var Window = function(tabs) {

this.tabs = tabs; // we keep a record of the array inside the object

};

// When you join two windows into one window

Window.prototype.join = function (otherWindow) {

this.tabs = this.tabs.concat(otherWindow.tabs);

return this;

};

// When you open a new tab at the end

Window.prototype.tabOpen = function (tab) {

this.tabs.push('new tab'); // let's open a new tab for now

return this;

};

// When you close a tab

Window.prototype.tabClose = function (index) {

var tabsBeforeIndex = this.tabs.splice(0, index); // get the tabs before the tab

var tabsAfterIndex = this.tabs.splice(index); // get the tabs after the tab

this.tabs = tabsBeforeIndex.concat(tabsAfterIndex); // join them together

return this;

};

// Let's create three browser windows

var workWindow = new Window(['GMail', 'Inbox', 'Work mail', 'Docs', 'freeCodeCamp']); // Your mailbox, drive, and other work sites

var socialWindow = new Window(['FB', 'Gitter', 'Reddit', 'Twitter', 'Medium']); // Social sites

var videoWindow = new Window(['Netflix', 'YouTube', 'Vimeo', 'Vine']); // Entertainment sites

// Now perform the tab opening, closing, and other operations

var finalTabs = socialWindow

.tabOpen() // Open a new tab for cat memes

.join(videoWindow.tabClose(2)) // Close third tab in video window, and join

.join(workWindow.tabClose(1).tabOpen());

alert(finalTabs.tabs);

4. Avoid Mutations and Side Effects Using Functional Programming

// the global variable

var fixedValue = 4;

function incrementer () {

// Add your code below this line

return fixedValue + 1;

// Add your code above this line

}

var newValue = incrementer(); // Should equal 5

console.log(fixedValue); // Should print 4

5. Pass Arguments to Avoid External Dependence in a Function

// the global variable

var fixedValue = 4;

// Add your code below this line

function incrementer (value) {

return value + 1;

// Add your code above this line

}

var newValue = incrementer(fixedValue); // Should equal 5

console.log(fixedValue); // Should print 4

6. Refractor Global Variables Out of Functions

// the global variable

var bookList = ["The Hound of the Baskervilles", "On The Electrodynamics of Moving Bodies", "Philosophiæ Naturalis Principia Mathematica", "Disquisitiones Arithmeticae"];

/\* This function should add a book to the list and return the list \*/

// New parameters should come before the bookName one

// Add your code below this line

function add (list, bookName) {

var addBooks = list;

return addBooks.concat(bookName);

// Add your code above this line

}

/\* This function should remove a book from the list and return the list \*/

// New parameters should come before the bookName one

// Add your code below this line

function remove (list, bookName) {

var removeBooks = [...list];

if (removeBooks.indexOf(bookName) >= 0) {

removeBooks.splice(removeBooks.indexOf(bookName),1);

return removeBooks;

// Add your code above this line

}

}

var newBookList = add(bookList, 'A Brief History of Time');

var newerBookList = remove(bookList, 'On The Electrodynamics of Moving Bodies');

var newestBookList = remove(add(bookList, 'A Brief History of Time'), 'On The Electrodynamics of Moving Bodies');

console.log(bookList);

7. Use the map Method to Extract Data from an Array

// the global variable

var watchList = [

{

"Title": "Inception",

"Year": "2010",

"Rated": "PG-13",

"Released": "16 Jul 2010",

"Runtime": "148 min",

"Genre": "Action, Adventure, Crime",

"Director": "Christopher Nolan",

"Writer": "Christopher Nolan",

"Actors": "Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen Page, Tom Hardy",

"Plot": "A thief, who steals corporate secrets through use of dream-sharing technology, is given the inverse task of planting an idea into the mind of a CEO.",

"Language": "English, Japanese, French",

"Country": "USA, UK",

"Awards": "Won 4 Oscars. Another 143 wins & 198 nominations.",

"Poster": "http://ia.media-imdb.com/images/M/MV5BMjAxMzY3NjcxNF5BMl5BanBnXkFtZTcwNTI5OTM0Mw@@.\_V1\_SX300.jpg",

"Metascore": "74",

"imdbRating": "8.8",

"imdbVotes": "1,446,708",

"imdbID": "tt1375666",

"Type": "movie",

"Response": "True"

},

…];

// Add your code below this line

var rating = [];

rating = watchList.map(function(i)

{

return {title: i.Title, rating: i.imdbRating};

});

// Add your code above this line

console.log(rating);

**8. Implement map on a Prototype**

// the global Array

var s = [23, 65, 98, 5];

Array.prototype.myMap = function(callback){

var newArray = [];

// Add your code below this line

for(var i = 0; i < this.length; i++){

newArray.push(callback(this[i]));

}

// Add your code above this line

return newArray;

};

var new\_s = s.myMap(function(item){

return item \* 2;

});

**9. Use the filter Method to Extract Data from an Array**

// Add your code below this line

var filteredList = [];

var newList = watchList.filter(function(i){

//var rating = Number(i.imdbRating);

if(i.imdbRating >= 8.0) {

return i;

}

});

filteredList = newList.map(function(i) {

return {title: i.Title, rating: i.imdbRating};

});

// Add your code above this line

console.log(filteredList);

Another Solution:

Var filteredList = watchlist.filter((elemJ) => Number(elem.imbRating) >= 8.0).map((elem => ({title: elem.Title, rating: elem.imdRating}));

10. Implement the filter Method on a Prototype

// the global Array

var s = [23, 65, 98, 5];

Array.prototype.myFilter = function(callback){

var newArray = [];

// Add your code below this line

for(var i = 0; i < this.length; i++){

if (callback(this[i])) {

newArray.push(this[i]);

}

}

// Add your code above this line

return newArray;

};

var new\_s = s.myFilter(function(item){

return item % 2 === 1;

});

**11. Return Part of an Array Using the slice Method**

function sliceArray(anim, beginSlice, endSlice) {

// Add your code below this line

return anim.slice(beginSlice,endSlice);

// Add your code above this line

}

var inputAnim = ["Cat", "Dog", "Tiger", "Zebra", "Ant"];

sliceArray(inputAnim, 1, 3);

**12. Remove Elements from an Array Using slice Instead of splice.**

function nonMutatingSplice(cities) {

// Add your code below this line

return cities.slice(0,3);

// Add your code above this line

}

var inputCities = ["Chicago", "Delhi", "Islamabad", "London", "Berlin"];

nonMutatingSplice(inputCities);

**13. Combine Two Arrays Using the concat Method**

function nonMutatingConcat(original, attach) {

// Add your code below this line

return original.concat(attach);

// Add your code above this line

}

var first = [1, 2, 3];

var second = [4, 5];

nonMutatingConcat(first, second);

**14. Add Elements tot eh End of an Array Using concat Instead of push**

function nonMutatingPush(original, newItem) {

// Add your code below this line

return original.concat(newItem);

// Add your code above this line

}

var first = [1, 2, 3];

var second = [4, 5];

nonMutatingPush(first, second);

**15. Use the reduce Method to Analyze Data**

// Add your code below this line

var averageRating;

var count = 0;;

averageRating = watchList.reduce(function(accumulator, current){

if(current["Director"] === 'Christopher Nolan'){

accumulator += Number(current["imdbRating"]);

count++;

}

return accumulator;

}, 0)/count;

// Add your code above this line

console.log(averageRating);

**16. Sort an Array Alphabetically using the sort Method**

function alphabeticalOrder(arr) {

// Add your code below this line

return arr.sort()

// Add your code above this line

}

alphabeticalOrder(["a", "d", "c", "a", "z", "g"]);

**17. Return a Sorted Array Without Changing the Original Array**

var globalArray = [5, 6, 3, 2, 9];

function nonMutatingSort(arr) {

// Add your code below this line

var newSort = {};

return newSort = arr.concat().sort();

// Add your code above this line

}

nonMutatingSort(globalArray);

**18. Split s String into an Array using the split Method**

function splitify(str) {

// Add your code below this line

return str.split(/\W/);

// Add your code above this line

}

splitify("Hello World,I-am code");

\*\*\*\w is used to find a word character (character from a-z, A-Z, 0-9, including the \_ (underscore) character).

\W is used to find a non-word character.

**19. Combine an Array into a String Using the join Method**

function sentensify(str) {

// Add your code below this line

return str.split(/\W/).join(" ");

// Add your code above this line

}

sentensify("May-the-force-be-with-you");

20. Apply Functional Programming to Convert Strings to URL Slugs

// the global variable

var globalTitle = "Winter Is Coming";

// Add your code below this line

function urlSlug(title) {

return title.toLowerCase().trim().split(/\s+/).join("-");

}

// Add your code above this line

var winterComing = urlSlug(globalTitle); // Should be "winter-is-coming"

\*\*\*\s means "one space", and \s+ means "one or more spaces"

20. Use the every Method to Check that Every Element in an Array Meets a Criteria

function checkPositive(arr) {

// Add your code below this line

return arr.every(function(value) {

return value > 0;

});

// Add your code above this line

}

checkPositive([1, 2, 3, -4, 5]);

\*\*\*why 2 return: the checkPositive function needs to return the result of the function “every()” which also needs a return statement for each iteration to do it’s job properly…

so it’s something like this:

every() is checking each element of the array and returning true / false to the surrounding checkPositive function…  
but if that surrounding checkPositive function isn’t returning anything, then it’s like the checkPositive function kind knows what every() returned, but it’s not telling anyone…

21. Use the some Method to Check that Any Elements in an Array Meet a Criteria

function checkPositive(arr) {

// Add your code below this line

return arr.some(function(value) {

return value > 0;

});

// Add your code above this line

}

checkPositive([1, 2, 3, -4, 5]);

22. Introduction to Currying and Partial Application

function add(x) {

// Add your code below this line

return function(y) {

return function(z) {

return x + y + z;

}

}

// Add your code above this line

}

add(10)(20)(30);