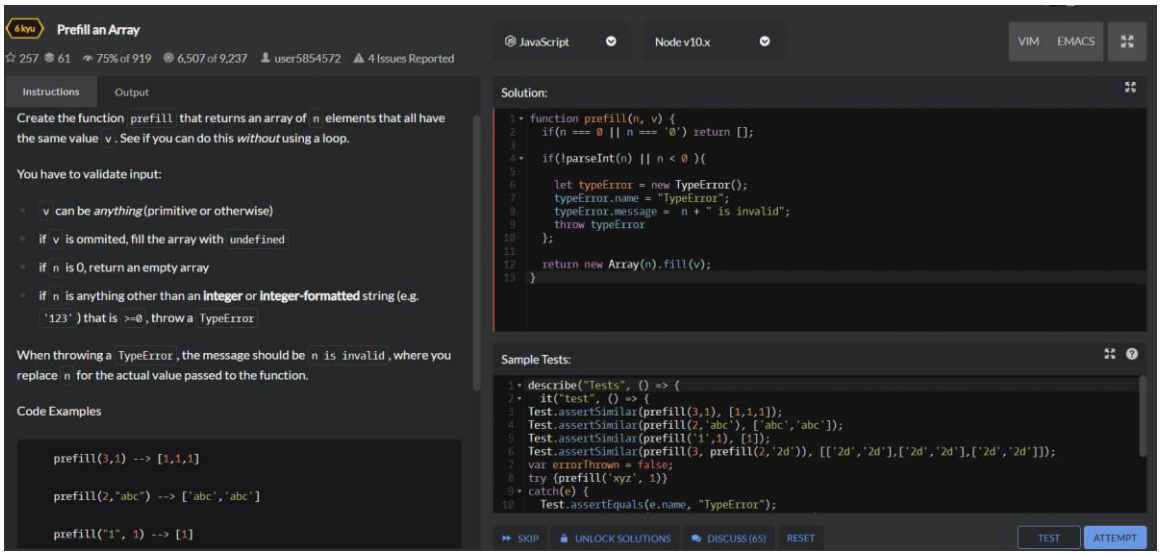
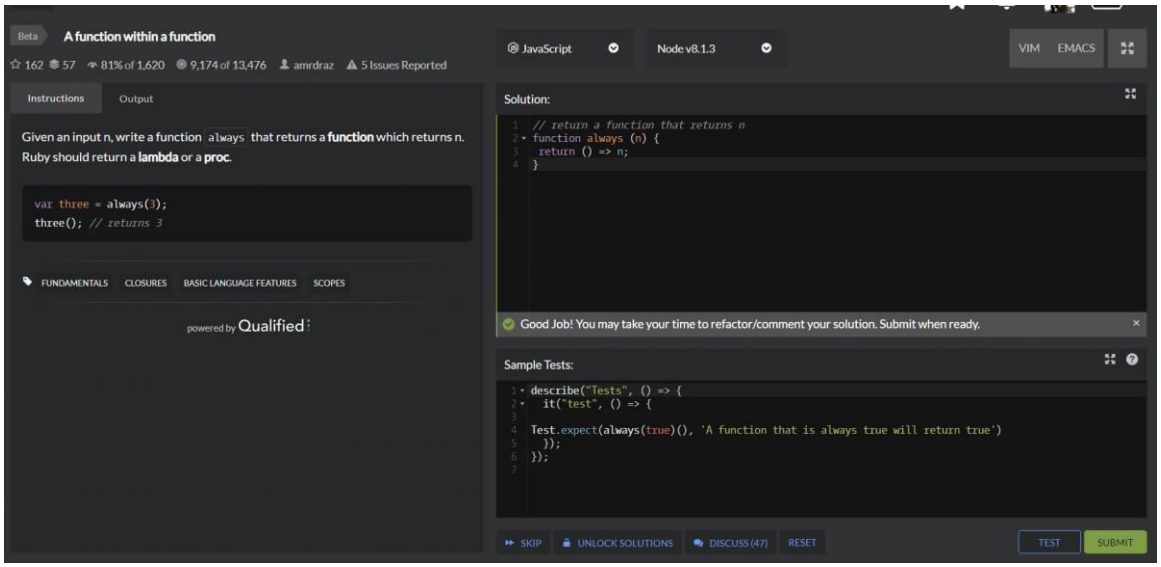
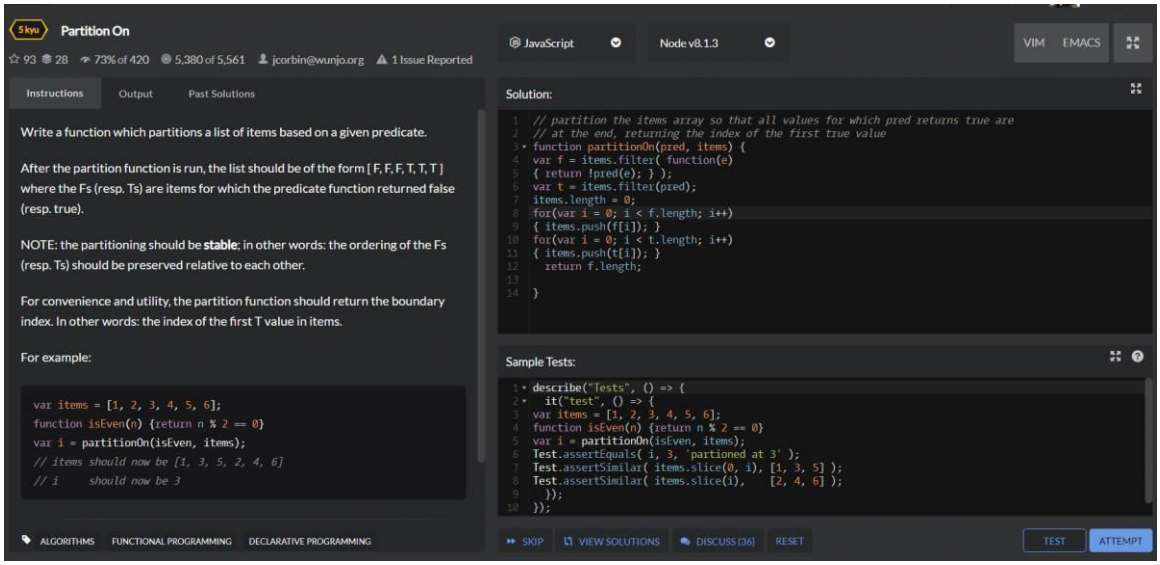
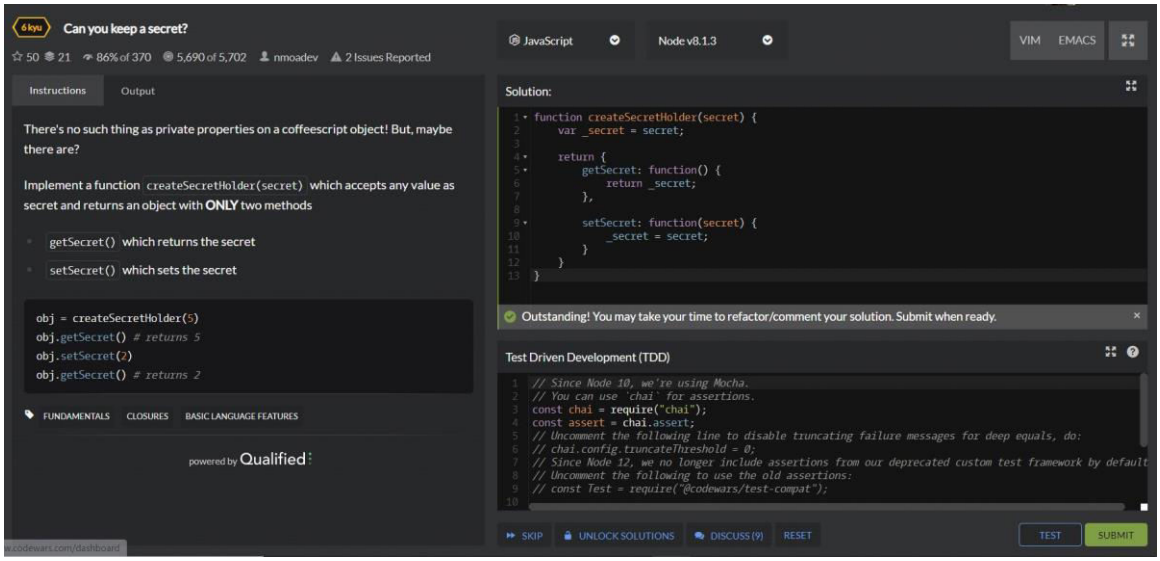
Home Task:


6 kyu
Prefill an Array
257 61 75% of 919 6,507 of 9,237 user5854572 4 Issues Reported
JavaScript
Node v10.x
VIM EMACS
Instructions
Output
Create the function prefill that returns an array of n elements that all have the same value v. See if you can do this without using a loop.
You have to validate input:
v can be anything (primitive or otherwise)
if v is ommited, fill the array with undefined
if n is 0, return an empty array
if n is anything other than an integer or integer-formatted string (e.g. '123') that is >=0, throw a TypeError
When throwing a TypeError, the message should be n is invalid, where you replace n for the actual value passed to the function.
Code Examples
prefill(3,1) --> [1,1,1]
prefill(2,"abc") --> ['abc','abc']
prefill("1", 1) --> [1]
Solution:
function prefill(n, v) {
if(n === 0 || n === '0') return [];
if(!parseInt(n) || n < 0){
let typeError = new TypeError();
typeError.name = "TypeError";
typeError.message = n + " is invalid";
throw typeError
};
return new Array(n).fill(v);
}
Sample Tests:
describe("Tests", () => {
it("test", () => {
Test.assertSimilar(prefill(3,1), [1,1,1]);
Test.assertSimilar(prefill(2,'abc'), ['abc','abc']);
Test.assertSimilar(prefill('1',1), [1]);
Test.assertSimilar(prefill(3, prefill(2,'2d')), [['2d','2d'],['2d','2d'],['2d','2d']]);
var errorThrown = false;
try {prefill('xyz', 1)}
catch(e) {
Test.assertEquals(e.name, "TypeError");
SKIP
UNLOCK SOLUTIONS
DISCUSS (65)
RESET
TEST
ATTEMPT


Beta
A function within a function
162 57 81% of 1,620 9,174 of 13,476 amrdraz 5 Issues Reported
JavaScript
Node v8.1.3
VIM EMACS
Instructions
Output
Given an input n, write a function always that returns a function which returns n. Ruby should return a lambda or a proc.
var three = always(3);
three(); // returns 3
FUNDAMENTALS
CLOSURES
BASIC LANGUAGE FEATURES
SCOPES
powered by Qualified
Solution:
// return a function that returns n
function always (n) {
return () => n;
}
Good Job! You may take your time to refactor/comment your solution. Submit when ready.
Sample Tests:
describe("Tests", () => {
it("test", () => {
Test.expect(always(true)(), 'A function that is always true will return true')
});
});
SKIP
UNLOCK SOLUTIONS
DISCUSS (47)
RESET
TEST
SUBMIT


5 kyu
Partition On
93 28 73% of 420 5,380 of 5,561 jcorbin@wunjo.org 1 Issue Reported
Instructions
Output
Past Solutions
Write a function which partitions a list of items based on a given predicate.
After the partition function is run, the list should be of the form [F, F, F, T, T, T] where the Fs (resp. Ts) are items for which the predicate function returned false (resp. true).
NOTE: the partitioning should be stable; in other words: the ordering of the Fs (resp. Ts) should be preserved relative to each other.
For convenience and utility, the partition function should return the boundary index. In other words: the index of the first T value in items.
For example:
var items = [1, 2, 3, 4, 5, 6];
function isEven(n) {return n % 2 == 0}
var i = partitionOn(isEven, items);
// items should now be [1, 3, 5, 2, 4, 6]
// i should now be 3
ALGORITHMS
FUNCTIONAL PROGRAMMING
DECLARATIVE PROGRAMMING
JavaScript
Node v8.1.3
VIM
EMACS
Solution:
// partition the items array so that all values for which pred returns true are
// at the end, returning the index of the first true value
function partitionOn(pred, items) {
var f = items.filter(function(e)
{ return !pred(e); });
var t = items.filter(pred);
items.length = 0;
for(var i = 0; i < f.length; i++)
{ items.push(f[i]); }
for(var i = 0; i < t.length; i++)
{ items.push(t[i]); }
return f.length;
}
Sample Tests:
describe("Tests", () => {
it("test", () => {
var items = [1, 2, 3, 4, 5, 6];
function isEven(n) {return n % 2 == 0}
var i = partitionOn(isEven, items);
Test.assertEquals(i, 3, 'partioned at 3');
Test.assertSimilar(items.slice(0, i), [1, 3, 5]);
Test.assertSimilar(items.slice(i), [2, 4, 6]);
});
});
SKIP
VIEW SOLUTIONS
DISCUSS (36)
RESET
TEST
ATTEMPT


6 kyu
Can you keep a secret?
50 21 86% of 370 5,690 of 5,702 nmoadev 2 Issues Reported
Instructions
Output
There's no such thing as private properties on a coffeescript object! But, maybe there are?
Implement a function createSecretHolder(secret) which accepts any value as secret and returns an object with ONLY two methods
getSecret() which returns the secret
setSecret() which sets the secret
obj = createSecretHolder(5)
obj.getSecret() # returns 5
obj.setSecret(2)
obj.getSecret() # returns 2
FUNDAMENTALS
CLOSURES
BASIC LANGUAGE FEATURES
powered by Qualified
JavaScript
Node v8.1.3
VIM
EMACS
Solution:
function createSecretHolder(secret) {
var _secret = secret;
return {
getSecret: function() {
return _secret;
},
setSecret: function(secret) {
_secret = secret;
}
}
}
Outstanding! You may take your time to refactor/comment your solution. Submit when ready.
Test Driven Development (TDD)
// Since Node 10, we're using Mocha.
// You can use `chai` for assertions.
const chai = require("chai");
const assert = chai.assert;
// Uncomment the following line to disable truncating failure messages for deep equals, do:
// chai.config.truncateThreshold = 0;
// Since Node 12, we no longer include assertions from our deprecated custom test framework by default
// Uncomment the following to use the old assertions:
// const Test = require("@codewars/test-compat");
SKIP
UNLOCK SOLUTIONS
DISCUSS (9)
RESET
TEST
SUBMIT

2.3