**Introduction to the Intermediate Algorithm Scripting Challenges**

**1. Sum All Numbers in a Range**

function sumAll(arr) {

var max = Math.max(arr[0], arr[1]);

var min = Math.min(arr[0], arr[1]);

var sum = 0;

for (var i = min; i <= max; i++){

sum += i;

}

return sum;

}

sumAll([1, 4]);

2. Diff Two Arrays

function diffArray(arr1, arr2) {

var newArr = arr1.concat(arr2);

return newArr.filter((item) => {

return (arr1.indexOf(item) === -1 || arr2.indexOf(item) === -1)

});

}

diffArray([1, 2, 3, 5], [1, 2, 3, 4, 5]);

**3. Seek and Destroy**

function destroyer(arr) {

// Remove all the values

var args = Array.prototype.slice.call(arguments);

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

for(var j = 0; j < args.length; j++) {

if(arr[i] === args[j]) {

delete arr[i];

}

}

}

return arr.filter(Boolean);

}

destroyer([1, 2, 3, 1, 2, 3], 2, 3);

Intermediate soln:

function destroyer(arr) {

var args = Array.from(arguments).slice(1);

return arr.filter(function(val) {

return !args.includes(val);

});

}

**Code Explanation:**

1. Declare a variable named args and set it equal to a new Array object from() the arguments passed into the function. On the same or next line, use the slice() method on args starting from the second index, 1. This separates the arguments used for filtering into their own array of args.

* Return the filtered array, using includes() in the callback function to check if val is *not* in args; returning true to keep the value in the original array or false to remove it.

4. Wherefore art thou

function whatIsInAName(collection, source) {

// What's in a name?

var arr = [];

// Only change code below this line

arr = collection.filter(function(item) {

for(var i in source) {

if(source[i] != item[i]) {

return false;

}

}

return true;

});

// Only change code above this line

return arr;

}

whatIsInAName([{ first: "Romeo", last: "Montague" }, { first: "Mercutio", last: null }, { first: "Tybalt", last: "Capulet" }], { last: "Capulet" });

**5. Spinal Tap Case**

function spinalCase(str) {

// "It's such a fine line between stupid, and clever."

// --David St. Hubbins

var regex = /\s+|\_|(?=[A-Z])/;

return str.split(regex).join('-').toLowerCase();

}

spinalCase('This Is Spinal Tap');

result: "this-is-spinal-tap"

**6. Pig Latin**

function translatePigLatin(str) {

var regexVowel = /[aieou]/;

var letters = str.split("");

if(str[0].match(regexVowel)){

return str + 'way';

}

else if (str.match(regexVowel)) {

str = str.slice(str.search(regexVowel)) + str.slice(0, str.search(regexVowel)) + "ay";

}

if(!regexVowel.test(str)) str += "ay";

return str;

}

translatePigLatin("consonant");

results: translatePigLatin("california")should return "aliforniacay"

translatePigLatin("glove")should return "oveglay"

translatePigLatin("algorithm")should return "algorithmway".

**7. Search and Replace**

function myReplace(str, before, after) {

if(before[0] === before[0].toUpperCase()) {

after = after.replace(after[0], after[0].toUpperCase());

}

return str.replace(before, after);

}

myReplace("A quick brown fox jumped over the lazy dog", "jumped", "leaped");

myReplace("He is Sleeping on the couch", "Sleeping", "sitting")should return "He is Sitting on the couch".

**8. DNA Pairing**

function pairElement(str) {

var pairing = [];

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

switch(str[i]) {

case "G":

pairing.push([str[i], "C"]);

break;

case "C":

pairing.push([str[i], "G"]);

break;

case "A":

pairing.push([str[i], "T"]);

break;

case "T":

pairing.push([str[i], "A"]);

break;

}

}

return pairing;

}

pairElement("GCG");

**9. Missing Letters**

function fearNotLetter(str) {

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

var letter = str.charCodeAt(i);

if(letter !== str.charCodeAt(0) + i) {

return String.fromCharCode(letter - 1);

}

}

return undefined;

}

fearNotLetter("abce");

10. Sorted Union

function uniteUnique(arr) {

var newArray = [];

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

var arrArguments = arguments[i];

for(var j = 0; j < arrArguments.length; j++) {

var valueIndexArr = arrArguments[j];

if(newArray.indexOf(valueIndexArr) < 0) {

newArray.push(valueIndexArr);

}

}

}

return newArray;

}

uniteUnique([1, 3, 2], [5, 2, 1, 4], [2, 1]);  [1, 3, 2, 5, 4].

**10. Convert HTML Entities**

function convertHTML(str) {

// &colon;&rpar;

var tempString = str.split('');

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

switch(tempString[i]) {

case "&":

tempString[i] = "&amp;";

break;

case "<":

tempString[i] = "&lt;";

break;

case ">":

tempString[i] = "&gt;";

break;

case '"':

tempString[i] = "&quot;";

break;

case "'":

tempString[i] = "&apos;";

break;

}

}

tempString = tempString.join("");

return tempString;

}

convertHTML("Dolce & Gabbana");

should return Dolce &​amp; Gabbana.

Another method:

**function** **convertHTML**(str) {

*//Chaining of replace method with different arguments*

str = str.replace(/&/g,'&amp;').replace(/</g,'&lt;').replace(/>/g,'&gt;').replace(/"/g,'&quot;').replace(/'/g,"&apos;");

**return** str;

}

**11. Sum All Fibonacci Numbers**

function sumFibs(num) {

var sum = 0;

var prevNum = 0;

var currentNum = 1;

while (currentNum <= num) {

if(currentNum % 2 !== 0){

sum += currentNum;

}

currentNum += prevNum;

prevNum = currentNum - prevNum;

}

return sum;

}

sumFibs(4); should return 5.

sumFibs(1)should return a number.

sumFibs(1000)should return 1785.

**12) Sum All Primes**

function sumPrimes(num) {

var sum = 0;

function checkPrime(number) {

for(var i = 2; i < number; i++) {

if(number % i === 0) {

return false;

}

}

return true;

}

for(var number = 2; number <= num; number++){

if(checkPrime(number)) {

sum += number;

}

}

return sum;

}

sumPrimes(10); should return a number.

sumPrimes(10)should return 17.

**13. Smallest Common Multiple**

function smallestCommons(arr) {

var max = Math.max(arr[0], arr[1]);

var min = Math.min(arr[0], arr[1]);

var commonMultiple = max;

for(var i = max; i >= min; i--) {

if(commonMultiple % i !== 0) {

commonMultiple += max;

i = max;

}

}

return commonMultiple;

}

smallestCommons([1,5]); should return 60.

smallestCommons([5, 1])should return 60.

smallestCommons([2, 10])should return 2520.

**14. Drop it**

function dropElements(arr, func) {

// Drop them elements.

var arrlength = arr.length;

for(var i = 0; i < arrlength; i++){

if(func(arr[0])) {

break;

} else {

arr.shift();

}

}

return arr;

}

dropElements([1, 2, 3], function(n) {return n < 3; });

dropElements([1, 2, 3, 4], function(n) {return n >= 3;})should return [3, 4].

dropElements([1, 2, 3], function(n) {return n > 0;})should return [1, 2, 3].

Note: It looks like arr.shift() removes the element which shortens the length of the array. So using arr.length in the for loop will not get calculated correctly. So thats why the original solution stores arr.length as times so that it’s consistent across all loop iterations and not get affected by changes from shift().

**15. Streamroller**

function steamrollArray(arr) {

// I'm a steamroller, baby

return arr.flat(3);

}

steamrollArray([1, [2], [3, [[4]]]]); should return [1, 2, 3, 4].

steamrollArray([[["a"]], [["b"]]])should return ["a", "b"].

**Note:** Flatten a nested array. You must account for varying levels of nesting.

**16. Binary Agents**

function binaryAgent(str) {

var tempStr = str.split(' ');

var newStr = [];

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

newStr.push(String.fromCharCode(parseInt(tempStr[i],2)));

}

return newStr.join('');

}

binaryAgent("01000001 01110010 01100101 01101110 00100111 01110100 00100000 01100010 01101111 01101110 01100110 01101001 01110010 01100101 01110011 00100000 01100110 01110101 01101110 00100001 00111111")should return "Aren't bonfires fun!?"

**17. Everything Be True**

function truthCheck(collection, pre) {

// Is everyone being true?

var count = 0;

for(var item in collection) {

if(collection[item].hasOwnProperty(pre) && Boolean(collection[item][pre])) {

count++;

}

}

return count == collection.length;

}

truthCheck([{"user": "Tinky-Winky", "sex": "male"}, {"user": "Dipsy", "sex": "male"}, {"user": "Laa-Laa", "sex": "female"}, {"user": "Po", "sex": "female"}], "sex"); should return true.

truthCheck([{"user": "Tinky-Winky", "sex": "male"}, {"user": "Dipsy"}, {"user": "Laa-Laa", "sex": "female"}, {"user": "Po", "sex": "female"}], "sex")should return false.

18. Arguments Optional

function addTogether() {

function checkNumber(num){

if(typeof num !== 'number') {

return undefined;

}else {

return num;

}

}

if(arguments.length > 1) {

var arg1 = checkNumber(arguments[0]);

var arg2 = checkNumber(arguments[1]);

if(arg1 === undefined || arg2 === undefined) {

return undefined;

} else {

return arg1 + arg2;

}

} else {

var arg3 = arguments[0];

if(checkNumber(arg3)) {

return function(arg2) {

if(arg3 === undefined || checkNumber(arg2) === undefined) {

return undefined;

} else {

return arg3 + arg2;

}

};

}

}

}

addTogether(2,3); should return 5.

addTogether(2)(3)should return 5.

addTogether("http://bit.ly/IqT6zt")should return undefined.

18. Make a Person

var Person = function(firstAndLast) {

// Complete the method below and implement the others similarly

var fullName = firstAndLast;

this.getFullName = function() {

return fullName;

};

this.getFirstName = function() {

return fullName.split(" ")[0];

};

this.getLastName = function() {

return fullName.split(" ")[1];

};

this.setFirstName = function(fName) {

fullName = fName + " " + fullName.split(" ")[1];

};

this.setLastName = function(lName) {

fullName = fullName.split(" ")[0] + " " + lName;

};

this.setFullName = function(name) {

fullName = name;

};

};

var bob = new Person('Bob Ross');

bob.getFullName(); should return "Bob Ross".

bob.getLastName()should return "Ross".

bob.getFullName()should return "Haskell Ross" after bob.setFirstName("Haskell").

19. Map the Debris

function orbitalPeriod(arr) {

var GM = 398600.4418;

var earthRadius = 6367.4447;

var pi = 2 \* Math.PI;

var newArray = [];

var getOrbitalPeriod = function(object) {

var a = Math.pow(earthRadius + object.avgAlt,3);

var sqroot = Math.sqrt(a/GM);

var orbPeriod = Math.round(pi \* sqroot);

delete object.avgAlt;

object.orbitalPeriod = orbPeriod;

return object;

};

for(var item in arr) {

newArray.push(getOrbitalPeriod(arr[item]));

}

return newArray;

}

orbitalPeriod([{name : "sputnik", avgAlt : 35873.5553}]);

should return [{name: "sputnik", orbitalPeriod: 86400}]