Lab 2

Re-submit Assignment

Due Feb 4 by 11:55pm **Points** 100 **Submitting** a file upload **File Types** zip

CS-546 Lab 2

Modules and Error Checking

This week, you will be creating your first package and first module!

You will write 3 files this week + provide a package.json file; two of which will be modules, and one of which will use that module. You will submit them in a zip file named `LastName_FirstName.zip`. You **should not** have any folders inside the zip file.

1. **index.js**: This file will require both `geometry.js` and `utilities.js` and will test each method you export 5 times.
2. **geometry.js**: This module will provide several methods to compute basic geometric formulas.
3. **utilities.js**: This module will provide several methods to perform basic utility methods.
4. **package.json**: This file will describe your package. You can accomplish most of the specified requirements with the `npm init` command (see below).

If an argument fails error checking, you should throw a string describing what's argument was wrong, and what went wrong.

Making a package

Make sure to initialize a package and name it `cs-546-lab-2` for this lab! You can use `npm init` to help get this started.

You must also include a proper `npm start` task that will run `index.js`, and an `author` field in your `package.json` file! The author field should have your first name, last name, and CWID.

geometry.js

This file will export 4 methods. Each method will have error checking on each argument.

volumeOfRectangularPrism(length, width, height)

This method will calculate the volume of a rectangular prism. You must check that each argument is provided, is a number, and is within proper bounds.

surfaceAreaOfRectangularPrism(length, width, height)

This method will calculate the surface area of a rectangular prism. You must check that each argument is provided, is a number, and is within proper bounds.

volumeOfSphere(radius)

This method will calculate the volume of a sphere. You must check that each argument is provided, is a number, and is within proper bounds. You **must** use `Math.PI` as the `pi` value.

surfaceAreaOfSphere(radius)

This method will calculate the surface area of a sphere. You must check that each argument is provided, is a number, and is within proper bounds. You **must** use `Math.PI` as the `pi` value.

utilities.js

This file will export 3 methods. Each method will have error checking on each argument.

deepEquality(obj1, obj2)

This method check each field (at every level deep) in `obj1` and `obj2` for equality. It will return `true` if each field is equal, and `false` if not.

For example, if given the following:

```
const first = {a: 2, b: 3};
const second = {a: 2, b: 4};
const third = {a: 2, b: 3};
console.log(deepEquality(first, second)); // false
console.log(deepEquality(first, third)); // true
```

You must check that each argument is provided and that each argument is an object.

uniqueElements(arr)

This method will iterate throughout the array provided in `arr` and return how many unique elements are in the array.

You must check that `arr` is provided and that it is an array. For example:

```
const testArr = ["a", "a", "b", "a", "b", "c"];
console.log(uniqueElements(testArr)); // outputs 3
```

countOfEachCharacterInString(str)

This method will traverse the string provided, `str`, and return an object. Each unique character in the array will be a key in the object, and the value will be how many times it appears in the string provided.

For example:

```
const test = "Hello, the pie is in the oven";
const charMap = countOfEachCharacterInString(test);
```

Would result in `charMap` having the value of:

```
{
    " ": 6,
    ",": 1,
    "H": 1,
    "e": 5,
    "h": 2,
    "i": 3,
    "l": 2,
    "n": 2,
    "o": 2,
    "p": 1,
    "s": 1,
    "t": 2,
    "v": 1
}
```

You must check that `str` is provided and that it is a string.

Requirements

1. You will have to write each function
2. You must submit all files, zipped up, not contained in any folders
3. You must not use any npm dependencies in this lab. You must do basic error checking in each function
    1. Check for arguments existing and of proper type.
    2. Throw if anything is out of bounds (ie, trying to perform an incalculable math operation or accessing data that does not exist)