A food restaurant has hired you to create a function for their website that allows them to set a meal and price each morning for SuperToday’s Special. Use your knowledge of getters and setters to make sure anyone who uses the new function can generate a meal and a price for SuperToday’s Special without any embarrassing errors!

**1.**

We’ll hold the meal, price, and their respective getters and setters in an object named menu.

In **app.js**, create an empty menu object.

**2.**

The menu object will hold the meal and price of SuperToday’s Special as properties and they shouldn’t be altered directly.

Within the menu object, create a \_meal property with a value of an empty string (''). This will eventually hold the name of the meal.

**3.**

Next, add a \_price property with a value of 0. This will eventually hold the price of the meal, and should also not be altered directly.

**4.**

We know properties that begin with \_ should not be directly manipulated. But just to validate this knowledge, let’s test it out!

Below the menu object, directly manipulate the two properties by assigning \_meal a number value and \_price a string value.

Then, below the new assignments, console.log() the menu object to see how not type checking these values could cause confusion for a website visitor!

**5.**

To safely reassign the two menu properties, we can add setters that type check the values being assigned.

Below the properties, use the set keyword to create a meal setter method with mealToCheck as a parameter. Leave the function body empty for now.

**6.**

In the body of the setter method, create an if statement that checks if mealToCheck is a string. If it is, return the object’s \_meal property with mealToCheck assigned as the value.

**7.**

Utilizing the same procedures as above, use the set keyword to create a price setter with priceToCheck as a parameter. This method should make sure the value associated with \_price is always a number.

**8.**

Below the menu object, set the values of \_meal and \_price using the newly created setter methods. Then, console.log() the menu object to check their functionality.

**9.**

Now it’s time to safely return the values of the \_meal and \_price properties in a readable form. Instead of directly accessing the properties, we can use a getter method that proactively checks if a meal and price have been properly set, before returning the values.

Below the setters, use the get keyword to create a todaysSpecial method. Leave the function body empty for now.

**10.**

In the body of the getter, create an if…else statement to check if \_meal and \_price values exist (or are truthy values). If so, return a string telling potential website visitors what SuperSuperToday’s Special is. For example: “SuperToday’s Special is Spaghetti for $5!”

If \_meal and \_price values do not exist (or are falsy) return the string 'Meal or price was not set correctly!'.

**11.**

Finally, use the getter method to console.log() SuperToday’s Special.

Assuming you used the meal setter to assign a string, and the price setter to assign a number in task 8, you should see SuperToday’s Special logged to the console.

If you want to extend your learning on this project, try adding an array of meals and prices to randomly set and get SuperToday’s Special!