**TypeMart**

Welcome to our new online shop: TypeMart! We’re ready to go live… well, almost… We’re attempting to get this shop up and going, but we need your help to code a crucial piece of our online mart: the purchase flow!

We think you’re the right person for the job because it’s really important to us that the purchase flow is type-safe. (All that type safety will allow us to sleep well at night.)

Currently, we are producing a few *types* of products, and we can’t wait to expand as we start to sell our first items. For our mart to be successful, we’ll have to calculate the correct price for shoppers, in addition to applying any specials or other charges.

Follow the tasks below to create a purchase flow for TypeMart visitors.

**1.**

Welcome to TypeMart’s codebase! Currently it consists of two files:

1. A **products.ts** file, which has a list of products we will sell.
2. An **index.ts** file, which we can use to create our a purchase flow for our users.

Take a moment to look at **products.ts** to get familiar with our products, then start by importing products from **products.ts** into **index.ts**.

**2.**

Next, we need to store the name of the product the user is trying to find.

Declare a variable named productName, annotate it with the appropriate type, and set its value to one of the product names in **products.ts**. Feel free to choose any of the products.

**3.**

Now that we have the product’s name in a variable, we need to find it by its name from our list of products. Use any method to find a product inside products that matches the string stored in the productName variable.

Once you have found the matching product, store that product in a variable named product.

**4.**

Let’s verify we found the correct product from the last step.

Use console.log() to print the product variable’s value.

Save your file in the code editor, run tsc on the command line to compile your code into JavaScript, then run node index.js to see the output. Verify that you see an object printed in the console.

**5.**

Now that we selected a product, we have some additional logic to implement.

We have a few items that are available for pre-order—shoppers can purchase them now, and we will ship them at a later date. If this is the case for a product, we want to notify the customer that we’ll send them a message when their product ships.

In **index.ts**, write a conditional that checks if the .preOrder property is true on the product, then log a message to the console to tell the customer that we’ll send them a message when it’s on the way.

**6.**

Once you’ve implemented this condition, you should test that it’s working as expected. Make sure that the 'fanny pack' product makes the message print and that the 'beanie' product does not (since it’s not on pre-order).

Use tsc to compile your code, then node index.js to run your program.

**7.**

Now we’re ready to start putting together all the data we’ll need to provide the shopper a receipt. The receipt will show the shipping address, price, tax, shipping cost, and total cost.

Declare four variables named: shipping (number), taxPercent (number), taxTotal (number), and total (number)—make sure to annotate all their types.

**8.**

Declare a variable shippingAddress. Its value should be a string containing the customer’s address. Make sure to annotate it correctly and assign shippingAddress a value of an address in a string.

**9.**

If the price of an item is over $25, we will provide free shipping.

Write a conditional that sets shipping to 0 if the price of the product is 25 or over. Include a console.log() stating that we provide free shipping for this product. If the product is under $25, set the value of shipping to 5.

Use tsc and node index.js to verify that 'fanny pack' prints the message about free shipping and 'shirt' does not.

**10.**

Since our shop operates out of New York City, we’re required to pay extra tax when someone from New York buys a product from TypeMart.

If the shipping address contains the value 'New York', then we need to set the taxPercent to 0.1 (a shocking 10%), otherwise, the taxPercent should be .05.

To check if the shippingAddress contains 'New York' you can use the [.match() method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/match) to check.

**11.**

Next, calculate the total.

First, multiply the product.price by the taxPercent, then assign that value to the taxTotal variable.

Then, add together the product.price, taxTotal, and shipping, and assign it to the total variable.

Run tsc to check for any type errors.

**12.**

Now that we have all the required data for the receipt, use console.log() to print:

* Product name
* Shipping address
* Price of the product
* Tax total
* Shipping
* Total amount

Then run tsc and node index.js to verify that the receipt is correct.

For example, the total cost for a 'fanny pack' bought inside New York should be $33.00, while a 'tote bag' bought outside of New York should be $26.00.

**13.**

Congratulations on completing this project! You practiced annotating types, fixing type errors with tsc, and creating a type safe program. The management at TypeMart is thrilled; we hope to see you at the grand opening!