**Step 14**

You are already familiar with an HTML class, but JavaScript also has a *class*. In JavaScript, a class is like a blueprint for creating objects. It allows you to define a set of properties and methods, and instantiate (or create) new objects with those properties and methods.

The class keyword is used to declare a class. Here is an example of declaring a Computer class:

Example Code

class Computer {};

Declare a ShoppingCart class.

**Step 15**

Classes have a special constructor method, which is called when a new instance of the class is created. The constructor method is a great place to initialize properties of the class. Here is an example of a class with a constructor method:

Example Code

class Computer {

constructor() {

}

}

Add an empty constructor method to the ShoppingCart class.

The this keyword in JavaScript is used to refer to the current object. Depending on where this is used, what it references changes. In the case of a class, it refers to the instance of the object being constructed. You can use the this keyword to set the properties of the object being instantiated. Here is an example:

Example Code

class Computer {

constructor() {

this.ram = 16;

}

}

In your constructor, use the this keyword to set the items property to an empty array. Also, set the total property to 0, and the taxRate property to 8.25.

**Step 17**

Your ShoppingCart class needs the ability to add items. Create an empty addItem method, which takes two parameters: id and products. Creating a method might look like this:

Example Code

class Computer {

constructor() {

this.ram = 16;

}

addRam(amount) {

this.ram += amount;

}

}

The first parameter, id, is the id of the product the user has added to their cart. The second parameter, products, is an array of product objects. By using a parameter instead of directly referencing your existing products array, this method will be more flexible if you wanted to add additional product lists in the future.

You need to find the product that the user is adding to the cart. Remember that arrays have a .find() method. In your addItem function, declare a product variable, and assign it the value of calling the .find() method on the products array.

For the callback to .find(), pass a function that takes a single parameter item, and returns whether the id property of item is strictly equal to the id parameter passed to addItem.

**Step 32**

There is still more functionality that your ShoppingCart class needs, but first you need to be able to test the code you have currently written. You'll need to *instantiate* a new ShoppingCart object and assign it to a variable. Here is an example of instantiating the Computer class from earlier examples:

Example Code

const myComputer = new Computer();

Declare a cart variable, and assign it a new ShoppingCart object. Note the use of the new keyword when instantiating the object.

# Step 46

Because of the way computers store and work with numbers, calculations involving decimal numbers can result in some strange behavior. For example, 0.1 + 0.2 is not equal to 0.3. This is because computers store decimal numbers as binary fractions, and some binary fractions cannot be represented exactly as decimal fractions.

We want to clean up the number result from your calculation. Wrap your calculation in parentheses (don't include the return statement!) and call the .toFixed() method on it. Pass the .toFixed() method the number 2 as an argument. This will round the number to two decimal places and return a string.

# Step 47

The issue with .toFixed() returning a string is that you want to be able to perform calculations with the tax rate. To fix this, you can pass the .toFixed() call (including the calculation) to the parseFloat() function. This will convert the fixed string back into a number, preserving the existing decimal places.

Pass your .toFixed() call to parseFloat().