IS 312: Web Design and Programming

PE06: Programming Exercise

> Revised: Fall 2022
> By: Tj Scharlau

Objective: Declare, Instantiate, and Extend a TypeScript Interface

For this exercise you will be declaring, instantiating, and extending an interface in TypeScript. There are three tasks for this exercise.

Task #1: Declaring an interface

1. Open your IDE and start a new project.
    1. You will first declare an interface named `Pizza` that includes two properties: `type` as a `string` and `slices` as a `number`.
2. You will next implement that interface.
    1. Start by using the `Pizza` interface as a type in a variable declaration.
    2. Declare a new variable called `myPizza` as *type* `Pizza`., and then assign values to the required properties.
    3. In VSCode you should see intellisense suggest the correct name and types.
    4. You can check your work by outputting one of the values to the console.
3. Take a screenshot of your code and include both the screenshot and your source code in your Lab Report under **Task 1**.

Task #2: Instancing an Interface and using a function to check a value

Task 2-1

For this task you will be creating a function that will compare the value of the slices in an instance of the `Pizza` interface and returning a message if the number is too high.

1. First, create a *function* called `checkSlices`.
    1. This uses the `Pizza` interface as a *`parameter type`* for the function
    2. You can call this parameter whatever you want, though something that keeps the pizza theme wouldn't be unusual.
2. This function should check the `number of slices` in the `Pizza object` and `returns` a message based on the result.
3. There are typically `8 slices in a pizza`, depending on how it's cut.
    1. Your function should consider `anything above 8` as too many slices and `output` that it's too many.
    2. Otherwise, it should return a message based on how many slices are left (1, 7/8, ¾, 5/8, ½, 3/8, ¼ or 1/8).

    3. This could simply be a `"There is _____ remaining of the pizza."`
4. You will run the code using an object passed into the function. This uses the same syntax as was used in PE04 for the slices, using `{}` and the name : `value pairs`.
5. Once you run your code, take a screenshot of both the code and the results, and put them in your lab report under `Task 2-1`.

Task 2-2

1. What happens if you add another property to the `Pizza` interface? Let's add a new property called `crust` and assign it the `string type`. Run your code, and report on the results. Include a screen shot of the results, and caption it `Task 2-2`.
2. Review the errors that appear in your code. To resolve them, you should add the `crust` property to the implementation details of the variable declaration and add it as a `parameter` passed to the function. For now, make the `crust` property optional by adding a question mark to the property name. All the errors should be resolved because the `crust` property is no longer required by the interface.

Task #3: Extending Interfaces

1. For this task you will declare a new interface called Toppings that extends the `Pizza` interface. Your extension should include the following parameters:
    - sauce with is a literal type 'tomato' | 'alfredo' | 'bbq sauce'
    - `pineapple` of type `Boolean` (optional)
    - `parmesan` of type `Boolean` (optional)
    - `crust` of type `Boolean`

Task 3-1

1. Note any errors or problems that TypeScripts mentions in the code, and correct them as appropriate.
2. ***Do not rename any of the properties.***
3. Take a screenshot of your source code and include it in your report under *Task 3*, with the caption Task 3-1.
4. Next change your `myPizza` variable to `type Toppings`.
    1. What happens when you do this?
    2. Why?
    3. Add your response and reasoning to your lab report under `Task 3`.
5. Now correct the error by modifying your `myPizza` variable by adding any missing required properties.
    1. You can also add any of the optional parameters that you want.
6. Now update your `checkSlices` function to use the `Toppings` interface.
    1. What happens when you update the code?
    2. What happens when you execute the code?
    3. Are there any errors?
        1. If so, fix them, and try running the code again.
        2. If not, why?
7. Take a screenshot of the output of your code and include your final source code in the word document.

Task 4: Learning & Challenges

1. In this task, you should simply discuss what you learned from the exercise. What was surprising, confusing, or problematic.
2. Add this write up to your Lab Report under *Task 4*.
3. You can include any screenshots if needed to clarify or demonstrate your point or issue.
4. Please caption any images in this section with `Task 4-#` (incrementing each # for each image).
5. Be sure to reference any screenshots by caption in your text.

Submission

To submit this assignment, you should submit your complete TypeScript source code file, and your Lab Report.