# JS Advanced: Regular Exam - 07.07.19

Exam problems for the ["JavaScript Advanced" course @ SoftUni](https://softuni.bg/courses/js-advanced). Submit your solutions in the SoftUni Judge system at <https://judge.softuni.bg/Contests/Compete/Index/1715#0>.

# Problem 2. PizzUni

### Use the provided PizzUni class to solve this problem.

### Your Task

Using **Mocha** and **Chai** write **unit** **tests** to test the entire functionality of the **PizzUni class**. Make sure instances of it have all the **required** **functionality** and **validation**. You may use the following code as a template:

describe(**"*Tests* …"**, **function**() {

describe(**"*TODO* …"**, **function**() {  
 ***it***(**"*TODO …*"**, **function**() {

*//* ***TODO:*** …

});

});  
 *//* ***TODO:*** …

#### });

### Functionality

The code inside the pizzuniClass.jsfile defines a class that contains information about an auto service. An instance of the class should support the following operations:

#### constructor()

An instance of **PizzUni class** should have the following **3** **properties**:

* **registeredUsers** - by default is an **empty** **array**
* **availableProducts** - by default isan **object** thatmust have only 2 properties:
  + **pizzas** - array with **3 strings:**("Italian Style", "Barbeque Classic" and "Classic Margherita")
  + **drinks** - array with **3 strings** ("Coca-Cola", "Fanta" and "Water")
* **orders** -by default is an **empty** **array**

#### registerUser({email})

This function **registers** a user by the given email (**string**) into **registeredUsers** **property**

* Receives a **string** (email)
* If the email is **already used** for registration before that, the expected behavior of the class is to **throw an error** with the following message:   
  `**This email address (${email}) is already being used!**`
* Otherwise, the given email is successfully registered to the **registeredUsers** **property** as an object. That object holds the given **email** and **orderHistory** which by default is an **empty** **array.**
* At the end, the **current** **object** should be **returned**.

#### makeAnOrder()

This function **makes** **an** **order** from an **already** **registered** user and **saves** the **current** **order** into their **orderHistory** property.

* Receives 3 parameters - **email**, **orderedPizza** and **orderedDrink**
  + If the incoming email is **NOT** registered in the **registeredUsers** **property**, the following error should be thrown:  
    "**You must be registered to make orders!**";
* An order is considered **valid** when the incoming **orderedPizza** is one of the **pizzas** in the **availableProducts** **property**. If it is not the following error is thrown:  
  "**You must order at least 1 Pizza to finish the order.**".
* The **orderedDrink** is **not required.** But if it is ordered, the drink should be one of the **drinks** in the **availableProducts** **property**.
* When the **order is valid** a new **object** which holds information about the current order (**the ordered pizza** and **the ordered drink, if any**) is pushed in the registered user **orderHistory** **property.** Also, a new **object** which holds information like (**ordered** **pizza**, **ordered** **drink** (if any), **email** and **status** which is **pending**) is pushed in the **orders** property.
* The following function **returns** a **number** (the **index** of the order in the **orders** array)

#### completeOrder()

This function **changes** the **status** of the **first** **object** with status "**pending**" to "**completed'** in the **orders** property and **returns** the **current** **object**.

#### detailsAboutMyOrder({id})

* Receives a **number** (id) which is the **index** for the current order
* If the current **order id** (index) is valid, the function returns the status of that order in format:  
  "**Status of your order: {pending/completed}**"

#### doesTheUserExist({email})

* Receives a **string** (email) and **returns** an **object** with all the registered users with that email

### Submission

Submit your tests inside a describe() statement, as shown above.

*GOOD LUCK!😊*