# JS Advanced: Exam Preparation 2

# Problem 1. Travel Agency

**Link on Judge: [https://judge.softuni.org/Contests/Practice/Index/3008#0](https://judge.softuni.org/Contests/Practice/Index/3008" \l "0)**

**Environment Specifics**

Please, be aware that every JS environment may **behave differently** when executing code. Certain things that work in the browser are not supported in **Node.js**, which is the environment used by **Judge**.

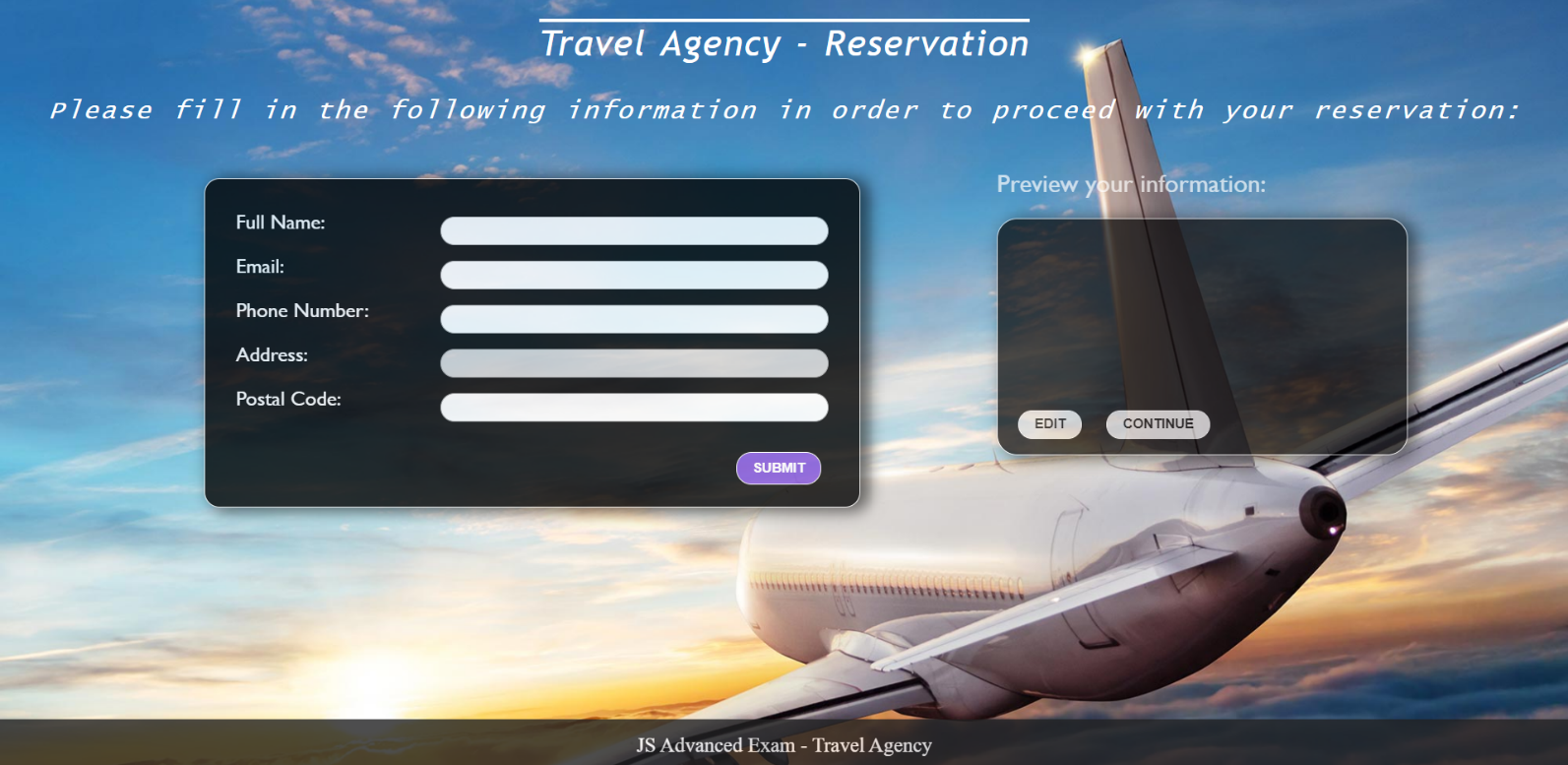
The following actions are **NOT** supported:

* **.forEach()** with **NodeList** (returned by **querySelector()** and **querySelectorAll()**)
* **.forEach()** with **HTMLCollection** (returned by **getElementsByClassName()** and **element.children**)
* Using the **spread-operator** (**...**) to convert a **NodeList** into an array
* **append()** in Judge (use only **appendChild()**)
* **replaceWith()** in Judge
* **replaceAll()** in Judge
* **closest()** in Judge
* **replaceChildren()**

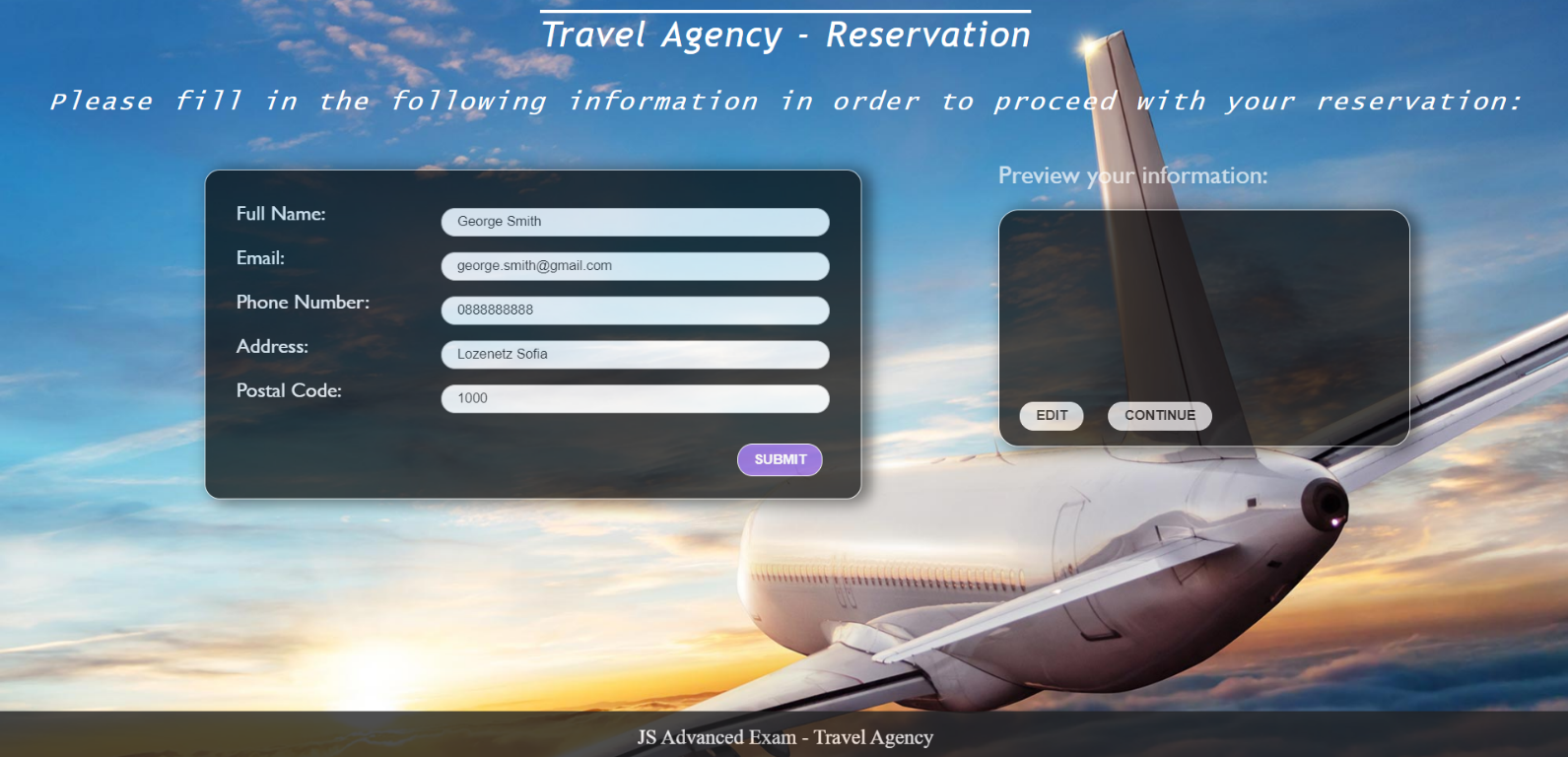
If you want to perform these operations, you may use **Array.from()** to first convert the collection into an array.

**Use the provided skeleton to solve this problem.**

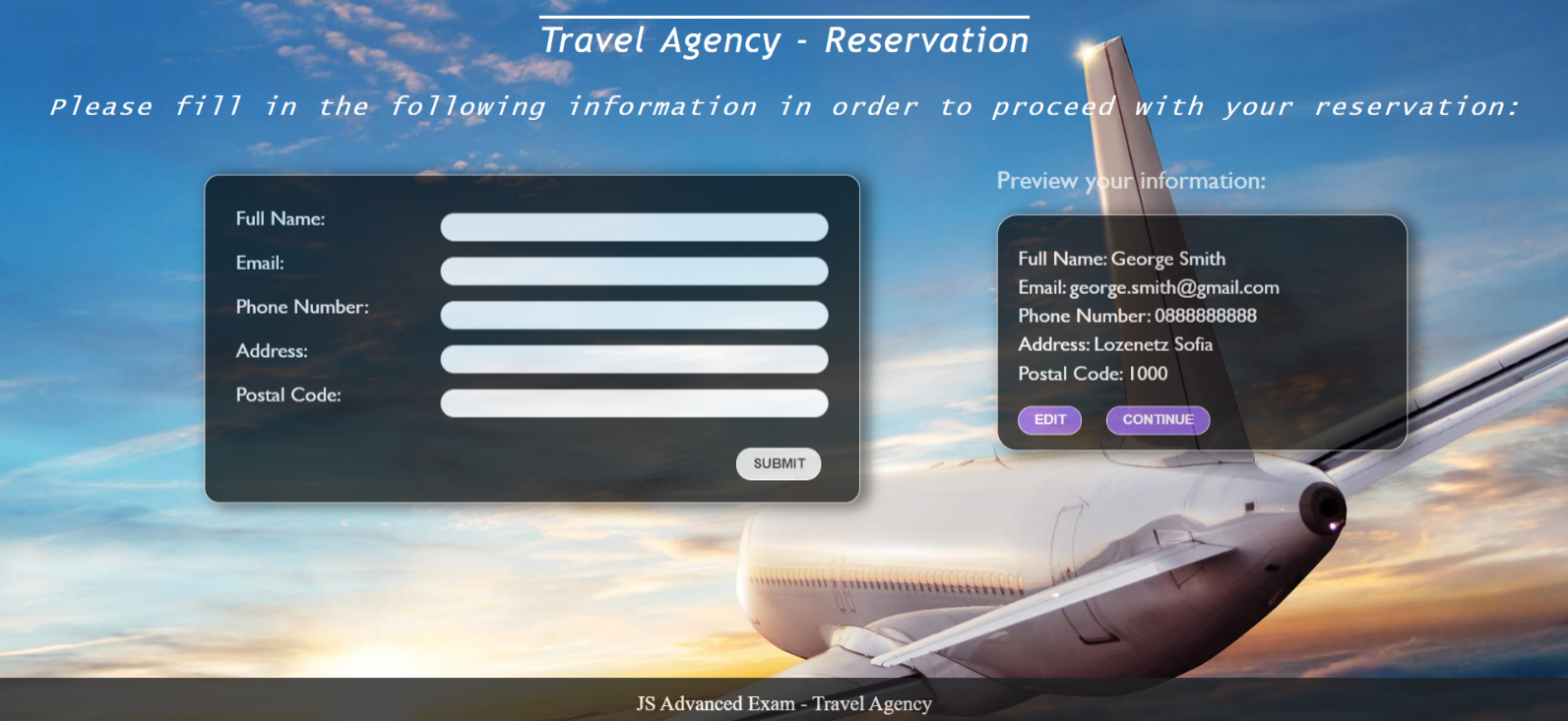
**Write the missing functionality** of this user interface. The functionality is divided in the following steps:



## ****Getting the user’s personal information****



* On clicking the “Submit” button the information from the input fields is listed in the “preview” section. For each input field a **list item** is **added** to the “infoPreview” unordered list.
* The text format and order for each list item should be the same as on the second picture below.
* When the button is clicked, the input fields must be cleared and the “Submit” button must be **disabled**. At the same time the “Edit” button and the “Continue” button must be **enabled**.
* One can only submit information if the **“Full Name”** and **“Email**” input fields are **not empty**.



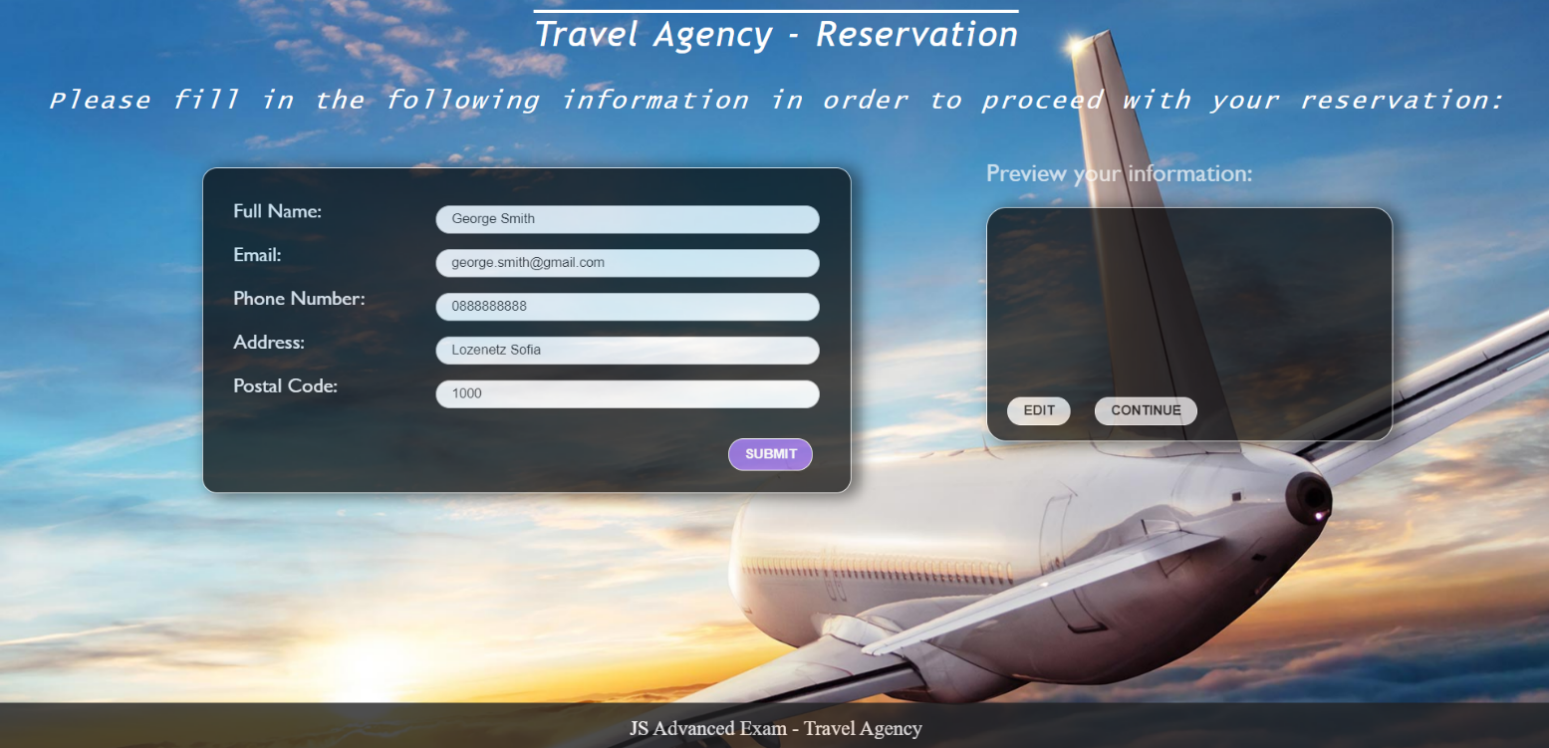
## ****Previewing the user’s personal information****

**This is an example for the preview section:**



**The functionality here is the following:**

* **When the “**Edit” **button is clicked, all of the personal information of the user is loaded in the input fields from step I and both the** “Edit” **and** “Continue” **buttons are disabled while the** “Submit” **button is enabled again.**

****

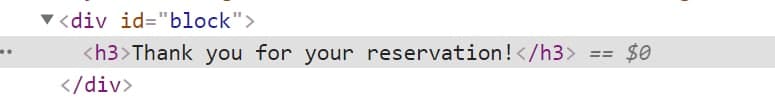
* **The list items must be removed from the** “infoPreview”**.**

****

* **When the** “Continue” **button is clicked,** **the reservation is completed. For you, this means removing everything inside of the** “block” **<div> and adding there only a <h3> tag: "Thank you for your reservation!" message:**

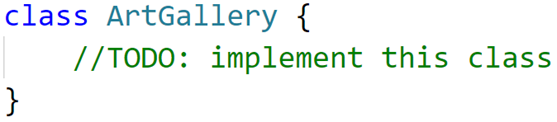
****

**This is everything your webpage must contain at this step:**



# Problem 2. Art Gallery

**Link on Judge: [https://judge.softuni.org/Contests/Practice/Index/3089#1](https://judge.softuni.org/Contests/Practice/Index/3089" \l "1)**



Write a **class Art Gallery**, which supports the described functionality below.

**Functionality**

**Constructor**

Should have these **4** properties:

* **creator - string**
* **possibleArticles - { "picture":200,"photo":50,"item":250 }**
* **listOfArticles - empty array**
* **guests - empty array**

**At the initialization** of the **ArtGallery** class, the **constructor** accepts only the **creator!**

The **possibleArticles** is an **object**, and the **submitted values** are **by default** and represent the **available article models** **("picture", "photo", "item")**, whichwill be displayed in the gallery and the necessary points for purchasing a specific article.

**Methods**

**addArticle( articleModel, articleName, quantity )**

This method adds article to the art gallery. Methodaccepts 3 arguments:

* **articleModel (string)**;
* **articleName (string);**
* **quantity (number)**;
* If the **articleModel**, is not present in **possibleArticles** object with specified default models, an error with the following message should be thrown:

**"This article model is not included in this gallery!"**

**Note** that the resulting **articleModel** argument can be submitted in both **lowercase and uppercase letters** and will **be correct**, and no error should be thrown see the **example below**:

**articleModel - ("picture") ->correct**

**articleModel - ("Picture") ->correct**

**articleModel - ("PICTURE") ->correct**

* If the **articleName** already exists in **listOfArticles array** andthe **articleModel is the same** just add the new quantity to the old one.
* Otherwise, should **add** the **articleModel, articleName, quantity** to **listOfArticles** arrayinfollowing **format**: **{articleModel, articleName, quantity}.** The **articleModel** must be **toLowerCase().**
* **And finally**, return the following message**:**

**"Successfully added article {articleName} with a new quantity- {quantity}."**

**inviteGuest ( guestName, personality)**

Accept 2 arguments: **guestName (string), personality (string)**

* If the **guestName** is already present in the **guests array**, throw a new error:

**"{guestName} has already been invited."**

* Otherwise, **create a new record** in the **guests array** in **following format: {guestName, points, purchaseArticle: default 0}.** Where the **points** are the **points that the guest has.** With them he can buy an article. They are **determined depending on personality** (see the table below).

**Example- (**"**Ivan**"**,** "**Vip**"**)** -> **the points are 500 [** If you get a **personality** that is **not present in the table below**, **put 50 points (**"**Petar**"**,** "**Normal**"**)->50 points)];**

**The property purchaseArticle** will record the number of **customer purchases, initially** at the invitation of the guest **the value is zero**.

* Finally, return the message:

**"You have successfully invited {guestName}!"**

|  |  |
| --- | --- |
| **Personality** | **Point** |
| **"Vip"** | **500** |
| **"Middle"** | **250** |

**buyArticle ( articleModel, articleName, guestName)**

Accept 3 arguments: **articleModel (string)**, **articleName (string)** and **guestName (string)**

* If the **articleName** is not found **in listOfArticles array or** the **articleModel doesn’t match**, throw a new error:

**"This article is not found."**

* If the **quantity** of the current **article is equal to 0,** return message:

**"The {articleName}** **is not available."**

* If the **guestName** is not present in the **guests array,** return message:

**"This guest is not invited."**

* Otherwise, you need to check if the **guest has the required number** of **points** to purchase the article. (The necessary points of the article are determined by the model in **possibleArticles array**)
  + If the **points** are **not enough to buy an article**, return the following message:

**"You need to more points to purchase the article."**

* + If **they are enough**, you need to **reduce the current points of the guest** by according to the points of model article in **possibleArticles array,** **reduce the quantity** of the current article and **increase the number of purchases** of the guest.
* Finally, return message:

**"{guestName} successfully purchased the article worth {articlePoint} points."**

The **articlePoint** is the value at which the article was purchased.

**showGalleryInfo (criteria)**

Accept 1 argument-**criteria.** This method **return gallery information** based on the criteria. Possible values for the **criterion** are two types:

* If the criterion is-**"article"-** then you need to **return** all the information about the articles saved in **listOfArticle** array in following format:
* On first line show the following message:

**"Articles information:"**

* On the lines, display information about each article:

**{articleModel} - {articleName} - {quantity}**

* If the criterion is-**"guest"-** then you need to **return** all the information about the guests saved in **guest** array in following format:
  + On first line show the following message:

**"Guests information:"**

* + On the lines, display information about each guest:

**{guestName} - {purchaseArticle}**

**Examples**

|  |
| --- |
| **Input 1** |
| **const *artGallery*** = **new** ArtGallery(**'Curtis Mayfield'**); ***console***.log(***artGallery*.addArticle**(**'picture', 'Mona Liza', 3**));  ***console***.log(***artGallery*.addArticle**(**'Item', 'Ancient vase', 2**));  ***console***.log(***artGallery*.addArticle**(**'PICTURE', 'Mona Liza', 1**)); |

|  |
| --- |
| **Output 1** |
| Successfully added article Mona Liza with a new quantity- 3.  Successfully added article Ancient vase with a new quantity- 2.  Successfully added article Mona Liza with a new quantity- 1. |

|  |
| --- |
| **Output 2** |
| You have successfully invited John!  You have successfully invited Peter!  John has already been invited. |

|  |
| --- |
| **Input 2** |
| **const *artGallery*** = **new** ArtGallery(**'Curtis Mayfield'**); ***console***.log(***artGallery*.inviteGuest**(**'John', 'Vip'**));  ***console***.log(***artGallery*.inviteGuest**(**'Peter', 'Middle'**));  ***console***.log(***artGallery*.inviteGuest**(**'John', 'Middle'**)); |

|  |
| --- |
| **Input 3** |
| **const *artGallery*** = **new** ArtGallery(**'Curtis Mayfield'**);  ***artGallery*.addArticle**(**'picture', 'Mona Liza', 3**);  ***artGallery*.addArticle**(**'Item', 'Ancient vase', 2**);  ***artGallery*.addArticle**(**'picture', 'Mona Liza', 1**);  ***artGallery*.inviteGuest**(**'John', 'Vip'**);  ***artGallery*.inviteGuest**(**'Peter', 'Middle'**); ***console***.log(***artGallery*.buyArticle**(**'picture', 'Mona Liza', 'John'**));  ***console***.log(***artGallery*.buyArticle**(**'item', 'Ancient vase', 'Peter'**));  ***console***.log(***artGallery*.buyArticle**(**'item', 'Mona Liza', 'John'**)); |

|  |
| --- |
| **Output 3** |
| John successfully purchased the article worth 200 points.  Peter successfully purchased the article worth 250 points.  This article is not found. |

|  |
| --- |
| **Input 4** |
| **const *artGallery*** = **new** ArtGallery(**'Curtis Mayfield'**);  ***artGallery*.addArticle**(**'picture', 'Mona Liza', 3**);  ***artGallery*.addArticle**(**'Item', 'Ancient vase', 2**);  ***artGallery*.addArticle**(**'picture', 'Mona Liza', 1**);  ***artGallery*.inviteGuest**(**'John', 'Vip'**);  ***artGallery*.inviteGuest**(**'Peter', 'Middle'**); ***artGallery*.buyArticle**(**'picture', 'Mona Liza', 'John'**);  ***artGallery*.buyArticle**(**'item', 'Ancient vase', 'Peter'**);  ***console***.log(***artGallery*.showGalleryInfo**(**'article'**));  ***console***.log(***artGallery*.showGalleryInfo**(**'guest'**)); |

|  |
| --- |
| **Output 4** |
| Articles information:  picture - Mona Liza - 3  item - Ancient vase - 1  Guests information:  John - 1  Peter - 1 |

# Problem 3. Unit Testing

**Link on Judge: [https://judge.softuni.org/Contests/Practice/Index/3235#2](https://judge.softuni.org/Contests/Practice/Index/3235" \l "2)**

**Your Task**

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **library**, which represents an object. You may use the following code as a template:

|  |
| --- |
| describe(**"*Tests* …"**, **function**() {  describe(**"*TODO* …"**, **function**() {  ***it***(**"*TODO …*"**, **function**() {  *//* ***TODO:*** …  });  });  *//* ***TODO:*** …  }); |

The object that should have the following functionality:

* **calcPriceOfBook (nameOfBook, year) -** A function that accepts a string and a number:
* The function calculates the price of the book depending on the **year** of publication
* The standard price of the book is 20 BGN
* If the **year** of publication is **less** than or **equal** to **1980**, there is a **50%** percent discount from the standard price
* The function calculated price of the book and **return**: **`Price of {nameOfBook} is {price}`**
* You need to validate the input, if **nameOfBook** is not a string, or the **year** is not an **integer** number, **throw** an error: "**Invalid input**"
* **findBook (booksArr,** **desiredBook)**- A function that accepts an array and string:
  + The array includes all available **books** in the library ([**"Troy", "Life Style", "Torronto", etc.**])
* If the length of the **booksArr** array is zero, **throw** an error in the following format: "**No books currently available**"
  + The function checks whether the submitted string **desiredBook** is present in the array **booksArr**.
* If present in the array, the function **return**: "**We found the book you want.**"
* Otherwise the function **return**: "**The book you are looking for is not here!**"
* There is no need for validation for the input, you will always be given an array and string
* **arrangeTheBooks (countBooks)** - A function accept a number:
  + You need to validate the input, if the **countBooks** is not an **integer** number, or is a negative number, **throw** an error: "**Invalid input**"
  + The library has 5 **shelves**, each shelf can hold 8 books. Distribute the books on the shelves
  + If all the books are arranged on the shelves, **return**: "**Great job, the books are arranged.**"
  + Otherwise, if no space has been reached, **return**: "**Insufficient space, more shelves need to be purchased.**"

**JS Code**

To ease you in the process, you are provided with an implementation that meets all of the specification requirements for the **library** object:

|  |
| --- |
| library.js |
| const library = {      calcPriceOfBook(nameOfBook, year) {          let price = 20;          if (typeof nameOfBook != "string" || !Number.isInteger(year)) {              throw new Error("Invalid input");          } else if (year <= 1980) {              let total = price - (price \* 0.5);              return `Price of ${nameOfBook} is ${total.toFixed(2)}`;          }          return `Price of ${nameOfBook} is ${price.toFixed(2)}`;      },      findBook: function(booksArr, desiredBook) {          if (booksArr.length == 0) {              throw new Error("No books currently available");          } else if (booksArr.find(e => e == desiredBook)) {              return "We found the book you want.";          } else {              return "The book you are looking for is not here!";          }      },      arrangeTheBooks(countBooks) {          const countShelves = 5;          const availableSpace = countShelves \* 8;          if (!Number.isInteger(countBooks) || countBooks < 0) {              throw new Error("Invalid input");          } else if (availableSpace >= countBooks) {              return "Great job, the books are arranged.";          } else {              return "Insufficient space, more shelves need to be purchased.";          }      }  }; |

**Submission**

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