# JS Advanced: Exam Preparation 1

# Problem 1. Furniture Store

**Link in Judge:** [**https://judge.softuni.org/Contests/Practice/Index/3860#0**](https://judge.softuni.org/Contests/Practice/Index/3860#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.**

**Note**: You **can't** and you have no permission to **change** directly the given HTML code (index.html file).

**Write the missing JavaScript code** to make the **Furniture** **Store** work as expected:



**Your Task**

* A**ll fields (model, year, description, and price)** are **filled with the correct input**
  + **Model and description** are **non**-**empty** **strings**
  + **Year** and **Price** need to be **positive** **numbers**
  + **All fields must be filled**

1. **Getting the furniture information**



* When you click the “Add” button, the information from the input fields must be added to the table and then clear input fields.
* The table contains **Model, Price of furniture** and **Actions** - **[More information], [Buy it]**. **The price** must be **rounded** to the **second** digit after the decimal point.



**Each furniture** must be appended to **"furniture-list"** and look like the picture below: 

Each piece of furniture has the main information line **(Model, Price)** and an additional information line. The additional information line stores **the description and year** of manufacture of the furniture **(hidden until the "More info" button is pressed**)**.**

When the **"More Info"** button is clicked, change button text from **"More Info"** to **"Less Info"** and style display of **"class = hide"** from **"none "** to **"contents".** The second **<td>** must-have attribute **colspan** with value **3.** When click **"Less Info"** button is clicked, change button text from **"Less Info"** to **"More Info"** and style from **"contents "** to **"none".**



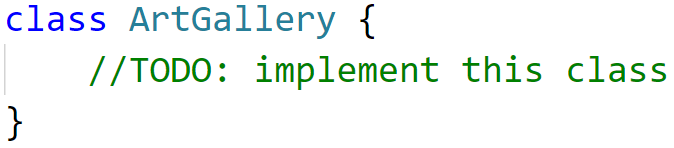


When the **"Buy it"** button is clicked, should have the following **functionality:**

* **The current furniture** must be **removed from the row** on the table
* You need to **change** the **total** profit in **the store.** Take the element with class **"total-price"** **and increase the current total price** **with** the **price of the furniture**.



## Problem 2. Art Gallery

**Link in Judge:** [**https://judge.softuni.org/Contests/Practice/Index/3860#1**](https://judge.softuni.org/Contests/Practice/Index/3860#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. |

|  |
| --- |
| **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'**)); |

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

|  |
| --- |
| **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. Cinema**

**Link in Judge:** [**https://judge.softuni.org/Contests/Practice/Index/3860#2**](https://judge.softuni.org/Contests/Practice/Index/3860#2)

**Your Task**

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **cinema**, 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:

* **showMovies(movieArr)**-A function that accepts an array:
* The array includes the available movies in the cinema ([‘King Kong’, ‘The Tomorrow War’, ‘Joker’,etc.])
* If the length of the input array is zero, the function returns the string: "**There are currently no movies to show.**"
* Otherwise, the function returns: an array of available movies, separated by a comma and space
* There is no need for validation for the input, you will always be given an array
* **ticketPrice(projectionType)**- A function that accept string:
  + The function checks whether the submitted projectionType is present in the schedule with the types of projections
  + If present in the schedule, return the price according to the type
* Otherwise, the function throws an error in the following format "**Invalid projection type.**"
* There is no need for validation for the input
* **swapSeatsInHall(firstPlace, secondPlace)**- A function that accepts two numbers
* The function swaps the seat number in the cinema hall, when two numbers are submitted.
* The exchange is not successful and the function returns "**Unsuccessful** **change of seats in the hall.**" :
  + If one of the two numbers do not exist
  + The numbers are not integers
  + If one of the numbers is greater than the capacity of the hall – **20**
  + Seats are less or equal of **0**
* Otherwise return: "**Successful change of seats in the hall.**"
* There is a need for validation for the input

**JS Code**

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

|  |
| --- |
| cinema.js |
| const cinema = {  showMovies: function(movieArr) {  if (movieArr.length == 0) {  return 'There are currently no movies to show.';          } else{           let result = movieArr.join(', ');              return result;          }      },    ticketPrice: function(projectionType) {    const schedule = {              "Premiere": 12.00,              "Normal": 7.50,              "Discount": 5.50          }          if (schedule.hasOwnProperty(projectionType)) {              let price = schedule[projectionType];              return price;          } else {              throw new Error('Invalid projection type.')          }      },    swapSeatsInHall: function(firstPlace, secondPlace) {          if (!Number.isInteger(firstPlace) || firstPlace <= 0 || firstPlace > 20 ||  !Number.isInteger(secondPlace) || secondPlace <= 0 || secondPlace > 20 || firstPlace === secondPlace) {              return "Unsuccessful change of seats in the hall.";          } else {              return "Successful change of seats in the hall.";          }      }  }; |

**Submission**

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