# JS Advanced: Exam Preparation 2

**Link to contest**: <https://judge.softuni.org/Contests/4535/JS-Advanced-Regular-Exam-17-February-2024>

# Problem 01. Snow Showdown

**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()**)
* **prepend()**
* **replaceWith()**
* **replaceAll()**
* **closest()**
* **replaceChildren()**
* **concat()**
* **at()**
* Always turn the collection into a **JS array** (forEach, forOf, et.)

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:

## Your Task

**Write the missing JavaScript code** to make the **Snow Showdown** work as expected:

All fields **(Snowman Name, Height (in centimeters), Location, Creator,** and **Special Attribute)** are **filled with the correct input**

* **Snowman Name, Height (in centimeters), Location, Creator,** and **Special Attribute** are **non**-**empty** **strings**. If any of them is empty, the program should not do anything.

## Getting the information from the form



• When the **["Add"]** button is clicked, the information from the input fields is listed in the "**Snowman Preview**" section. A **list item** is added to the **"snowman-preview "** unordered list.

* The list item should follow the same text format and order as shown in the provided picture.
* When the button is clicked, the **input** fields must be **cleared**, and the **["Add"]** button should be **disabled**. Additionally, the **"Edit"** and **"Next"** buttons need to be added.

The HTML structure looks like this:

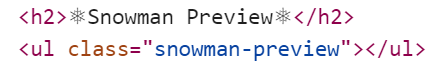


## Edit Snowman

**The functionality here is the following:**

* **When the "**Edit" **button is clicked, all of the information is loaded in to the input fields from step 1 and all the buttons in Snowman Preview section are removed while the** ["Add"] **button is enabled again.**

****

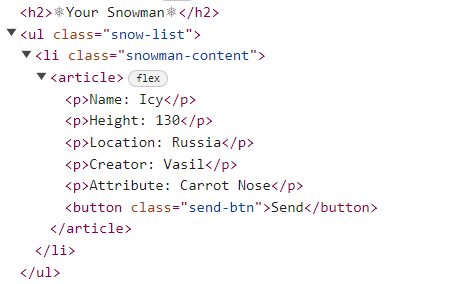
* **The list items must be removed from the** "snowman-preview" ****

## Next

* **When the** "Next" **button is clicked,** **the information from** "snowman-preview" unordered list must be transferred to "snow-list". **For you, this means removing everything inside of the <ul> with class =** "snowman-preview" **and adding in** "snow-list", **the list item with same information** The "Send" button must be **added.**

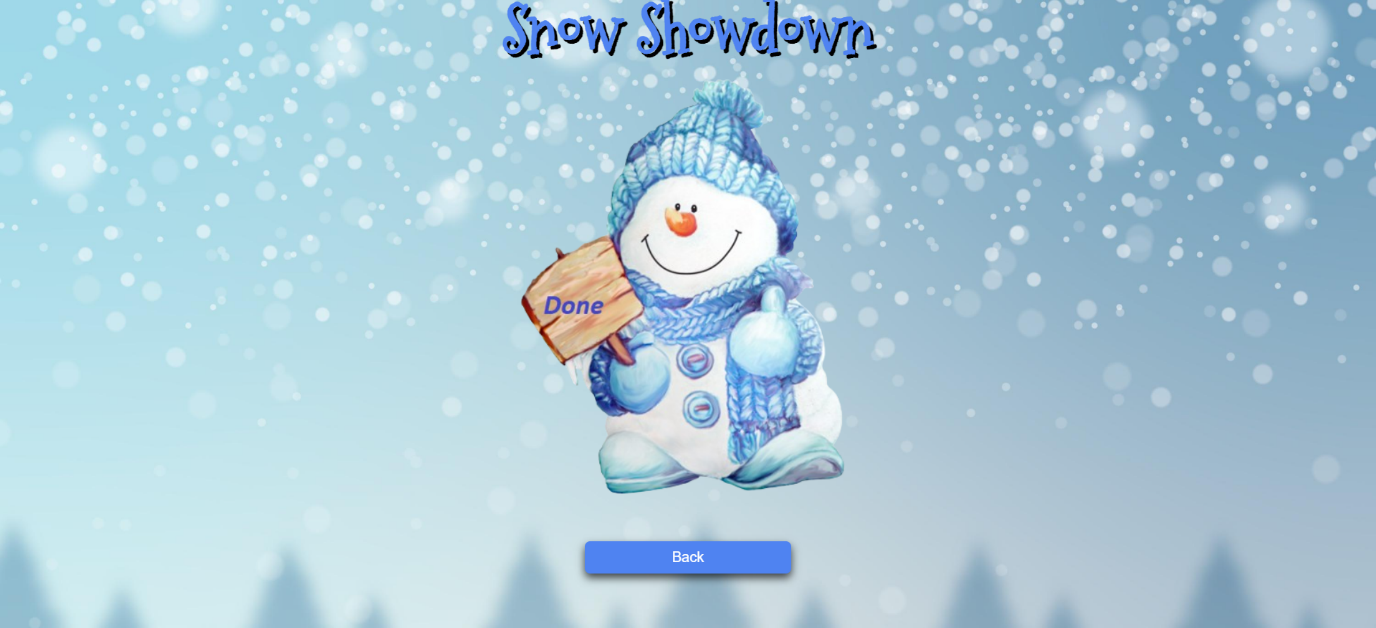
****

* **This is the HTML** structure **of the** "snow-list" unordered list**:**



## Sending

* **When the** "Send" **button is clicked,** you must remove **main** element from the HTML structureand then **add** "Back" **button** to the **body** and make **visible** the image with **id "back-img"**

****

* **This is the HTML** structure **of the** body**:**



## Back

* **When the** "Back" **button is clicked,** you must **reload** the page.

****

## Submission

Submit only yours **solve()** function.

# Problem 02. BookClub

class BookClub {

//TODO Implement this class

}

Write a class **BookClub**, which implements the following functionality:

## Functionality

**Constructor**

Should have these **3** properties:

* **library –** String
* **books –** Array (empty)
* **members** **–** Array (empty)

The constructor should accept the initial **library** name when the **BookClub** is initialized.

### addBook (title, author)

Both **title** and **author** are of type **string**.

If the book is already in the books array, **return:**

"**The book "{title}" by {author} is already in {library} library.**"

* Otherwise, this function should add the book with the properties: **title** and **author** to the books array, and **return**:

"**The book "{title}" by {author} has been added to {library} library.**"

### addMember (memberName)

The **memberName** is of type **string**.

* If the member is already a part of the book club, **return**:

"**Member {memberName} is already a part of the book club.**"

* Otherwise, this function should **add** the member to the **members** array and **return:**

"**Member {memberName} has been joined to the book club.**"

### assignBookToMember (memberName, bookTitle)

* If the **memberName** is **not found** in **members** array, throw **an Error:**

"**Member {memberName} not found.**"

* If the **bookTitle** is **not found** in **books** array, throw **an Error:**

"**Book "{bookTitle}" not found.**"

* Otherwise, this function should **remove the book** from the books array**,** and **return:**

"**Member {memberName} has been assigned the book "{assignedBook}" by {author}.**"

### generateReadingReport ()

This method should **return** the complete information about the book club:

* If the club does **not** have any members, **return:**

**"No members in the book club."**

* If there are no available books, **return:**

"**No available books in the library.**"

* Otherwise **return**:

"**Available Books in {library} library:**"

* And on the **new line**, display information about each book in the club library:

**""{book}" by {author}"**

## Examples

|  |
| --- |
| **Input 1** |
| const myBookClub = new BookClub('The Bookaholics');  console.log(myBookClub.addBook("The Great Gatsby", "F. Scott Fitzgerald"));  console.log(myBookClub.addBook("To Kill a Mockingbird", "Harper Lee"));  console.log(myBookClub.addBook("1984", "George Orwell"));  console.log(myBookClub.addMember("Alice"));  console.log(myBookClub.addMember("Peter"));  console.log(myBookClub.assignBookToMember("Mary", "The Great Gatsby")); |

|  |
| --- |
| **Output 1** |
| The book "The Great Gatsby" by F. Scott Fitzgerald has been added to The Bookaholics library.  The book "To Kill a Mockingbird" by Harper Lee has been added to The Bookaholics library.  The book "1984" by George Orwell has been added to The Bookaholics library.  Member Alice has been joined to the book club.  Member Peter has been joined to the book club.  Uncaught Error Error: Member Mary not found. |

|  |
| --- |
| **Input 2** |
| const myBookClub = new BookClub('The Bookaholics');  console.log(myBookClub.addBook("The Great Gatsby", "F. Scott Fitzgerald"));  console.log(myBookClub.addBook("To Kill a Mockingbird", "Harper Lee"));  console.log(myBookClub.addBook("1984", "George Orwell"));  console.log(myBookClub.addMember("Alice"));  console.log(myBookClub.addMember("Alice"));  console.log(myBookClub.assignBookToMember("Alice", "The Great Gatsby"));  console.log(myBookClub.generateReadingReport()); |

|  |
| --- |
| **Output 2** |
| The book "The Great Gatsby" by F. Scott Fitzgerald has been added to The Bookaholics library.  The book "To Kill a Mockingbird" by Harper Lee has been added to The Bookaholics library.  The book "1984" by George Orwell has been added to The Bookaholics library.  Member Alice has been joined to the book club.  Member Alice is already a part of the book club.  Member Alice has been assigned the book "The Great Gatsby" by F. Scott Fitzgerald.  Available Books in The Bookaholics library:  "To Kill a Mockingbird" by Harper Lee  "1984" by George Orwell |

|  |
| --- |
| **Input 3** |
| const myBookClub = new BookClub('The Bookaholics');  console.log(myBookClub.addBook("The Great Gatsby", "F. Scott Fitzgerald"));  console.log(myBookClub.addBook("To Kill a Mockingbird", "Harper Lee"));  console.log(myBookClub.addBook("1984", "George Orwell"));  console.log(myBookClub.addMember("Alice"));  console.log(myBookClub.addMember("Peter"));  console.log(myBookClub.assignBookToMember("Peter", "The Da Vinci Code")); |

|  |
| --- |
| **Output 3** |
| The book "The Great Gatsby" by F. Scott Fitzgerald has been added to The Bookaholics library.  The book "To Kill a Mockingbird" by Harper Lee has been added to The Bookaholics library.  The book "1984" by George Orwell has been added to The Bookaholics library.  Member Alice has been joined to the book club.  Member Peter has been joined to the book club.  Uncaught Error Error: Book "The Da Vinci Code" not found. |

|  |
| --- |
| **Input 4** |
| const myBookClub = new BookClub('The Bookaholics');  console.log(myBookClub.addBook("To Kill a Mockingbird", "Harper Lee"));  console.log(myBookClub.addBook("1984", "George Orwell"));  console.log(myBookClub.addMember("Alice"));  console.log(myBookClub.addMember("Peter"));  console.log(myBookClub.assignBookToMember("Peter", "1984"));  console.log(myBookClub.assignBookToMember("Alice", "To Kill a Mockingbird"));  console.log(myBookClub.generateReadingReport()); |

|  |
| --- |
| **Output 4** |
| The book "To Kill a Mockingbird" by Harper Lee has been added to The Bookaholics library.  The book "1984" by George Orwell has been added to The Bookaholics library.  Member Alice has been joined to the book club.  Member Peter has been joined to the book club.  Member Peter has been assigned the book "1984" by George Orwell.  Member Alice has been assigned the book "To Kill a Mockingbird" by Harper Lee.  No available books in the library. |

# Problem 3.Plan Your Trip

## Your Task

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

* **choosingDestination (destination, season, year) -** A function that accepts **three** parameters: **string**, **string**, and **number**.
* If the **year** is different than **2024**, **throw** an error: **"Invalid Year!"**
* If the value of the string **destination** is different from "**Ski Resort**", **throw** an error:

**"This destination is not what you are looking for."**

* To be picked, the **destination** must meet the **following requirement**:
  + If the **season** is **Winter**, **return** the string:

**"Great choice! The ${season} is the perfect time to visit the ${destination}."**

* Otherwise, if the above condition in **not** met, **return** the following message:

**"Consider visiting during the Winter for the best experience at the ${destination}."**

* There is **no** need for **validation** for the **input**, you will always be given two strings, and number.
* **exploreOptions (activities, activityIndex) -** A function that accepts an **array** and **number**. The **activities** array will store the different activities (["**Skiing** ", "**Snowboarding** ", "**Winter Hiking** "…]).
  + You must **remove** an **element** (activity) from the **array** that is located on the **index** specified as a parameter.
  + Finally, **return** the changed array of activities as a string, joined by a comma and a space.
  + There is a **need for validation** for the input, an **array** and index may not always be valid. In case of submitted **invalid** parameters, **throw** an error **"Invalid Information!"**:
    - If passed **activities** parameteris **not** an array.
    - If the **activityIndex** is not a number and is outside the limits of the array.
    - If the **activityIndex** is not a integer number.
* **estimateExpenses (distanceInKilometers, fuelCostPerLiter) -** A function that accepts two parameters: **number, number**.
* You need to **calculate** the **cost** of the travel.
  + - **The result must be formatted to the second digit after the decimal point.**
* If the total cost is **less** or **equal** to $500. **return** the following message:

**"The trip is budget-friendly, estimated cost is $${totalCost}."**

* Else, **return** the following message:

**"The estimated cost for the trip is $${totalCost}, plan accordingly."**

* You **need to validate** the input, if the **distanceInKilometers** and **fuelCostPerLiter** are not **numbers**, or are **negative** numbers or **zero**, **throw** an error: **"Invalid Information!"**.

## JS Code

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

|  |
| --- |
| planYourTrip.js |
| const planYourTrip = {    choosingDestination(destination, season, year) {      if (year != 2024) {        throw new Error(`Invalid Year!`);      } else {        if (destination == "Ski Resort") {          if (season === "Winter") {            return `Great choice! The ${season} is the perfect time to visit the ${destination}.`;          } else {            return `Consider visiting during the Winter for the best experience at the ${destination}.`;          }        } else {          throw new Error(`This destination is not what you are looking for.`);        }      }    },    exploreOptions(activities, activityIndex) {      let result = [];      if (        !Array.isArray(activities) ||        !Number.isInteger(activityIndex) ||        activityIndex < 0 ||        activityIndex >= activities.length      ) {        throw new Error("Invalid Information!");      }      for (let i = 0; i < activities.length; i++) {        if (i !== activityIndex) {          result.push(activities[i]);        }      }      return result.join(", ");    },    estimateExpenses(distanceInKilometers, fuelCostPerLiter) {      let totalCost = (distanceInKilometers \* fuelCostPerLiter).toFixed(2);      if (        typeof distanceInKilometers !== "number" ||        distanceInKilometers <= 0 ||        typeof fuelCostPerLiter !== "number" ||        fuelCostPerLiter <= 0      ) {        throw new Error("Invalid Information!");      } else if (totalCost <= 500) {        return `The trip is budget-friendly, estimated cost is $${totalCost}.`;      } else {        return `The estimated cost for the trip is $${totalCost}, plan accordingly.`;      }    },  }; |

## Submission

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