# JS Advanced: Exam Preparation 1

**Link to contest**: <https://judge.softuni.org/Contests/3565>

**Problem 1. Dish Manager**

**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
* **replaceChildren()** in Judge
* **replaceAll()** in Judge
* **closest()** in Judge

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).

Картина, която съдържа текст

Описанието е генерирано автоматично

**Your Task**

**Write the missing JavaScript code** to make the **Dish** **Manager** work as expected:

All fields **(First Name, Last Name, Age** and **Dish description)** are **filled with the correct input**

* + **First Name, Last Name, Age** and **Dish description** are **non**-**empty** **strings**. If any of them are empty, the program should not do anything.

1. **Getting the information from the form**

When you click the **["Submit"]** button, the information from the input fields must be added to the **ul** with the **id** **"in-progress"** and **then clear input fields**. Also the counter with **id "progress-count"** should be increased by 1.

The HTML structure looks like this:

Картина, която съдържа текст

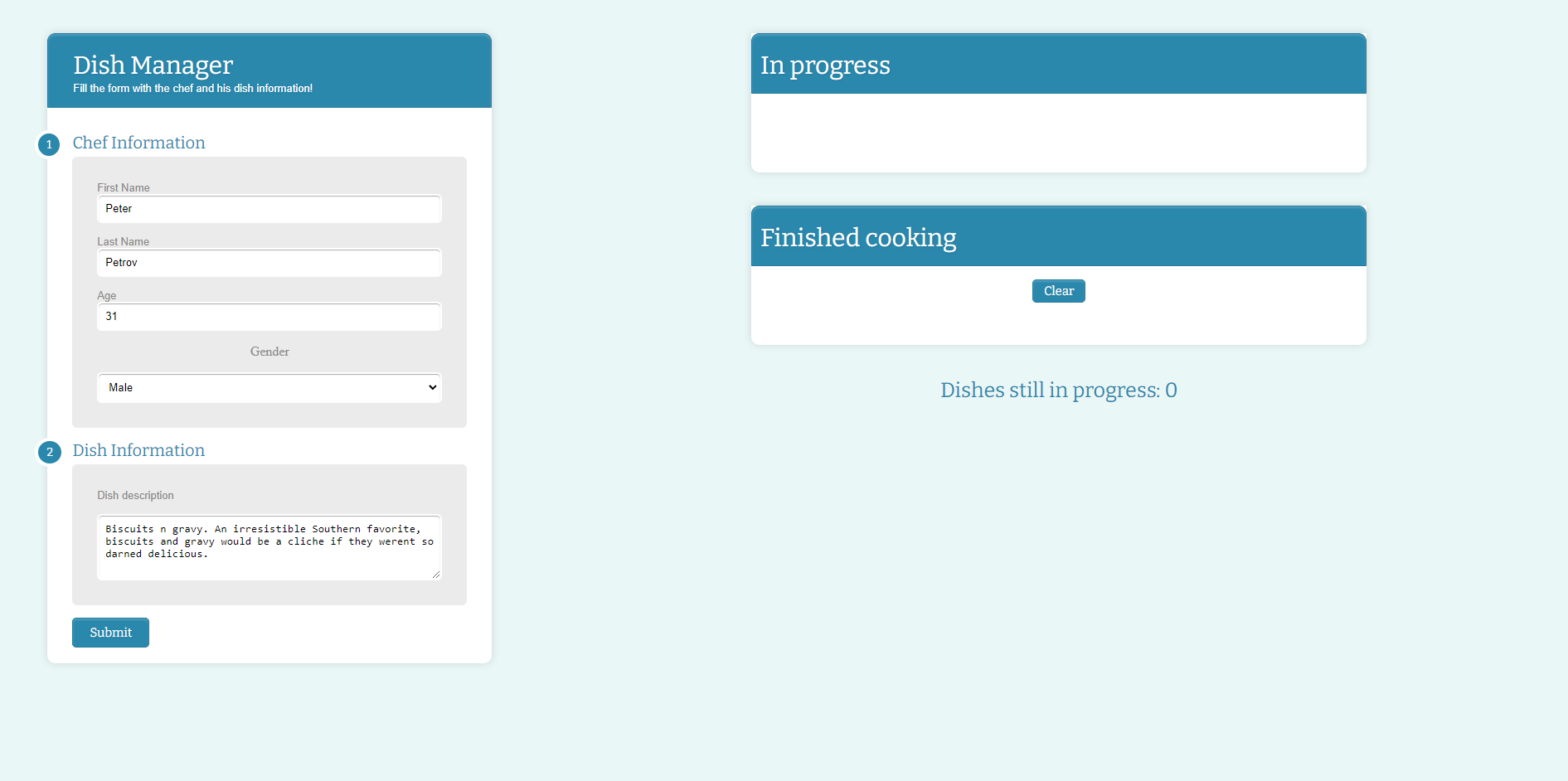
Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматично

1. **Edit information for posts**

When the **["Edit"]** button is clicked, the information from the post must be sent to the input’s fields and the record should be deleted from the **ul** **"in-progress"**. Also the counter with **id "progress-count"** should be decreased by 1.



After editing the information make a new record to the **ul** with updated information and increase the counter.

Картина, която съдържа текст

Описанието е генерирано автоматично

1. **Complete posts**

When you click the **["Mark as complete"]** button, the record must be **deleted** from the **ul** with **id**

**"in-progress"** , appended to the **ul** with the **id** **"finished"** and the counter with **id "progress-count"** should be decreased by 1.

The **buttons [“Edit”]** and **[“Mark as complete”]** should be removed from the **li** element.

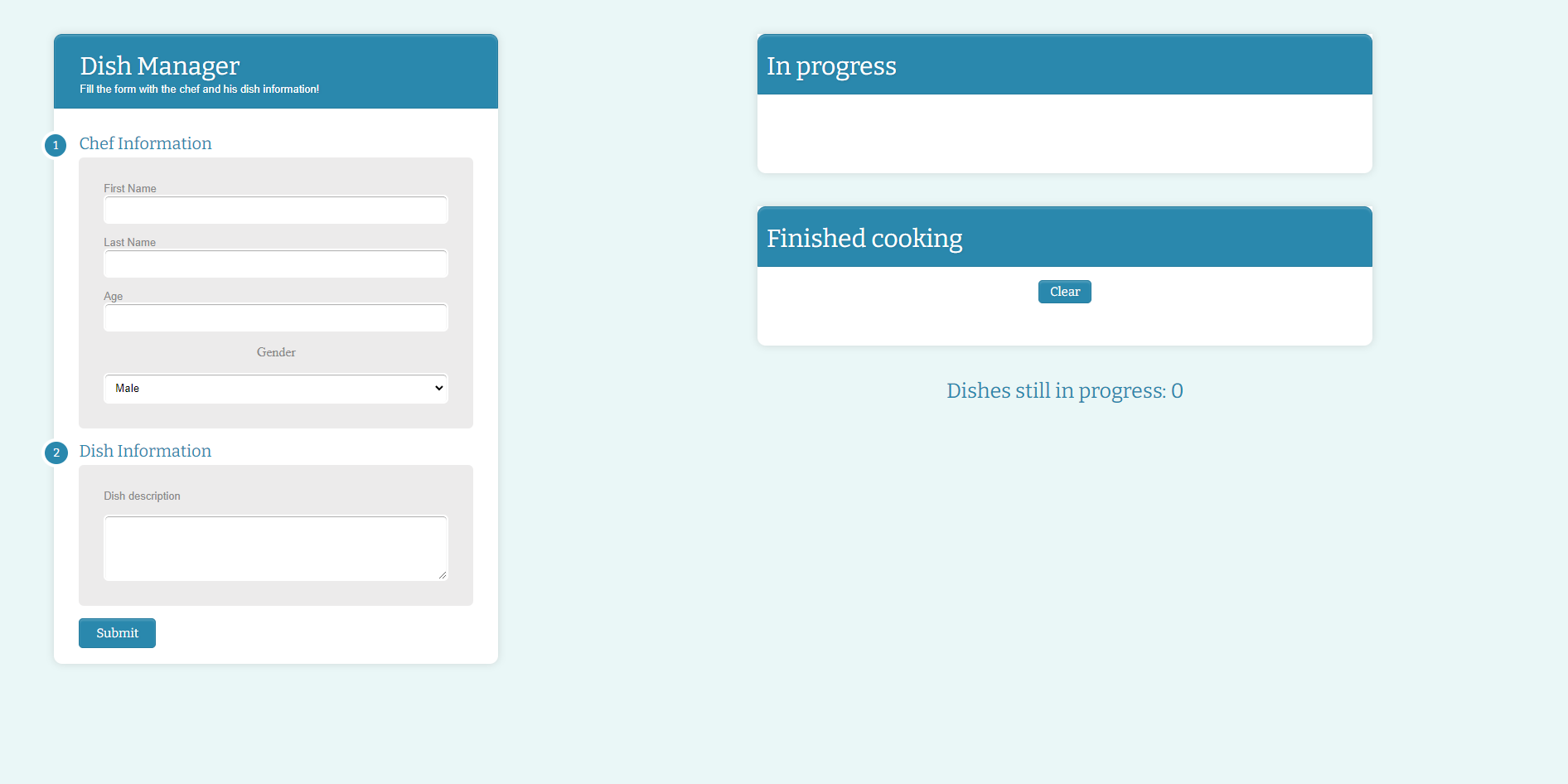
Картина, която съдържа текст

Описанието е генерирано автоматично Картина, която съдържа текст

Описанието е генерирано автоматично

1. **Clear posts**

When you click the **["Clear"]** button, the record for all posts must be **deleted** from the **ul** with the **id** **"finished".**



# Problem 2.Triathlon

## Triathlon Class

class Triathlon {

//TODO: Implement this class…

}

Implement a **class Triathlon**, which supports the functionality described below.

### Functionality

### Constructor

The constructor has **3** properties:

* **competitionName – a string**
* **participants – an empty object**
* **listOfFinalists – an empty array**

**At the initialization** of the **Triathlon** class, the **constructor** accepts only one parameter **competitionName** which is a **string**!

The **participants** property is an **object**, representing a key-value pair of a participant name and its gender.

### Methods

#### addParticipant (participantName, participantGender)

This method adds a new participant to the **participants** object. The methodaccepts 2 arguments:

* **participantName – a string**
* **participantGender – a string**

If the **participant exists in the participants object**, **return** the following message:

**`${participantName} has already been added to the list`**

Otherwise, **add the new participant** to the **participants** object in the following format **{‘Peter’ : ‘male’}** and **return** the following message:

**`A new participant has been added - ${participantName}`**

#### completeness (participantName, condition)

Accept 2 arguments:

* **participantName – a string**
* **condition – a number (0 – 100)**

If the participant doesn’t exist in the participants object, **throw new Error**:

**`$****{participantName} is not in the current participants list`**

There are three disciplines in the triathlon and each requires the participant condition to be at least **30** for a discipline to be completed.

If the participant exists in the participants object but his condition is lower than **30**, **throw new Error**:

**`${participantName} is not well prepared and cannot finish any discipline`**

Afterwards, find how many disciplines is the participant able to complete depending on his **condition** (**condition divided by 30**). If they completed only one or two disciplines, **return** the following message:

**`${participantName} could only complete ${completedCount} of the disciplines`**

Otherwise extract the participant from the participants object and add him to the **list of finalists** in the following format: **{ participantName, participantGender }**

**Return** the following message:

**`Congratulations, ${participantName} finished the whole competition`**

#### rewarding (participantName)

Accept 1 argument:

* **participantName – a string**

If the **participantName** is not present in the **list of finalists, return** the following message:

**`${participantName} is not in the current finalists list`**

Otherwise, **return** the following message:

**`${participantName} was rewarded with a trophy for his performance`**

#### showRecord (criteria)

Accept 1 argument:

* **criteria – a string**

This method **returns information** based on the criteria. The three possible types of criteria are: **"male"**, **"female"** or **"all".**

If the **list of finalists is empty**, return the following message:

**`There are no finalists in this competition`**

If there are no finalists with the given **criteria**, **return**:

**`There are no ${participantGender}'s that finished the competition`**

If there are finalists with the given **criteria, return** a message with the **firstly added participant** in the following format**:**

**`${participantName} is the first ${criteria} that finished the ${competitionName} triathlon`;**

Otherwise if the criteria is **all,** returnall finalists from the **list of finalists** array in following format:

* On first line show the following message:

**`List of all ${competitionName} finalists:`**

* On the following lines, display the names of each finalist sorted in ascending order:

**`${participantName}`**

### Examples

|  |
| --- |
| **Input 1** |
| const contest = new Triathlon("Dynamos");  console.log(contest.addParticipant("Peter", "male"));  console.log(contest.addParticipant("Sasha", "female"));  console.log(contest.addParticipant("Peter", "male")); |

|  |
| --- |
| **Output 1** |
| A new participant has been added - Peter  A new participant has been added - Sasha  Peter has already been added to the list |

|  |
| --- |
| **Input 2** |
| const contest = new Triathlon("Dynamos");  console.log(contest.addParticipant("Peter", "male"));  console.log(contest.addParticipant("Sasha", "female"));  console.log(contest.addParticipant("George", "male"));  console.log(contest.completeness("Peter", 100));  console.log(contest.completeness("Sasha", 70));  console.log(contest.completeness("George", 20)); |

|  |
| --- |
| **Output 2** |
| A new participant has been added - Peter  A new participant has been added - Sasha  A new participant has been added - George  Congratulations, Peter finished the whole competition  Sasha could only complete 2 of the disciplines  Uncaught Error: George is not well prepared and cannot finish any discipline |

|  |
| --- |
| **Input 3** |
| const contest = new Triathlon("Dynamos");  console.log(contest.addParticipant("Peter", "male"));  console.log(contest.addParticipant("Sasha", "female"));  console.log(contest.completeness("Peter", 100));  console.log(contest.completeness("Sasha", 70));  console.log(contest.rewarding("Peter"));  console.log(contest.rewarding("Sasha")); |

|  |
| --- |
| **Output 3** |
| A new participant has been added - Peter  A new participant has been added - Sasha  Congratulations, Peter finished the whole competition  Sasha could only complete 2 of the disciplines  Peter was rewarded with a trophy for his performance  Sasha is not in the current finalists list |

|  |
| --- |
| **Input 4** |
| const contest = new Triathlon("Dynamos");  console.log(contest.showRecord("all")); |

|  |
| --- |
| **Output 4** |
| There are no finalists in this competition |

|  |
| --- |
| **Input 5** |
| const contest = new Triathlon("Dynamos");  console.log(contest.addParticipant("Peter", "male"));  console.log(contest.addParticipant("Sasha", "female"));  console.log(contest.completeness("Peter", 100));  console.log(contest.completeness("Sasha", 90));  console.log(contest.rewarding("Peter"));  console.log(contest.rewarding("Sasha"));  console.log(contest.showRecord("all")); |

|  |
| --- |
| **Output 5** |
| A new participant has been added - Peter  A new participant has been added - Sasha  Congratulations, Peter finished the whole competition  Congratulations, Sasha finished the whole competition  Peter was rewarded with a trophy for his performance  Sasha was rewarded with a trophy for his performance  List of all Dynamos finalists:  Peter  Sasha |

|  |
| --- |
| **Input 6** |
| const contest = new Triathlon("Dynamos");  console.log(contest.addParticipant("Peter", "male"));  console.log(contest.addParticipant("Sasha", "female"));  console.log(contest.addParticipant("George", "male"));  console.log(contest.completeness("Peter", 100));  console.log(contest.completeness("Sasha", 90));  console.log(contest.completeness("George", 95));  console.log(contest.rewarding("Peter"));  console.log(contest.rewarding("Sasha"));  console.log(contest.rewarding("George"));  console.log(contest.showRecord("male")); |

|  |
| --- |
| **Output 6** |
| A new participant has been added - Peter  A new participant has been added - Sasha  A new participant has been added - George  Congratulations, Peter finished the whole competition  Congratulations, Sasha finished the whole competition  Congratulations, George finished the whole competition  Peter was rewarded with a trophy for his performance  Sasha was rewarded with a trophy for his performance  George was rewarded with a trophy for his performance  Peter is the first male that finished the Dynamos triathlon |

# Problem 3. Movie Theater

**Your Task**

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

**ageRestrictions (movieRating) -** A function that accepts one parameter: **string**.

* If the value of the parameter **movieRating** is equal to "**G**", return:

**`All ages admitted to watch the movie`**

* If the value of the parameter **movieRating** is equal to "**PG**", return:

**`Parental guidance suggested! Some material may not be suitable for pre-teenagers`**

* If the value of the parameter **movieRating** is equal to "**R**", return:

**`Restricted! Under 17 requires accompanying parent or adult guardian`**

* If the value of the parameter **movieRating** is equal to "**NC-17**", return:

**`No one under 17 admitted to watch the movie`**

* Otherwise, if the above conditions are not met, **return** the following message:

**`There are no age restrictions for this movie`**

* There is **no** need for **validation** for the **input**, you will always be given a string.

**moneySpent (ticket**

**s, food, drinks) -** A function that accepts three parameters: **number, array** and **array**.

* Calculate the **total cost** you are going to pay depending on the **tickets** count, purchased **food** and **drinks:**
  + Each **ticket** costs **15** levs
* The theater offers **two** options for **foods** and **drinks**:
  + The two options for foods are:
    - **Nachos,** which costs **6** levs
    - **Popcorn,** which costs **4.50** levs
  + The two options for drinks are:
    - **Soda**, which costs **2.50** levs
    - **Water**, which costs **1.50** levs
* If the **total cost** is bigger than **50** a **discount** of **20%** shouldbeapplied**.** Then **return** the following message with the cost **rounded** to the **second** **digit** after the decimal point**:**

**`The total cost for the purchase with applied discount is ${totalCost}`**

* Otherwise, return:

**`The total cost for the purchase is ${totalCost}`**

* You need to validate the input, if the **tickets, food** and **drinks** are not a **number, array and array**, **throw** an error: "**Invalid input**"

**reservation (rowsArray, neededSeatsCount) -** A function that accepts **array** and **number**.

* + The **rowsArray** array will store the rows and the free seats for them ([{ **rowNumber**: **1**, **freeSeats**: **7** }, { **rowNumber: 2, freeSeats: 5** }...])
  + The **neededSeatsCount** will be a number higher than **0**, representing the count of the needed seats for the reservation.
  + You must iterate through both the **rowsArray** and extract the **numbers** of the **rows** that have enough free spaces
* Finally, **return** the row with the largest number.
  + There is a need for validation for the input, may not always be valid. In case of submitted **invalid** parameters, **throw** an error "**Invalid input**":
    - If passed **rowsArray** or **neededSeatsCount** parameters are not an array and number.

**JS Code**

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

|  |
| --- |
| **movieTheater.js** |
| const movieTheater = {    ageRestrictions(movieRating) {      switch (movieRating) {        case "G":          return "All ages admitted to watch the movie";        case "PG":          return "Parental guidance suggested! Some material may not be suitable for pre-teenagers";        case "R":          return "Restricted! Under 17 requires accompanying parent or adult guardian";        case "NC-17":          return "No one under 17 admitted to watch the movie";        default:          return "There are no age restrictions for this movie";      }    },    moneySpent(tickets, food, drinks) {      if (        typeof tickets !== "number" ||        !Array.isArray(food) ||        !Array.isArray(drinks)      ) {        throw new Error("Invalid input");      }      let bill = 0;      bill += tickets \* 15;      food.forEach((element) => {        switch (element) {          case "Nachos":            bill += 6;            break;          case "Popcorn":            bill += 4.5;            break;        }      });      drinks.forEach((element) => {        switch (element) {          case "Soda":            bill += 2.5;            break;          case "Water":            bill += 1.5;            break;        }      });      if (bill > 50) {        bill -= bill \* 0.2;        return `The total cost for the purchase with applied discount is ${bill.toFixed(          2        )}`;      } else {        return `The total cost for the purchase is ${bill.toFixed(2)}`;      }    },    reservation(rowsArray, neededSeatsCount) {      if (!Array.isArray(rowsArray) || typeof neededSeatsCount !== "number") {        throw new Error("Invalid input");      }      let availableRows = [];      rowsArray.forEach((row) => {        if (row.freeSeats >= neededSeatsCount) {          availableRows.push(row.rowNumber);        }      });      return Math.max(...availableRows);    },  }; |