# JS Advanced: Exam Preparation 2

**Link to contest**: https://judge.softuni.org/Contests/3682

# Problem 1. Scary Story

**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**.

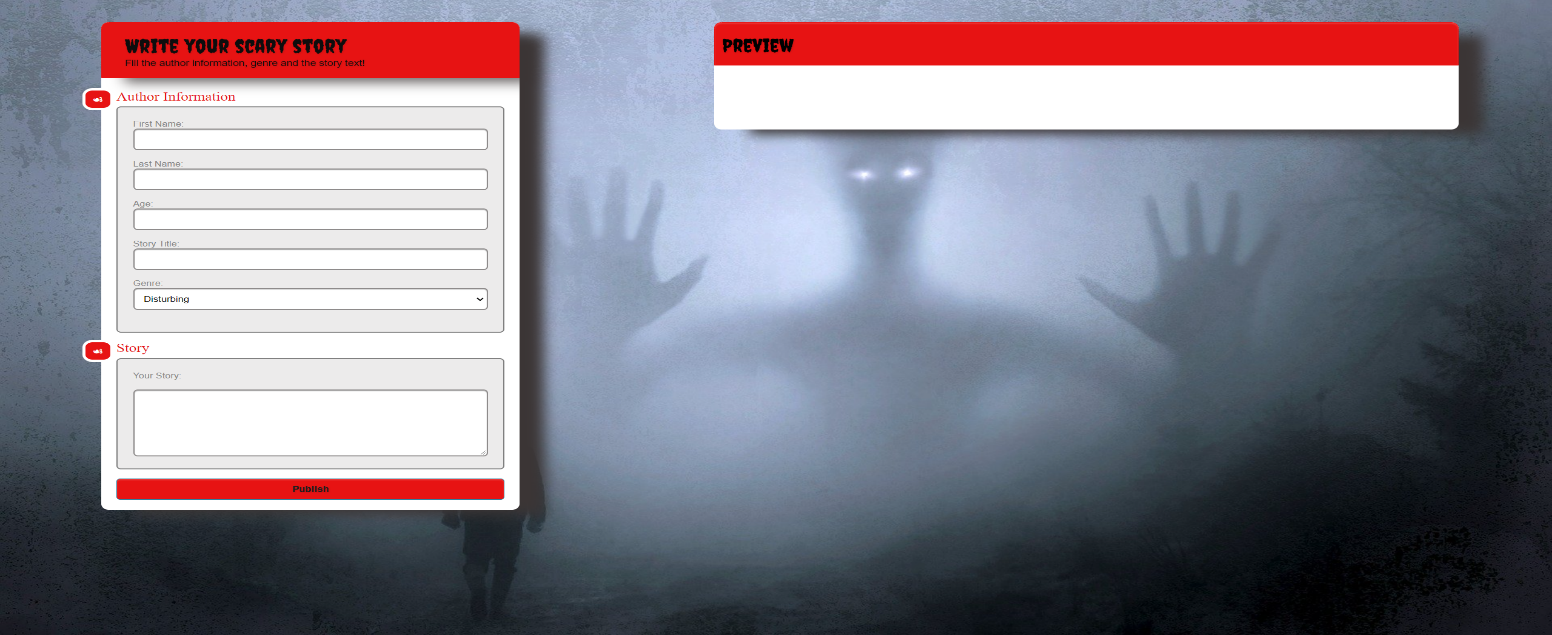
The following actions are **NOT** supported:

* **.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()**
* 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 **Scary Story** work as expected:

All fields **(First Name, Last Name, Age, Genre, Story title** and **Story text)** are **filled with the correct input**

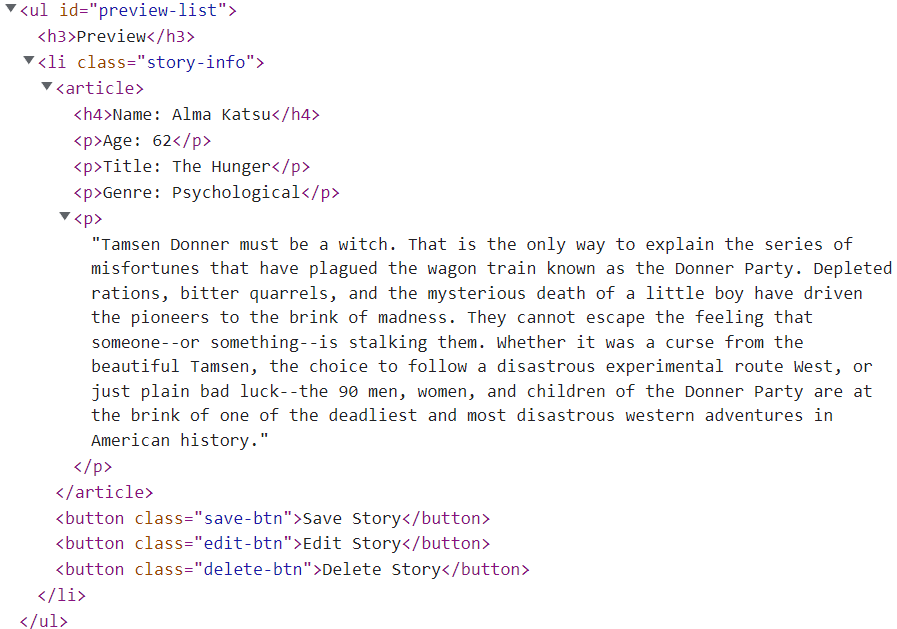
* **First Name, Last Name, Age, Story title** and **Story text** are **non**-**empty** **strings**. If any of them is empty, the program should not do anything.

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

#### 

* On clicking the [“Publish”] button the information from the input fields is listed in the **"**preview" section. For the input fields a **list item** is **added** to the **"**preview-list" unordered list.
* The text format and order for each list item should be the same as on the picture below.
* When the button is clicked, the input fields must be cleared and the ["Publish"] button must be **disabled**. At the same time the "Save", "Edit" and the "Delete" buttons must be **enabled**.

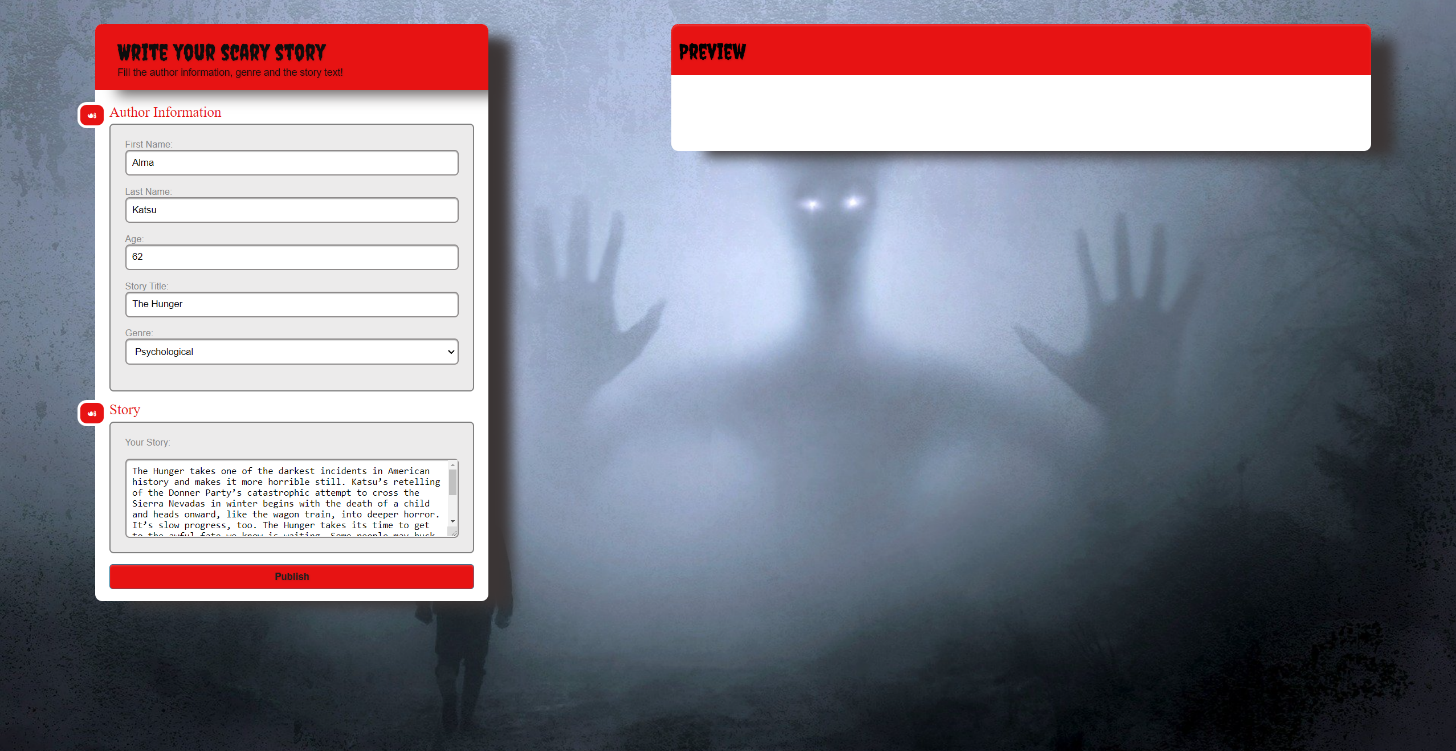
The HTML structure looks like this:



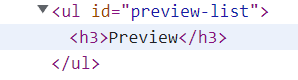
1. **Editing the information**

**The functionality here is the following:**

* **When the "**Edit" **button is clicked, all of the information is loaded in the input fields from step 1 and all buttons in preview section are disabled while the** ["Publish"] **button is enabled again.**

****

* **The list items must be removed from the** "preview-list" and **all of the information must go back to the input fields again**.

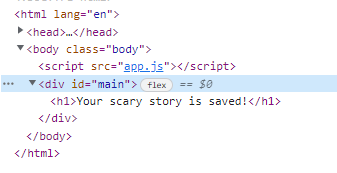
****

1. **Saving the information**

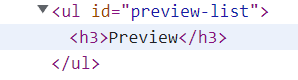
* **When the** "Save" **button is clicked,** **the story is completed. For you, this means removing everything inside of the div with id =** "main" **and adding there only a <h1> tag. With message: "Your scary story is saved!"**

****

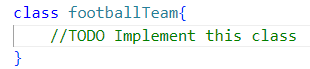
* **This is everything your webpage must contain at this step:**



* **When the** "Delete" **button is clicked,** **the list item must be removed from the** "preview-list" **and the** ["Publish"] **button is enabled again**.

****

# Problem 2. Football team



Write a **class Football team**, which supports the described functionality below:

**Functionality**

**Constructor**

Should have these **3** properties:

* **clubName - string**
* **country - string**
* **invitedPlayers - empty array**

**At the initialization** of the **FootballTeam** class, the **constructor** accepts the **clubName** and **country.**

**Hint:** You can add more properties to help you finish the task.

### newAdditions(footballPlayers)

This method adds players to the invitation list. The method takes **one argument**: **footballPlayers (array of strings)**.

* **Every element** into this array is information about player in the format:

**"{name}/{age}/{playerValue}"**

* They are **separated** by slash symbol **"/"**. The **playerValue** is in millions. **Example**: ["**Kylian Mbappé/23/160**"**,** "**Lionel Messi/35/50**"**,** "**Pau Torres/25/52**"**…**]
* If the **name** of the current player is already present in **invitedPlayers** array, update the old **playerValue only if** the current one is **higher**.
* Otherwise, should **add** the player, with properties: **{name, age, playerValue}** to the **invitedPlayers array**.
* In **all cases**, you must **finally return a string** in the following format:

**"You successfully invite {name1}, {name2}, …{name**N}**."**

**Note**: When returning the **string**, keep in mind that the different **names** of **football Players** **must** be:

* **Unique** -for instance**:** 
  + **" You successfully invite Kylian Mbappé, Lionel Messi, Pau Torres "** - is a correctly returned string
  + **" You successfully invite Kylian Mbappé, Lionel Messi, Kylian Mbappé "**- is not a correctly returned string
* **Separated** by **comma** and **space (, )**

### signContract(selectedPlayer)

With this method, manager can sign contracts with player from the invited list. The method takes **one argument**: **selectedPlayer (string)**.

* Тhe stringabout the selected player is in the format:

**"{name}/{playerOffer}"**

* Check:
  + If the **name** of the current player is not present in **invitedPlayers** array, an **error** with the following message should be **thrown**:

**"{name} is not invited to the selection list!"**

* + If the **playerOffer** selected by the manager for a given player **is less** than the value recorded in the array **invitedPlayers**,an **error** with the following message should be **thrown**:

**"The manager's offer is not enough to sign a contract with {name}, {priceDifference} million more are needed to sign the contract!"**

* + - **priceDifference -** is the difference between **playerValue** and **playerOffer.**
  + Otherwise, if the above conditions are not met, you must replace **playerValue** with the string "**Bought**"
  + Finally, you need to **return** the string in the following format:

**"Congratulations! You sign a contract with {BoughtPlayer} for {BuyingPrice} million dollars."**

### ageLimit(name, age)

With this method, we make sure that the players are young enough to sign **a five-year** contract with the team, a **reduced-time** contract, or no contract at all. The method takes **two arguments**:

* **name (string)**
* **age (number)**
* If the submitted **name** is not present in the **invitedPlayers** array, an **error** with the following message should be **thrown**:

**"{name} is not invited to the selection list!"**

* If the age recorded in the **invitedPlayers** array is **less than** the age submitted as an argument, you must check the difference between **limit age** and **player age**. If the difference is **less** **than** 5 years, return the following string: **"{name} will sign a contract for {ageDifference} years with {clubName} in {country}!"**
  + **ageDifference -** is the difference between limit age and player age.
* If the age difference is **more** than **5** years, **return** the following message:

**"{name} will sign a full 5 years contract for {clubName} in {country}!"**

* If the player age from the **invitedPlayers** array is **greater** or **equal** than the age submitted as an argument, **return** the following message: **"{name} is above age limit!"**

### transferWindowResult()

* This method **returns** **all players**, The first line shows the following message:

**"Players list:"**

* On the new line, display information about each player sorted in **ascending** order of **name**:

**"Player {name}-{playerValue}"**

### Example

|  |
| --- |
| **Input 1** |
| let fTeam = new footballTeam("Barcelona", "Spain");   console.log(fTeam.newAdditions(["Kylian Mbappé/23/160", "Lionel Messi/35/50", "Pau Torres/25/52"])); |

|  |
| --- |
| **Output 1** |
| You successfully invite Kylian Mbappé, Lionel Messi, Pau Torres. |

|  |
| --- |
| **Input 2** |
| let fTeam = new footballTeam("Barcelona", "Spain");  console.log(fTeam.newAdditions(["Kylian Mbappé/23/160", "Lionel Messi/35/50", "Pau Torres/25/52"]));  console.log(fTeam.signContract("Lionel Messi/60"));  console.log(fTeam.signContract("Kylian Mbappé/240"));  console.log(fTeam.signContract("Barbukov/10")); |

|  |
| --- |
| **Output 2** |
| You successfully invite Kylian Mbappé, Lionel Messi, Pau Torres.  Congratulations! You sign a contract with Lionel Messi for 60 million dollars.  Congratulations! You sign a contract with Kylian Mbappé for 240 million dollars.  Uncaught Error: Barbukov is not invited to the selection list! |

|  |
| --- |
| **Input 3** |
| let fTeam = new footballTeam("Barcelona", "Spain");  console.log(fTeam.newAdditions(["Kylian Mbappé/23/160", "Lionel Messi/35/50", "Pau Torres/25/52"]));  console.log(fTeam.ageLimit("Lionel Messi", 33 ));  console.log(fTeam.ageLimit("Kylian Mbappé", 30));  console.log(fTeam.ageLimit("Pau Torres", 26));  console.log(fTeam.signContract("Kylian Mbappé/240")); |

|  |
| --- |
| **Output 3** |
| You successfully invite Kylian Mbappé, Lionel Messi, Pau Torres.  Lionel Messi is above age limit!  Kylian Mbappé will sign a full 5 years contract for Barcelona in Spain!  Pau Torres will sign a contract for 1 years with Barcelona in Spain!  Congratulations! You sign a contract with Kylian Mbappé for 240 million dollars. |

|  |
| --- |
| **Input 4** |
| let fTeam = new footballTeam("Barcelona", "Spain");  console.log(fTeam.newAdditions(["Kylian Mbappé/23/160", "Lionel Messi/35/50", "Pau Torres/25/52"]));  console.log(fTeam.signContract("Kylian Mbappé/240"));  console.log(fTeam.ageLimit("Kylian Mbappé", 30));  console.log(fTeam.transferWindowResult()); |

|  |
| --- |
| **Output 4** |
| You successfully invite Kylian Mbappé, Lionel Messi, Pau Torres.  Congratulations! You sign a contract with Kylian Mbappé for 240 million dollars.  Kylian Mbappé will sign a full 5 years contract for Barcelona in Spain!  Players list:  Player Kylian Mbappé-Bought  Player Lionel Messi-50  Player Pau Torres-52 |

# Problem 3. Choose Your Car

**Your Task**

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

* **choosingType (type, color,** **year) -** A function that accepts **three** parameters: **string**, **string**, and **number**.
* If the **year** is **less** than 1900 and the **year** is **more** than 2022, **throw** an error: **"Invalid Year!"**
* If the value of the string **type** is different from "**Sedan**", **throw** an error: **"This type of car is not what you are looking for."**
* To be picked, the **car** must meet the **following requirement**:
  + If the **year** of the car is **greater** or **equal** to **2010**, **return** the string:

**"This ${color} ${type} meets the requirements, that you have."**

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

**"This ${type} is too old for you, especially with that ${color} color."**

* There is **no** need for **validation** for the **input**, you will always be given two strings, and number.
* **brandName** **(brands, brandIndex) -** A function that accepts an **array** and **number**. The **brands** array will store the brand names (["**BMW**", "**Toyota**", "**Peugeot**"…]).
  + You must **remove** an **element** (brand) from the **array** that is located on the **index** specified as a parameter.
  + Finally, **return** the changed array of brands 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 **brands** parameteris **not** an array.
    - If the **index** is not a number and is outside the limits of the array.
* **CarFuelConsumption (distanceInKilometers, consumptedFuelInLitres) -** A function that accepts two parameters: **number, number**.
* You test drive the car to find out what its consumption is.
* You need to **calculate** liters per 100 kilometers consumption by **dividing** the fuel consumption by 100 and then **multiply** by distance.
  + - **The result must be formatted to the second digit after the decimal point.**
* If the liters/100km is **less** or **equal** to 7L. **return** the following message:

**"The car is efficient enough, it burns ${litersPerHundredKm} liters/100 km."**

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

**"The car burns too much fuel - ${litersPerHundredKm} liters!"**

* You **need to validate** the input, if the **distanceInKilometers** and **consumptedFuelInLitres** are not a **numbers**, or are a **negative** numbers, **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 **chooseYourCar** object:

|  |
| --- |
| chooseYourCar.js |
| onst chooseYourCar = {      choosingType(type, color, year) {          if (year < 1900 || year > 2022) {              throw new Error(`Invalid Year!`);          } else {              if (type == "Sedan") {                  if (year >= 2010) {                      return `This ${color} ${type} meets the requirements, that you have.`;                  } else {                      return `This ${type} is too old for you, especially with that ${color} color.`;                  }              }              throw new Error(`This type of car is not what you are looking for.`);          }      },    brandName(brands, brandIndex) {          let result = [];          if (!Array.isArray(brands) || !Number.isInteger(brandIndex) || brandIndex < 0 || brandIndex >= brands.length) {              throw new Error("Invalid Information!");          }          for (let i = 0; i < brands.length; i++) {              if (i !== brandIndex) {                  result.push(brands[i]);              }          }          return result.join(", ");      },      carFuelConsumption(distanceInKilometers, consumptedFuelInLiters) {         let litersPerHundredKm =((consumptedFuelInLiters / distanceInKilometers)\* 100).toFixed(2);          if (typeof distanceInKilometers !== "number" || distanceInKilometers <= 0 ||              typeof consumptedFuelInLiters !== "number" || consumptedFuelInLiters <= 0) {              throw new Error("Invalid Information!");          } else if (litersPerHundredKm <= 7) {              return `The car is efficient enough, it burns ${litersPerHundredKm} liters/100 km.`;          } else {              return `The car burns too much fuel - ${litersPerHundredKm} liters!`;          }      }  } |