JS Advanced: Exam 26 April 2018

Problems for exam preparation for the "JavaScript Advanced" course @ SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/1004/.

Problem 4. Public Transport (Object Interacting with DOM)

Write a JS class with the name "PublicTransportTable" that generates and controls a public transport table. It contains two main columns for transport data (Type, Name) and one column for Actions - [Search], [Clear] and [More/Less Info] buttons. The second row of the table consists of two input fields, [Search] and [Clear] button. The main table's body consists of rows with types of vehicles with additional inner tables for more information. Every **vehicle** has a **main info row** and an **additional info row**. See the examples and templates for more details.

The constructor of your class needs to take one argument – town (string). The town should replace in the table's title (<caption>) the part in the curly brackets. See the main table's HTML for more details below.

Additionally, the **class should contain** the following **functionality**:

Function addVehicle(obj) – takes one (object) argument, creates the vehicle's main info row (see the template) and appends it to the main table's body with ID of "vehicles-info". The argument is in the following format:

```
{ type: String, name: String, route: String, price: Number, driver: String }
```

With this object's data you should create a table's row following this HTML structure:

```
Template Vehicle's Main Info Row
<!-- Vehicle's type -->
  <!-- Vehicle's name -->
  <button>[More/Less] Info</button>
```

When a vehicle's row is created its initial button's text should be "More Info". Also, when clicked a [More Info] button should have the following functionality:

- Changes its text to "Less Info"
- Creates below the table's row where it is placed on a new row with the vehicle's additional info data. You should use the following HTML structure for generating the new row with inner table:

```
Template Vehicle's Additional Info Row
Route: {vehicle's route}
     Price: {vehicle's price}
      Driver: {vehicle's driver}
```

An additional vehicle's row consists of a row with a single column that have an inner table with four rows each having a **single column** with a vehicle data.

















Furthermore, when a [Less Info] button is clicked, it should have the following functionality:

- Changes its text to "More Info"
- Removes the row (below the table's row where it is placed on) with the extra vehicle's data

Also, when clicked the [Search] and [Clear] buttons should have the following functionality:

[Search] button:

- **Filters** the **vehicles** by the **text** received from the **input boxes** (**case-sensitive**). Every **input box** corresponds to the vehicle's column it is placed on. You should show a row (remove "display: none;") If its columns include the text from the their corresponding text box or hide it if they do not include (set "display: none;")
- Removes any additional vehicle rows, that are displaying the extra data (route, price, driver), if the main row should be removed and sets its button text from [Less Info] to [More Info]
- Do nothing if both input boxes are empty

[Clear] button:

- Removes all filters and displays all main vehicle rows
- Does nothing to additional vehicle rows
- Clears the input boxes

Submission

Submit only your **PublicTransportTable** class.

Examples

You should use the following HTML skeleton to test your functionality:

```
index.html
<!DOCTYPE html>
<html Lang="en">
<head>
  <meta charset="UTF-8">
  <title>Payment Manager</title>
  <style>
    body{padding:30px}caption,table{border:1px solid #b1cbbb}table{border-
collapse:collapse;margin:auto}caption{font-size:20px;font-
weight: 700; color: #DEEAEE; background: #485788; border-
bottom:none;padding:5px}td,th{padding:5px 20px;text-align:center}tfoot
td,th{background:#b1cbbb}tr:nth-child(odd){background:#EAF7FB}tr:nth-
child(even){background:#C6D1D4}button{margin:auto;cursor:pointer}input[type=text]{width:
100px}tbody>.more-info{background:#E6F1B4!important}tbody tr table{margin:5px auto 5px
20px;width:330px;}tbody tr table td{background:#eea29a99!important;text-
align:left;width:170px}
  </style>
  <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
</head>
<body>
      <caption>{Town name}'s Public Transport</caption>
      <thead>
```















```
Type
            Name
            Actions
         <input name="type" type="text">
            <input name="name" type="text">
              <button class="search-btn">Search</putton>
              <button class="clear-btn">Clear</putton>
            </thead>
     <script src="./solution.js"></script>
 <script>
   $(function() {
     let publicTransportTable = new PublicTransportTable('Sofia')
     let vehiclesDataArr = [
       { type: 'bus', name: '84', route: 'Sofia airport - Gen. Gurko str.', price:
1.60, driver: 'Pesho' },
       { type: 'tram', name: '7', route: 'Borovo - metro station Han Kubrat', price:
13.37, driver: 'Misho' },
       { type: 'bus', name: '280', route: 'Student city - SU Kliment Ohridsky', price:
4.20, driver: 'Gosho' },
       { type: 'trolleybus', name: '11', route:' Drujba 1 - Stochna station sq.',
price: 2.60, driver: 'Tosho' },
       { type: 'metro', name: 'M1', route: 'Slivnica - Business Park', price: 3.50,
       driver: 'Petq' },
       { type: 'bus', name: '404', route: 'AP Drujba - Central station', price: 1.60,
       driver: 'Silviq' },
       { type: 'metro', name: 'M2', route: 'Sofia airport - James Bourchier', price:
5.99, driver: 'Krum' },
      // You can add/remove any vehicles data objects here
     for (let vehicleObj of vehiclesDataArr) {
       publicTransportTable.addVehicle(vehicleObj)
   })
 </script>
</body>
</html>
```

The sample page contains the main table element whit head, body and a script. The script will populate the table's body with seven main vehicles info rows that should look like the examples, if your code is correct. You need to put









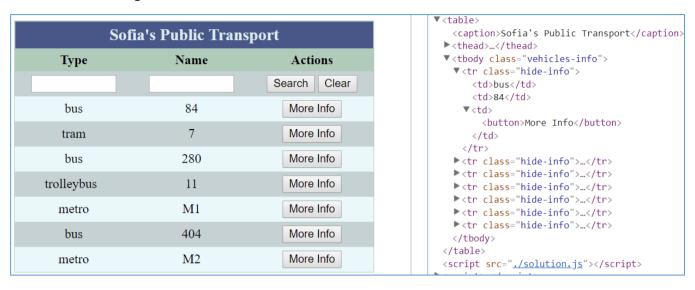




the files – index.html and solutions.js (where is your PaymentManager class) – in the same directory for them to work with the default code.

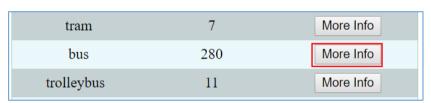
In the example we will use only the seven objects with vehicle's data in the script to generate and manipulate the table's functionality.

Initial table rendering with vehicles:

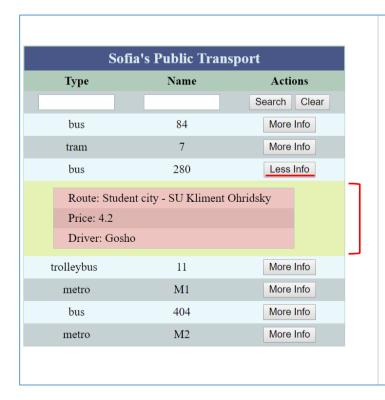


Use the "Template Main Info Rows" skeleton to build your main vehicles rows.

Click the [More Info] button on bus 280's row:



After the click we change the button's text to "Less Info" and add the additional vehicle's info row below the main one. Thus becoming 7 main and 1 additional vehicle info rows (8 total) in the main table (vehicles-info). As shown by the red lines below.

















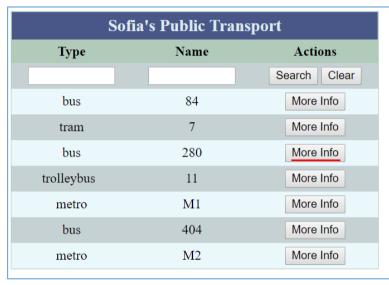




Click the [Less Info] button on bus 280's row:

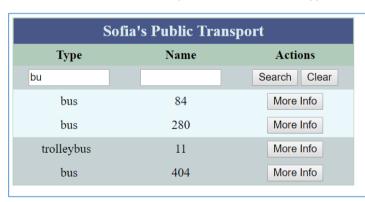


After the click we change the button's text to "More Info" and remove the additional vehicle's info row below the main one. Thus becoming 7 main and no additional vehicle info rows in the main table (vehicles-info) again. As shown by the red lines below.



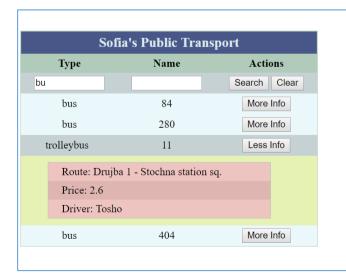


Then we enter "bu" in the input box on column "Type" and click the [Search] button:





After that, we click the [More Info] button on trolleybus 11's row:



```
▶...
▶...
▼
trolleybus 
11
▼
<button>Less Info</putton>
▼
▼
▼
 ▼
 ▶...
 ▶...
 ▶...
 ▶...
▶...
▶...
```

8 rows -4 visible; 3 hidden; 1 extra (additional)



© Software University Foundation. This work is licensed under the CC-BY-NC-SA license.







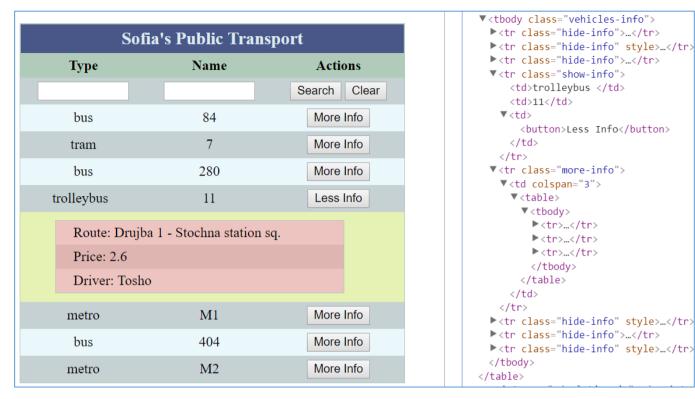








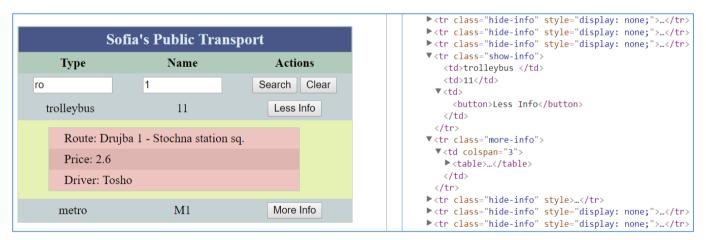
Then we click the [Clear] button:



Chain of commands to the table:

- Click [More Info] button on bus 404, metro M2, tram 7, bus 84
- Enter "ro" in the input box on column "Type", "1" in the input box on column "Name" and then click the [Search] button

Chain result:



Chain of commands to the table:

- Click [Less Info] button on trolleybus 11
- Click [More Info] button on metro M1
- Click [Clear] button
- Click [More Info] button on bus 404, bus 84, metro M2

Scroll down to see result:



