JS Advanced: Exam 18 March 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/974/.

Problem 4. Payment Manager (Object Interacting with DOM)

Write a JS class with the name "PaymentManager" that generates and controls a payment manager table. It contains three columns for payments data (Name, Category, Price) and one column for Actions - [Add] and [Delete] buttons. The last row of the table consists of three input fields and a [Add] button. There can be multiple payment manager tables – i.e. instances of the JS class. See the examples and templates for more details.

The constructor of your class needs to take one argument – title (string). The title should be used in the table's title (<caption>). See the table's HTML template for more details below.

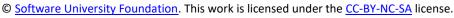
Additionally, the class should contain the following functionality:

Function render(id) – takes one (string) argument, generates the table's HTML element and appends it to the element in the DOM with ID equal to the argument

The table consists of a title, four columns (name, category, price and actions), three input fields and two types of buttons (add, delete). You should use the following HTML structure for generating the table:

```
Template Table
<caption>{Title} Payment Manager</caption>
  <thead>
    >
       Name
       Category
       Price
      Actions
    </thead>
  <!-- Payment's name -->
       <!-- Payment's category -->
       <!-- Payment's price -->
       <button>Delete</button>
    ...
  <tfoot class="input-data">
    <input name="name" type="text">
       <input name="category" type="text">
       <input name="price" type="number">
       </tfoot>
```

















Types of values in a payment's columns:

- Name string
- Category string
- Price number

Every input box corresponds to the payment's column it is placed on, where the data should be added.

Furthermore, when **clicked** the table's **buttons** should have the following **functionality**:

[Add] button:

- Appends the text received from the input boxes and a [Delete] button as a new row at the end of the table's payments
- Clears the input boxes after the payment's information is submitted
- Do nothing if either of the text boxes is empty

[Delete] button:

Should **remove** the table's **row**, where it is placed on

Note that, every button's functionality should work only for the table where it is located on.

Submission

Submit only your **PaymentManager** class.

Examples

You can 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}div{margin-bottom:20px}table{border:1px solid #000;border-
collapse:collapse;margin:auto}caption{font-size:20px;font-
weight:700;color:#fff;background:#000;padding:5px}td,th{padding:5px 20px;text-
align:center}tfoot td,th{background:#ccc}tr:nth-child(odd){background:#eee}tr:nth-
child(even){background:#ddd}tfoot{border-top:1px solid #000}tfoot>tr td{font-
style:italic;font-weight:700}button{margin:auto;cursor:pointer}
 </style>
 <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
</head>
<body>
  <div id="amazon"></div>
  <div id="ebay"></div>
  <div id="walmart"></div>
  <script src="./solution.js"></script>
```

















```
<script>
    $(function() {
      let paymentMangersData = [
        ['amazon', 'Amazon'],
        ['ebay', 'eBay'],
        ['walmart', 'Walmart'],
        // Others
     for (let [target, title] of paymentMangersData) {
        let paymentManager = new PaymentManager(title)
          paymentManager.render(target)
    })
  </script>
</body>
</html>
```

The sample page contains three main table's element and a script. The script will create two different payment manager tables that should look like the examples, if your code is correct. There is a third commented out manager's data that you can use in the script. 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 first example we will use only the first payment manager's data in the script and generate only one table.

Initial table rendering:



Use the "Template Table" skeleton to build your table.

Adding the first payment:



We add the first payment and two more after that, becoming the total of three payments.



Then we add two more payments - "JavaScript The Good Parts" and "Audeze LCD-3" and click to delete a payment:











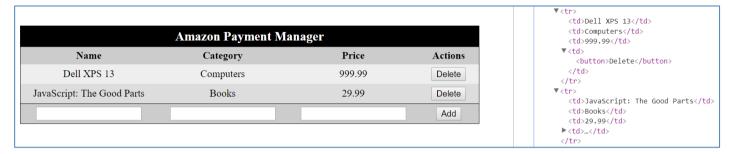






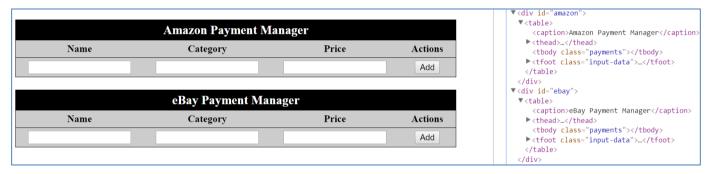


After deleting the payment with column name's value - "Audeze LCD-3":



In the **second example** we will use two payment manager's data in the script to generate two tables.

Initial tables rendering:



Chain of commands to the tables:

- Adding payments to Amazon Payment Manager
 - ["Logitech MX Master", "Mouses", "149.99"]
 - ["Dell XPS 13", "Computers", "999.99"]
 - o ["Oree Board", "Keyboards", "80"]
- Adding payments to eBay Payment Manager
 - ["V60 Cherry Dolch", "Keyboards", "105"]







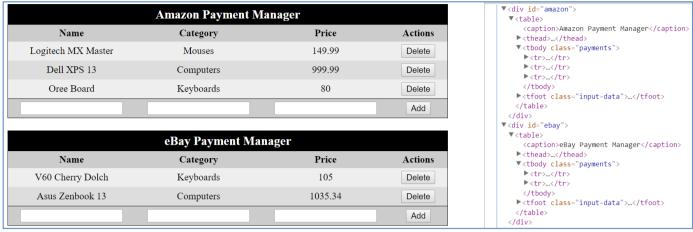








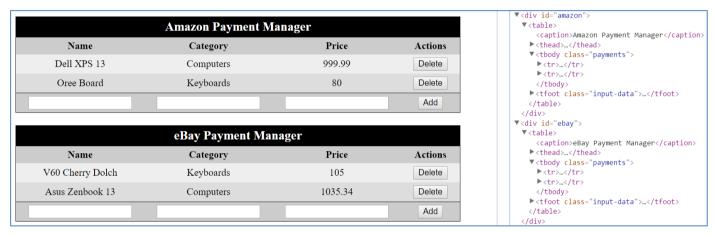




["Asus Zenbook 13", "Computers", "1035.34"]

More commands:

Delete the payment with column name "Logitech MX Master" from Amazon Payment Manager



More commands:

- Adding a payment to eBay Payment Manager:
 - ["Oree Board", "Keyboards", "80"]
- Delete the payment with column name "Oree Board" from Amazon Payment Manager
- Adding a payment to Amazon Payment Manager:
 - ["V60 Cherry Dolch", "Keyboards", "105"]
- Delete the payment with column name "Asus Zenbook 13" from eBay Payment Manager
- Delete the payment with column name "V60 Cherry Dolch" from eBay Payment Manager





