# JS Advanced: Exam 18 March 2018

Problems for exam preparation for the [“JavaScript Advanced” course @ SoftUni](https://softuni.bg/courses/javascript-advanced). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/756/>. // TODO: Change contests number

# Problem 4. Payment Manager (Object Interacting with DOM)

Write a JS **class** with name **"PaymentManager"** that generates and controls a **payment manager** table. It contains **three columns** for payments data **(Name, Category, Price)** each of them has **sorting** **(arrow)** **buttons** and **one column** for **Actions - [Add]** and **[Delete]** buttons**.** The **last row** of the table consists of **three** **input fields** and the **[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** arguments – **title** (string). The **title** is 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 addProduct() – **appends** a new **row at the end** of the **table’s body** with the **payment’s data** (name, category, price) and a **[Delete]** action button
* Function attachSortEvents()– **attaches all event listeners** to the **sorting** (arrow) **buttons** with their intended functionality **– ordering** table’s **payments** in **ascending** or **descending** way
* 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 **three** **types** of **buttons** (add, delete and sort). You **should** **use** the following **HTML structure** **for** generating the **table**:

|  |
| --- |
| Template Table |
| // TODO: Add template-table.html |

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** 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

**Arrow button:**

* **Sorts** thetable’s **payments by** the **column’s values** where it is placed on through:
* **Up arrow – sorts** in **ascending** order
* **Down arrow – sorts** in **descending** order
* **Columns "Name"** and **"Category" sort** by **alphabetical** order
* **Column "Price" sort** by **numerical** order

Note that, **every button’s functionality** is **only** for the payment manager’s **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 |
| // TODO: Add input.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.

In the **first example** we will use only the first payment manager’s data in the script and generate only one table.

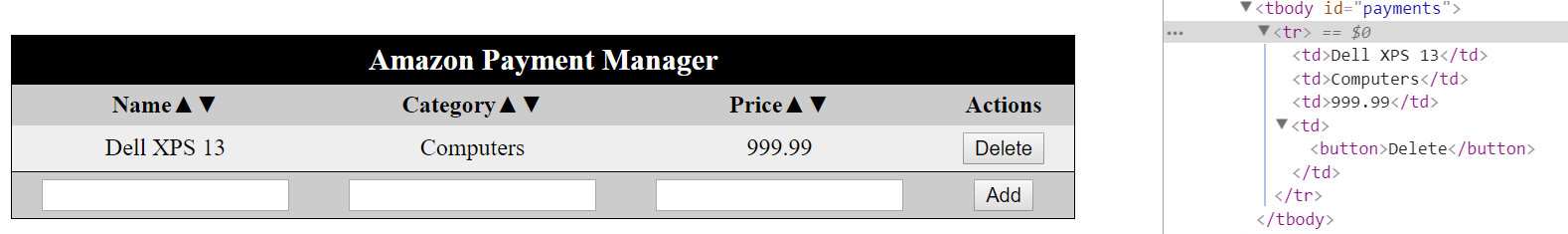
**Initial table rendering**:



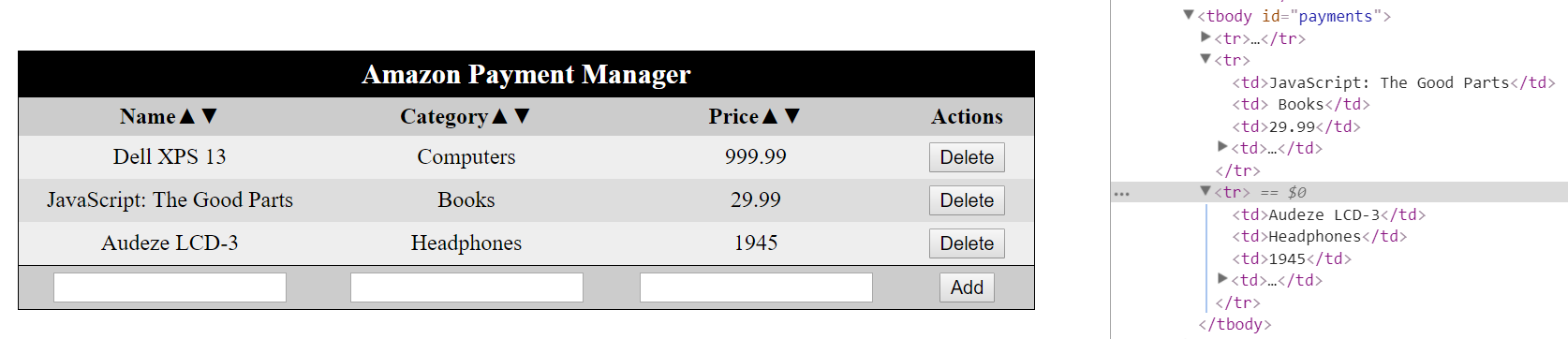
**Adding** the **first payment**:

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

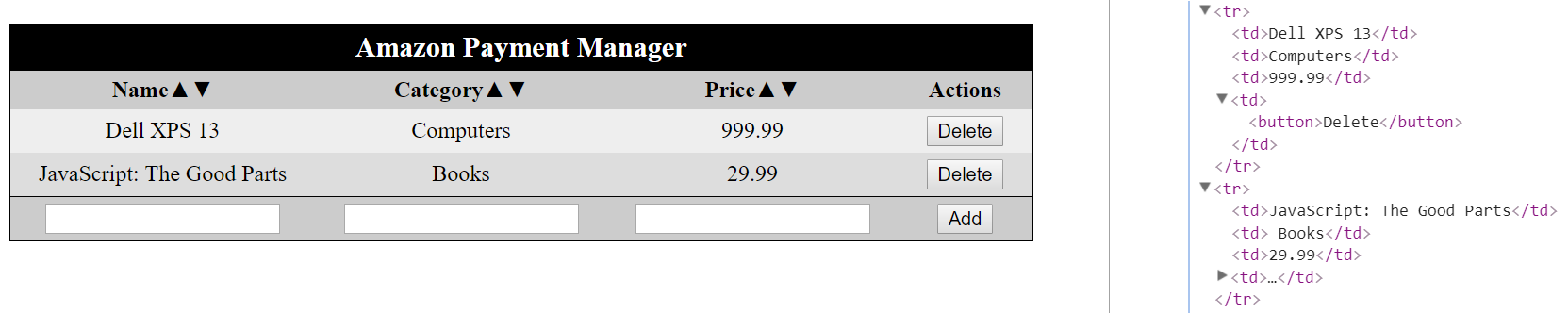




We adding two more payments – "JavaScript The Good Parts" and "Audeze LCD-3" and click to delete a payment:

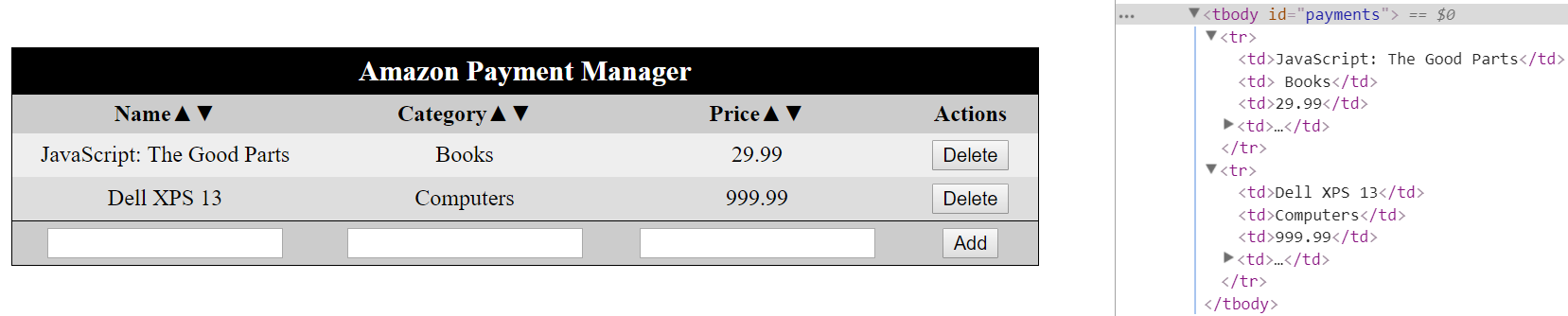


After deleting the last added payment – "Audeze LCD-3":



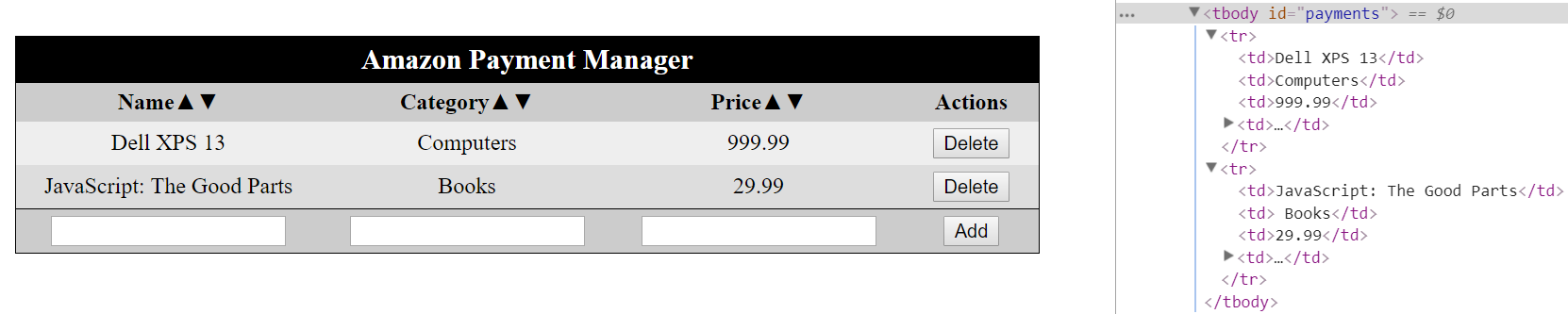
Sorted payments by "Name" in descending alphabetical order after clicking the down arrow:

Clicked



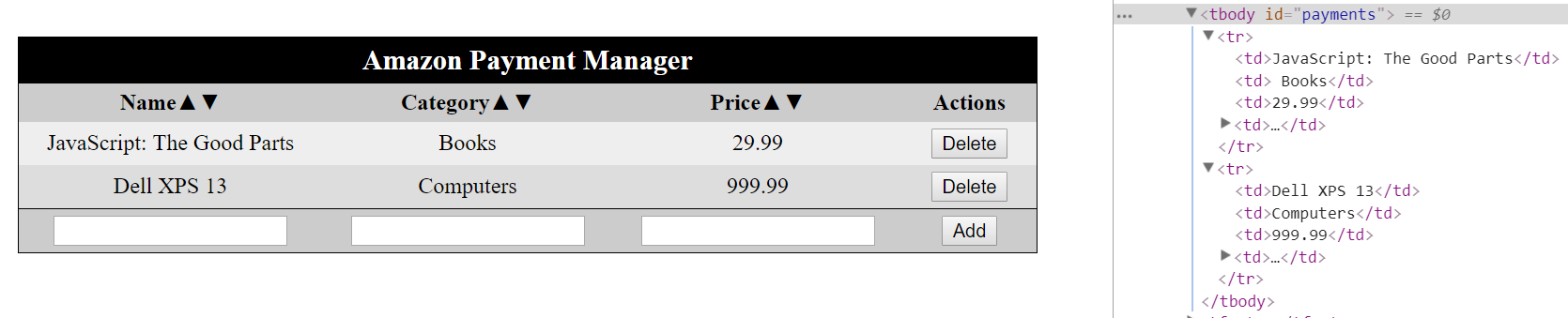
Sorted payments by "Price" in descending order after clicking the down arrow:

Clicked



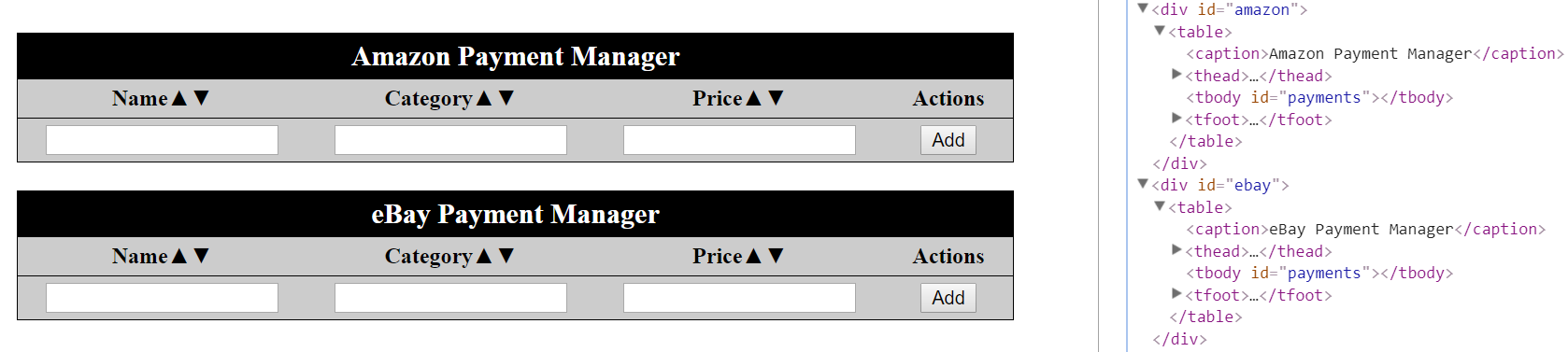
Sorted payments by "Category" in ascending alphabetical order after clicking the up arrow:

Clicked



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

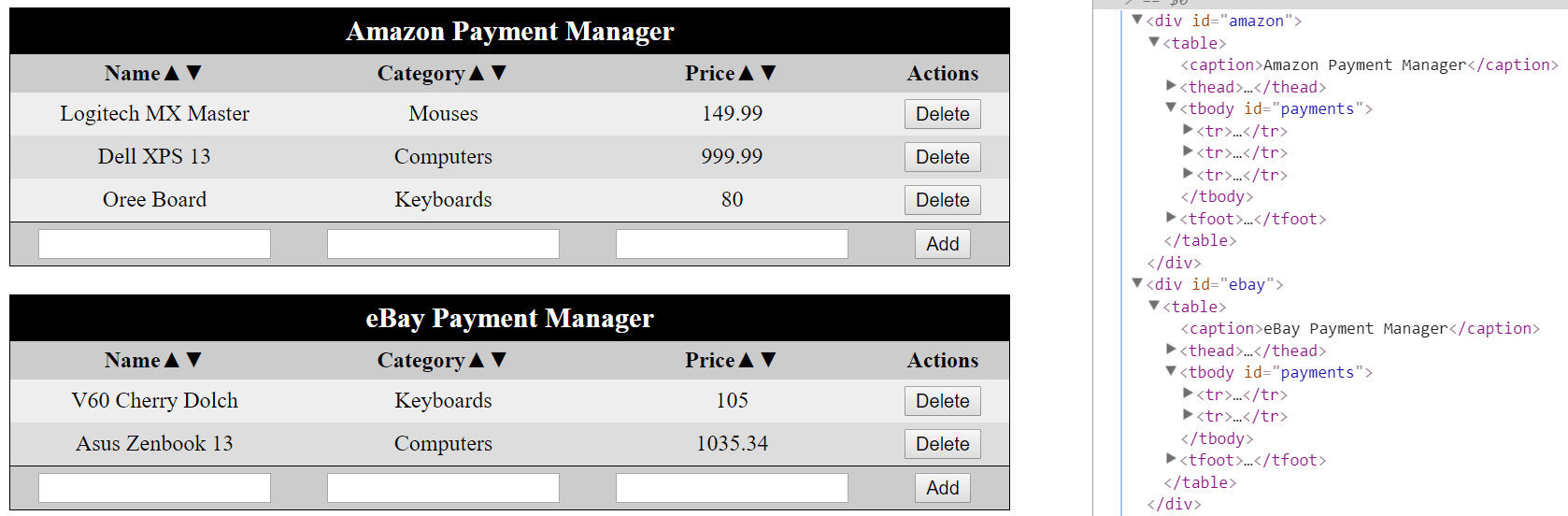
**Initial tables rendering**:



Chain of **commands** to the tables:

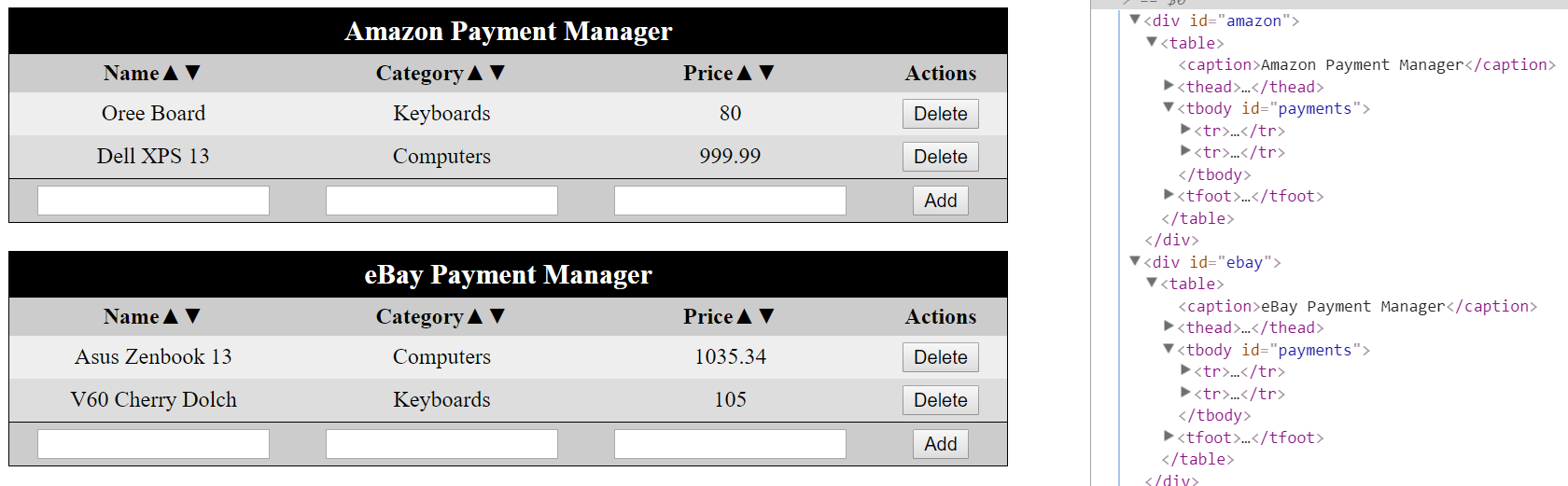
* **Adding** payments to Amazon Payment Manager
* [ "Logitech MX Master", "Mouses", "149.99" ]
* [ "Dell XPS 13", "Computers", "999.99" ]
* [ "Oree Board", "Keyboards", "80" ]
* **Adding** payments to eBay Payment Manager
* [ "V60 Cherry Dolch", "Keyboards", "105" ]
* [ "Asus Zenbook 13", "Computers", "1035.34" ]

Scroll down to see the results after adding the payments.



More commands:

* **Delete** **payment** with name "Logitech MX Master" from **Amazon Payment Manager**
* **Sort** payments in **Amazon Payment Manager** by **category** in **descending** order (down arrow)
* **Sort** payments in **eBay Payment Manager** by **name** in **ascending** order (up arrow)



More commands:

* **Adding payment** to **eBay Payment Manager**:
* [ "Oree Board", "Keyboards", "80" ]
* **Sort** payments in **eBay Payment Manager** by **price** in **ascending** order (up arrow)
* **Delete** **payment** with name "Oree Board" from **Amazon Payment Manager**
* **Adding payment** to Amazon Payment Manager:
* [ "Asus Zenbook 13", "Computers", "1035.34" ]
* **Sort** payments in **Amazon Payment Manager** by **name** in **ascending** order (up arrow)
* **Delete** **payment** with name "V60 Cherry Dolch" from **eBay Payment Manager**

Scroll down to see the results after these commands.

