Client-Side Web Technologies – Homework Assignment 3

Due: April 30 @ 11:59 PM (syllabus says April 27 but I am giving a few extra days)
Value: 20% of total grade

Overview

In this assignment you will create a reusable component using JavaScript and some CSS. Reusable components are extremely useful because they can allow developers that make use of them to greatly speed up the time it takes to build whatever software they are constructing.

The component that you will create will be a reusable data grid. Developers that make use of your component will specify several parameters such as column widths, header names, and the data to display.

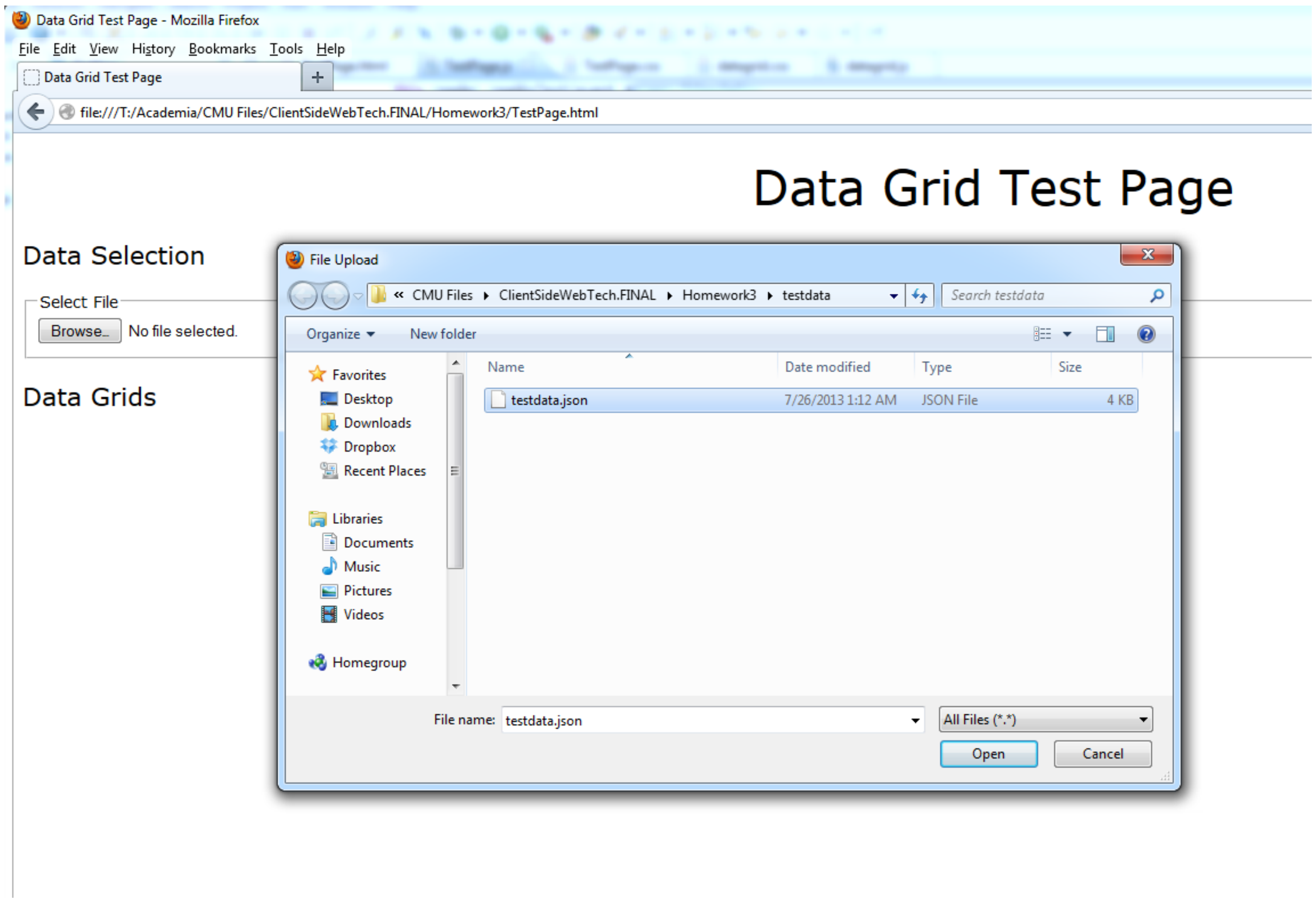
I have created a test page that you will use to test your component with. I have also created a CSS file to set some basic page styles as well as a JavaScript file that will handle testing your component. Additionally, there is a file that contains test data to populate your grid. Finally, there is a JSHint configuration file (discussed later). Here is the file structure:

```
testpage.html
.jshintrc
css
|-------- testpage.css
js
|-------- testpage.js
testdata
|-------- testdata.json
```

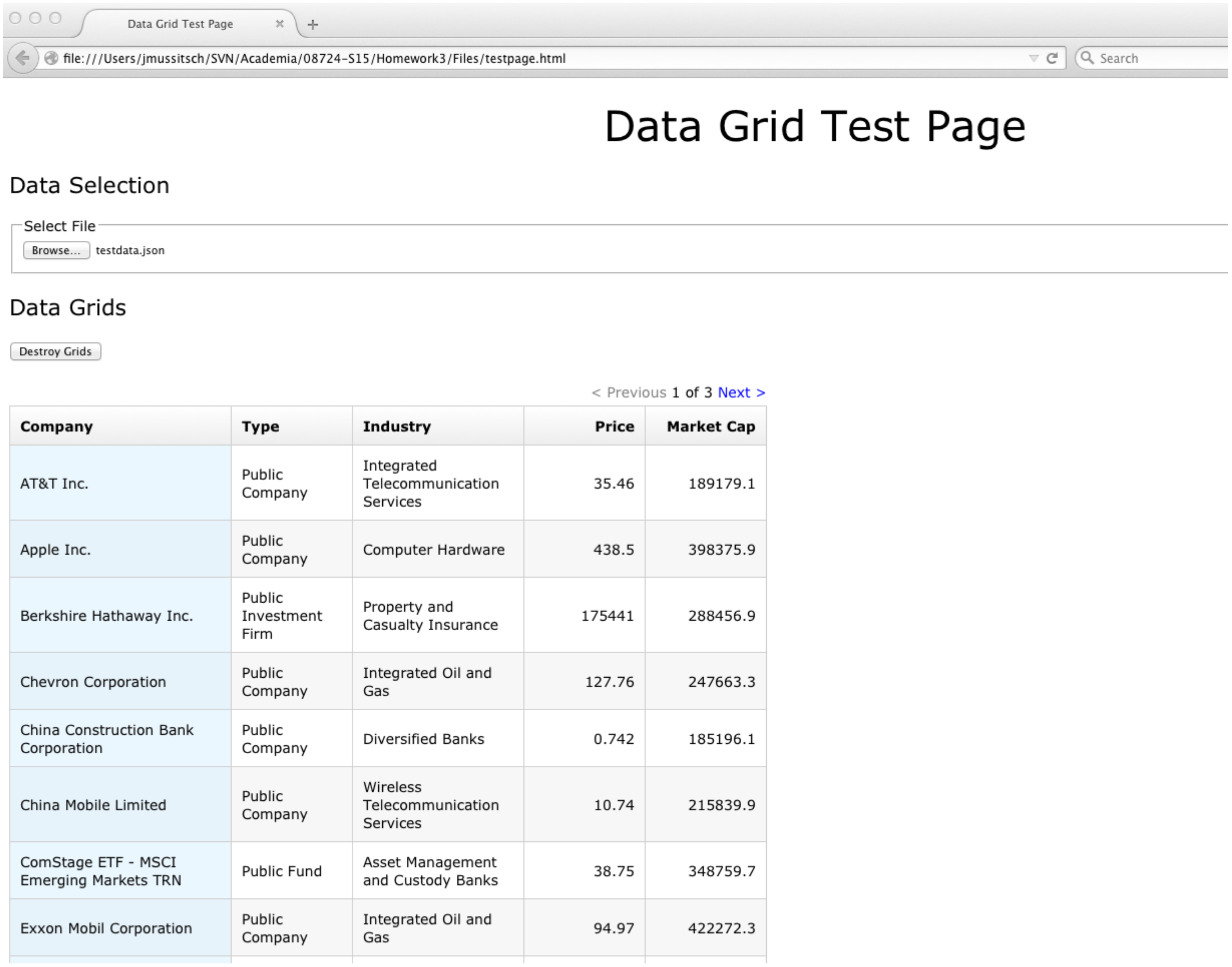
You wil create an additional directory called "datagrid". Inside that directory you will have two subdirectories called "css" and "js". Inside the directories you will create your "datagrid.css" and "datagrid.js" files respectively. The final directory structure for your assignment will look like this:

```
testpage.html
.jshintrc
css
  |-------- testpage.css
js
  |-------- testpage.js
testdata
  |-------- testdata.json
datagrid
  |-------- css
              |-------- datagrid.css
  |-------- js
              |-------- datagrid.js
```

The test page contains a file input control that you will use to select the testdata.json file to load data into your component grid:


Data Grid Test Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Data Grid Test Page
file:///T:/Academia/CMU Files/ClientSideWebTech.FINAL/Homework3/TestPage.html
Data Grid Test Page
Data Selection
Select File
Browse... No file selected.
Data Grids
File Upload
« CMU Files ▸ ClientSideWebTech.FINAL ▸ Homework3 ▸ testdata
Search testdata
Organize ▾ New folder
Favorites
Desktop
Downloads
Dropbox
Recent Places
Libraries
Documents
Music
Pictures
Videos
Homegroup
Name Date modified Type Size
testdata.json 7/26/2013 1:12 AM JSON File 4 KB
File name: testdata.json
All Files (*.*)
Open
Cancel

The testpage.js file provided adds an event listener on the input element and handles its *change* event. This will then create two different instances of your data grid component using the data file (the below screenshot shows part of the first grid):


Data Grid Test Page
file:///Users/jmussitsch/SVN/Academia/08724-S15/Homework3/Files/testpage.html
Search

Data Grid Test Page

Data Selection

Select File

Browse... testdata.json

Data Grids

Destroy Grids

< Previous 1 of 3 Next >

Company	Type	Industry	Price	Market Cap
AT&T Inc.	Public Company	Integrated Telecommunication Services	35.46	189179.1
Apple Inc.	Public Company	Computer Hardware	438.5	398375.9
Berkshire Hathaway Inc.	Public Investment Firm	Property and Casualty Insurance	175441	288456.9
Chevron Corporation	Public Company	Integrated Oil and Gas	127.76	247663.3
China Construction Bank Corporation	Public Company	Diversified Banks	0.742	185196.1
China Mobile Limited	Public Company	Wireless Telecommunication Services	10.74	215839.9
ComStage ETF - MSCI Emerging Markets TRN	Public Fund	Asset Management and Custody Banks	38.75	348759.7
Exxon Mobil Corporation	Public Company	Integrated Oil and Gas	94.97	422272.3

Each column in the grid will be sortable. Clicking on the different column headers will sort their respective column. The grid will also have an optional pager that allows users to view subsets of all the data.

Requirements

There are a number of requirements that your grid component must adhere to. These are outlined below.

No External Libraries, APIs, Etc.

For this assignment, you are **NOT** permitted to use any external libraries or APIs such as jQuery, Underscore, etc. All code must be written by you.

Creating a New Grid Component

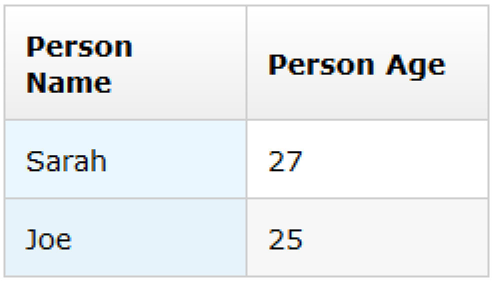
The code for your component must be contained in the datagrid.js file. The **ONLY** global identifier that your code should add is that of the `DataGrid` constructor function. Your `DataGrid` component must have the following interface:

- A constructor function `DataGrid` that will take in an object that sets the various options that the grid uses to work; this object is defined as follows:
  - `data` — an array of objects that represent the rows in the grid; the objects in this array will have properties that represent the column data
  - `rootElement` — the DOM element that will contain the grid (you **cannot** make any assumptions about the id of the element, the element's location in the DOM, etc.)
  - `columns` — an array of objects that define column properties; the column objects will look like this:
    - `name` — Text to display in the column header
    - `align` — How column data should be horizontally aligned; acceptable values are `left` or `right`
    - `dataName` — The property name of the row object that represents the column's data
  - `pageSize` — an OPTIONAL page size that specifies the number of rows to show on each "page" of data
  - `onRender` — an OPTIONAL function that will get called by your grid every time it is:
    - initially displayed
    - sorted
    - paged
- Instances of your `DataGrid` type must have a method called `destroy` that when called, removes the grid from the DOM

For example, to create a new grid I could call your constructor as follows:

```
let myGrid = new DataGrid(
   {
          "data": [
                 {"Name": "Joe", "Age": 25},
                 {"Name": "Sarah", "Age": 27}
          ],
          "rootElement": document.getElementById("gridwrapper1"),
          "columns": [
                 {"name": "Person Name", "align": "left", "dataName": "Name"},
                 {"name": "Person Age", "align": "left", "dataName": "Age"}
          ],
          "onRender": function() { console.log("Grid rendered..."); }
   }
);
```

The preceding code would create a grid that looks like this (note that since no pageSize was specified we do not show a pager):



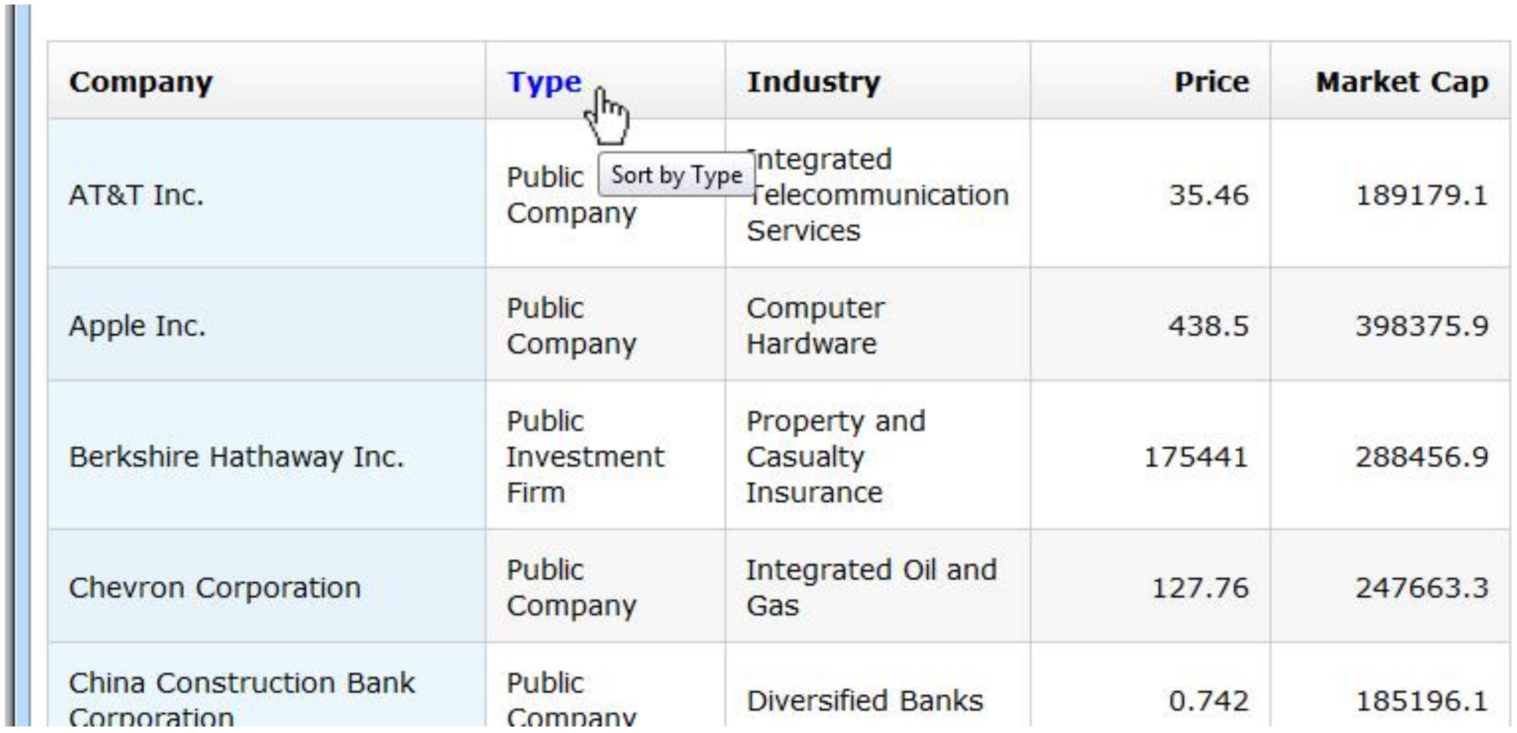
Person Name	Person Age
Sarah	27
Joe	25

When the constructor is called, the grid should be immediately rendered. By default, when rendered initially, the first column should be sorted ascending (strings alphabetically “a” to “z”, numbers in ascending order of value). Also, when initially rendered if the grid has a pager then the first page of data should be displayed.

Your grid should not have any restrictions on number of columns or rows. Obviously, it should work with the test data and columns used by my test script I have provided, but it should also work if I used a different set of data with different columns. You can assume, however, that the data will always have strings or numbers for column values.

Sorting

As stated above, by default the grid will be sorted by the first column in ascending order when first rendered. Each column must be sortable. The currently sorted column must have a different background color than the other columns, similar to what is shown in the screenshot below (HINT: if you give the sorted column a background color that contains an alpha value you will retain the alternating row color differences as well since the column color will have some transparency). Hovering over a column header must change the header text to blue, the cursor to a pointer, and also display a tool tip “Sort by [column name]” as shown below:



Company	Type	Industry	Price	Market Cap
AT&T Inc.	Public Company	Integrated Telecommunication Services	35.46	189179.1
Apple Inc.	Public Company	Computer Hardware	438.5	398375.9
Berkshire Hathaway Inc.	Public Investment Firm	Property and Casualty Insurance	175441	288456.9
Chevron Corporation	Public Company	Integrated Oil and Gas	127.76	247663.3
China Construction Bank Corporation	Public Company	Diversified Banks	0.742	185196.1

When the header is clicked, the grid should be rerendered and be sorted by the sorted column. If the header of the currently sorted column is clicked, the direction of the sort should change (so if the column is currently sorted in ascending order, clicking the header should change the sort to descending order). If a different column than the currently sorted one is clicked the data should sort by the newly sorted column and sort in ascending order (HINT: remember, the Array type has a sort function that may be useful).

Paging

When a pageSize value is specified, your grid should divide the data such that each page displays that number of rows. If no pageSize is specified or the total number of rows <= the pageSize then do not render a pager. Since the total number of all elements in the grid may not be evenly divisible by the pageSize value, the last page of the grid may have fewer rows than the pageSize value. To determine the total number of pages your grid will have you divide the total number of rows by the pageSize and round up to the nearest integer:

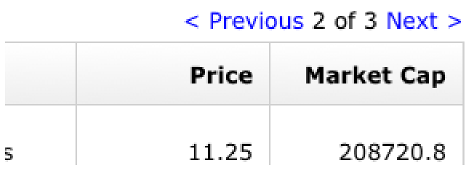
$$totalPages = \lceil totalRows / pageSize \rceil$$

JavaScript's `Math` reference type has a `ceil` method that will be useful.

Paging works such that on the first page the grid will show rows 1 to pageSize, the second page will show rows pageSize + 1 to 2 * pageSize, etc. So, for example if the grid has 25 rows and the pageSize is 10:

- Page 1 will show rows 1 to 10
- Page 2 will show rows 11 to 20
- Page 3 will show rows 21 to 25

The pager should display a clickable Previous element, the current page out of the total number of pages, and a clickable Next element as show below:

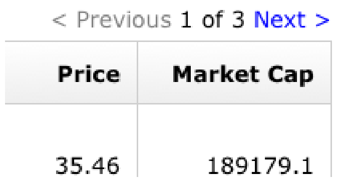


< Previous 2 of 3 Next >

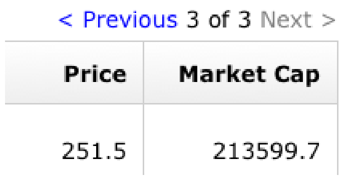
	Price	Market Cap
s	11.25	208720.8

The pager should be positioned above the grid and aligned to the right (although it is cheating a bit semantically you may find the caption element and related CSS caption properties to be a useful container for your paging elements). Clicking "Previous" should display the grid showing the previous page's data. Clicking "Next" should display the grid showing the next page's data. Hovering over the "Previous" and "Next" elements should change the cursor to a pointer (similar to sorting).

On the first page of data, the "Previous" element should be disabled and indicate to the user that it can't be clicked. Similarly, on the last page the "Next" element should be disabled. Here is how the pager would look on the first and last pages respectively:


< Previous 1 of 3 Next >

Price	Market Cap
35.46	189179.1


< Previous 3 of 3 Next >

Price	Market Cap
251.5	213599.7

Styles

You must create a datagrid.css file that contains style rules for your grid. In addition to the sort-specific and paging-specific styles discussed above, you must at the very least:

- Have collapsed borders
- Have some cell padding
- Have alternating row background colors
- Have a distinctive background for your header row cells (I have added a gradient for the example in my screenshots but you are not required to use a gradient)

Feel free to add any other style rules that you would like. However, you are only required to do what is outlined above.

Also, keep in mind that you must make sure that your column alignments adhere to the values provided by the developer using your grid (as discussed above).

The onRender Callback

If a developer using your grid sets the `onRender` property of the object passed to the `DataGrid` constructor, you should call this function whenever you render the grid. This would be when the grid is first rendered as well as any time after it is sorted or paged.

The test functionality that I provide will log to the console each time my code receives the callback for the first grid that I create. You can use this to see if you are doing things correctly. You can view the console output in Firebug and Chrome's Developer Tools.

Performance Considerations

To help with the performance of your grid, any properties and methods that **can be shared** amongst instances of your `DataGrid` type should be put on its prototype. Be careful that you **do not** put properties or methods that **cannot be shared** on the prototype.

JSHint

JSHint is an extremely useful tool that can detect errors and potential problems in JavaScript code. A requirement of this assignment is that running JSHint on your datagrid.js file returns no errors. You should run it on your code regularly as you develop because it will help you a lot in finding errors and bugs in your code.

To install JSHint, you will need to install Node.js so you get NPM (the Node Package Manager). You can get Node.js here if you do not already have it installed:

https://nodejs.org/en/

You want to download the LTS version (current version is 6.10.2). After installing Node.js you need to install JSHint. Run this command from the command line:

```
npm install -g jshint
```

This will install JSHint globally. Then in the datagrid/js directory run:

```
jshint datagrid.js
```

JSHint will look for a configuration file called ".jshintrc" in the current directory and then all ancestor directories so it should find the configuration file two directories up. The same configuration file will be used when grading your assignment. This will output any errors or warnings. Read more about JSHint and the configuration options in the provided file here:

http://jshint.com/docs/

Submission

To submit your assignment, add your “datagrid” folder that contains the “css” and “js” subfolders and their respective “datagrid.css” and “datagrid.js” files to a ZIP archive, name the ZIP file ***Homework3_[andrewid].ZIP***, and upload to Blackboard under Assignment 3 by the due date/time above. For example, my ZIP file name would be ***Homework3_jmussits.ZIP***.

Grading Rubric

This assignment is worth 100 points (20% of your total grade). The following is how the assignment will likely be scored:

- The DataGrid type’s interface has been implemented correctly (constructor and `destroy` method work correctly) **[10 points]**
- The grid renders the HTML correctly using the data and options provided via the constructor **[10 points]**
- Sorting works correctly **[20 points]**
- Paging works correctly **[20 points]**
- CSS styles done correctly **[10 points]**
- The onRender callback done correctly **[10 points]**
- No global identifiers other than the DataGrid constructor introduced **[10 points]**
- No errors when JSHint is run on datagrid.js is [**10 points**]

Hints

- The table element has special methods for creating table related child elements. Check out the HTML specs or W3Schools for help: http://www.w3schools.com/jsref/dom_obj_table.asp.
- Since you will be building your grid from an array of data it may be a good idea to store that array in your grid object and sort/page the array and render your HTML from that.
- Remember, arrays have a sort function that you can pass a function as a parameter to do comparisons (see my array examples).
- Maintaining some state in your grid object about what column is currently sorted, what direction, and what page is being displayed may be useful.
- Use HTML custom data attributes where useful.
- This assignment does not require a lot of code. But it does require a good bit of thought. Do not wait until the last minute to start or you will find yourself in trouble.