Client-Side Web Technologies – Homework Assignment 3

Due: February 24 @ 11:59 PM

Value: 15% 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.

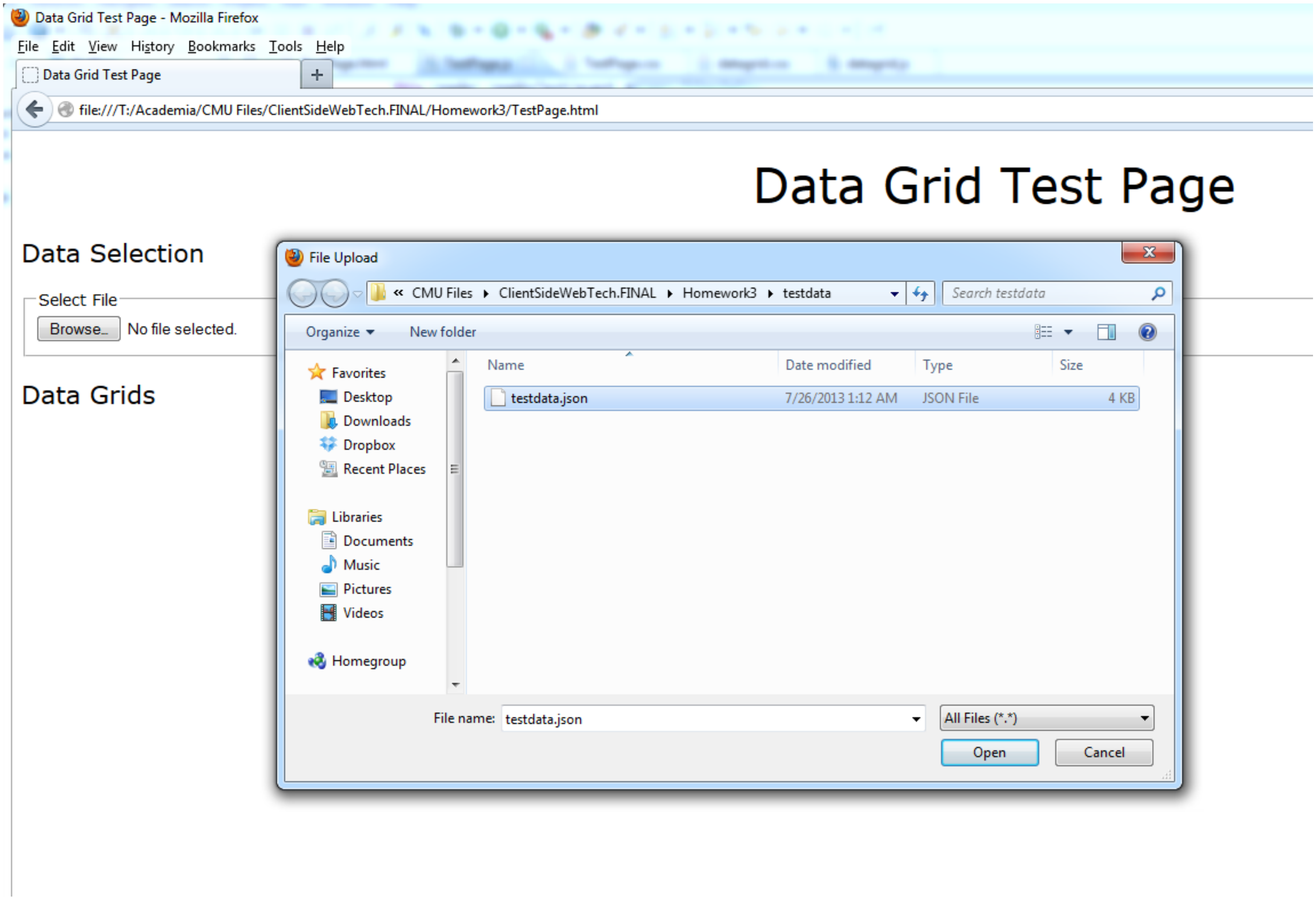
I have created a test page that you will use to test your component with. I have also created a CSS file to set some basic page styles as well as a JavaScript file that will handle testing your component. Additionally, there is a file that contains test data to populate your grid. Here is the file structure:

```
testpage.html
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
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:///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

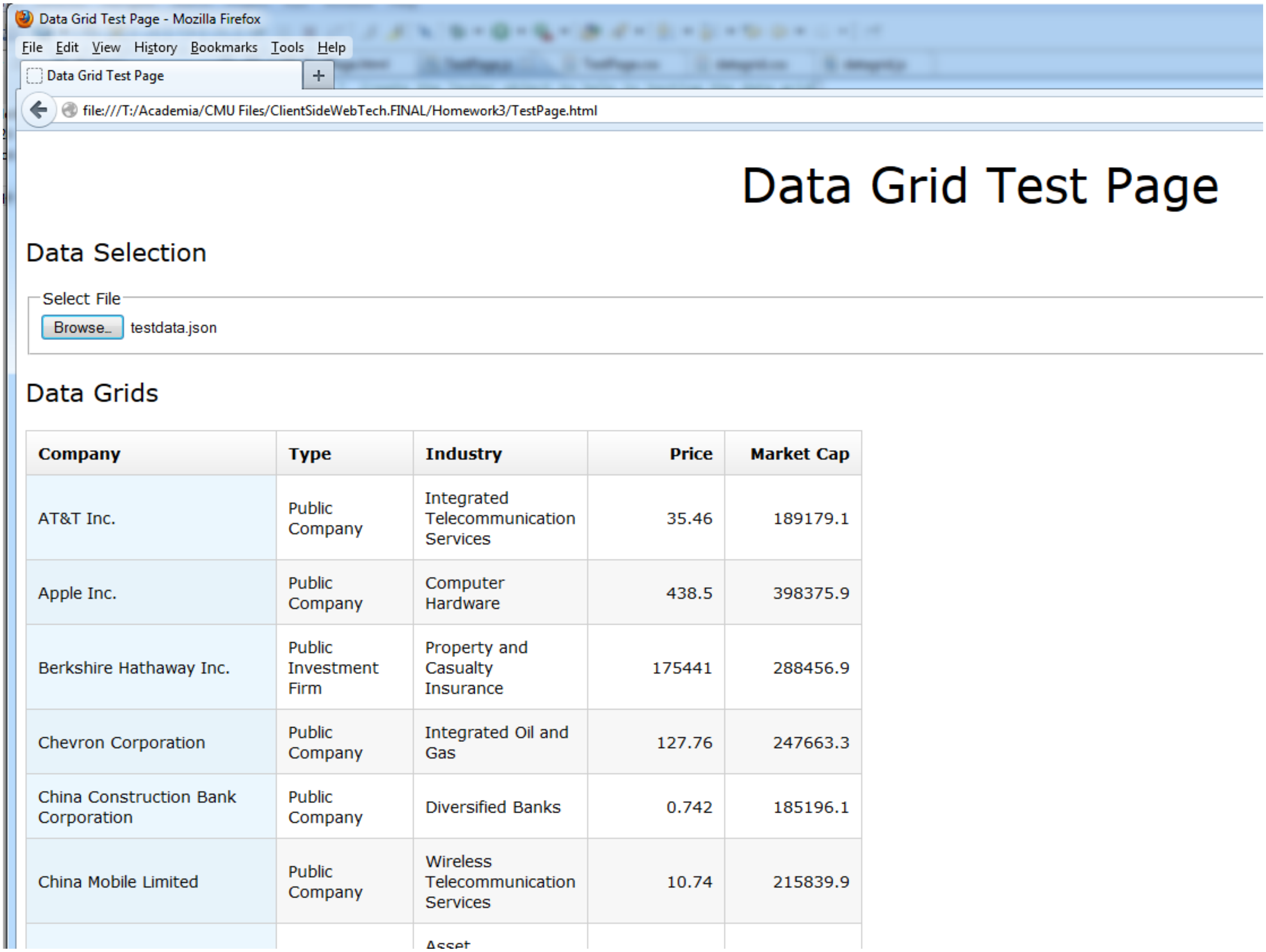
« CMU Files ▸ ClientSideWebTech.FINAL ▸ Homework3 ▸ testdata

Name	Date modified	Type	Size
testdata.json	7/26/2013 1:12 AM	JSON File	4 KB

File name: testdata.json

All Files (*.*)

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 - 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... testdata.json

Data Grids

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
		Asset		

Each column in the grid will be sortable. Clicking on the different column headers will sort their respective column.

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. All code must be written by you. You will get to use jQuery in the next assignment.

Creating a New Grid Component

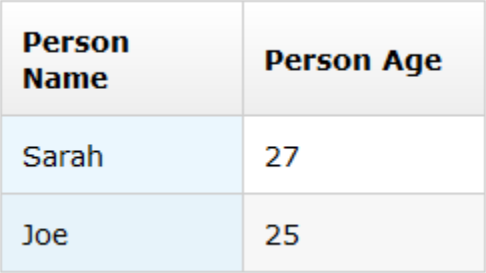
The code for your component must be contained in the datagrid.js file. The ONLY global identifier that your code should add is that of the `DataGrid` constructor function. Your `DataGrid` component must have the following interface:

- 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
  - `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`
    - `width` The width of the column in pixels
    - `dataName` The property name of the row object that represents the column's data
  - `onDraw` an OPTIONAL function that will get called by your grid every time it is drawn (rendered)
- 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:

```
var myGrid = new DataGrid(
   {
         "data": [
               {"Name": "Joe", "Age": 25},
               {"Name": "Sarah", "Age": 27}
         ],
         "rootElement": doc.getElementById("gridwrapper1"),
         "columns": [
               {"name": "Person Name", "align": "left", "width": 100, "dataName": "Name"},
               {"name": "Person Age", "align": "left", "width": 100, "dataName": "Age"}
         ],
         "onDraw": myGrid_onDraw
   }
);
```

The preceding code would create a grid that looks like this:



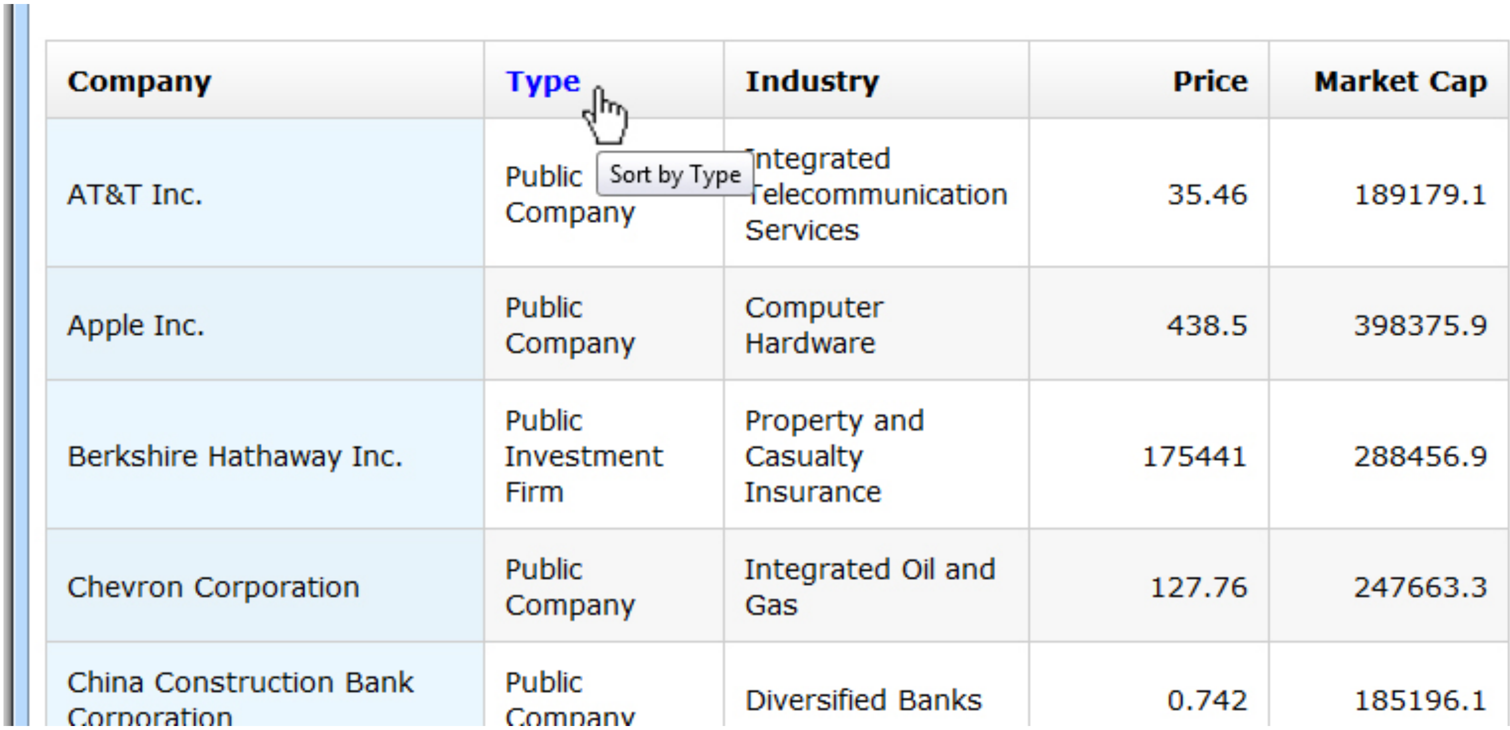
Person Name	Person Age
Sarah	27
Joe	25

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).

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 a 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 redrawn 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).

Styles

You must create a datagrid.css file that contains style rules for your grid. In addition to the sort-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 widths and alignments adhere to the values provided by the developer using your grid (as discussed above).

The onDraw Callback

If a developer using your grid sets the `onDraw` 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. You must pass as a parameter to the callback the instance of your `DataGrid` object.

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.

Submission

To submit your assignment, add your "datagrid" folder than 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 150 points (15% 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) **[20 points]**
- The grid renders the HTML correctly using the data and options provided via the constructor **[40 points]**
- Sorting works correctly **[40 points]**
- CSS styles done correctly **[20 points]**
- The onDraw callback done correctly **[20 points]**
- No global identifiers other than the DataGrid constructor introduced **[10 points]**