Front-End test

Using React or Angular 1.5+, write an app that will display various tables depending on resource availability on the server. The table resources are static JSON files, and they will imitate backend API endpoints on the server, that can be fetched via XHR requests. Their file names are in format `{tableName}.json`, where `{tableName}` is variable and can be used in the URL route, page title or page heading, and one could add more JSON files on the server to represent additional tables. Adding JSON files should not require any changes on the frontend.

You will be provided three example JSON files that would represent resources for three tables. The JSON files will be formatted as follows:

```
[{
      field1: value1,
      field2: value2,
      ...
},{
      field1: value1,
      field2: value2,
      ...
},{
      field1: value1,
      field2: value2,
      ...
},...]
```

These are the tasks that need to be completed:

1. You need to have a router in format `/table/{tableName:string}`. If someone writes an URL for a JSON resource, he/she should write its table name. Otherwise show 404 page. For example. If I have priceList.json available on the server, I would write `/table/priceList` in the address bar to show that data.
2. Write a service for fetching data by `{tableName}`
3. The cells in the table should be in formatted as follows:
    a. If it's an integer number, write it aligned to the right
    b. If it’s a decimal number, write it with a dollar sign prefixed, fixed to two decimals and aligned to the right
    c. If it’s an URL, write it inside an anchor, linking to that URL

    d. If it’s a date, use angular-moment library to filter it in relative time (from-now). Also, moment has a parser for dates that are strings in ISO standard, you can use that for validation.
    e. If it's an object, write it as a top-down list with pairs `{subFieldName: value}`
    f. Any other case is a simple string.

  Note that you don't have to understand the context of the data, just following the rules per value recognized is accepted (for example, if column name money_paid is not a decimal, but an integer, write by rule a., not b.)

4. Every column needs to have an input for filtering, and clicking on column title should sort the list.
5. The column title (field name from JSON) should be formatted in human readable manner, removing _ from it, and the first character should be uppercase.

You can use lodash for helper functions if you need it, and add helper tools for complex validation or type recognition.

Some rules to follow:
1. Use bootstrap v3 or v4 for UI elements
2. For Angular, use Jon Papa style guide, otherwise for React use Airbnb style guide. This is not a strict rule, you can apply even just a few rules from these guides in order help the reviewer understand which one you used. Other style guides are accepted if the guide is standardized in the community, and if you are using it, please mention that in project readme.md.
3. Use ES6+, Babel and separate builds for production and for development (production build needs to be uglified). You can choose any build system, as long as the project is initialized via NPM and all the tools are fetched via NPM (not cdn links).
4. If Angular is used, organize the architecture in components, and keep template and controllers for a component in the same folder.