WalletHub

Frontend Angular Test

Create an angular app that meets the following requirements:

1) Use ui-route to create at least 3 pages that share a single state for loading them. To achieve this, have state url parameters by dynamic and load views/controllers based on those parameters.

Create a method which can load the controller and template of each page on demand, when route changes. Make the state url option have 2 parameters with the second one optional (this can be done by using regexp on the url parameters) so that one of the pages can load additional information inside another ui-view based on that parameter.

2) Provide example on how to pass data between page controllers. Have an input field in one page of which contents is displayed in the second page. Please do not use $rootScope.

3) Have page transition animations.

4) Create an array of 50 items, each item having 3 properties. Create an option to sort the array by one or two properties. Make the array be filterable by a search input.

5) Have a directive that is dependent of ngModel, that restricts the input to numbers only and auto formats the input as a phone number like (541) 754-3010. The field must change its format as user types. Do not create a $filter for this directive as that does not mean the directive itself is doing the format.

6) Have a directive that is dependent of ngModel, that restricts the input to numbers only and auto formats the input into currency, ex: $1,234. The field must change its format as user types. Do not create a $filter for this directive as that does not mean the directive itself is doing the format.

7) Create 3 number inputs that sum up into an additional field. Editing each of the 3 inputs will always update the sum value. The fourth field can also be editable, and whenever user changes this one its value must be spread across the other 3 fields. Example:

Initial values:
#1 = 100 #2 = 100 #3 = 100 #4 = 300

User interacts with field #1:
#1 = 200 #2 = 100 #3 = 100 #4 = 400

User interacts with field #4 after editing #1:
#1 = 150 #2 = 75 #3 = 75 **#4 = 300**

8) Create 3 inputs each with maximum char. length 5. When entering the 5th char. in an input the cursor should jump to the next input right after the char. has been inserted. When deleting the 1st char. of an input the cursor should jump at the end of previous input right after that char. has been deleted.

9) Make use of require.js to load the app resources.