Objective

Please implement the functionality and features defined in this section using the steps provided below.
A demo of the working page is shown in the video (markup-exercise-functionality.swf) that is included with the exercise to give you an idea of the expected functionality.

1. Create an HTML page that matches that attached screenshots found in the screenshots folder.
2. Carve up and extract any extra images that you may need from the attached screenshots. We have provided an API server to serve most of the text you will need for the different sections as well as images for the map, thumbnail (see public/assets/images), and icon font (public/assets/fonts).
3. You are free to use a JS library/framework of your choosing ***unless we requested that you use a specific one***.
4. To make things a little simpler, there is a http node server for serving files out of the public folder. To start the server, you need at least Node v4 installed. From the command line, navigate to the *simple-server* folder, and type: ***node server*** and then hit enter***.***
5. Feel free to restructure the folders to your preference, but keep in mind if you use the node server that it serves files from the public folder. You may also use any file server and build set up you like, but make sure we can build and run it on our end. If you use any node modules, **DO NOT** include them in your returned package. Instead, include a package.json file.

Requirements

- ***Tabbed Content***
    - The page should have functional tabs. When a tab is clicked the tab's content should be shown and the old tab's content should hide.
    - **Default tab selected should be the Description Tab**
- ***Expandable/Collapsible Description and Details Section***
    - The content in Description and Details tabs should be an expandable and collapsible section with a View more/View less link. If the section is expanded and the VIEW LESS TEXT is clicked, only 200px of the content should be visible in that section and the VIEW LESS TEXT should change to the VIEW MORE TEXT. If the section is collapsed and the VIEW MORE TEXT is clicked, all content should become visible for that section and the VIEW MORE TEXT should change to the VIEW LESS TEXT.
        - Description expandable trigger text:
            - VIEW LESS TEXT = SHOW FULL DESCRIPTION
            - VIEW MORE TEXT = HIDE FULL DESCRIPTION
        - Details expandable trigger text:
            - VIEW LESS TEXT = VIEW MORE DETAILS
            - VIEW MORE TEXT = VIEW FEWER DETAILS
    - Initial state for the expandable/collapsible section is collapsed.
- ***Location Link***
    - The map marker and text "Strip" under the product name, should open the map tab and scroll the content in to view when clicked.
- ***All available hotels list*** *(left column under hotel image)*
    - The list returned from the API will be unordered and contain duplicates. You will need to order them alphabetically and remove the duplicates.

All other links on the page are not expected to be functional. The map should just be a static image scraped from our screenshot (also included under public/assets/images/map_venetian.png). Do not try to pull the map data from Google.

Expectations

- Markup should be semantic and accessible.
- Should display correctly on IE11, Edge, Firefox and Chrome.
- The review is based on the HTML, JavaScript, and CSS produced. The coding should be completed as if it were going to be used on a real website. Consider how to make the code reusable for other hotels as well as how to make the page load as fast as possible.

API Details

Included with the node http server (mentioned above) is an API server. Once running, it provides 2 endpoints for retrieving data:

- Venetian hotel details: http://localhost:8888/api/hotels/venetian
- All available hotel list (left column in screenshots): http://localhost:8888/api/hotels/

It's a simple *GET* server and just serves the JSON files located in /public/api/hotels. Start the server and navigate to those endpoints to understand how the data is structured. Please try to fetch all data through the API without hard coding any of the content in the application itself.