1. Open index.html (in the root of the project) and app.js in the JS folder
2. Starting with index.html - Ember starts off normal, you can see the DOCTYPE, html, head and body tags, all the normal linked in CSS, notice in this example we are using bootstrap also.
3. Scroll down to line 16
   1. Between line 16 and 43 is your base layout.
   2. If you flip to the app.js file, line 5, this line is what starts the ember app
4. Back to index.html line 16, in this tutorial we are using the built in handlebars templating system. Ember views do exist but theyre more for complex views
5. So to start, each page will be in a script block with the type text/x-handlebars
6. Then you have normal html and bootstrap tags and classes until line 20. This is a {{link-to}} helper.
7. You can see the same on lines 27 – 30.
8. Since the entire site is in one html file, the link to helper links to a route (in app.js) which then looks for its data-template-name companion in index.html
9. ---walk though {{#link-to "about" class="btn btn-primary"}}About{{/link-to}} into app.js line 11.
10. This is where you build your routes, or links. I will explain the resource in a minute.
11. ----pull up screen shot of ember inspector
    1. after adding these routes, you would pull up ember inspector and see these pre-built for you by simply creating the route. This is how ember works for you with its convention over configuration
12. back to index.html, line 38 where you see {{outlet}}, this is where each template will load. From ASP you can relate to this top script tag being your master page, and the below script tags being your content pages.
13. Open index.html in any browser, notice nothing on the pages
14. So first we will uncomment lines 55 to 68 to show the home template.
    1. If you recall the index route built in app.js…WAIT we didn’t make the index route! The index is one of the few default, pre-built routes made for you.
    2. So as long as the data-template-name is equal to index, this will be the first page loaded in the outlet above.
15. Moving on to line 77 to 93, uncomment these to build the about page
    1. notice line 76 uses the data-template-name but the line 77 uses id with the same route, either will work.
16. Uncomment lines 104 – 121 and this will build the collections page.
    1. This is where it was a bit tricky for me. This section uses a custom component (derived from web components)
    2. In the id you must reference a component with component/1stword-2ndword
    3. This allows you to connect to what looks like a 2nd view, note line 116 where you create a loop with the data making a single collection in collections.
    4. Back to line 107 {{yield}} will return all the data from the collections below
    5. Back again to line 106, this bind-attr is used because you must bind external resources (like this example is the src is the img given from the data)
    6. Moving to app.js line 26, this is where the data came from, this JSON array. Note the image path used, and others used in index.html
    7. Now refresh your browser, and view the page source. If you see divs everywhere, that’s because unless otherwise told, this is what ember uses…not very semantic is it.
    8. So back to app.js, uncomment lines 50-53. This will take the single collection component made in index.html and change the tagName to an article, and add some classes for extra styles
17. Back to index.html, uncomment lines 130-155. Here we have our exhibits.
    1. Its similar to the collections but instead of showing all the exhibts at once, we create links to them on line 144-146. Before we get too far, go to app.js line 60.
    2. You can see this is the Exhibits route, returning exhibits, which are built on line 75 as another JSON array. This time you have info and an id per object.
    3. In your browser, navigate to the exhibits
       1. To start, you can see the count in the page is 3, this is hardcoded in index.html line 136
       2. Moving back to app.js, uncomment lines 111 – 113, this will return our total exhibits and we will use this as a handlebar to dynamically show the exhibits
       3. Back to line 136 in index.html , remove the 3 and type {{totalExhibits}} save and refresh your browser
       4. To show you it is dynamic, go back to the app.js and modify the JSON by cutting the last object (make sure to remove the comma) save and refresh and your browser should now show only 2 exhibits
       5. Now we can totally remove that JSON var (lines 75-93) as well as lines 60-64
       6. Uncomment lines 66-72
       7. This same exhibits file is located in js/exhibits.json so this route will now use the jQuery getJson method to get the file, and then use the .then() method to wait for it to load, and lastly it will return the data. This is an instance of JS promises also.
18. Now, line 144, link-to ‘exhibit’ this in index.html, coordinates with the route in app.js lines 97-105. Specifically line 99, from the json file we get each exhibit\_id (-1 because its 0 based) this then links to the correct exhibit to return to the EXHIBIT template which will load into the {{outlet}} handlebar
    1. exhibitTitle was made in app.js lines 118-120.
19. Lastly the notes page.
    1. Explain on the fly
20. All other script tags loaded at bottom