Mob Programming Demonstration for CSUSM Students

August 3, 2018

Initial Setup – previously completed

The environment you are working within has been modified to support this demonstration with the installation of three tools:

- Atom – this is the code editor that was used in class. (https://Atom.io) At Hunter we typically use Visual Studio Code (https://code.visualstudio.com)
- NodeJS – this is an open source web host that allows you to reference web pages locally. (https://nodejs.org)
- Angular CLI – the Command Line Interface for Angular. (https://cli.angular.io/) – note the command line to install this can be run from any folder it's: "npm i -g @angular/cli". These instructions were created with Angular CLI version 6.1.2.

Angular applications are built to create dynamic web applications and turn into HTML, CSS and JavaScript. This walkthrough focused on CSS and with a small amount of scripting.

Getting Started

Once you have connected to the development machine you'll want to open a command window to get started. For this first section most of the navigation will come from this script as we introduce the generate and introduce the base website.

1. Navigate to: C:\ProjectFiles
2. On the command line type: "ng new CSUSM" - this will generate a new web app called CSUSM.
3. After the project has finished generating, Navigate to the newly created CSUSM sub-folder.

The goal of these steps is to introduce the basic elements of testing all actions occur on the command line:

4. End-to End testing example, type: "ng e2e"
5. This will take between 30 seconds to a minute to run.
6. What is occurring on-screen? The application is compiled, Chrome opens with the website and quickly closes. Once completed you are back on the command line.
7. In the text you should see and indication that a single test ran and completed successfully.
8. Unit Test example, type: "ng test"
9. Once again there will be a slight delay as the application is built.
10. Chrome will open, but this time it will stay on screen after running 3 unit tests.
11. Review with the students that the software the test engine ran three pre-build unit tests. Chrome will be displayed indicating it is being driven by Karma v1.7.1, below the information for Chrome you'll see that the tests are Jasmine tests.
12. Hunter uses Test Driven Development for all our projects, and the CLI helps use with basic example of key forms of testing.
13. Do not close chrome, instead return to the command window and type Ctrl-C twice which will close Chrome and allow termination of the test batch job. One way to develop with Angular is to leave the test engine running while changes are made it was outside our scope.

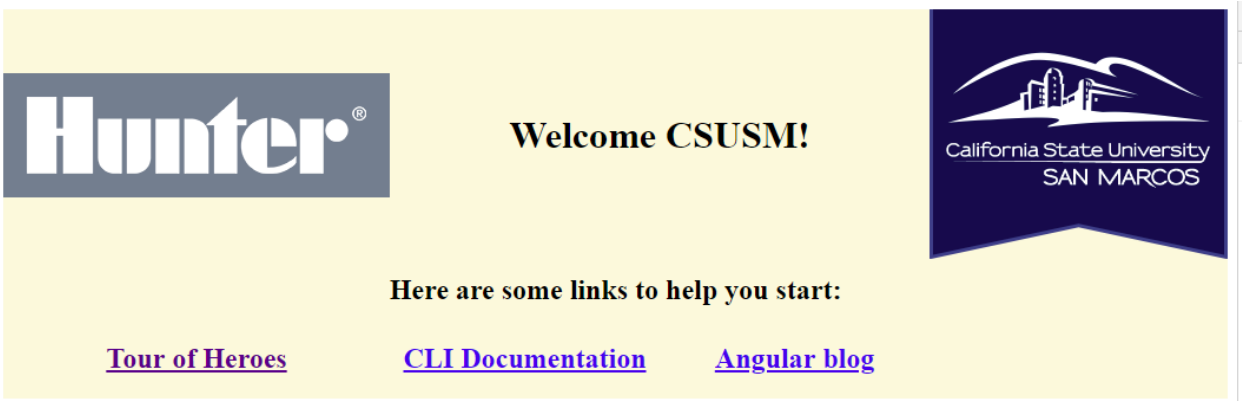
14. To finish this section on the command line type: ng serve
15. This starts the angular server, plan to wait about 30 second for the server to be running.
16. Open Chrome and have the students navigate to: http://localhost:4200

Working with HTML and CSS

The goal in this section is to introduce the code and allow navigation by the students.

17. Open the Atom editor and point it to the folder at: C:/ProjectFiles/CSUSM
18. Within the folder hierarchy you can note for the students that there are several JSON configuration files at the top which have been generated for them and that describe the dependencies of their website.
19. Navigate into the folder "src".
20. Navigate to the "app" folder.
21. Within the "app" folder open the **app.component.html** and **app.component.css**
22. Within the html, on line 4, remove the word "to", and save the change.
23. Draw the student's attention to how the display within Chrome is updated.
24. The next step is for the students to replace the Angular graphic from their page with the CSUSM banner logo:

25. *The students should update the image tag independently, however, if they are stuck,.navigate to CSUSM.edu. This logo is on the homepage, right click on the logo. Chrome will offer the option to Open the image in a new tab, which will display the path: https://www.csusm.edu/_resources/images/homepage/csusmribbon.png*
26. The goal is to implement a new CSS Grid on the default page.
27. The following image shows how the page should look when complete.
28. The grid has 4 columns, and 3 rows.
29. The top row is 300px, the center row is 100px and the bottom row is auto.
30. The columns are all defined as 1fr.



31. Content-Hunter.txt provides the HTML <img> tag and CSS for the hunter image. It is the source for this content and should be used to provide a graphic (students could provide their own.)
32. Copy the <img> tag from the Content-Hunter.txt file and paste it on the line before the title <h1> tag.
33. Next paste the CSS class from Content-Hunter.txt into the app.component.css file and save that – the image should now be visible above the text.
34. CSS Grid works by placing elements that are located within its display area into cells. The students should have everything they need to create the display above. Hints on the grid:
    a. The grid for this display is the outer-most tag.
    b. One good way to see what's actually in the grid is to give it a background color.
    c. Some elements from the original such as the <ul> and <li> tags should be deleted.
    d. Very few CSS classes are needed, one such would apply to the "title" to the h1 section.
    e. To have the text for "Here are some…" span all the columns, the students will need to apply a class that defines how many columns the text should occupy.
35. *Content-CSS contains the solution for the CSS Grid, it should not be opened unless needed..*
36. Once the students have a grid the students should be allowed to experiment further. What happens if they add 3 more links into the bottom section of the div? 10 more?

Scripting

Most students have familiarity with <script> tags within their pages. This is not how Angular approaches scripting. In fact the process through which you build and publish Angular sites strips out embedded <script> tags.

Point out that the generated site already uses a title that is 'scripted'. The page title is set within the Typescript file associated with the HTML. The typescript file actually drives the placement and contents of the HTML interpreted by the browser.

Because for this demonstration the students hadn't really worked with either typescript and had only minimal exposure to scripting, this part of the demonstration – which was meant to close out the presentation is written more prescriptive, but can be done independently.

37. Open the file app.component.ts file. The property title is assigned your project name: CSUSM.
38. There is a Content – Typescript.txt file on the desktop that can be opened.

39. The typescript in the content file will demonstrate changing CSS settings dynamically via script.
40. The code first creates a variable in the App component; 'titleStyle', which is initially undefined.
41. Below the definition of 'titleStyle is a new method called "changeColor".
42. "changeColor" changes the background-color style property for the title, by assigning a value to 'titleStyle'.
43. Note that the comma that follows the property in the sample script. It is possible to define multiple style properties at the same time. Anything defined in HTML or CSS can be changed from script.
44. Add the Typescript into the class. Place it on the line immediately below the title definition.
45. While this provides a way to update the page style dynamically – it doesn't actually hook into the HTML. Switch to the app.component.html to now link this script with the page.
46. A common event is the user clicking the mouse. Students might be familiar enough to know about 'onClick'. However, in Angular the way to attach is slightly different.
47. Go to the <h1> tag used for the title, for this demonstration the changeColor method will be used to update only this element.
48. Within the <H1> tag define a (click) handler. Set it to the name of the method "changeColor()"

    <h1 class="title" (click)="changeColor()">

49. This calls the method but doesn't map the behavior of the html to the resulting property in the script. To do that the [ngStyle] property needs to be data bound to the 'titleStyle' property.
50. Continuing to work within the tag, you will should have a tag that looks similar to the following:

```
<h1 class="title" (click)="changeColor()" [ngStyle]="titleStyle">
```

51. At this point if you've saved the fully updated application when they click on the title a "teal" bar should appear around the title, clicking a second time should clear that style from the screen.
52. This completes the planned change, however, if there was a problem you may want to debug.
53. One way to see what's happening in the script is to console log raw data.
54. Use the F12 key in the browser. This allows you to view raw data in the browser's console.
55. In the 'changeColor' method you can add the following line as the first line of code : console.log("!!!", this.titleStyle);
56. Note the "!!!" is optional but helps highlight the value, console log allows you to pass any number of different values separated by comma's which it will interpret and output as strings.

That completes what was put together for this 2 hour visit. Working as mobs and growing communication skills were also part of this demonstration so this was more than enough of an introduction to Angular development. In watching the teams work with this one of the more interesting parts was seeing them attempt to develop the CSS on their own. Most groups were approaching 60+ lines of CSS to try and get the behavior described. This highlights my 1st rule of CSS: - Less is More. Unfortunately it is also common behavior for most people working with CSS.