Mob Programming Demonstration for CSUSM Students

July 26, 2019

Initial Setup – possibly completed previously

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 8.1.2.

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.
   a. You will be prompted to add routing – this is optional, we won't use it today.
   b. You will be asked for the type of CSS file – for simplicity choose the default of CSS.
3. After the project has finished generating, Navigate to the newly created CSUSM sub-folder.

The students are new to development and are unfamiliar with testing. The goal of these steps is to introduce the basic elements of testing. All of these actions occur on the command line if you get an error you probably haven't navigated to the new sub-folder in step 3:

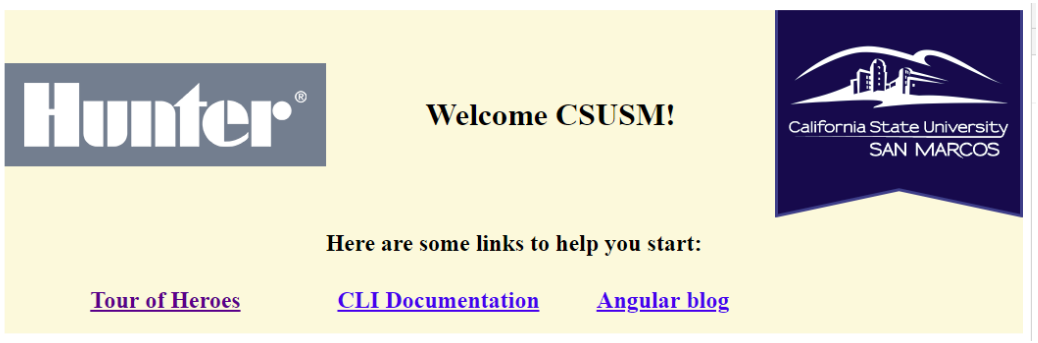
4. End-to End testing example, type: "ng e2e"
5. This will take between 30 seconds to a minute to run. What is occurring on-screen during this time? - The application is compiled, Chrome opens with the website and quickly closes. Once completed you are back on the command line with your test results.
6. In the text you should see and indication that a single test ran and completed successfully.
7. Unit Test example, type: "ng test"
8. Once again there will be a slight delay as the application is built, during which you can discuss how e2e and unit testing are different.
9. On completion Chrome will open, and this time it will stay on screen running 3 unit tests.
10. Review with the students that the software the test engine ran three pre-built unit tests. Chrome will be displayed indicating it is being driven by Karma v4.1.0, below the information for Chrome you'll see that the tests are Jasmine tests and the test results.

11. Hunter uses Test Driven Development for all our projects, and the CLI allows us to demonstrate basic testing behavior and how easy testing can be. One way to develop with Angular is to leave the test engine running while changes are made it is however outside the scope of what we'll be doing today.
12. To clean up from the test you do not need to close chrome. Return to the command window and type Ctrl-C twice which will close halt the test engine and Chrome.
13. In order to view our changes while we work, from the command line now type: ng serve
14. This starts the angular server, plan to wait about 30 second for the server to be running.
15. 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.

16. Open the Atom editor and point it to the folder at: C:/ProjectFiles/CSUSM
17. 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.
18. Navigate into the folder "src".
19. Navigate to the "app" folder.
20. Within the "app" folder open the **app.component.html** and **app.component.css**
21. Within the html, on line 4, remove the word "to", and save the change.
22. Draw the student's attention to how the display within Chrome is updated in real-time as changes are saved.
23. The next step is for the students to replace the Angular graphic from their page with the CSUSM banner logo available from CSUSM.edu:
24. *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*
25. The goal is to implement a new CSS Grid on the default page.
26. The following image shows how the page should look when complete.
27. The grid has 4 columns, and 3 rows.
28. The top row is 300px, the center row is 100px and the bottom row is auto.
29. The columns are all defined as 1fr.
30. The next goal is to create an image tag for the Hunter logo.
31. The image tag should be added on the line before the title's <h1> tag.



Welcome CSUSM!

Here are some links to help you start:

Tour of Heroes CLI Documentation Angular blog

32. The following HTML tag provides a link to a Hunter logo – note the image is white and has a transparent background so it may be difficult to see until we create CSS after this step:

```
 <img  class="hunterImage" alt="Hunter Logo"
src="https://www.hunterindustries.com/sites/all/themes/hunter_responsive/image
s/logo_hunter.png"  />
```

33. To see this switch to the CSS file and paste the following CSS:

```
.hunterImage {
  padding: 20px;
  background-color: slategray;
}
```

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.

    ```
    .title {
      grid-column-start: span 2;
    }
    ```

    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.

    ```
    .intro {
      grid-column-start: span 4;
      padding-left: 30px;
    }
    ```

35. 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.

For security, Angular sites strip out embedded <script> tags.

Note that the generated site already has a title that is 'scripted'. The page title is set within the Typescript file associated with the HTML. With Angular the backing typescript file can drive the placement and contents of the HTML. The data which is used can be bound and everything from content to style can be modified when events are raised from the HTML to the backing script.

Because our expectation in planning for the students was that they will have little experience with typescript and only minimal exposure to scripting, this part of the demonstration is very prescriptive (it spells out simple steps).

36. Open the file **app.component.ts** file. The property title is assigned your project name: CSUSM.
37. The first step is to create a property in the App component.
38. 'titleStyle' should be used as the name and it can be left untyped or typed as a string.
39. 'titleStyle' should be located on the line below the existing definition of "title", and should be left undefined.
40. Below the definition of 'titleStyle' declare a method called 'changeColor'.
41. 'changeColor' will change the background-color style property for the title.
42. It will do this, by assigning a value to 'titleStyle'.

```
1. changeColor(): void {
2.     if (this.titleStyle === undefined) {
3.       this.titleStyle = { 'background-color': 'teal' };
4.     } else {
5.       this.titleStyle = undefined;
6.     }
7. }
```

43. As you enter the code, consider: the method does not return a value.
44. The first line is a conditional which checks the current status of the 'titlestyle' property to see if it is defined.
    a. This is a good point to discuss how Javascript and by extension Typescript treat undefined as something different from null – although the net result is often the same.
45. Line 3 of the method updates the property 'titleStyle' if it was undefined. If you are familiar with "in-line styling" of HTML tags you'll see that's what is being set.
    a. Note that the comma that follows the property in the sample script. It is possible to define multiple style properties at the same time.
46. Line 4 provides an else so that this change can be toggled by clicking multiple times, and the remaining lines close out the method.
47. While we have updated a property, there isn't anything tying this property back into the HTML which the browser is interpreting. To do this you need to switch to the html.

48. Switch to the **app.component.html** to now link this script with the page.
49. Go to the <h1> tag used for the title, for this demonstration the changeColor method will be used to update this tag.
50. Within the <H1> tag define a (click) handler. Set (click) to the name of the method 'changeColor()'

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

51. This calls the method whenever the title is clicked. If you run the code now, it will call, but nothing will happen.
52. That's because the HTML also needs to have a link to the property in the script file which was updated.
53. To map the behavior of the html to the resulting property in the script, the [ngStyle] property associated with this HTML tag needs to be data bound to the 'titleStyle' property of the Typescript.
54. 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">
```

55. At this point if you've saved the fully updated application when the title is clicked on, a "teal" bar should appear around the title. Clicking a second time should clear that style from the screen.

Debug with Console.Log() and F12

56. In theory this completes the planned change, however, what if there was a problem you may want to debug – and certainly it would be good or us to introduce the most common way of handling this.
57. One way to see what's happening in the script is to console log raw data.
58. Use the F12 key in the browser. This allows you to view raw data in the browser's console.
59. In the 'changeColor' method you can add the following line as the first line of code : console.log("!!!", this.titleStyle);
60. 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.
61. Once the page refreshes with your new code, you can see the log message appear in the console that is part of the F12 display.
62. Next Inspect an element on the page. There are two ways to do this:
    a. Right click on any element in the page and select inspect.
    b. In the top left corner of the F-12 window is a small arrow. Clicking this allows you to select something currently in the page which might not otherwise accept a right click.
63. Demonstrate for the students how you can edit the CSS on an element from the F-12 window, effectively allowing you to check potential changes without the need to rebuild the full web page.

That completes what was put together for this 2 hour visit.