# REVIEW DRILLS

There is no better way to learn to code than to "just do it". Here are some practice tasks to verify your own understanding and to keep it fresh over the coming week.

## HTML & CSS

1) Create a skeleton HTML page from scratch
2) Add 3 paragraphs to your page with some text in each. Make the 1st and 3rd paragraphs have italicized text. Prefer to use a CSS pseudo class to do this.
3) Add a div in the 2nd paragraph which should contain 2 new spans. Style the 2nd contained span in this new div to have a black background and a white text color. Make the 1st span in this div have a text size of 20pt. Test yourself by adding another div with another 2 spans and verify that the spans in the 2nd div do *not* have the styles intended for the 1st div applied.
4) Make the cursor switch to a pointer when hovering over the 2nd div added above. Also, change the background color to red and the text color to white when the cursor is over the box.
5) Move your styles to a separate .css file if you don't already have them in one and link it in the head of your html page.

## JAVASCRIPT WITH JQUERY

6) Add a script tag to reference jquery in your page
7) Add a script tag to reference jgrowl(https://github.com/stanlemon/jGrowl) or toastr (https://github.com/CodeSeven/toastr) to your page
8) Add a form that consists of a textbox and a button. When the button is clicked show a toast/growl with the text in the textbox.
9) Add a SELECT (dropdown) html element to your form that provides 3 visual options (error, warning, success). Now when the button is clicked show the toast/growl differently according to the selection in the dropdown.
10) Add another dropdown with a label of Page Background Color and an accompanying button that is labeled "Set". When the page loads use a javascript array of strings to add elements to the dropdown list. When the Set button is clicked apply the selected background color to the page.
11) Add an html table to the page that includes tbody, th, tbody and td tags. The table will have 3 columns. API Name, Vote Count and an empty header column. Each of the rows will have a button labeled "Delete" in this 3rd column. On page render add rows into the table body using the JSON returned from an ajax GET to this URL http://voteapi.empower2018.us/tally. (Open that url in a browser to inspect the JSON structure returned so that you can write code to process the structure into html table rows.)
12) When a delete button is clicked within a row delete the entire row of the table in which the button exists. (HINT: use closest()).

## BOOTSTRAP

13) In the HEAD of your page add references to the bootstrap CSS and bootstrap JS files.
14) Improve the layout of your page using divs that implement bootstrap's grid layout (container, row and a div per needed column).
15) Apply bootstrap table classes to your table and add them in your JS that render the rows dynamically so that the styling of some of the rows are either green, yellow, red, blue or other with the "Contextual Classes" section under the tables documentation here (https://getbootstrap.com/docs/4.0/content/tables/).
16) Show a success alert upon deletion of a row (https://getbootstrap.com/docs/4.0/components/alerts/)

17) Use a bootstrap modal (https://getbootstrap.com/docs/4.0/components/modal/) to collect an api name in a textbox. If the user clicks a cancel button do nothing but if the user clicks the OK or Save button add another row to the table using the text they provided as the API name value and a 0 as the count. (Do not copy/paste what you did before, refactor it to reuse the code you wrote earlier into a function that can be called from both use cases.)

## JQUERY UI

18) In the HEAD of your page add references to the jquery ui CSS and JS files (https://jqueryui.com/themeroller/). Unlike bootstrap jquery UI provides 25 different skins on their components. Use that link to examine the different look and feel options and choose the one that you like best before downloading. The download will contain CSS that will style your jquery ui components to your preference.
19) Replace the bootstrap modal with a jquery ui dialog component to serve the same purpose and to see the difference between the two.
20) Implement tabs. Place all of the html UI you've already got into the first tab and create at least a 2nd tab.
21) Add a datepicker to the 2nd tab within the first/top segment of an accordion. Just put a paragraph tag and some text into a 2nd and 3rd accordion segment.

## C#

22) Open Visual Studio and create a new project of type Console App naming it CPractice. Add a 2nd project of type Class Library, name it OOPPractice.
23) Add a reference to your OOPPractice project from your CPractice console app. All of the classes you define below should be created within this OOPPractice class library project, not the console app.
24) Create an abstract base class named Animal that consists of
    a. a name property (string)
    b. a gender property (TypeOfGender enum that you need to define)
    c. an overridable "public relations" property (a QualityOfPublicRelations enum that you need to define with 4 values: horrible, poor, decent, excellent)
    d. an abstract method named Speak that returns a string
    e. and a concrete, overriddable method named Move which returns an integer indicating how fast. It should have a default implementation that returns a 5 (out of 10). If you want to make it even nicer create an enum with mappings to the integers 1 – 10 that describes each speed level and use it instead of the int value.
25) Add a child class to Animal that is also abstract named Rodent.
26) Add a concrete (normal, non-abstract) child class to Rodent named Squirrel that has excellent public relations; implement any other required properties or methods.
27) Add a concrete child class to Rodent named Rat that has horrible public relations; implement any other required properties or methods.
28) Add a concrete child class to Rodent named Chipmunk that has decent public relations; implement any other required properties or methods.
29) Add an abstract class named Bird that is child to Animal
30) Add a concrete class named Sparrow that is child to Bird that has decent public relations; implement any other required properties or methods.
31) Add a Velociraptor concrete class that is child to Animal. Add a method named hunt that takes a parameter of type Animal and returns a bool indicating if the hunt was successful or not. Use an instance of Random to decide the fate of the hunt.
32) In your console app create a list of Animal and select 3 random animals from the list that will be hunted by an instance of velociraptor. Write the names of the animals who successfully escaped and those who were not so lucky to the console.
33) Add an interface named ISoarable which consists of a Fly method. The method takes no parameters but returns an int indicating the distance covered. Squirrel and Sparrow must now implement this interface.

34) Filter your list of animals for any squirrels or sparrows and put them into a 2nd list of type ISoarable named fliers using LINQ. Iterate through the fliers list and write the name of the animal, the animal type and distance covered to the console for each member of the list. (HINT: you can inspect the type of an object in C# by saying "variablename is TypeName".)

35) Now, make this a simple database application. Add a 3rd project named DBPractice. Add a single class to that project named DatabaseServices.

36) Create a database using your relational database platform of choice that has a single table with the following 2 tables:

    a. Animal, which has the following columns
        i. TypeName
        ii. Name
        iii. Gender
    b. Performance, which has the following columns
        i. AnimalId
        ii. Escapes (a number)
        iii. Captures (a number)

37) Now add a list of animals tot your animal table and convert your console application to load your list of animals from your database table. Do this by adding a GetAnimals method to your DatabaseServices class that returns a list of animal instances that you load from your database table. Store your connectionstring in your console app's config file (if you don't have a config file, you can add one using Add New Item and choosing Application Configuration).

38) Store the outcomes of the velociraptor's hunt to the performance table by adding a 2nd method to your DatabaseServices class that saves a record to the Performance table.