HTML5 & CSS-S1- Practice Exercise

Note: Every Problem Starts in a new page

Problem Statement 1: Creating Blog Using HTML5 Semantic Markup and CSS.

Use HTML5 Semantic markup tags such as <header>, <nav>, <section>, <aside>, <article>, <ul>, <a>, <footer>, and Appropriate CSS

Note:

* Write a HTML code in index.html
* Create external CSS file styles.css to style the blog using CSS and link to index.html



Problem Statement 2: Showing Progress toward a goal with the <meter> Element.

1. Create a new HTML5 document html5\_meter/index.html to represent our meter to demonstrate and how it works with below code, hard-coding 2500.00 as our current value to demonstrate and how it works.



Problem Statement 3: Defining an FAQ with a Description List

Develop a Web page to display Description List as below.

1. Create a new HTML5 Document html5\_descriptionlist\_faq/index.html to use the <dl> tag to define the FAQ itself, <dt> tag for each question and <dd> tag for answer to each question as below.



Problem Statement 4: Create User-Friendly Web Forms.

Create a form contains fields as name, a contact, email address, and staging URL, Project start date, priority, and estimated number of hours that project should calculate, and like to give a color to identify the project.

1. Create a new HTML5 page html\_forms/index.html with basic HTML form that does a POST request with the assumption that at some point there’ll be a page that processes this form as below.
2. Create a slider using range to accept the priority as shown below
3. Create a number field to accept the estimated hours as shown below.
4. Create a date field to record the date of the project as shown below.
5. Create an email field to accept the email as shown below.
6. Create an URL field to accept project’s staging URL as shown below.
7. Create color field to accept project’s color code as shown below.



Problem Statement 5: Styling the Web Form

Style the Web Form created in Problem Statement 4 using CSS

1. Create a new file called stylesheets/style.css and link it in the <head> section of the form’s page in the index.html as below.
2. Remove the numbering, margins, and padding from the list using below CSS code.
3. Align the labels and the input fields and add a little styling to the input fields.



Problem Statement 6: Jumping to the First Field with Autofocus

Create a HTML Page to demonstrate Jumping to the First Field with Autofocus as shown below.

1. Create a new file called html\_forms/autofocus/index.html to demonstrate jumping to the First Field with autofocus.



Problem Statement 7: Providing Hints with Placeholder Text

Placeholder text to give an little guidance on our password requirements and add consistency.

1. Create html5\_placeholder/index.html for sign up using below source code.



Problem Statement 8: Validating User input without Javascript

Validation need to be done for all the inputs in the forms such as Name is required. And Valid Email Address such that.

1. Create an HTML5 form html\_validation/index.html where user fills in a form field by adding the required attribute to the element just like we do with the attribute. So, we can add the required attribute to the First Name, Last Name, and Email fields on the original signup form.



Problem Statement 9: Validation with Regular Expressions:

Pattern attribute lets us specify a regular expression against which we can validate the user data. The browser knows how to validate the password as below condition using patten attribute

* Password must be 8 or more characters.
* It should contain at least one number, an uppercase letter and one special character.
* Password and Confirm password must be same.

1. Create an HTML5 form html\_validation/index.html to validate the password using pattern attribute and regular expression as shown below.
2. Use the same pattern for Password Confirmation field as below.



Problem Statement 10: Styling the Fields:

:Valid and :InValid is used to style the validation in the Form Field.

1. Create a CSS html\_validation/stylesheets/style.css to provide instant feedback to users using pseudo classes :valid and :invalid and link it to the index.html created in Pronblem Statement 9.



Problem Statement 11: Styling Tables with Pseudo classes

Produce an Standard HTML Table.

1. Create css3\_advanced\_selectors/index.html to create a table as shown below.
2. Create a new file called stylesheets/style.css and link it up as shown below.
3. Create a new file called stylesheets/style.css as shown below.
4. Apply that style and the table looks like this.



Problem Statement 12: Striping Rows with :nth-of-type

1. Update a file stylesheets/styles.css created win Problem Statement 12 with below code



Problem Statement 13: Bolding the Last Row with :last-child

The Bottom row of the table need to be bolder than other rows.

1. Update a file stylesheets/style.css created with Problem Statement 13 with below code.

Problem Statement 14: Making Links Printable with :after content

:after is used to add the after content in the data for the particular tag.

1. Create css3\_print\_links/index.html with links as shown below.
2. Create css3\_print\_links/stylesheets/print.css with below code.
3. Link print.css in index.html as shown below.



Problem Statement 15: Building Mobile Interfaces with Media Queries

The blog need to be good in Smart phones.

1. Create css3\_mediaquery/stylesheets/style.css with below code and link it to the index.html created in Problem Statement 1 for blog.
2. Create stylesheets/mobile.css with above code and link in html as below.
3. With this our blog immediately becomes more readable on tiny screens, although the viewport is still zoomed way out. We can fix that by adding this tag right below the web page’s <title> tag.

