WDD 231

Home W1 W2 W3 W4 W5 W6 W7

W01 Assignment: Course Home Page

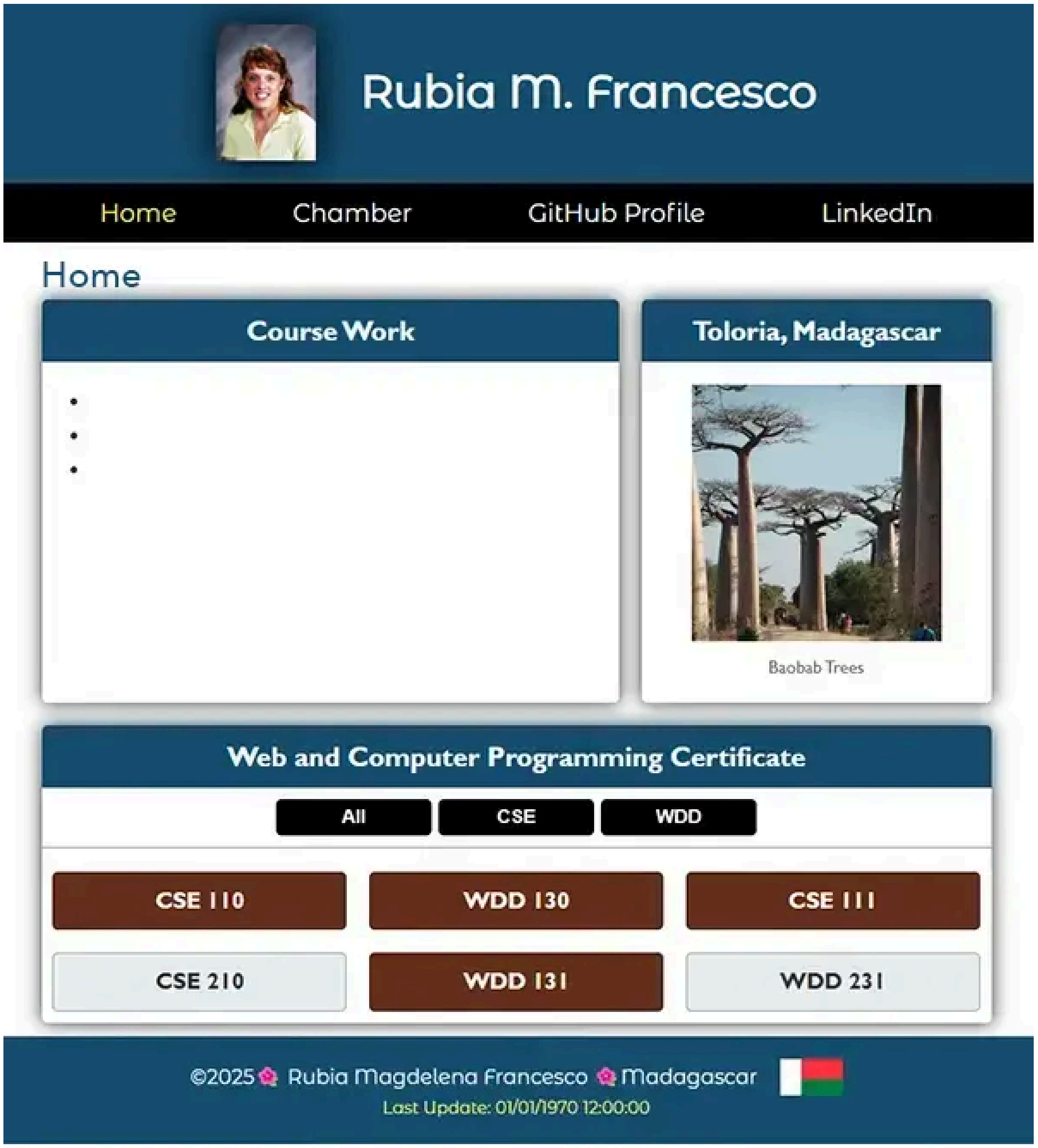
Overview

This assignment demonstrates your prerequisite knowledge by applying HTML, CSS, and JavaScript to the design and development a course home page.

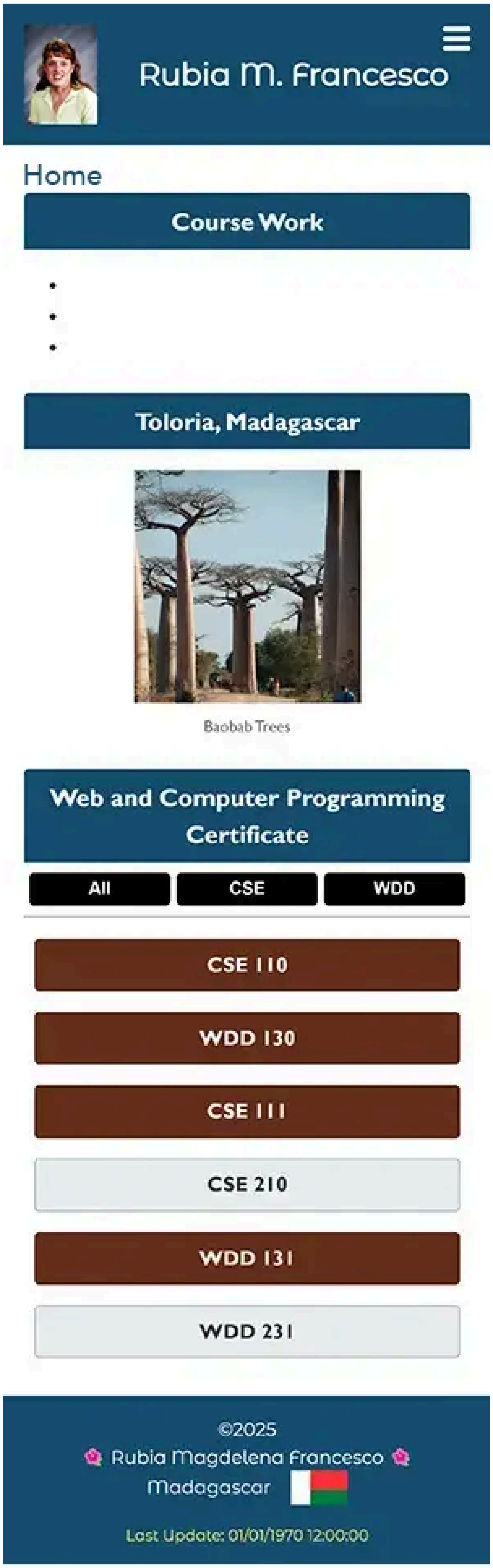
Task

Design and develop the a course home page using the following wireframe as a guide for the layout and content. The page must be **responsive** and pass **accessibility**, **best practice**, and **SEO** tests from DevTools Lighthouse audits.

▼ Screenshots



Example Home Page – Desktop



Rubia M. Francesco

Home

Course Work

-
-
-

Toloria, Madagascar

Baobab Trees

Web and Computer Programming Certificate

CSE 110

WDD 130

CSE 111

CSE 210

WDD 131

WDD 231

©2025
Rubia Magdelena Francesco
Madagascar

Last Update: 01/01/1970 12:00:00

Example Home Page – Mobile

Instructions

Step 1: File and Folder Setup

1. In VS Code, open your `wdd231` folder that you cloned from your GitHub account during the Setup: Hosting on GitHub activity.
2. Create a default home page for this repository so that when published to GitHub, will render at `https://yourgithubusername.github.io/wdd231`. What must this file be named?

▼ Check Your Understanding

`index.html`

3. Create appropriately named, supporting folders for images, CSS, and JavaScript files. These sub-folders must be placed in the root directory, `wdd231`.

▼ Check Your Understanding

- Images will be stored in a folder named `images`
- CSS files will be stored in a folder named `styles`
- JavaScript files will be stored in a folder named `scripts`

Reference: Naming Conventions learning activity.

Note that the words *directory* and *folder* have essentially the same meaning. Directory is the more accurate term for file systems while "folder" 📁 refers to the graphical metaphor that is generally accepted because it is highly related to the term "file" in the organized world.

4. Add CSS files as needed.

▼ Check Your Understanding

Students are required to write their own custom CSS in this course. Frameworks are **not** allowed at this point.

The assignment requires you to create a responsive layout for all screen widths from (320px) and up. For this class, use a normalize.css followed by a small.css followed by a larger.css file. The larger css file will contain a media query.

5. Add multiple JavaScript files as needed. For example, create files to support the responsive navigation (navigation.js), dates (date.js), and course information cards (course.js). Reference these in the `head` using the `defer` attribute method. As you move through this course, you will need to add additional JS files. Only load the files for each page that are needed for that page!

Step 2: HTML

1. Add the standard **HTML document structure** to the document.
2. Add the required elements to the `<head>` including the `title`, `meta description`, and the `meta author`.
3. Use **semantic HTML** to create the basic layout structure of the page. Consider using a `header`, a `nav`, a `main`, and a `footer` element.
4. Add the **content** as shown in the example screenshots above.
5. Note that all images used on your pages must be optimized.

In this class, the image memory threshold is **125 kB** or less per image.

6. The `<header>` tag should contain a small photo and your name rendered using a `<span>` tag.
7. The responsive navigation `<nav>` menu contains four links:

   ✔ **Home** – this current page.

   ✔ **Chamber** – will link to the Chamber of Commerce website project (future assignment).

   ✔ **GitHub Profile** – will link to your GitHub Profile page (future assignment).

   ✔ **LinkedIn** – links to your LinkedIn profile page.

   If you do not have a free LinkedIn account profile, consider making one or just provide a general link to LinkedIn.

8. The `<main>` element contains the following:

   ✔ Home rendered using an `<h1>` tag. MDN Reference

- ✔ Three `<section>` tags with headings `<h2>` as shown in the example screenshots above.

9. The `<footer>` has two paragraphs `<p>`.
   - ✔ The first paragraph contains the following:
     - the copyright symbol and current year that are written dynamically using JavaScript

       ▼ Check Your Understanding

       In HTML, provide a placeholder that can easily be selected using JavaScript.

       Example: `<span id="currentyear"></span>`

     - your name
     - your state or country
   - ✔ The second paragraph has an id of "**lastModified**" and will be populated with JavaScript code.

Step 3: CSS

1. Use your own color schema and typography choices.

   You are responsible to practice good design principles including alignment, color contrast, proximity, use of white space, repetition, and typography.

2. Use the Google Fonts API to select one or two fonts to use on the page.

   Video Demonstration: ▶ Using Google Fonts

3. Responsive navigation must, at least, support the following features:
   - ✔ small-screen hamburger like navigation
   - ✔ large-screen horizontal navigation using CSS Flex

     Video Demonstration: ▶ CSS Flex Navigation Menu
     Source code: CSS Flex Menu – CodePen

   - ✔ wayfinding

Wayfinding supports user friendly navigation by providing a clear indication of the user's current location within a website.

Here is an example of designing **wayfinding** into a navigation menu: Responsive Menu.

✔ hover effect for menu items

Responsive design includes providing a positive user experience of readability, usability, accessibility, visual appeal that adapts to the user's device by size, orientation, and resolution.

4. Your design cannot just be a simple one column layout (default flow layout) in all views in order to warrant full credit consideration.

Step 4: JavaScript

1. Reference JavaScript files by using a **`<script>`** reference in the **`head`** of the HTML file and using the attribute **`defer`**.

Why is the defer boolean attribute important?

2. Use JavaScript to support the responsive menu.
3. Use JavaScript to dynamically output the following:

✔ the copyright year (the current year) in the footer's first paragraph

Note this CodePen summary of the **`Date`** object.

✔ the date the document was last modified in the second paragraph

You can use the lastModified property of the document object to get this date/time dynamically.

Note that the **`document.lastModified`** returns a simple string. Therefore, you do not need to manipulate its output.

4. Copy this array of course objects into a JavaScript file: Course List Array

This array contains the course information for the required courses that are in the first certificate called Web and Computer Programming of the Software Development degree.

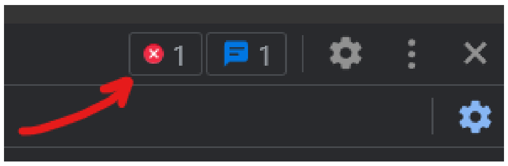
5. Modify the **courses** array content in your script file by changing the `completed` property to `true` if you have completed a course.
6. Dynamically display all the courses in the certificate section as shown in the example above. The courses that you have completed must be marked in a different way versus those that you have not completed. Use your page color scheme. The page should adjust automatically if the data source changes.
7. Using buttons, allow the user to select to display all courses, WDD courses, or CSE courses. Use the array (`filter`) method.

▶ Course Filter Buttons Demonstration

8. Design the course cards to indicate those courses that you have **completed** in a complimentary, but different style than the rest.
9. Provide a total number of **credits** required dynamically by using a `reduce` function (not shown on the screenshots). The number of credits shown should reflect just the courses currently being displayed.

Step 5: Testing

1. Continuously check your work by having it loaded in your browser using Local Server.
2. Use the browser's DevTools to check for JavaScript runtime errors in the console or click the red, error icon in the upper right corner of DevTools.



"DevTools" is an abbreviation for the browser's "Developer Tools." It refers to a set of tools or utilities provided by web browsers to help developers debug, profile, and analyze web pages during the development process. The tools are typically accessed by pressing the F12 function key or selecting the menu option for the browser's developer tools.

3. Use DevTools **CSS Overview** to check your color contrast.
4. Generate the DevTools **Lighthouse** report and run diagnostics for **Accessibility**, **Best Practices**, and **SEO** in both the mobile and Desktop views.

Use the **Private** or **Incognito** mode to test your page when using Lighthouse. The standard is to reach a score of 95+ in each of these categories.

Submission and Audit

1. Commit and sync your local work to your a GitHub Pages enabled **wdd231** repository
2. Use this ✓ audit tool to self-check your work for some of the required HTML elements and CSS content. This audit tool is also used by the assessment team.
3. Return to Canvas and submit your GitHub Pages enabled URL for wdd231, e.g.,

```
https://your-github-username.github.io/wdd231
```

Do not need to include **index.html** in the URL reference because index.html is the default file retrieved with a folder request on GitHub Pages.