Lab 9: Sticky Navigation and GitHub Pages

Overview

In this lab, you will enhance a simple website by implementing a sticky navigation menu and then learn how to deploy your site to GitHub Pages. The lab is divided into two main parts:

1. **Part 1: Sticky Navigation** - You'll implement a modern, responsive sticky navigation bar that remains at the top of the page while scrolling
2. **Part 2: GitHub Pages Deployment** - You'll learn how to create a GitHub account, set up a repository, and publish your website using GitHub Pages

Getting Started

1. Download these starter files from Canvas and save them to your lab9 folder:
    - `index.html`: Basic HTML structure with content sections
    - `citations.html`: A page for image and video source citations
    - `citations-styles.css`: CSS file with basic styling
2. Preview the starter code using Live Server:
    - If images are not displaying:
        - Verify the images folder is in the same directory as your HTML file
        - Check that image filenames match exactly with the HTML src attributes
        - Make sure all image files were downloaded successfully
3. Review the starter code:
    - Open `index.html` and `citations.html` in your code editor
    - Look for TODO comments indicating where you need to add code
    - Open `citations-styles.css` in your code editor
    - Find the sections marked with TODO comments
    - Note the existing styles that are already in place
4. Verify file structure:

```
lab9/
├── index.html
├── citations.html
├── citations-styles.css
└── images/
    ├── AbeautifulSunsetOverMountains.png
    └── uo_duck.png
```

Task 1: Create a Sticky Navigation Menu

Step 1: Add Navigation HTML

In both `index.html` and `citations.html`, you need to add a sticky navigation menu right after the opening <body> tag:

1. Add the following HTML structure right after the opening <body> tag in both HTML files:

```html
<!-- Sticky Navigation Menu -->
<nav class="navbar">
    <a href="index.html" class="logo">PROJECT</a>
    <div class="nav-links" id="navLinks">
        <a href="index.html">Home</a>
        <a href="#nature">Nature</a>
        <a href="#video">Video</a>
        <a href="citations.html">Citations</a>
    </div>
    <button class="mobile-menu-btn" onclick="toggleMenu()">☰</button>
</nav>
```

   Understanding the navigation links:

   - `href="index.html"`: This is a page-level link that navigates to the index.html file (the home page)
   - `href="#nature"`: This is a section-level link (note the # symbol) that scrolls to an element with `id="nature"` within the current page
   - `href="#video"`: This is a section-level link that scrolls to an element with `id="video"` within the current page
   - `href="citations.html"`: This is a page-level link that navigates to the citations.html file (a different page)

2. For `citations.html`, modify the logo and navigation links to be appropriate for that page:

   - Change the logo text to "CITATIONS"
   - Update the nav links to include sections specific to the citations page:
     - `#image-sources`
     - `#video-sources`

   Important distinction: On the citations.html page, links like `#image-sources` and `#video-sources` will scroll to sections within the citations.html page, while the `index.html` link will navigate back to the home page. This demonstrates the difference between section-level navigation (within the same page) and page-level navigation (between different HTML files).

3. Add `id` attributes to the corresponding sections in both HTML files:

   - In `index.html`, add `id="nature"` to the section with the nature image
   - In `index.html`, add `id="video"` to the section with the video
   - In `citations.html`, add `id="image-sources"` to the image sources section
   - In `citations.html`, add `id="video-sources"` to the video sources section

Step 2: Add Sticky Navigation CSS

In your CSS file, you need to add styling for the sticky navigation menu. Add the following CSS rules to your `citations-styles.css` file:

1. Style the main navbar container:

```
/* Sticky Menu Styles */
.navbar {
    position: sticky;
    top: 0;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem 2rem;
    background-color: rgba(255, 255, 255, 0.9);
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    z-index: 1000;
}
```

2. Style the logo:

```
.logo {
    font-size: 1.8rem;
    font-weight: bold;
    color: #007b00;
    text-decoration: none;
    letter-spacing: 1px;
}
```

3. Style the navigation links:

```
.nav-links {
    display: flex;
    gap: 2rem;
}

.nav-links a {
    color: #333;
    text-decoration: none;
    font-weight: 500;
    transition: color 0.3s ease;
}

.nav-links a:hover {
    color: #007b00;
}
```

4. Style the mobile menu button (hidden by default):

```css
.mobile-menu-btn {
    display: none;
    background: none;
    border: none;
    font-size: 1.5rem;
    cursor: pointer;
    color: #333;
}
```

5. Add responsive design for mobile devices:

```css
/* Responsive Design */
@media (max-width: 768px) {
    .nav-links {
        position: fixed;
        top: 70px;
        right: -100%;
        flex-direction: column;
        background: white;
        width: 70%;
        height: 100vh;
        padding: 2rem;
        box-shadow: -2px 0 5px rgba(0, 0, 0, 0.1);
        transition: right 0.3s ease;
        gap: 1rem;
    }

    .nav-links.active {
        right: 0;
    }

    .mobile-menu-btn {
        display: block;
    }
}
```

Note: We will implement the JavaScript functionality for the mobile menu in the lecture. For now, focus on creating the HTML structure and CSS styling for the sticky navigation bar. The mobile menu button will be visible on smaller screens but won't be functional yet.

Step 3: Test Your Sticky Navigation

1. Open your HTML files in a browser and test the following:
   - The navigation bar should "stick" to the top of the page as you scroll
   - Clicking on navigation links should scroll to the appropriate section
   - The navigation bar should be responsive on different screen sizes
   - When viewing on mobile (or resizing to a small window), the navigation should collapse to a menu button

Task 2: Prepare and Deploy to GitHub Pages

Step 1: Create a GitHub Account

1. Visit GitHub's website in your web browser
2. Click the "Sign up" button in the top-right corner of the page
3. Enter your email address in the provided field
4. Create a strong password and enter it in the password field
5. Choose a unique username that will identify you on GitHub
6. Complete the verification puzzle to prove you're not a robot
7. Click the "Create account" button
8. You'll receive a verification code via email - enter this code to verify your email address
9. Answer a few onboarding questions about your programming experience and intended use of GitHub
10. Your GitHub account is now created and ready to use!

Step 2: Setting Up a GitHub Pages Repository

1. **Create a new repository**:
   - Click the "+" icon in the top-right corner of GitHub and select "New repository"
   - Name your repository as: `yourusername.github.io` (replace "yourusername" with your actual GitHub username)
   - Make sure the repository is set to "Public"
   - Click "Create repository"
2. **Add content to your repository**:
   - Go to your GitHub repository
   - Click on the "Add file" button and select "Upload files"
   - Drag and drop or select all your lab files (HTML, CSS, images folder)
   - Add a commit message like "Initial commit of my website"
   - Click "Commit changes"
3. **Enable GitHub Pages**:
   - Go to your repository's "Settings" tab
   - Scroll down to the "Pages" section
   - Under "Source", select "main" branch if not selected
   - Click "Save"
4. **Access your published site**:
   - GitHub will display a message saying "Your site is published at https://yourusername.github.io"
   - Click the link or enter it in your browser to visit your new GitHub Pages site
   - Note that it may take a few minutes for your site to be available online

Step 3: Create Folders in Your Repository

If you need to organize your files into folders:

1. Go to your GitHub repository
2. Click on the “Add file” button and select “Create new file”
3. In the “Name your file...” field, type your folder name followed by /, e.g., “images/”
4. GitHub will recognize it as a new folder
5. Enter a file name (GitHub does not allow empty folders)
6. Add some content to the file, then click “Commit new file”

Step 4: Test Your Deployed Website

1. Visit your GitHub Pages URL: `https://yourusername.github.io`
2. Your HTML file MUST BE named index.html
3. Test all navigation links and features to ensure everything works properly
4. Check that all images are displaying correctly
5. Test responsive design by resizing your browser window

Troubleshooting

- If your site isn’t publishing, check that your repository is named correctly: `yourusername.github.io`
- Ensure your main HTML file is called `index.html` and is in the root directory
- Check the GitHub Pages section in your repository settings to verify it’s enabled
- Allow a few minutes for changes to propagate after committing new files
- If images aren’t displaying, check that the file paths are correct relative to your repository structure

Validation Requirements

HTML Validation

1. Use the W3C Markup Validation Service (https://validator.w3.org):
   - Go to “Validate by File Upload”
   - Upload your HTML files
   - Fix any validation errors or warnings

CSS Validation

1. Use the W3C CSS Validation Service (https://jigsaw.w3.org/css-validator/):
   - Go to “File Upload” tab
   - Upload your CSS files
   - Fix any validation errors or warnings

Submission Requirements

Submit only the URL of your deployed GitHub Pages site (example: https://yourusername.github.io).

Your submission should:

- Link to a fully functional website
- Include all required files in the GitHub repository:
- Completed `index.html` - Completed `citations.html`
- Completed `citations-styles.css`
- Images folder with all required images
- Have a sticky navigation menu working correctly
- Be mobile-responsive
- Pass HTML and CSS validation

Grading Criteria

- Sticky Navigation Implementation (40%)
- GitHub Pages Deployment (40%)
- Code Organization and Comments (20%)

Notes

- The `position: sticky` property is the key to creating a sticky navigation bar
- Responsive design is crucial for a professional website experience
- GitHub Pages is a free hosting service that's perfect for static websites
- The repository name must follow the pattern `username.github.io` for the main user site

Challenge

For extra practice (not graded):

- Add smooth scrolling behavior when clicking on navigation links
- Enhance the mobile navigation with animations
- Add a "back to top" button that appears when scrolling down
- Customize your navigation with additional styling or dropdown menus