# Navigation Bar

The navigation bar allows users of your website to find their way around. It gives quick access to important links for getting through your site that consistently appears at the top of the webpage.

In order to keep your navigation bar at the top of every page on your site, write the code for it inside of `app > views > layouts > application.html.erb.` Insert the code within the `<body>` element, before `<%= yield %>`

## Implementation

Figure 1 shows the code to produce a basic HTML navigation bar. *This bar will later be styled using bootstrap elements.*

```
<nav>
  <a href="/">
    <img src="<%= asset_path('MSU_Denver.png') %>" alt="MSU Denver Logo" width="60" height="60">
    Student Portfolios
  </a>
  <button type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
    <span></span>
  </button>
  <div id="navbarNav">
    <ul>
      <li>
        <%= link_to 'Home', students_path %>
      </li>
      <!-- Add other nav items here if needed -->
    </ul>
  </div>
</nav>
```

*(Figure 1)*

### Implementation Breakdown

`<nav>` is an HTML element for navigation on the web. It makes it easier for browsers, screen readers, and SEO link importance. It will, of course, contain all the stuff for your navigation bar.

`<a href="/">` is a wrapper that will make the logo a clickable link. The `"/"` specifies that it links to the the website's root, which is usually the homepage.

`<img src=“”…>` is the element that displays your logo. I recommend putting your image file into `app > assets > images` or making that folder if you don’t have one. I thinks thats how the `asset_path` rails method works… IDK haha!

`<button…>` is another HTML element for, you guessed it: buttons! The `<span>` field will be needed later when bootstrapping.

The `<div id=“navbarNav”>` will contain your links around the website. `<ul>` represents an unordered list container, which will then contain `<li>` list items. Adding new list items will allow you to add more links into the nav bar.

# Bootstrap That Thang

The basic HTML navigation bar is pretty unattractive, but bootstrap can help you fix that and make your website more flexible for all screen sizes.

**Attention:** Make sure bootstrap is installed in your project! If you haven’t already, you should follow Professor Harding’s guide to *Setting Up Bootstrap With Sprockets* found in the *CS3170 Shared Student Materials* drive under *Rails Set Up*.

Figure 2 gives you look at a basic setup to make this navigation bar more dynamic and appealing to your users.

```
<nav class="navbar navbar-expand-lg">
  <div class="container-fluid">
    <a class="navbar-brand" href="/">
      <img src="<%= asset_path('MSU_Denver.png') %>" alt="MSU Denver Logo" width="60" height="60"
      class="d-inline-block align-center">
      Student Portfolios
    </a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item">
          <%= link_to 'Home', students_path, class: 'nav-link' %>
        </li>
        <!-- Add other nav items here if needed -->
      </ul>
    </div>
  </div>
</nav>
```

*(Figure 2)*

**Bootstrap Breakdown**

Bootstrap is pretty much based on adding `class` attributes to everything; check out the differences between Figure 1 and Figure 2.

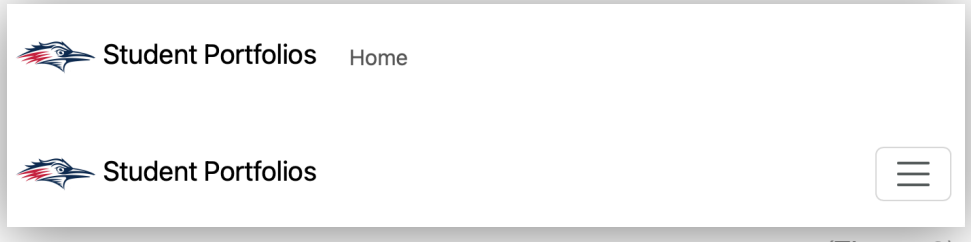
In the `<nav>` element, there is now a `class="navbar navbar-expand-lg"` tag inside the opener. This is a built-in bootstrap component that will make your navigation bar dynamically expand depending on screen size. You can choose the screen size in which the navigation bar will expand/collapse by choosing `-sm`, `-md`, `-lg`, or `-xl` on the end.

Inside of of that. You'll find a new `<div class="container-fluid">` element, encapsulating the rest of our code. This is another bootstrap component that allows the nav bar to stretch across the whole screen. A normal container would have margins on both sides.

Our `<a href="/">` element now contains a `class="navbar-brand"` modifier, which helps make sure the logo is displayed properly in a bootstrap navigation bar.

The `<img src…>` line also now contains a `class="d-inline-block mx-auto"` element. The `d-inline-block` part makes it a block inline with text. The `mx-auto` part centers it in its container.

`<button>` now contains a `class="navbar-toggler"` modifier, and inside of `<span>`, there is now a `class="navbar-toggler-icon"` component. These work together with `navbar-expand` to show the `navbar` icon on smaller windows or screens. Figure 3 displays two nav bars, the top had `"navbar-expand-sm"` instead of `-lg`.




*(Figure 3)*

The `class="collapse navbar-collapse"` inside of `<div>` helps the collapsable items collapse.

In the unordered list, when you combine `navbar-nav` with `nav-item`, you get a well-structured and styled navigation menu. The `navbar-nav` class handles the overall layout and styling, while nav-item specifies how each link should behave and be presented within that layout.

# ARIA Labeling

Adding ARIA labeling to your website will help boost accessibility on your website. Labeling essentially boils down to adding a series of `aria-label` attributes to things in your site. Figure 4 shows the same navigation bar with added ARIA elements.

```
<nav class="navbar navbar-expand-sm" aria-label="Main Navigation">
  <div class="container-fluid">
    <a class="navbar-brand" href="/" aria-label="Homepage">
      <img src="<%= asset_path('MSU_Denver.png') %>" alt="MSU Denver Logo" width="60" height="60"
       class="d-inline-block align-center">
      Student Portfolios
    </a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
     aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon" aria-hidden="true"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item">
          <%= link_to 'Home', students_path, class: 'nav-link', aria_label: "Go to Home" %>
        </li>
        <!-- Add other nav items here if needed -->
      </ul>
    </div>
  </div>
</nav>
```

**ARIA Breakdown**

`aria-label=“”` adds the tag that people using screen readers will use. You can see it being used to identify `“Main Navigation”`, `“Homepage”` etc. There are also labels for `aria-controls` as well as tags to help with expanding/collapsing menus.

Honestly, I just copied my code snippet into ChatGPT and asked for it to add ARIA labeling where it made sense because I found it a little overwhelming.