**Andrea Ramsey**
andrea.jos.ramsey@gmail.com

# Creating a Responsive Navigation Menu with CSS

---

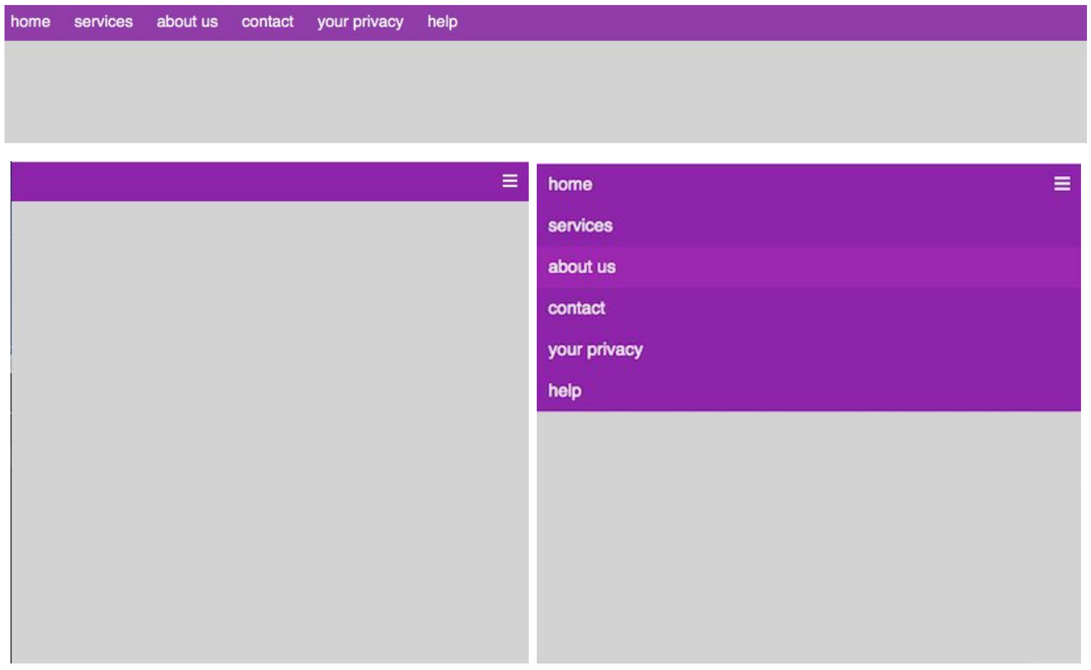
## Introduction

There are many ways to make a responsive  navigation menu for your website, often using a small amount of JavaScript. For this documentation we will create a responsive menu using pure HTML and CSS with no JavaScript. We will accomplish this using the “checkbox trick”

The checkbox trick (sometimes referred to as checkbox hack) uses a label and checkbox input to determine how elements are styled on a page based on whether or not the checkbox is “checked” or “unchecked”. Normally we would use an “on click” function in JavaScript, but this is a sneaky way of avoiding the use of JavaScript altogether.

After reading this documentation, the user should be able to create a basic responsive navigation bar. When on mobile, the navigation bar will include a hamburger-menu icon that toggles a dropdown menu with the page links.

We will not cover animations in this documentation but it is equally doable without the help of JavaScript.



## Prerequisites

It is recommended that the user knows some HTML and CSS

---

## 1. Setting Up

Start off by opening up your text editor and setting up your HTML and CSS environment, you will only need an HTML file and a CSS file for this. Please see the code below and follow the HTML  structure, notice we have named our css file “main.css” and have it linked in the HTML head. Please note we will be grabbing our icon from font-awesome, make sure to include the font-awesome link in your head tag if you want to use it as well.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">

 <head>
<!-- CSS link -->
  <link href="css/main.css" rel="stylesheet">

  <!-- Meta tag -->
  <meta charset="utf-8" name="viewport" content="width=device-width,
initial-scale=1.0">

<!-- Icon link -->
  <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.4.1/css/all.css"
integrity="sha384-5sAR7xN1Nv6T6+dT2mhtzEpVJvfS3NScPQTrOxhwjIuvcA67KV2R5Jz6kr4a
bQsz" crossorigin="anonymous">

  <title> CSS Only Nav Bar</title>
  </head>

  <body>
 </body>

</html>
```

# 2. HTML

## Adding the Basic Navigation Bar Structure:

- Create a nav tag and give it a class of “top-nav”
- Inside the nav tag put a div tag and give it the class “nav-links”
- Fill the “nav-links” div with the links you will need for your nav bar. For this documentation we are using: home, services, about us, contact, your privacy, and help

Your structure should look something like this in the end:

```
<body>
  <nav class="top-nav">
    <div class="nav-links">
      <a href="#home">home</a>
      <a href="#services">services</a>
      <a href="#about">about us</a>
      <a href="#contact">contact</a>
      <a href="#your-privacy">your privacy</a>
      <a href="#help">help</a>
    </div>
  </nav>
</body>
```

## Adding the Checkbox:

Next we are going to add the checkbox trick.

- Right inside the nav tag, above the div we will put a label tag and an input type tag
- Inside the label, grab a hamburger icon from Font-Awesome:

  `<i class="fa fa-bars"></i>` and put it inside

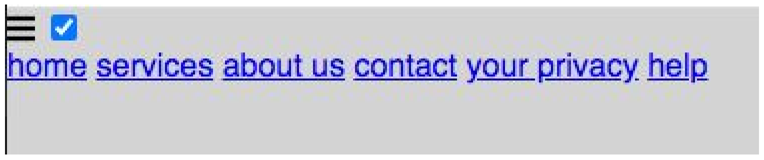
- The input tag will need to be of type “checkbox” with an id of “toggle”
- The label will need to be for “toggle” with a class of “icon”, please see the code below:

```
<label for="toggle" class="icon">
   <i class="fa fa-bars"></i>
</label>
<input type="checkbox" id="toggle">
```

### What exactly is going on here?

- Input elements can be paired with labels
- We are telling the browser that our icon label belongs to the toggle input by using the “for” attribute and assigning it to the “toggle” id
- When we click on the label (The hamburger icon in our case), it focuses/ toggles the checkbox input.

Here’s what we should have so far if you run the code in the browser:



It doesn’t look like much because we have not added the CSS yet. You’ll notice if you click on the hamburger menu, the checkbox will toggle ( we will eventually remove the checkbox!)

Next we will do the CSS, here we will style the elements. We will also be able to style different elements based on whether or not the checkbox is checked, this is how we will get our drop down menu without using JavaScript.

## 3. CSS

### Styling the Navigation Bar

That’s it for HTML! Head over to the main.css file. Below is the code for the basic (non responsive) navigation bar styles

- We’ve applied a transition to all elements, you may want to change this if you plan on adding animations to your project
- We’ve made the icon display-none so it is not viewable on desktop
- The rest are basic custom styles, please customize it how you would like or copy this code

```
body {
  margin: 0;
  font-family: Helvetica, sans-serif;
  background: lightgray;
}

* {
   transition: all 0.3s;
}

/* ************ Nav bar styles *********** */
```

```
.top-nav {
  overflow: hidden;
  background-color: #8e24aa;
  z-index:2;
}

.top-nav a, label{
  float: left;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 10px 12px;
  text-decoration: none;
  font-size: 16px;
}

 a:hover {
  background-color: #9c27b0;

}

/* hide menu on desktop */
.top-nav .icon {
  display: none;
}

.nav-icon {
  margin: 1em;
  width: 40px;
}
```

### Making the Navigation Bar Responsive:

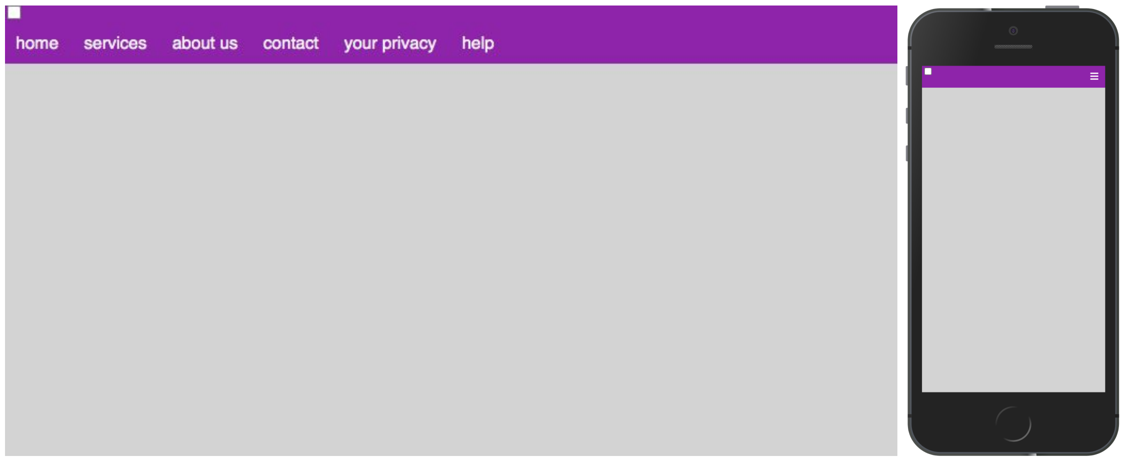
Next we will add the media queries. Media queries allow us to apply different styles to different devices based on the screen size

Make sure to place the media queries at the bottom of your CSS document.

Below is the code for the responsive styles:

- Here we determine the size of the screen that these styles are applied to with "max-width"
- When we are looking at a mobile device, the links in the top nav will disappear, the icon in the navigation will appear on the right side of the screen

```
/* ************ Responsive Styles *********** */

@media screen and (max-width: 600px) {

  .top-nav a{display: none;}

  .top-nav label.icon {
    float: right;
    display: block;
    color: white;
  }

  .top-nav{position: relative;}

}
```



That looks better, except the check box is still there and if we click the menu all it does is toggle the checkbox.
Next we will add the checkbox styles and use the “checked” keyword to make it all work.

### Checkbox Styles:

Now we only need to add a couple more lines to make this all come together

- First, in the navigation bar styles, add this code:

```
input[type=checkbox] {
display:none;
}
```

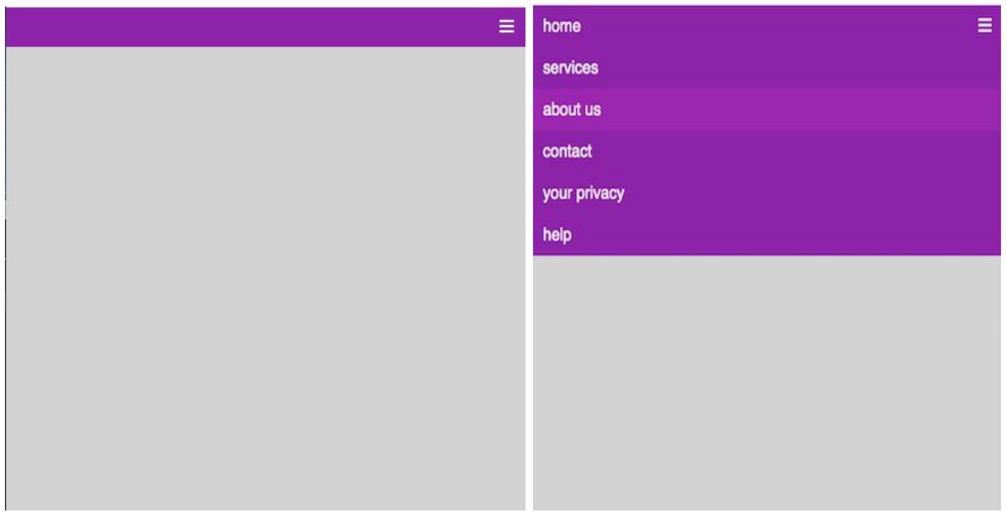
This will make the checkbox disappear

- Next in the in the Responsive styles/ media queries, add these lines:

```
#toggle:checked + .nav-links a {
    float: none;
    display: block;
    text-align: left;
  }
```

When our toggle input is checked, it will style the links as shown

Now when the checkbox is “checked” (or when we click the hamburger menu) it will bring our page links back, when we click it again, they will disappear. We’ve added the styles: float: none, display:block so the links appear to stack vertically instead of a horizontal line, providing an easier user experience on a mobile device.



## Conclusion

Here we have provided a fairly simple and effective way to make a navigation bar using pure CSS using the checkbox trick.

There are many reasons that this method may work for you:

- whether you want an extra challenge by not using JavaScript’s click functions
- whether you want to explore what CAN be done in CSS and/or to gain extra CSS practice
- or whether you are just not wanting/ready to use JavaScript

Is this method right for you? Check out the pros and cons below to see if this works for you and your project

## Pros

- No JavaScript needed
- Easy to implement
- Not much extra CSS and HTML

## Cons

- Does not work for some Android browsers(Android <4.1.2), and Mobile Safari browsers (IOS < 6.0)
- It's generally seen as best practice to use JavaScript for functionality, using the input/label trick is a fun and smart but the use of JavaScript is generally preferred
- Could potentially have a lot of extra CSS and HTML and not the best separation of concerns if more styles and animations were to be added