**University of Wolverhampton**

**School of Engineering, Computational and Mathematical Sciences**

**5CS020 Human-Computer Interaction**

**Workshop 8**

In this workshop you will be creating a Responsive Web app. Creating a responsive web app that caters to all common sizes of displays from small mobile phones to typical laptop and desktop computer screens to very large screens is an essential aspect of modern web development. In this comprehensive tutorial, we will explore how to create a responsive web app using HTML, CSS, and JavaScript.

**Understanding Responsive Web Design**

Responsive web design is a design approach that aims to create web apps that adapt to various screen sizes and devices. Responsive web apps ensure that users can easily access and navigate the app regardless of the device they are using.

To create a responsive web app, we need to use a combination of HTML, CSS, and sometimes JavaScript, to ensure that our web app responds to different screen sizes.

**Step 1: Create the HTML Markup**

The first step is to create the basic HTML markup for the navigation bar. We'll use the nav element to wrap the entire navigation bar, and we'll include a logo on the left and a horizontal list of links on the right. Here's an example:

**<nav>**

**<div class="logo">**

**<a href="#">Logo</a>**

**</div>**

**<ul>**

**<li><a href="#">Home</a></li>**

**<li><a href="#">About</a></li>**

**<li><a href="#">Services</a></li>**

**<li><a href="#">Contact</a></li>**

**</ul>**

**</nav>**

In this example, the logo is wrapped in a div element with a class of logo, and the links are contained within a ul element with individual li elements for each link. Note that we've included placeholder links with "#" as the href value - you'll want to replace these with the actual links to your website pages.

**Step 2: Style the Navigation Bar**

Next, we'll use CSS to style the navigation bar. We'll start with some basic styles for the nav element, and then add styles for the logo and links. Here's an example:

**nav {**

**display: flex;**

**justify-content: space-between;**

**align-items: center;**

**background-color: #333;**

**color: #fff;**

**padding: 10px;**

**}**

**nav ul {**

**margin: 0;**

**padding: 0;**

**list-style: none;**

**display: flex;**

**flex-direction: row;**

**}**

**nav ul li {**

**margin: 0 10px;**

**}**

**nav ul li a {**

**color: #fff;**

**text-decoration: none;**

**font-weight: bold;**

**font-size: 16px;**

**padding: 10px;**

**border-radius: 5px;**

**transition: background-color 0.3s ease-in-out;**

**}**

**nav ul li a:hover {**

**background-color: #666;**

**}**

In this example, we're using Flexbox to horizontally align the logo and links within the nav element. We've set the background colour to black (#333) and the text colour to white (#fff), and added some padding to provide some spacing.

For the ul element, we've removed the default margin and padding, and set it to display as a horizontal row using Flexbox. We've added some margin to the li elements to provide some spacing between them.

For the link elements, we've set the text colour to white, added some padding and border radius to create clickable buttons, and added a hover effect to change the background color when the mouse is over the link.

**Step 3: Add a Media Query for Small Screens**

Now we'll add a media query to adjust the layout of the navigation bar for small screens. We'll switch the layout from horizontal to vertical, and hide the links by default, adding a responsive menu button to toggle the display of the links. Here's an example:

**@media screen and (max-width: 768px) {**

**nav {**

**flex-direction: column;**

**align-items: stretch;**

**}**

**nav ul {**

**flex-direction: column;**

**display: none;**

**}**

**nav ul.show {**

**display: flex;**

**}**

**nav ul li {**

**margin: 10px 0;**

**}**

**.menu-button {**

**display: block;**

**background-color: transparent;**

**border: none;**

**color: #fff;**

**font-size: 16px;**

**padding: 10px;**

**cursor: pointer;**

**margin-top: 10px;**

**}**

**}**

In this example, we're using a media query to target screens with a maximum width of 768 pixels. Within the media query, we're setting the `flex-direction` property of the `nav` element to `column` to switch the layout from horizontal to vertical.

We're also setting the `display` property of the `ul` element to `none` to hide the links by default. We've added a new class of `show` to the `ul` element, which we'll use to toggle the display of the links when the menu button is clicked.

We've adjusted the margin of the `li` elements to provide some spacing between them in the vertical layout.

Finally, we've added a new class of `menu-button` to the button element that we'll use to toggle the display of the links. We've set the `display` property to `block` to make the button a block-level element, and added some basic styles to make it look like a menu button (three horizontal lines).

## Step 4: Add JavaScript for the Responsive Menu Button

Finally, we'll add some JavaScript to toggle the display of the links when the menu button is clicked. Here's an example:

**<script>**

**document.querySelector(".menu-button").addEventListener("click", function() {**

**document.querySelector("nav ul").classList.toggle("show");**

**});**

**</script>**

In this example, we're using the querySelector method to select the menu button element, and adding an event listener for the click event. When the menu button is clicked, we're using querySelector again to select the ul element, and toggling the show class on it using the classList property. This will either show or hide the links, depending on their current display state.

**Final Code**

Here's the final code for the responsive navigation bar with a responsive menu button:

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Responsive Navigation Bar</title>**

**<meta name="viewport" content="width=device-width, initial-scale=1">**

**<style>**

**nav {**

**display: flex;**

**justify-content: space-between;**

**align-items: center;**

**background-color: #333;**

**color: #fff;**

**padding: 10px;**

**}**

**nav ul {**

**margin: 0;**

**padding: 0;**

**list-style: none;**

**display: flex;**

**flex-direction: row;**

**}**

**nav ul li {**

**margin: 0 10px;**

**}**

**nav ul li a {**

**color: #fff;**

**text-decoration: none;**

**font-weight: bold;**

**font-size: 16px;**

**padding: 10px;**

**border-radius: 5px;**

**transition: background-color 0.3s ease-in-out;**

**}**

**nav ul li a:hover {**

**background-color: #666;**

**}**

**@media screen and (max-width: 768px) {**

**nav {**

**flex-direction: column;**

**align-items: stretch;**

**}**

**nav ul {**

**flex-direction: column;**

**display: none;**

**}**

**nav ul.show {**

**display: flex;**

**}**

**nav ul li {**

**margin: 10px 0;**

**}**

**.menu-button {**

**display: block;**

**background-color: transparent;**

**border: none;**

**color: #fff;**

**font-size: 16px;**

**padding: 10px;**

**cursor: pointer;**

**margin-top: 10px;**

**}**

**}**

**</style>**

**</head>**

**<body>**

**<nav>**

**<div class="logo">**

**<a href="#">Logo</a>**

**</div>**

**<ul>**

**<li><a href="#">Home</a></li>**

**<li><a href="#">About</a></li>**

**<li><a href="#">Services</a></li>**

**<li><a href="#">Contact</a></li>**

**</ul>**

**<button class="menu-button">&#9776;</button>**

**</nav>**

**<script>**

**document.querySelector(".menu-button").addEventListener("click", function() {**

**document.querySelector("nav ul").classList.toggle("show");**

**});**

**</script>**

**</body>**

**</html>**

**Summary**

We have walked through how to create a simple responsive navigation bar using HTML and CSS, and how to add a responsive menu button using JavaScript. By using media queries and Flexbox, we were able to create a navigation bar that switches between horizontal and vertical layouts depending on the screen size, and added a menu button to toggle the display of the links on small screens. This is just one example of how you can create a responsive navigation bar - there are many other approaches and variations you can try depending on your specific needs and preferences.