**Exercises on Media Queries and Fluid Layouts in CSS3**

**Exercise 1: Responsive Navigation Menu**

**Objective**: Create a responsive navigation menu that adjusts based on screen width.

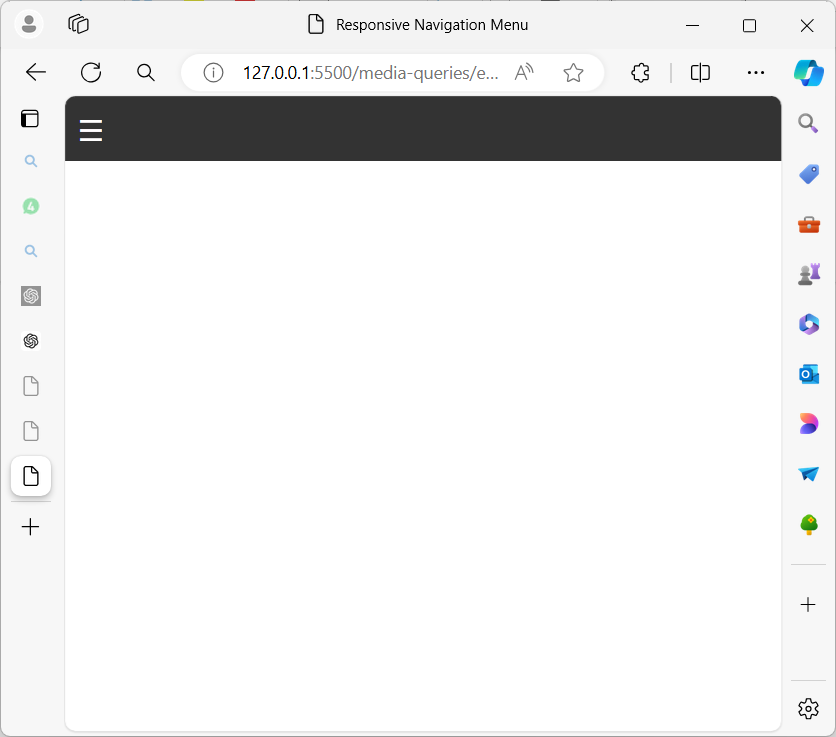
**Requirements**:

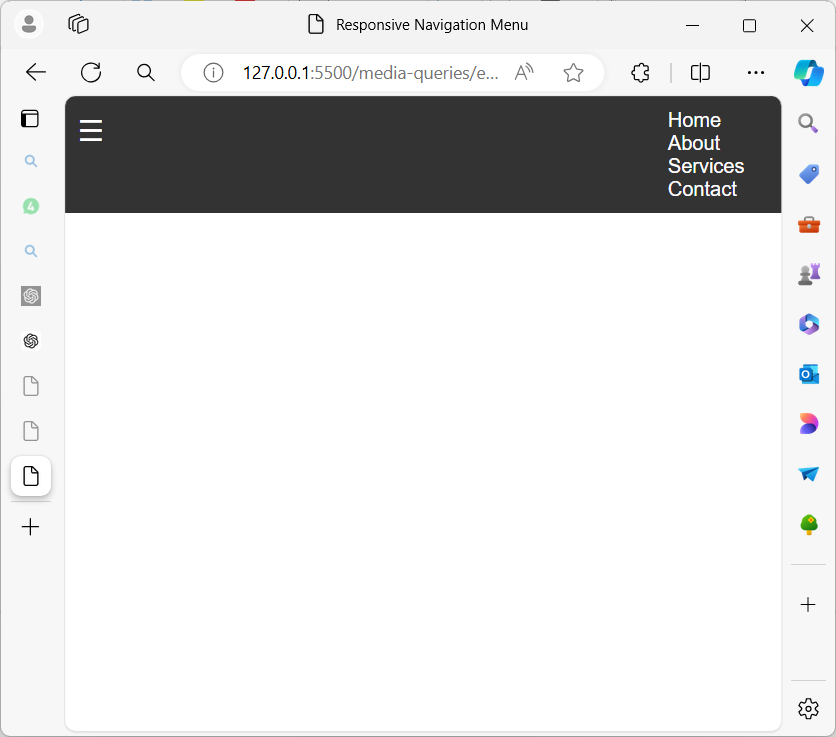
1. Create a horizontal navigation menu with links (Home, About, Services, Contact).
2. On screens wider than 768px, display the menu items in a horizontal row.
3. On screens 768px or narrower, display the menu items in a vertical column.
4. Add a hamburger icon that toggles the visibility of the menu items on small screens.

**Hints**:

* Use @media queries to adjust the layout for different screen sizes.
* Consider using Flexbox or Grid for layout adjustments.







**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Responsive Navigation Menu</title>

<style>

body {

margin: 0;

font-family: Arial, sans-serif;

}

.nav {

display: flex;

justify-content: space-between;

background-color: #333;

padding: 10px;

color: white;

}

.nav ul {

list-style: none;

margin: 0;

padding: 0;

display: flex;

}

.nav ul li {

margin-right: 20px;

}

.nav ul li a {

color: white;

text-decoration: none;

}

.hamburger {

display: none;

font-size: 1.5em;

cursor: pointer;

}

.nav-menu {

display: flex;

}

@media (max-width: 768px) {

.nav ul {

display: none;

flex-direction: column;

}

.nav ul.show {

display: flex;

}

.hamburger {

display: block;

}

}

</style>

</head>

<body>

<header class="nav">

<div class="hamburger" onclick="toggleMenu()">☰</div>

<ul class="nav-menu">

<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>

</header>

<script>

function toggleMenu() {

document.querySelector('.nav-menu').classList.toggle('show');

}

</script>

</body>

</html>

**Exercise 2: Fluid Grid Layout**

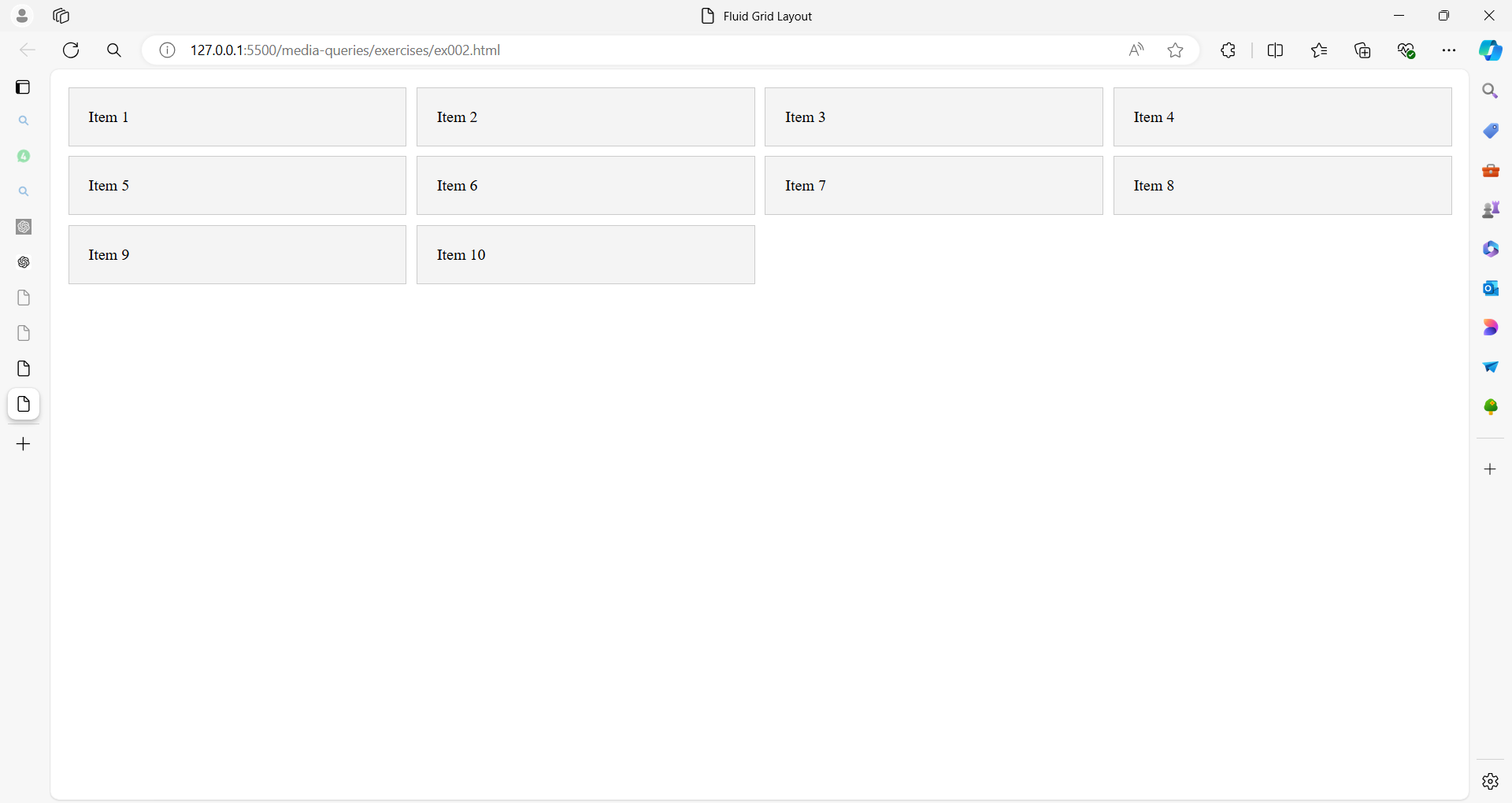
**Objective**: Create a fluid grid layout that adjusts the number of columns based on the screen width.

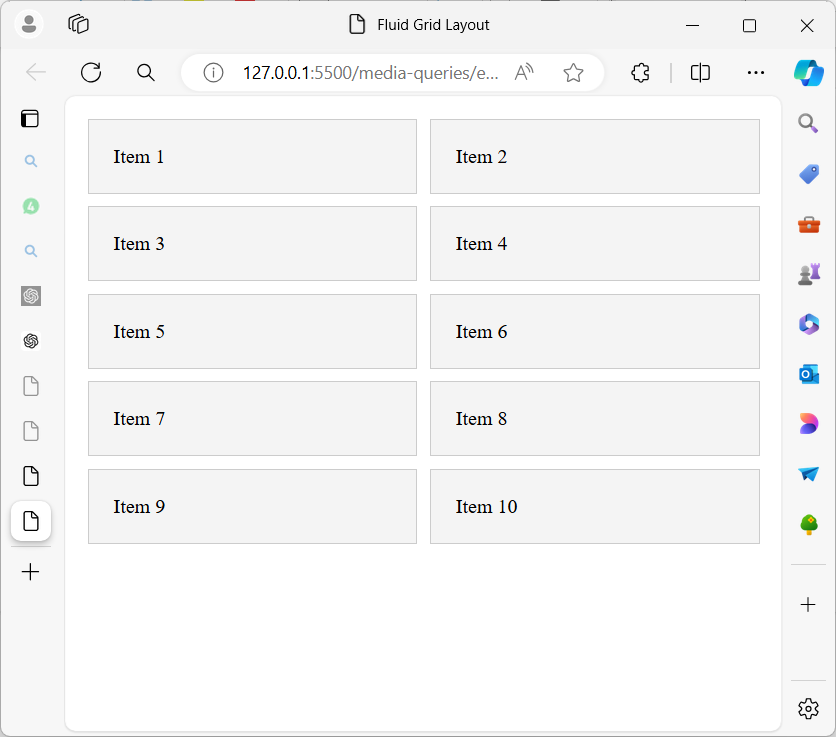
**Requirements**:

1. Create a grid layout with a variable number of columns.
2. On screens wider than 1024px, display 4 columns.
3. On screens between 768px and 1024px, display 3 columns.
4. On screens narrower than 768px, display 2 columns.

**Hints**:

* Use @media queries to adjust the grid layout.
* Use percentage-based widths for fluid layouts.





**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Fluid Grid Layout</title>

<style>

.grid {

display: grid;

gap: 10px;

padding: 10px;

}

.grid-item {

background-color: #f4f4f4;

border: 1px solid #ccc;

padding: 20px;

box-sizing: border-box;

}

@media (min-width: 1024px) {

.grid {

grid-template-columns: repeat(4, 1fr);

}

}

@media (min-width: 768px) and (max-width: 1023px) {

.grid {

grid-template-columns: repeat(3, 1fr);

}

}

@media (max-width: 767px) {

.grid {

grid-template-columns: repeat(2, 1fr);

}

}

</style>

</head>

<body>

<div class="grid">

<div class="grid-item">Item 1</div>

<div class="grid-item">Item 2</div>

<div class="grid-item">Item 3</div>

<div class="grid-item">Item 4</div>

<div class="grid-item">Item 5</div>

<div class="grid-item">Item 6</div>

<div class="grid-item">Item 7</div>

<div class="grid-item">Item 8</div>

<div class="grid-item">Item 9</div>

<div class="grid-item">Item 10</div>

</div>

</body>

</html>

**Exercise 3: Responsive Typography**

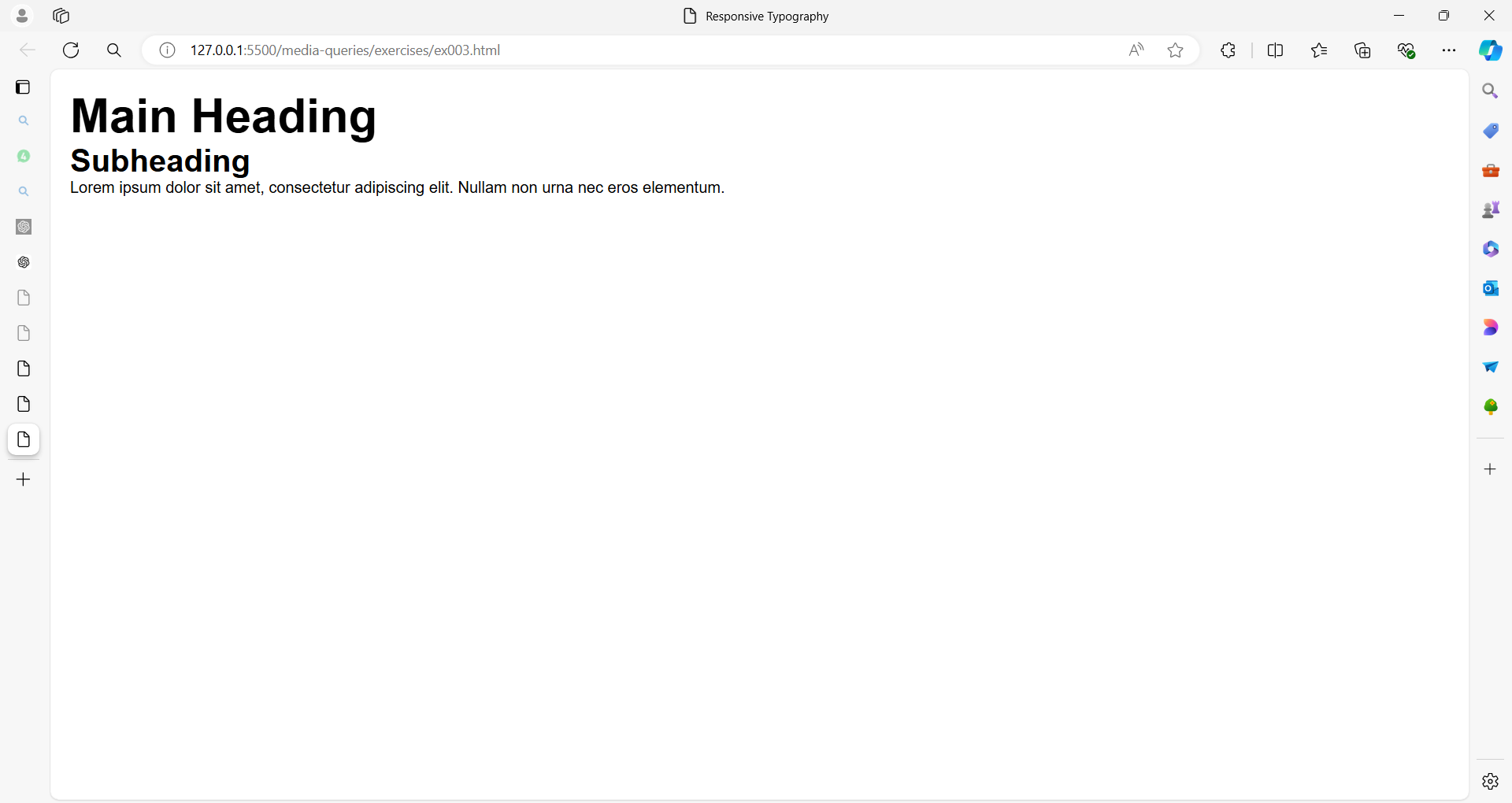
**Objective**: Adjust the typography based on the viewport width.

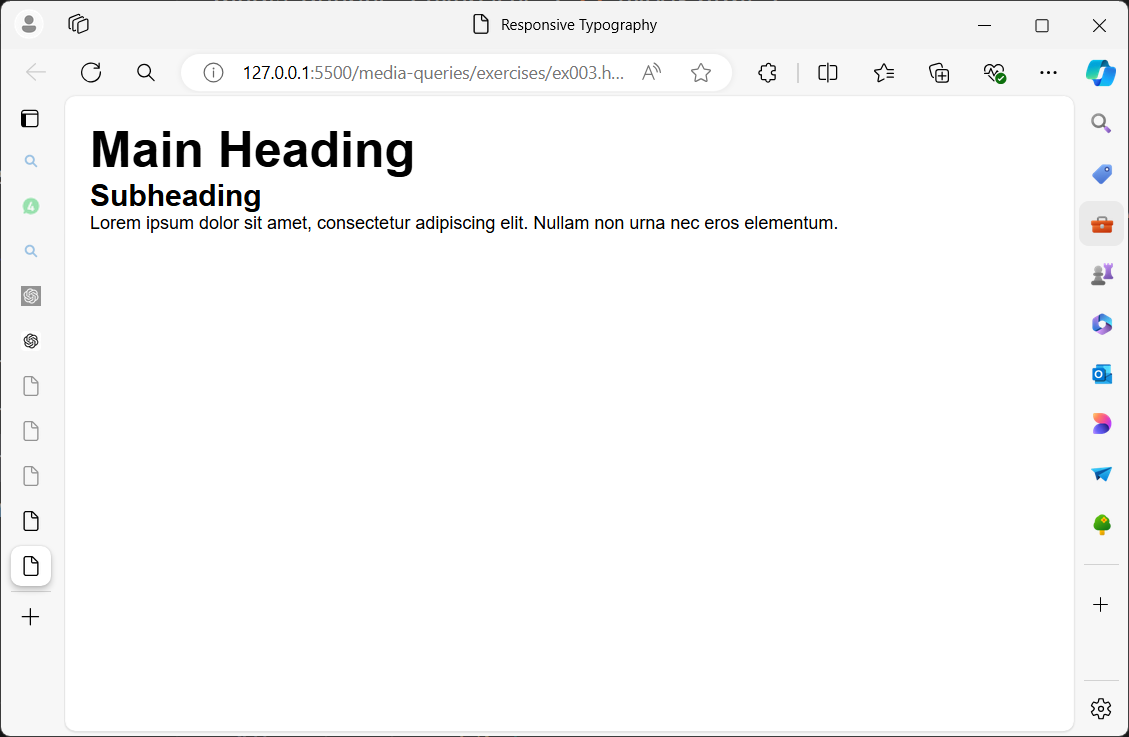
**Requirements**:

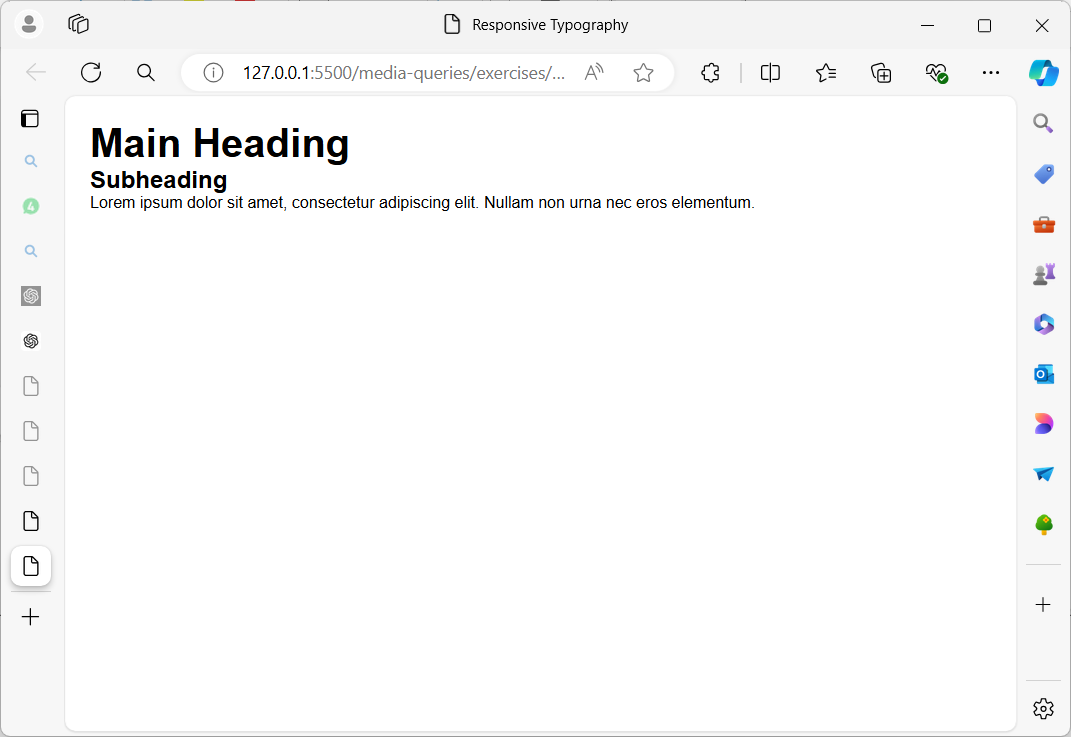
1. Create a page with a heading, subheading, and paragraph text.
2. On screens wider than 1200px, set the heading font size to 3em, the subheading to 2em, and the paragraph to 1em.
3. On screens between 768px and 1200px, set the heading font size to 2.5em, the subheading to 1.5em, and the paragraph to 0.9em.
4. On screens narrower than 768px, set the heading font size to 2em, the subheading to 1.2em, and the paragraph to 0.8em.

**Hints**:

* Use @media queries to adjust font sizes.







**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Responsive Typography</title>

<style>

body {

margin: 0;

font-family: Arial, sans-serif;

padding: 20px;

}

h1 {

margin: 0;

}

h2 {

margin: 0;

}

p {

margin: 0;

}

@media (min-width: 1200px) {

h1 {

font-size: 3em;

}

h2 {

font-size: 2em;

}

p {

font-size: 1em;

}

}

@media (min-width: 768px) and (max-width: 1199px) {

h1 {

font-size: 2.5em;

}

h2 {

font-size: 1.5em;

}

p {

font-size: 0.9em;

}

}

@media (max-width: 767px) {

h1 {

font-size: 2em;

}

h2 {

font-size: 1.2em;

}

p {

font-size: 0.8em;

}

}

</style>

</head>

<body>

<h1>Main Heading</h1>

<h2>Subheading</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam non urna nec eros elementum.</p>

</body>

</html>

**Notes:**

* **Responsive Navigation Menu**: This exercise introduces basic responsive design principles and JavaScript for interaction.
* **Fluid Grid Layout**: Demonstrates how to create a flexible layout that adapts to different screen sizes.
* **Responsive Typography**: Focuses on adjusting text sizes based on screen width, an essential aspect of responsive design.

**Exercise 4: Responsive Card Layout**

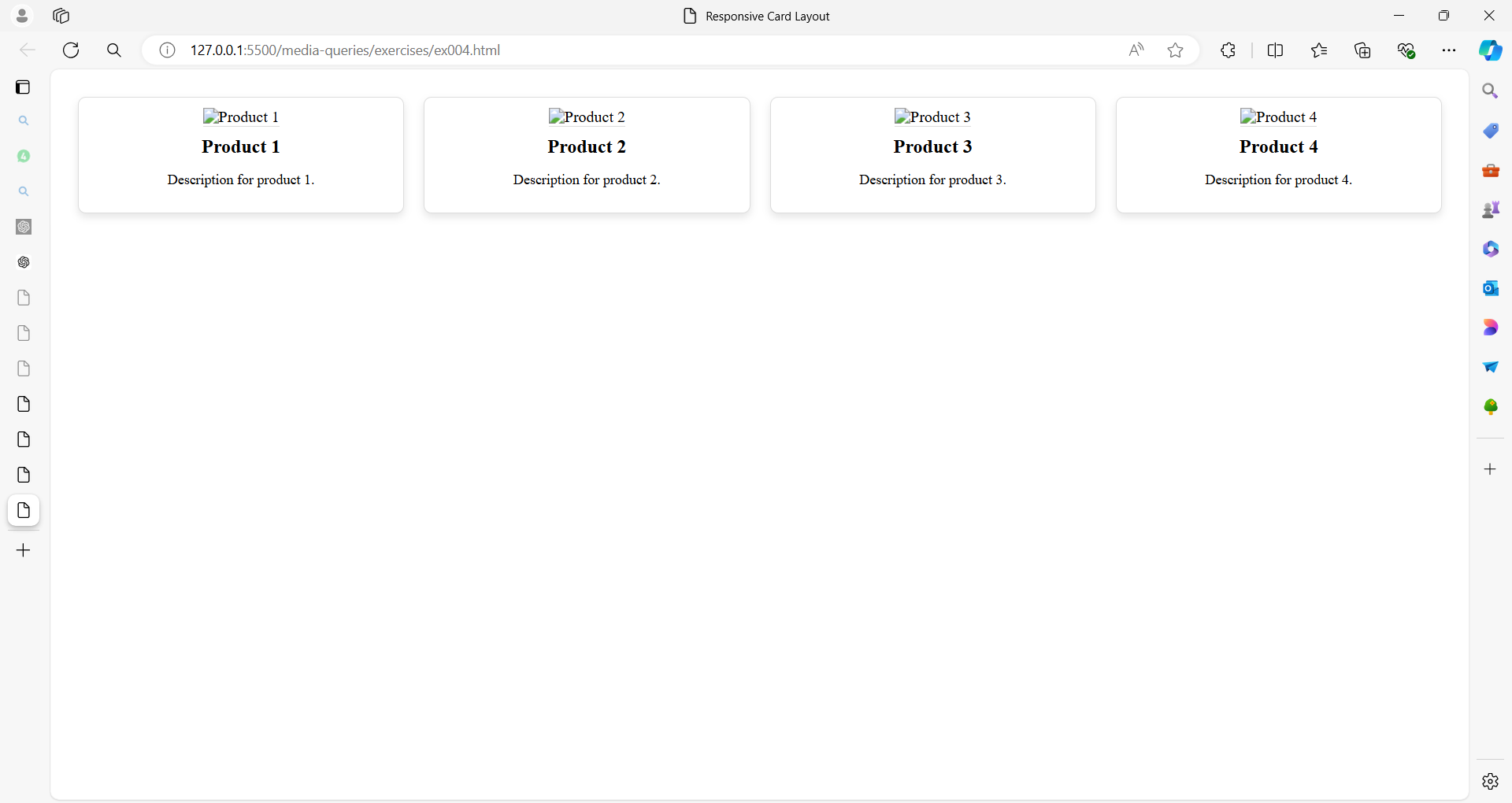
**Objective**: Create a responsive card layout that adapts to different screen sizes.

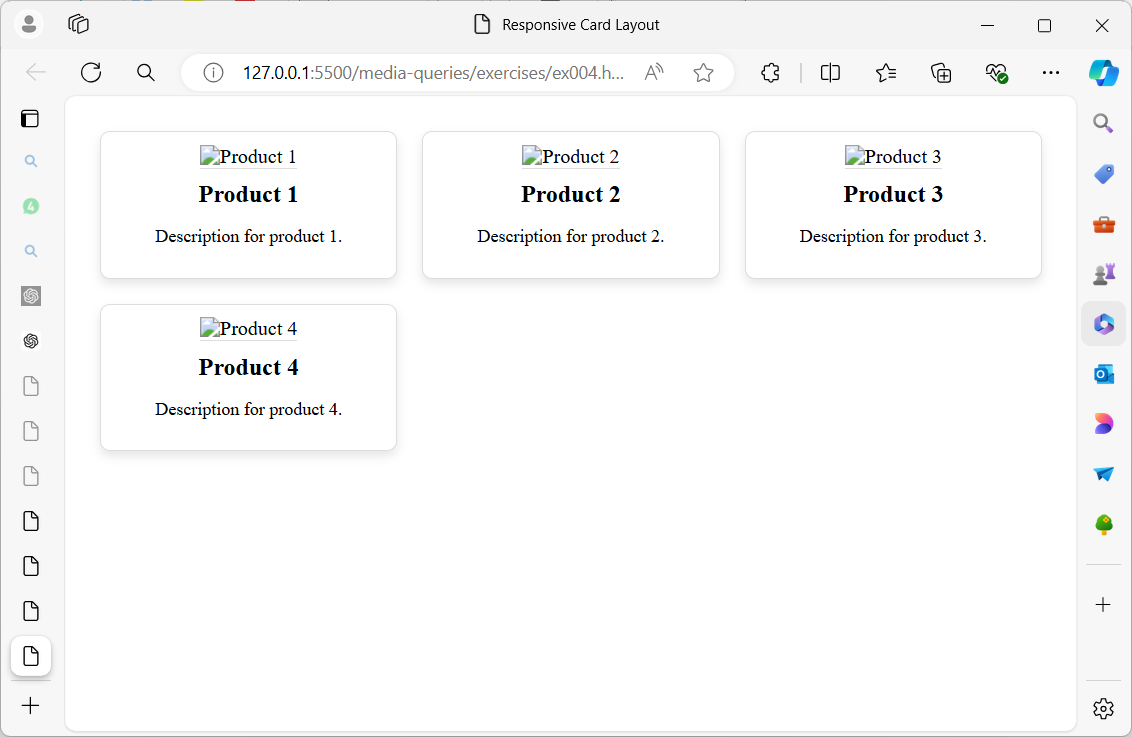
**Requirements**:

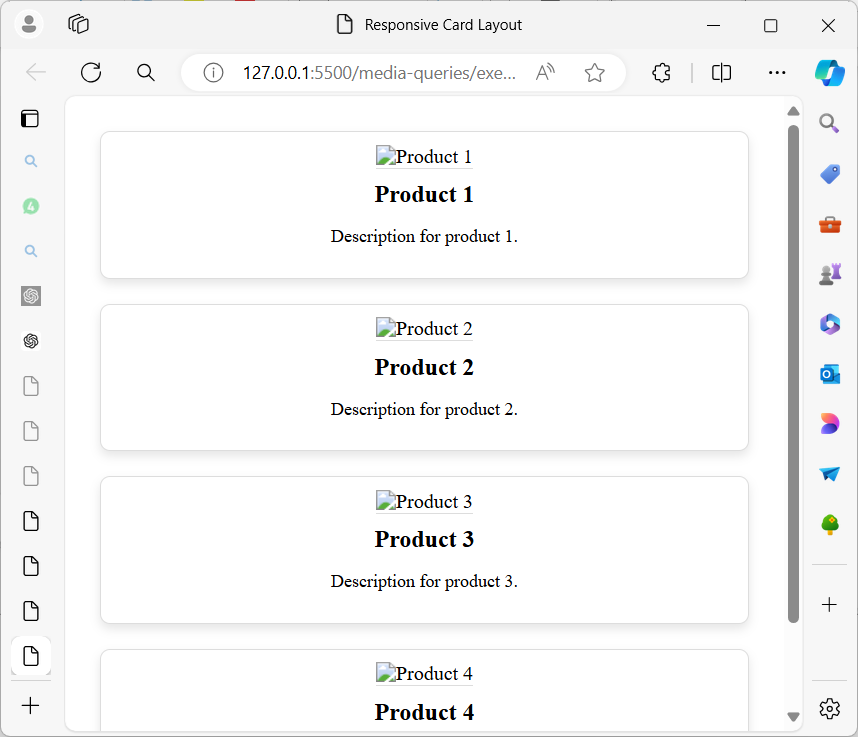
1. Create a grid of cards, each with an image, title, and description.
2. On screens wider than 1200px, display 4 cards per row.
3. On screens between 768px and 1200px, display 3 cards per row.
4. On screens narrower than 768px, display 1 card per row.

**Hints**:

* Use CSS Grid to layout the cards.
* Adjust grid-template-columns in @media queries for different screen sizes.







**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Responsive Card Layout</title>

<style>

.card-container {

display: grid;

gap: 20px;

padding: 20px;

}

.card {

background-color: #fff;

border: 1px solid #ddd;

border-radius: 8px;

overflow: hidden;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

text-align: center;

padding: 10px;

}

.card img {

width: 100%;

height: auto;

border-bottom: 1px solid #ddd;

}

.card h3 {

margin: 10px 0;

font-size: 1.2em;

}

.card p {

font-size: 0.9em;

}

@media (min-width: 1200px) {

.card-container {

grid-template-columns: repeat(4, 1fr);

}

}

@media (min-width: 768px) and (max-width: 1199px) {

.card-container {

grid-template-columns: repeat(3, 1fr);

}

}

@media (max-width: 767px) {

.card-container {

grid-template-columns: 1fr;

}

}

</style>

</head>

<body>

<div class="card-container">

<div class="card">

<img src="product1.jpg" alt="Product 1">

<h3>Product 1</h3>

<p>Description for product 1.</p>

</div>

<div class="card">

<img src="product2.jpg" alt="Product 2">

<h3>Product 2</h3>

<p>Description for product 2.</p>

</div>

<div class="card">

<img src="product3.jpg" alt="Product 3">

<h3>Product 3</h3>

<p>Description for product 3.</p>

</div>

<div class="card">

<img src="product4.jpg" alt="Product 4">

<h3>Product 4</h3>

<p>Description for product 4.</p>

</div>

<!-- Add more cards as needed -->

</div>

</body>

</html>

**Exercise 5: Fluid Typography with Viewport Units**

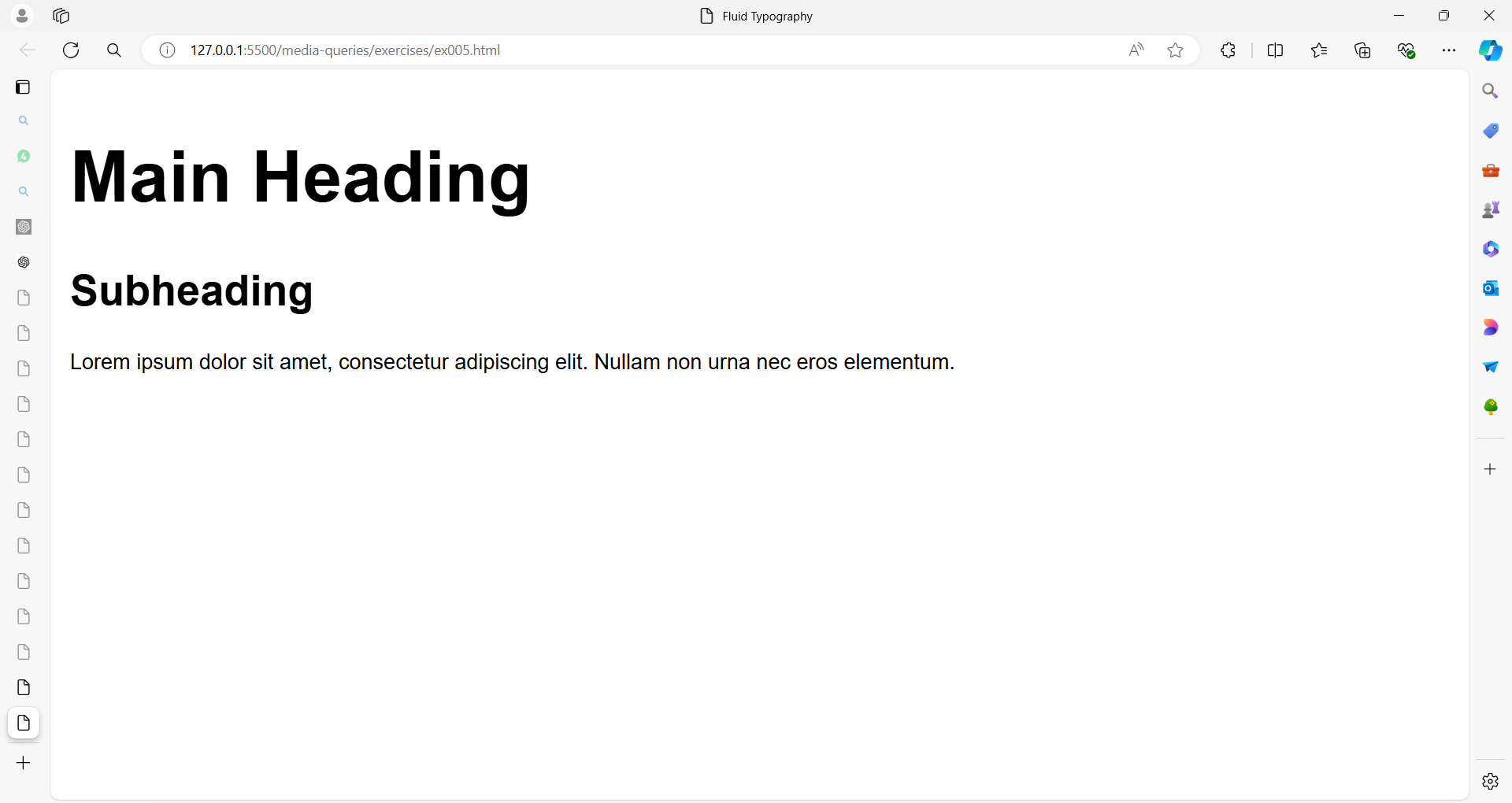
**Objective**: Create a page with typography that scales based on the viewport size using viewport units.

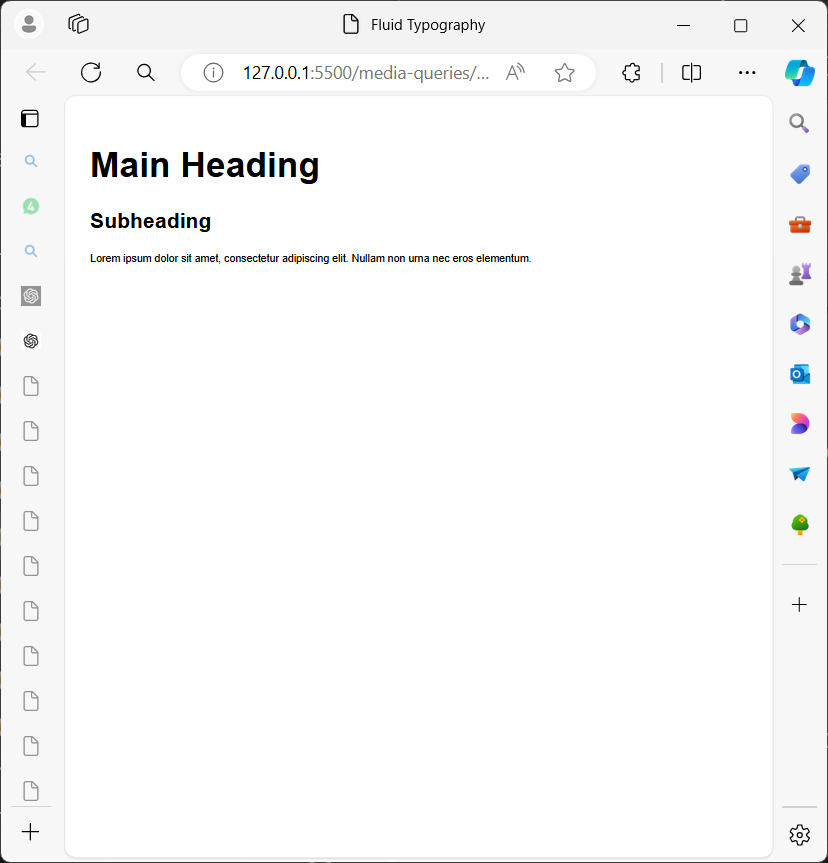
**Requirements**:

1. Create a heading, subheading, and paragraph text.
2. Use viewport units (vw and vh) to size the text so it scales fluidly with the viewport size.
3. Ensure the text remains readable on both very large and very small screens.

**Hints**:

* Use vw (viewport width) and vh (viewport height) units for font sizes.
* Combine viewport units with em or rem units to maintain readability.





**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Fluid Typography</title>

<style>

body {

margin: 0;

font-family: Arial, sans-serif;

padding: 20px;

}

h1 {

font-size: 5vw; /\* 5% of viewport width \*/

}

h2 {

font-size: 3vw; /\* 3% of viewport width \*/

}

p {

font-size: 1.5vw; /\* 1.5% of viewport width \*/

}

</style>

</head>

<body>

<h1>Main Heading</h1>

<h2>Subheading</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam non urna nec eros elementum.</p>

</body>

</html>

**Exercise 6: Responsive Image Gallery**

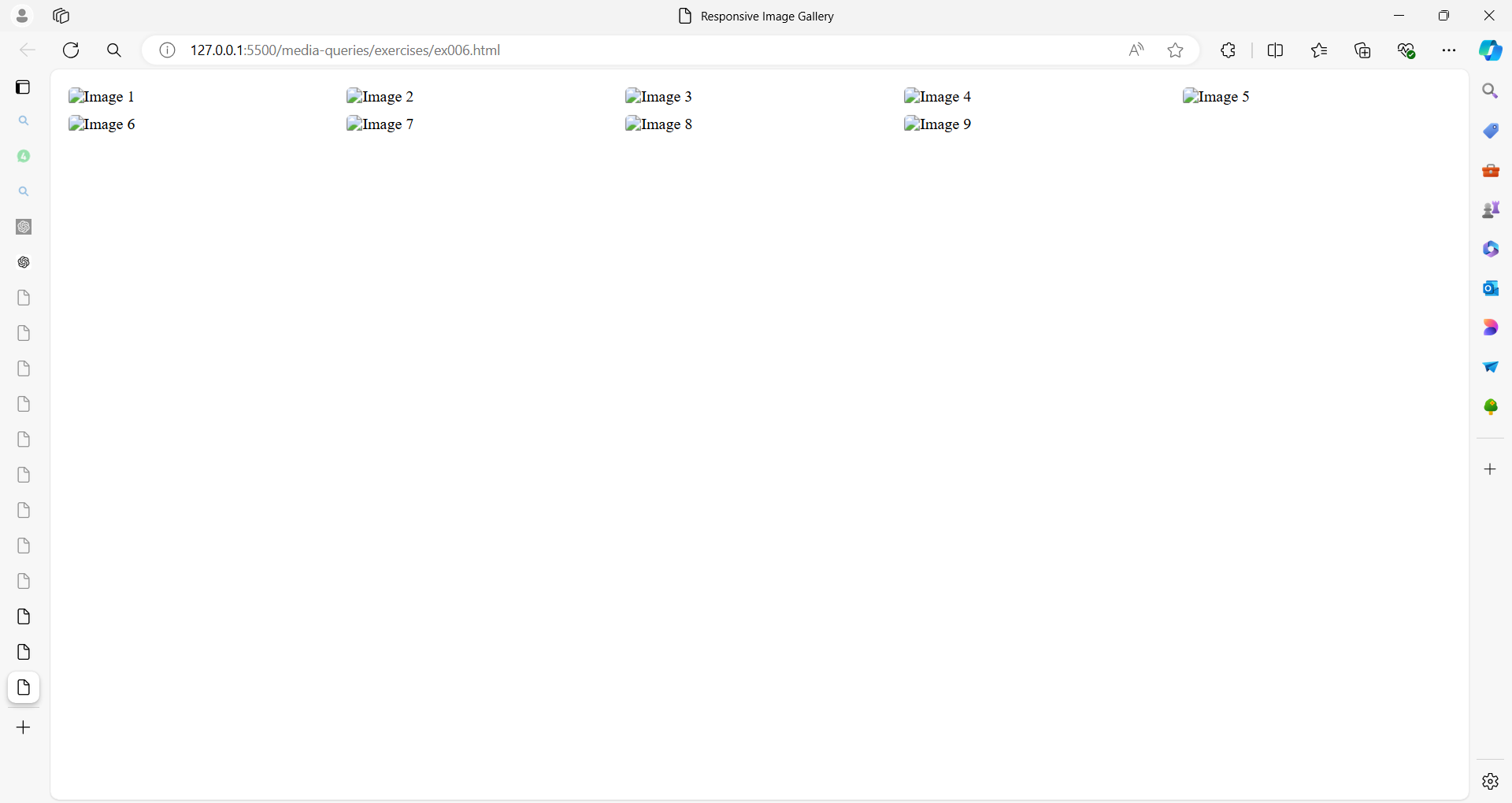
**Objective**: Create a responsive image gallery that adjusts the number of images per row based on screen width.

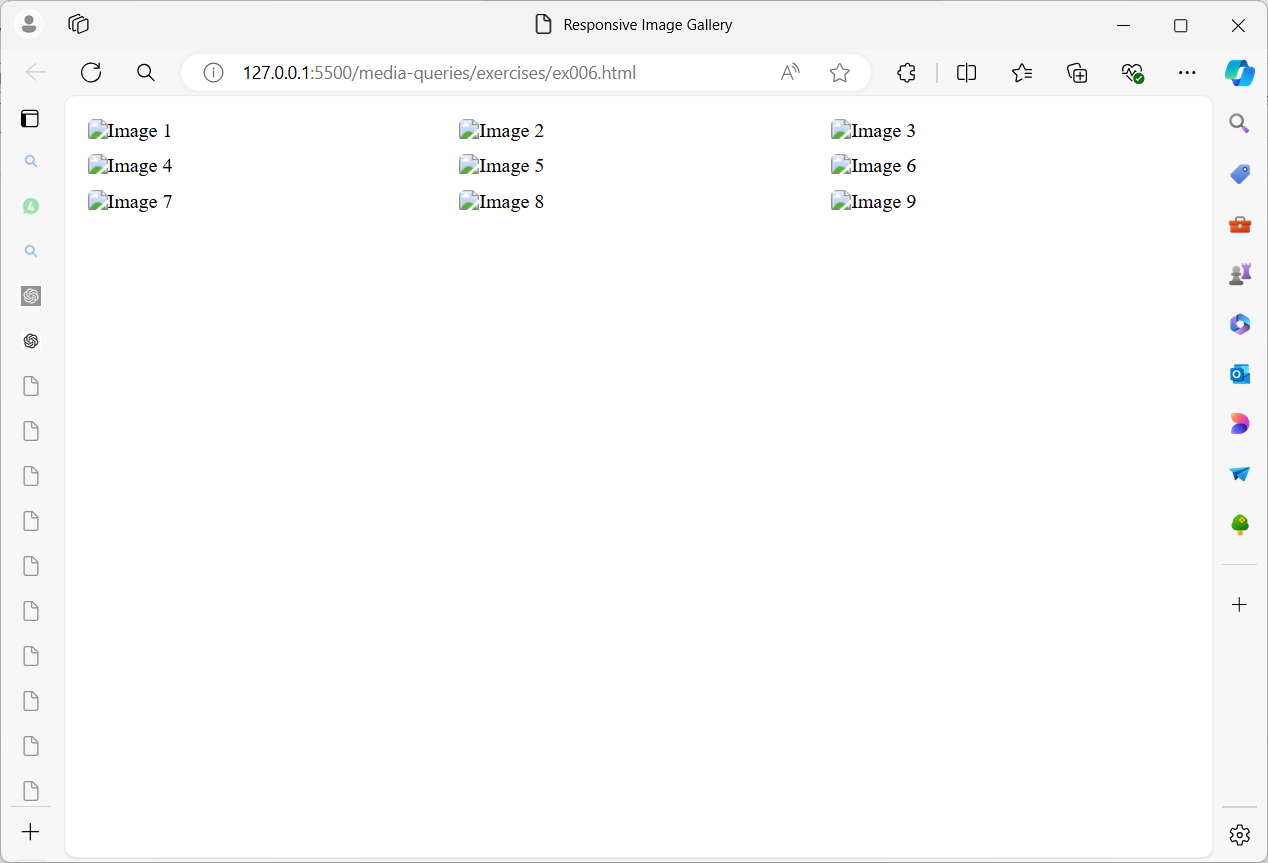
**Requirements**:

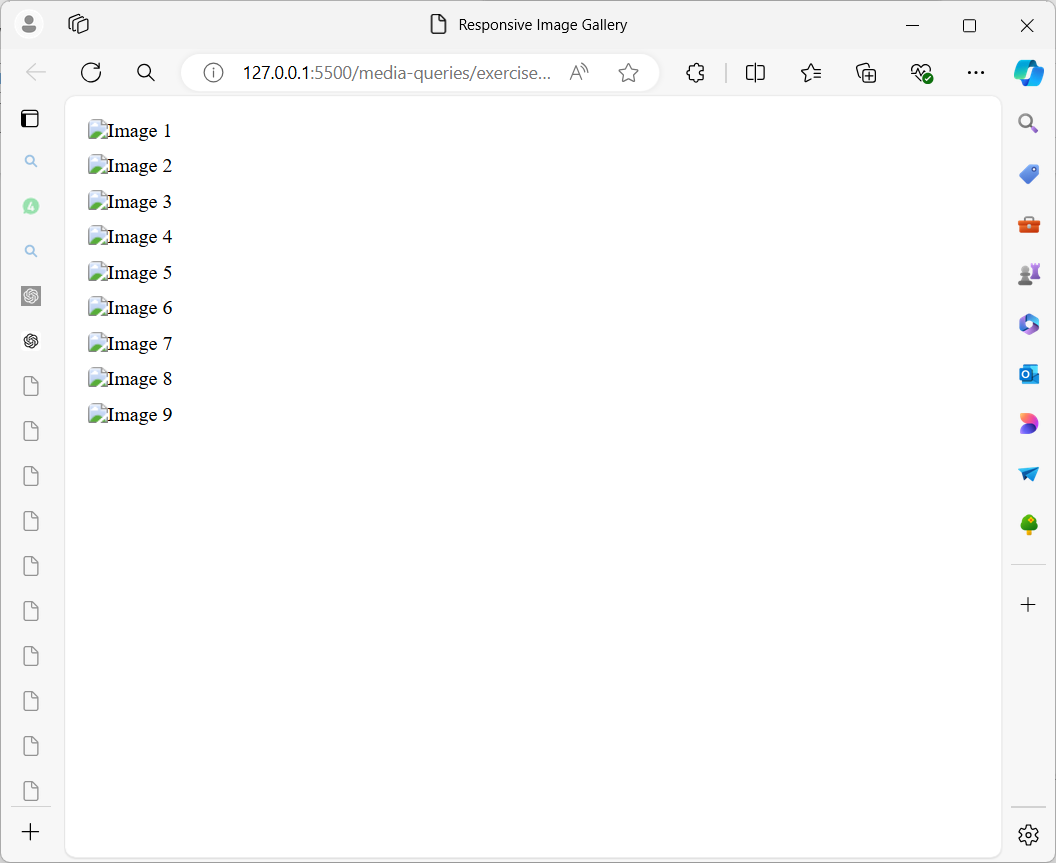
1. Create a gallery with images.
2. On screens wider than 1024px, display 5 images per row.
3. On screens between 768px and 1024px, display 3 images per row.
4. On screens narrower than 768px, display 1 image per row.

**Hints**:

* Use CSS Grid to layout the images.
* Adjust grid-template-columns in @media queries for different screen sizes.







**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Responsive Image Gallery</title>

<style>

.gallery {

display: grid;

gap: 10px;

padding: 10px;

}

.gallery img {

width: 100%;

height: auto;

border-radius: 5px;

}

@media (min-width: 1024px) {

.gallery {

grid-template-columns: repeat(5, 1fr);

}

}

@media (min-width: 768px) and (max-width: 1023px) {

.gallery {

grid-template-columns: repeat(3, 1fr);

}

}

@media (max-width: 767px) {

.gallery {

grid-template-columns: 1fr;

}

}

</style>

</head>

<body>

<div class="gallery">

<img src="image1.jpg" alt="Image 1">

<img src="image2.jpg" alt="Image 2">

<img src="image3.jpg" alt="Image 3">

<img src="image4.jpg" alt="Image 4">

<img src="image5.jpg" alt="Image 5">

<img src="image6.jpg" alt="Image 6">

<img src="image7.jpg" alt="Image 7">

<img src="image8.jpg" alt="Image 8">

<img src="image9.jpg" alt="Image 9">

</div>

</body>

</html>

**Exercise 7: Fluid Layout with Aspect Ratio**

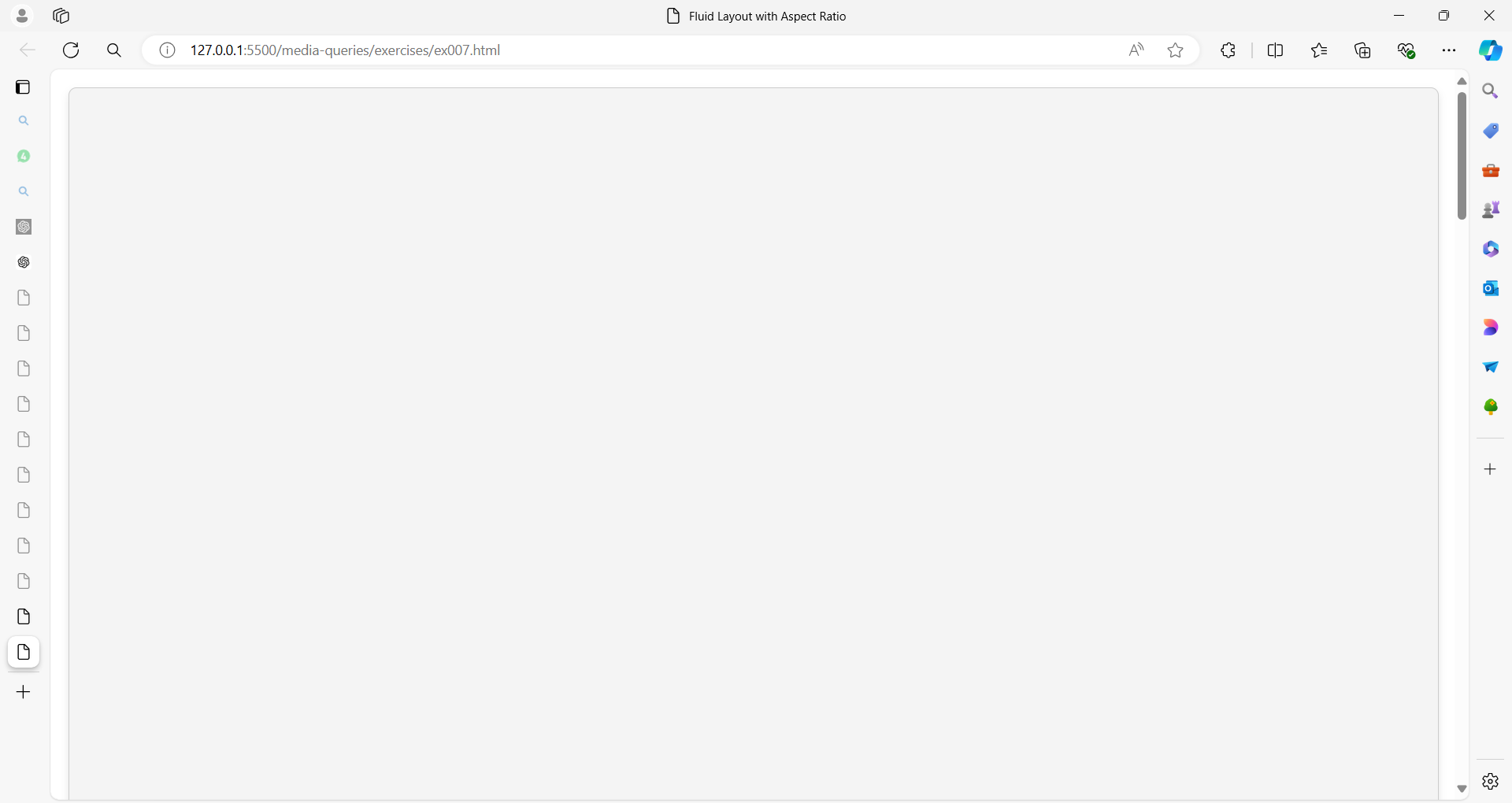
**Objective**: Create a layout with elements that maintain a specific aspect ratio and adjust based on the screen width.

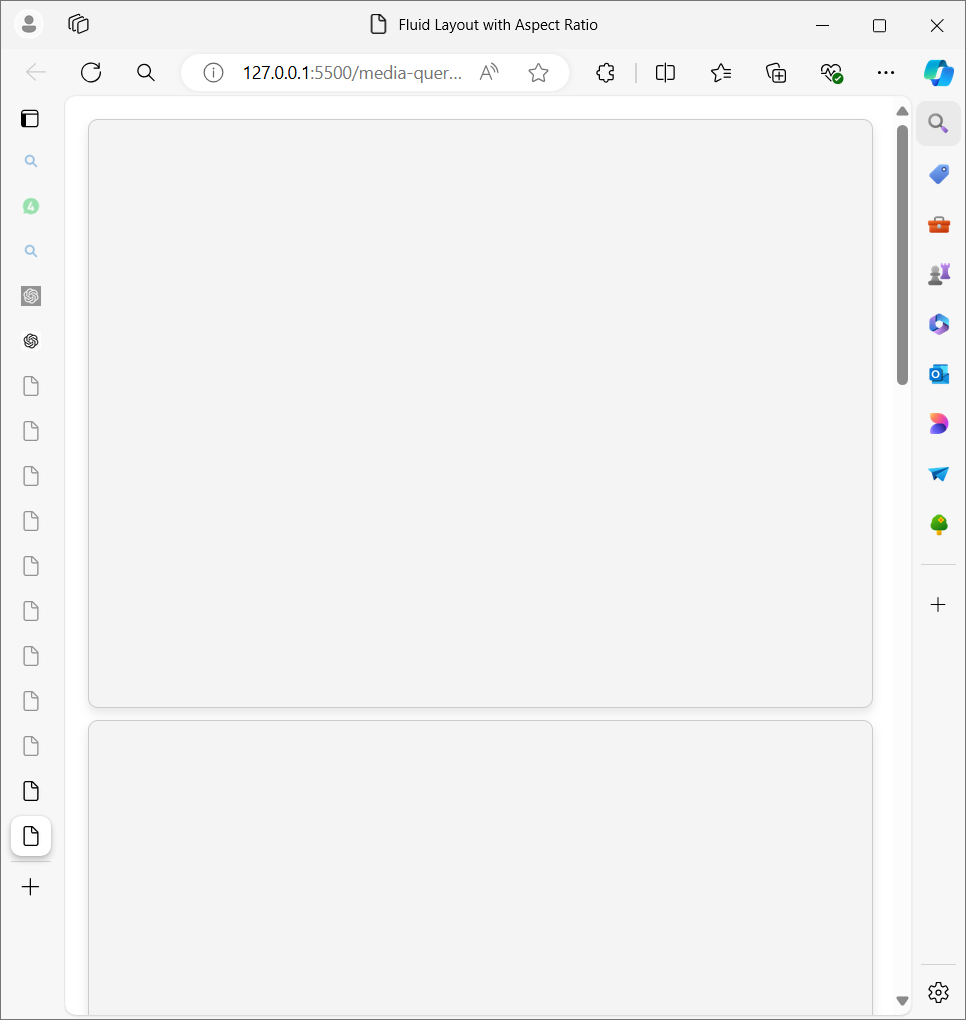
**Requirements**:

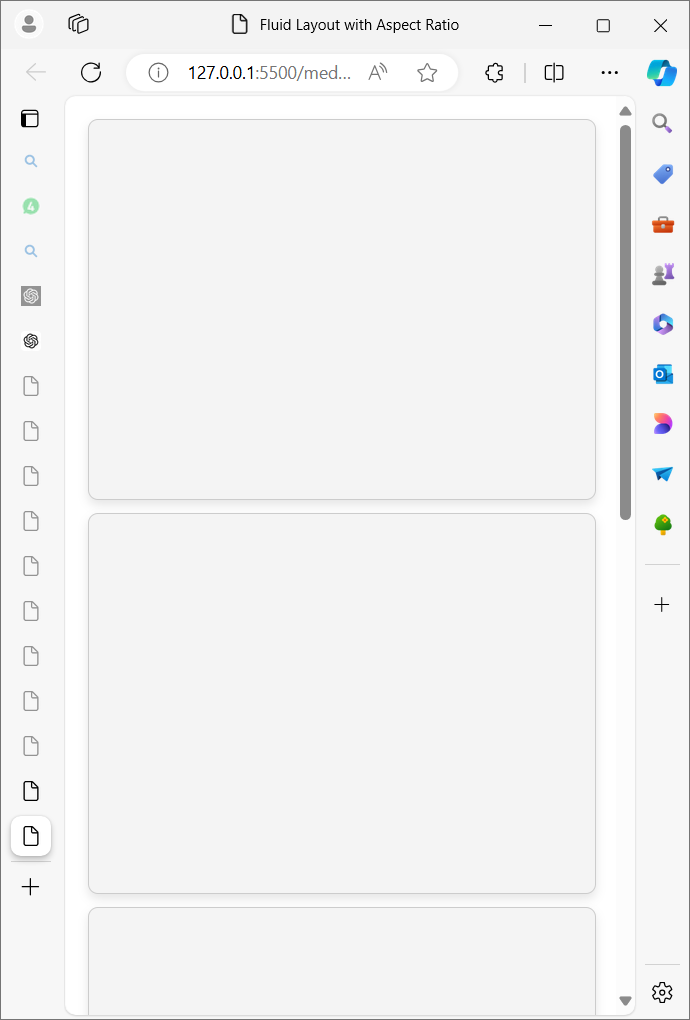
1. Create a container with a fixed aspect ratio.
2. Inside the container, add multiple items that maintain the aspect ratio of the container.
3. On screens wider than 1200px, the container should maintain a 16:9 aspect ratio.
4. On screens narrower than 1200px, adjust the aspect ratio to 4:3.

**Hints**:

* Use the aspect-ratio property to maintain the aspect ratio of elements.
* Adjust the aspect ratio in @media queries.







**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Fluid Layout with Aspect Ratio</title>

<style>

.container {

display: grid;

gap: 10px;

padding: 10px;

}

.item {

background-color: #f4f4f4;

border: 1px solid #ccc;

border-radius: 8px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

overflow: hidden;

position: relative;

width: 100%;

aspect-ratio: 16 / 9; /\* Default aspect ratio \*/

}

@media (max-width: 1200px) {

.item {

aspect-ratio: 4 / 3; /\* Adjusted aspect ratio \*/

}

}

</style>

</head>

<body>

<div class="container">

<div class="item"></div>

<div class="item"></div>

<div class="item"></div>

<div class="item"></div>

<div class="item"></div>

</div>

</body>

</html>

**Exercise 8: Responsive E-commerce Product Page**

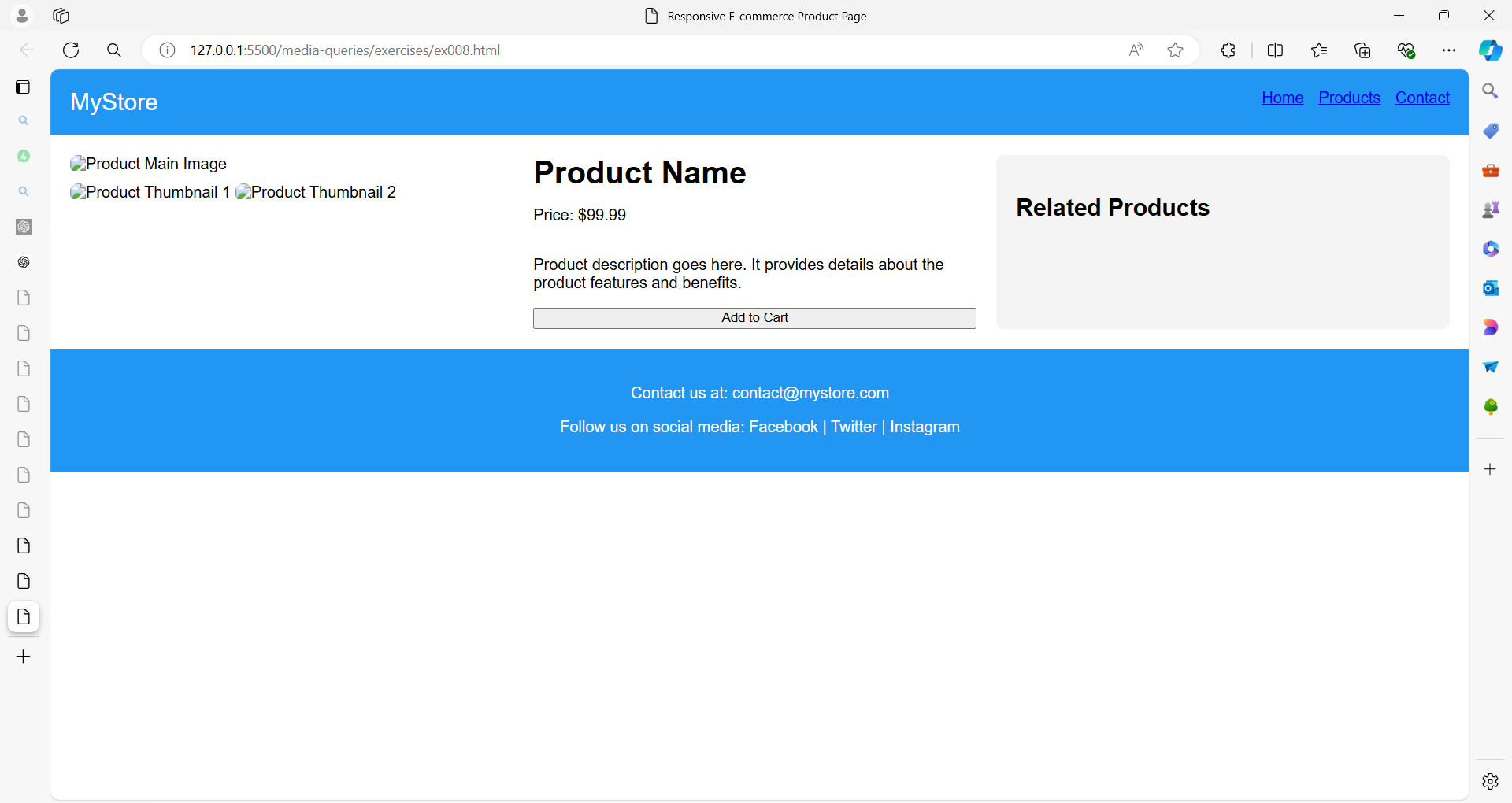
**Objective**: Design a responsive e-commerce product page that adapts to different screen sizes with a fluid layout.

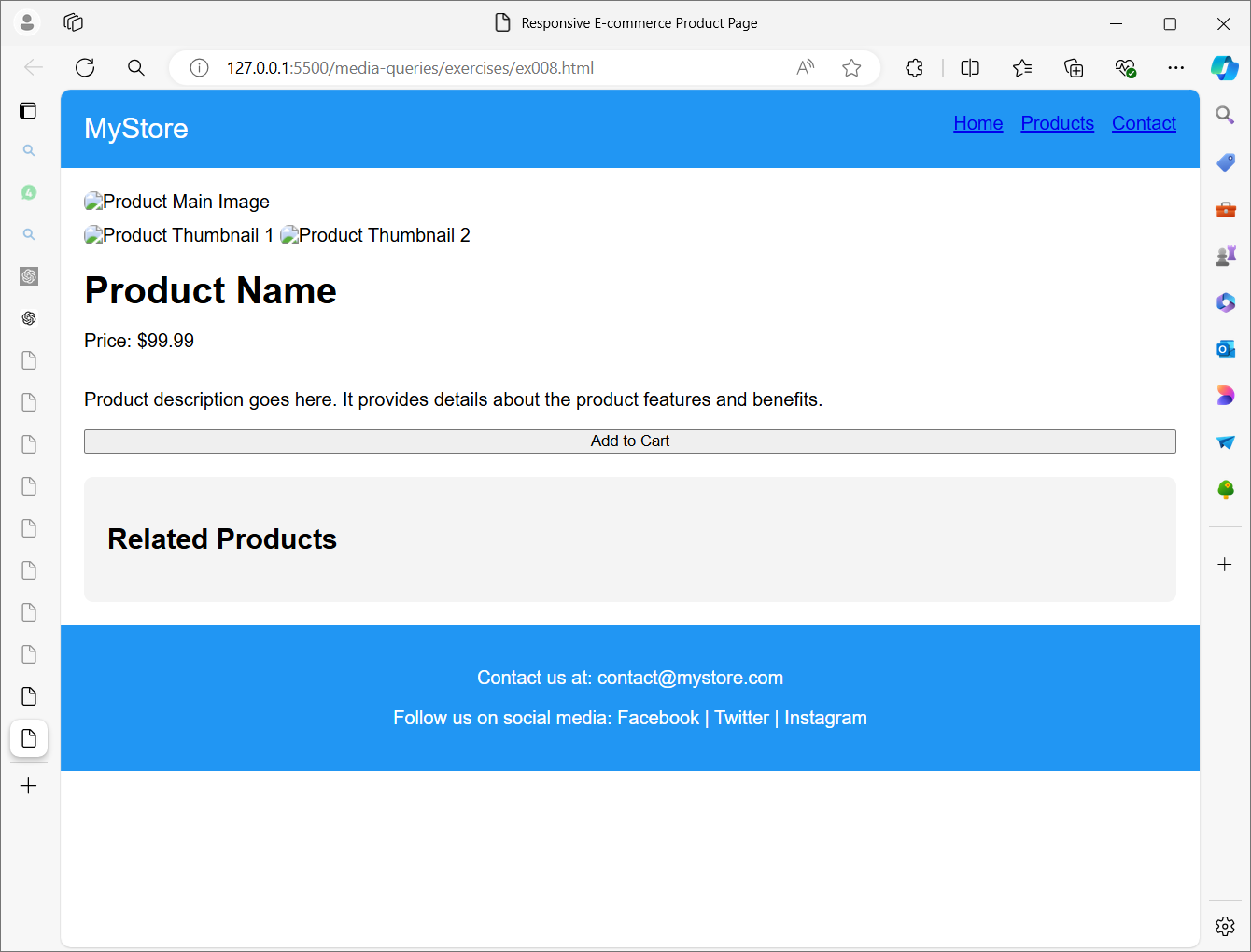
**Requirements**:

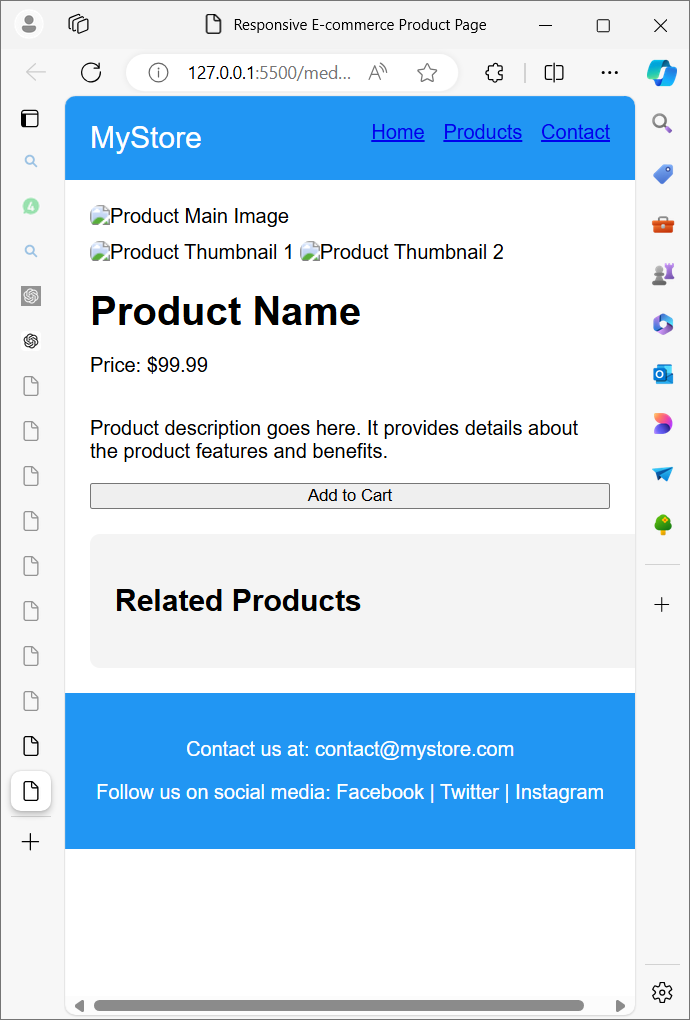
1. **Header**: Contains a logo on the left and navigation links on the right.
2. **Product Section**:
   * **Image Gallery**: Display a main product image and thumbnail images below it.
   * **Product Details**: Include product name, price, description, and add-to-cart button.
3. **Sidebar**: Show related products on the side.
4. **Footer**: Include contact information and social media links.
5. **Responsive Behavior**:
   * **Desktop (min-width: 1024px)**: Display the product image gallery and details side-by-side with the sidebar.
   * **Tablet (min-width: 768px and max-width: 1023px)**: Stack the product details below the image gallery and place the sidebar below the product section.
   * **Mobile (max-width: 767px)**: Stack all sections vertically and ensure images and text fit within the screen width.

**Hints**:

* Use CSS Grid or Flexbox for layout.
* Utilize @media queries to adjust layout and element sizes.







**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Responsive E-commerce Product Page</title>

<style>

body {

margin: 0;

font-family: Arial, sans-serif;

}

header {

display: flex;

justify-content: space-between;

padding: 20px;

background-color: #2196F3;

color: white;

}

header .logo {

font-size: 1.5em;

}

header .nav {

display: flex;

gap: 15px;

}

.container {

display: grid;

gap: 20px;

padding: 20px;

}

.product-section {

display: grid;

grid-template-columns: 1fr;

gap: 20px;

}

.product-gallery {

display: flex;

flex-direction: column;

gap: 10px;

}

.product-gallery img {

width: 100%;

height: auto;

border-radius: 8px;

}

.product-details {

display: flex;

flex-direction: column;

}

.product-details h1 {

margin: 0;

}

.related-products {

background-color: #f4f4f4;

padding: 20px;

border-radius: 8px;

}

.footer {

background-color: #2196F3;

color: white;

padding: 20px;

text-align: center;

}

@media (min-width: 1024px) {

.container {

grid-template-columns: 2fr 1fr;

}

.product-section {

grid-template-columns: 1fr 1fr;

grid-template-areas: "gallery details";

}

.product-gallery {

grid-area: gallery;

}

.product-details {

grid-area: details;

}

.related-products {

grid-column: 2 / 3;

}

}

@media (min-width: 768px) and (max-width: 1023px) {

.product-section {

grid-template-columns: 1fr;

}

.product-gallery {

order: 1;

}

.product-details {

order: 2;

}

.related-products {

order: 3;

grid-column: 1 / 2;

}

}

@media (max-width: 767px) {

.product-section {

grid-template-columns: 1fr;

}

.product-gallery {

width: 100%;

}

.product-details {

width: 100%;

}

.related-products {

width: 100%;

}

}

</style>

</head>

<body>

<header>

<div class="logo">MyStore</div>

<nav class="nav">

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

<a href="#">Products</a>

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

</nav>

</header>

<div class="container">

<div class="product-section">

<div class="product-gallery">

<img src="product-main.jpg" alt="Product Main Image">

<div class="thumbnail-gallery">

<img src="product-thumbnail1.jpg" alt="Product Thumbnail 1">

<img src="product-thumbnail2.jpg" alt="Product Thumbnail 2">

</div>

</div>

<div class="product-details">

<h1>Product Name</h1>

<p>Price: $99.99</p>

<p>Product description goes here. It provides details about the product features and benefits.</p>

<button>Add to Cart</button>

</div>

</div>

<aside class="related-products">

<h2>Related Products</h2>

<!-- List related products here -->

</aside>

</div>

<footer class="footer">

<p>Contact us at: contact@mystore.com</p>

<p>Follow us on social media: Facebook | Twitter | Instagram</p>

</footer>

</body>

</html>

**Exercise 9: Responsive Dashboard Layout**

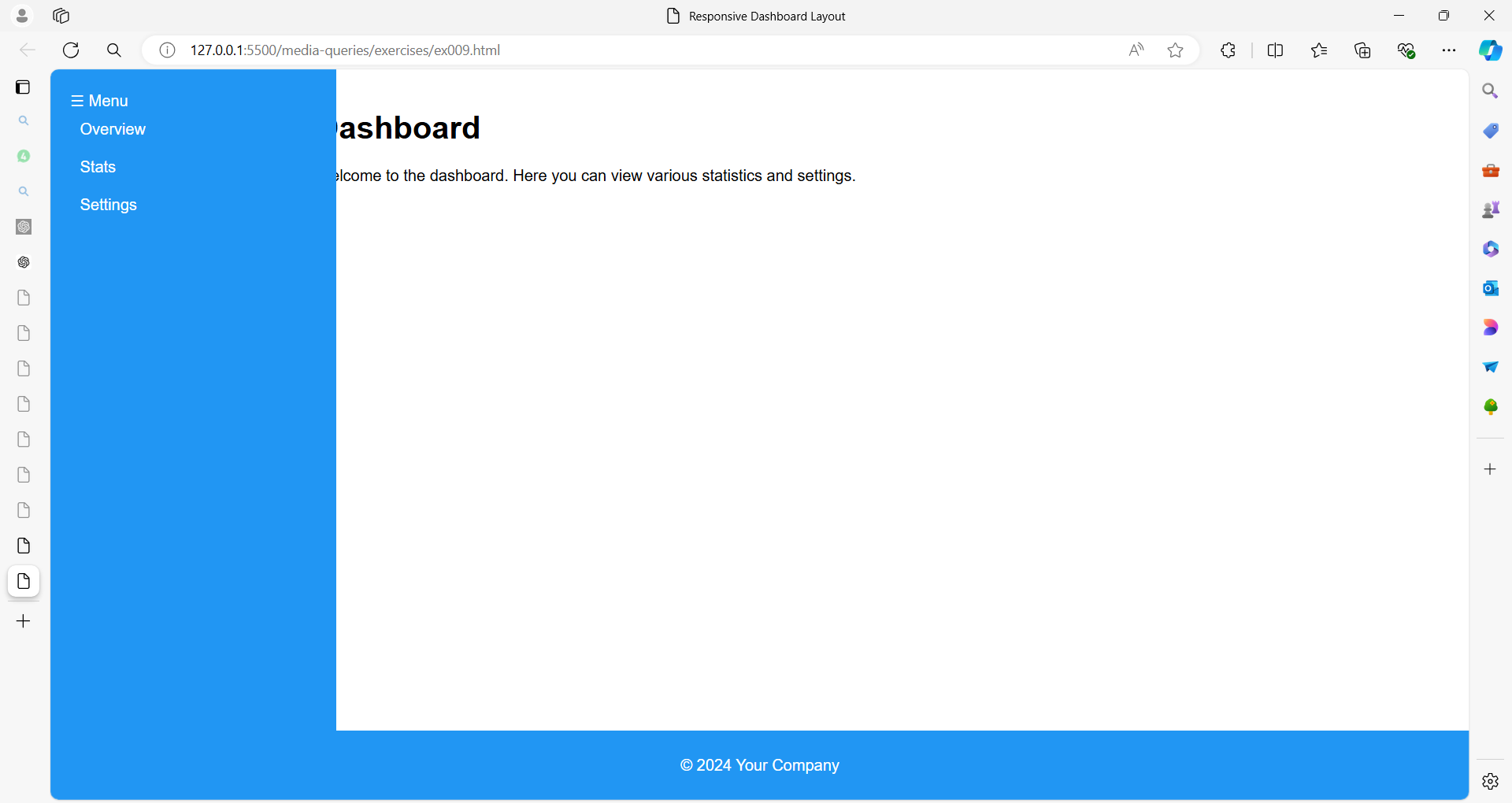
**Objective**: Create a responsive dashboard layout with a sidebar menu, main content area, and footer.

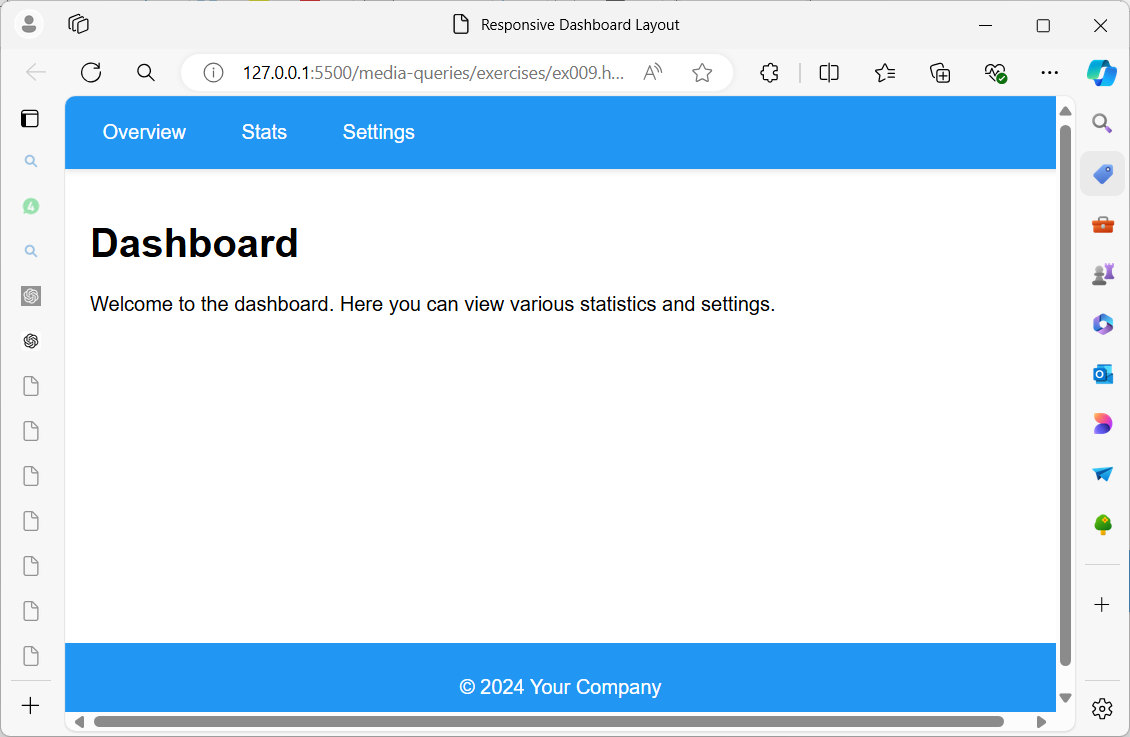
**Requirements**:

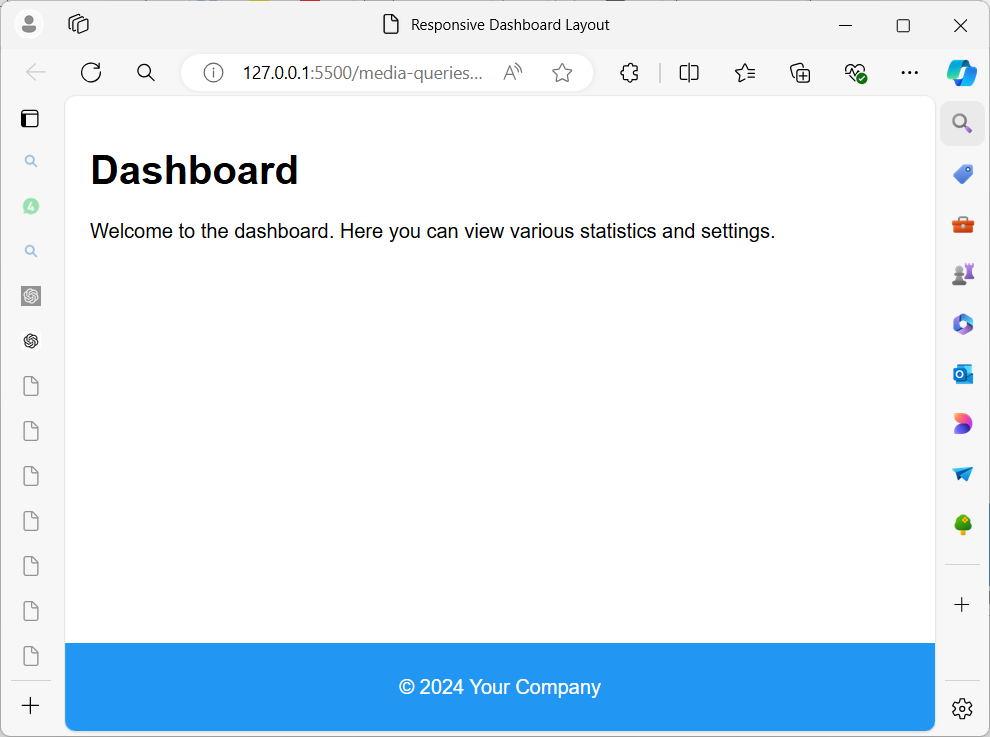
1. **Sidebar**: Contains navigation links.
2. **Main Content Area**: Displays different sections (e.g., overview, stats, settings).
3. **Footer**: Contains copyright information.
4. **Responsive Behavior**:
   * **Desktop (min-width: 1024px)**: Sidebar is fixed on the left, and the content area adjusts accordingly.
   * **Tablet (min-width: 768px and max-width: 1023px)**: Sidebar collapses into a top menu and content area adjusts to full width.
   * **Mobile (max-width: 767px)**: Sidebar collapses into a hamburger menu with a toggle button, content area is full width.

**Hints**:

* Use Flexbox or Grid for layout.
* Apply @media queries to handle different screen sizes and responsive behavior.







**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Responsive Dashboard Layout</title>

<style>

body {

margin: 0;

font-family: Arial, sans-serif;

display: flex;

flex-direction: column;

min-height: 100vh;

}

.sidebar {

background-color: #2196F3;

color: white;

padding: 20px;

width: 250px;

position: fixed;

top: 0;

bottom: 0;

overflow: auto;

}

.sidebar a {

color: white;

text-decoration: none;

display: block;

padding: 10px;

}

.sidebar a:hover {

background-color: #1976D2;

}

.main-content {

margin-left: 250px;

padding: 20px;

flex: 1;

}

.footer {

background-color: #2196F3;

color: white;

text-align: center;

padding: 10px;

}

@media (max-width: 1023px) {

.sidebar {

position: relative;

width: 100%;

height: auto;

display: flex;

justify-content: space-between;

flex-direction: row;

padding: 10px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

.sidebar a {

display: inline-block;

padding: 10px 20px;

}

.main-content {

margin-left: 0;

}

.sidebar-toggle {

display: none;

}

}

@media (max-width: 767px) {

.sidebar {

display: none;

}

.sidebar-toggle {

display: block;

background-color: #2196F3;

color: white;

padding: 10px;

text-align: center;

cursor: pointer;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

.sidebar-menu {

display: none;

}

.sidebar-menu.active {

display: block;

background-color: #2196F3;

color: white;

padding: 10px;

position: absolute;

top: 0;

left: 0;

width: 100%;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

}

</style>

</head>

<body>

<div class="sidebar">

<div class="sidebar-toggle">☰ Menu</div>

<div class="sidebar-menu">

<a href="#">Overview</a>

<a href="#">Stats</a>

<a href="#">Settings</a>

</div>

</div>

<div class="main-content">

<h1>Dashboard</h1>

<p>Welcome to the dashboard. Here you can view various statistics and settings.</p>

<!-- Additional content sections -->

</div>

<footer class="footer">

<p>&copy; 2024 Your Company</p>

</footer>

<script>

document.querySelector('.sidebar-toggle').addEventListener('click', function() {

document.querySelector('.sidebar-menu').classList.toggle('active');

});

</script>

</body>

</html>

**Exercise 10: Fluid Grid Layout with Aspect Ratio Control**

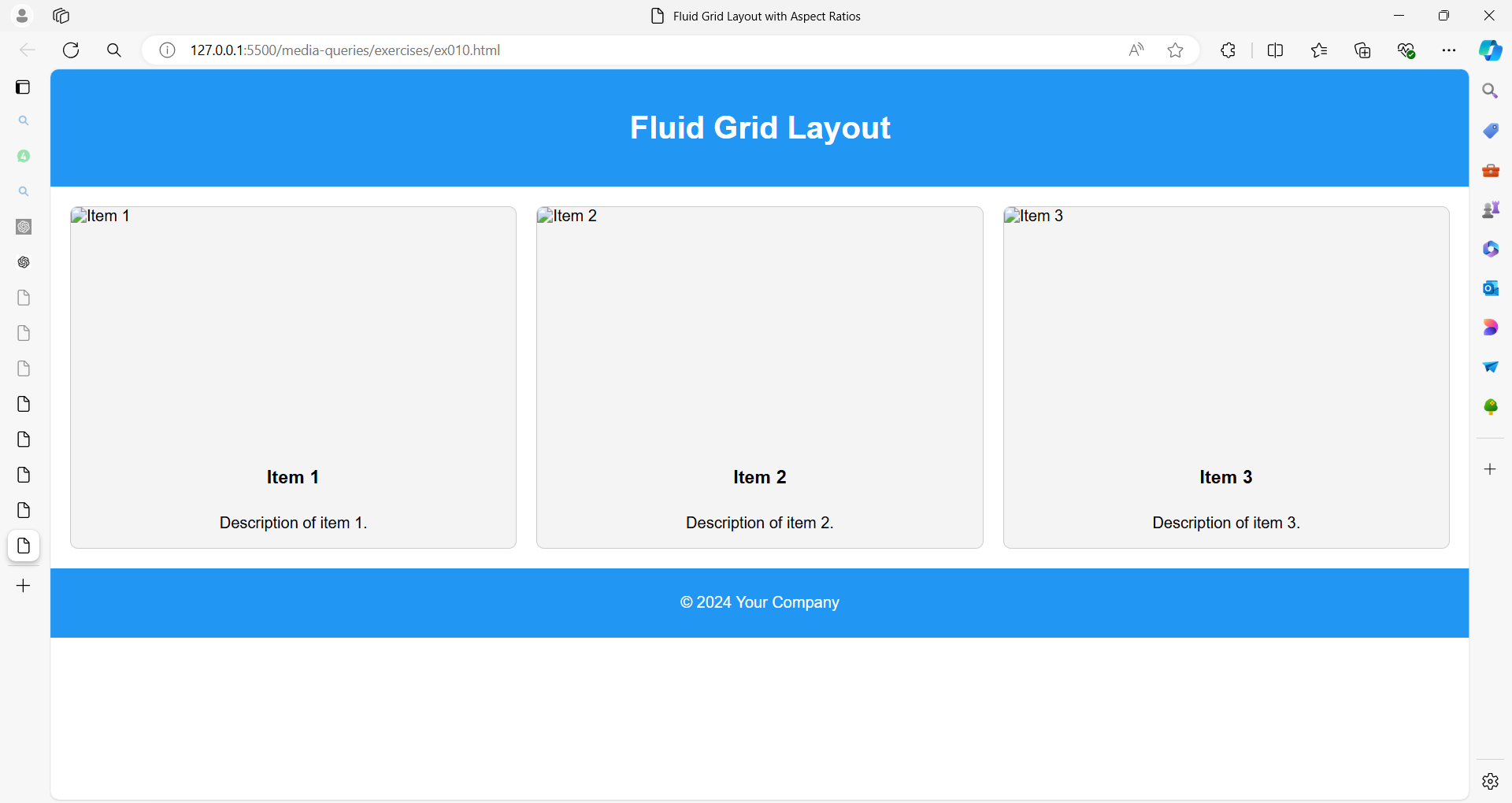
**Objective**: Build a fluid grid layout with aspect ratio control for images that maintain a consistent aspect ratio across different screen sizes.

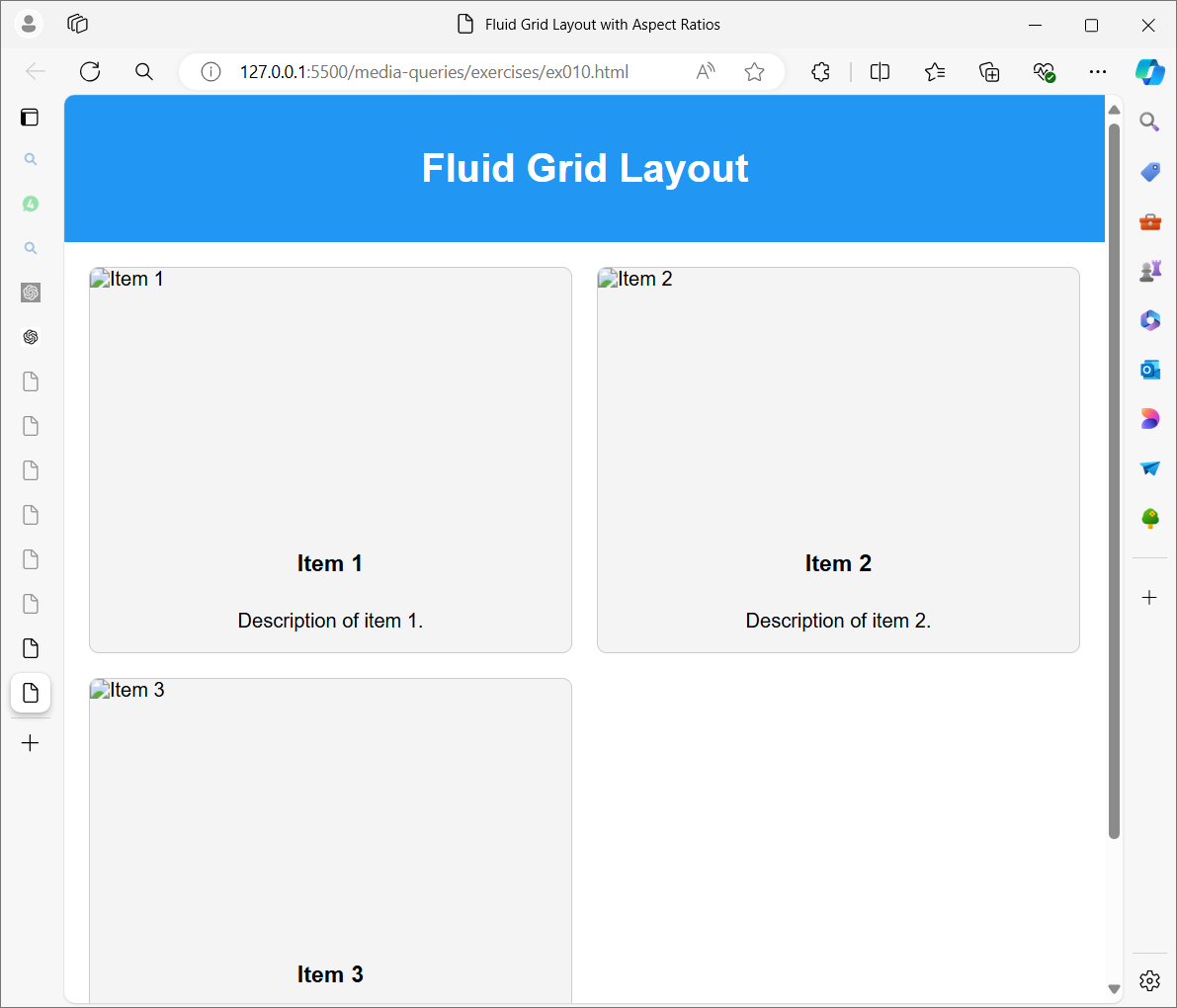
**Requirements**:

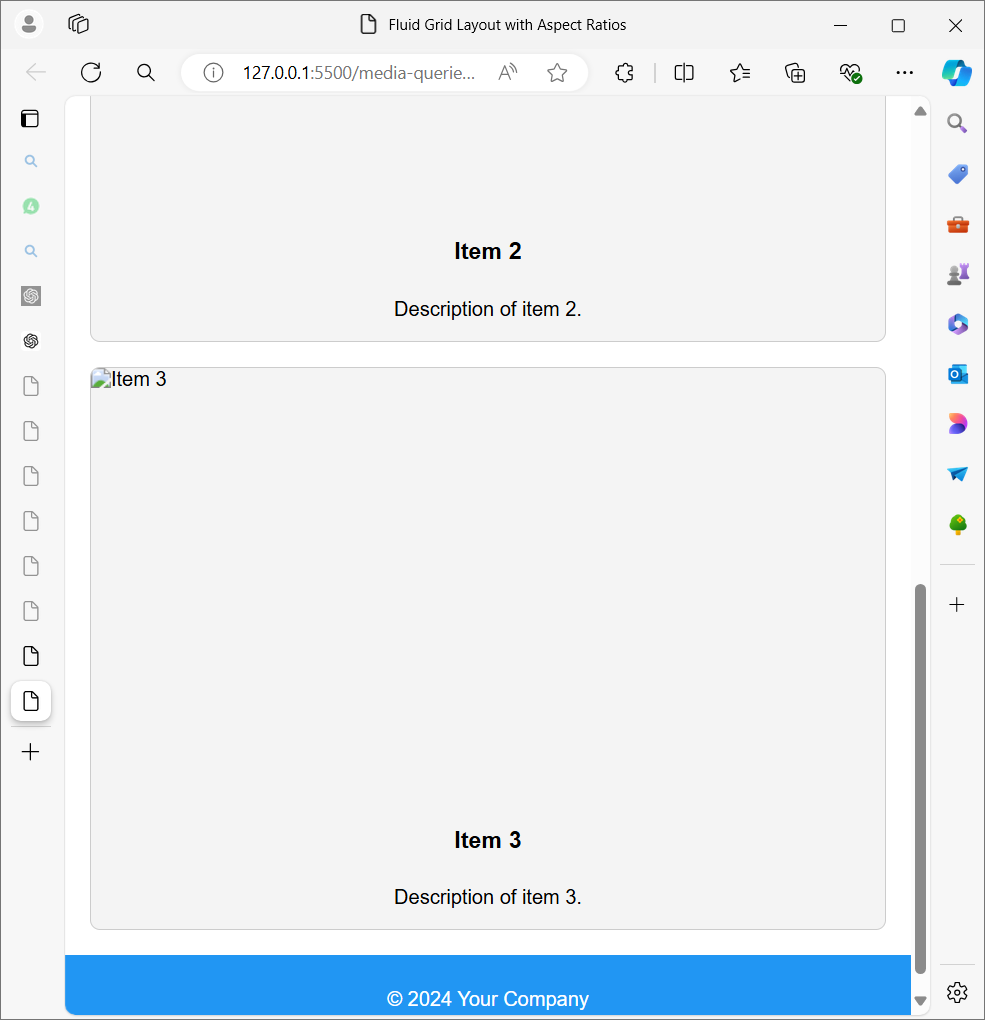
1. **Header**: Includes navigation and branding.
2. **Content Grid**: Displays items in a grid with controlled aspect ratios.
3. **Footer**: Contains copyright and additional links.
4. **Responsive Behavior**:
   * **Desktop (min-width: 1024px)**: Display items in a 3-column grid.
   * **Tablet (min-width: 768px and max-width: 1023px)**: Display items in a 2-column grid.
   * **Mobile (max-width: 767px)**: Display items in a single-column grid.

**Hints**:

* Use aspect-ratio property for images to maintain consistent ratios.
* Utilize @media queries to adjust grid layout and item sizing.







**Solution**:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Fluid Grid Layout with Aspect Ratios</title>

<style>

body {

margin: 0;

font-family: Arial, sans-serif;

}

header {

background-color: #2196F3;

color: white;

padding: 20px;

text-align: center;

}

.grid {

display: grid;

gap: 20px;

padding: 20px;

}

.grid-item {

background-color: #f4f4f4;

border: 1px solid #ccc;

border-radius: 8px;

overflow: hidden;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

}

.grid-item img {

width: 100%;

height: auto;

aspect-ratio: 16 / 9; /\* Maintain a 16:9 aspect ratio \*/

object-fit: cover;

}

.grid-item h3 {

margin: 10px 0;

}

.grid-item p {

padding: 0 10px;

}

footer {

background-color: #2196F3;

color: white;

text-align: center;

padding: 10px;

}

@media (min-width: 1024px) {

.grid {

grid-template-columns: repeat(3, 1fr);

}

}

@media (min-width: 768px) and (max-width: 1023px) {

.grid {

grid-template-columns: repeat(2, 1fr);

}

}

@media (max-width: 767px) {

.grid {

grid-template-columns: 1fr;

}

}

</style>

</head>

<body>

<header>

<h1>Fluid Grid Layout</h1>

</header>

<div class="grid">

<div class="grid-item">

<img src="image1.jpg" alt="Item 1">

<h3>Item 1</h3>

<p>Description of item 1.</p>

</div>

<div class="grid-item">

<img src="image2.jpg" alt="Item 2">

<h3>Item 2</h3>

<p>Description of item 2.</p>

</div>

<div class="grid-item">

<img src="image3.jpg" alt="Item 3">

<h3>Item 3</h3>

<p>Description of item 3.</p>

</div>

<!-- More items as needed -->

</div>

<footer>

<p>&copy; 2024 Your Company</p>

</footer>

</body>

</html>

These exercises cover a range of scenarios from creating responsive layouts with grids and fluid images to managing aspect ratios and ensuring consistent display across devices. Feel free to modify the exercises to better fit specific needs or to add additional challenges.