**1. Fixed Positioning**

**Functionality**: Keeps certain elements (like the header, subnav, and sidebar) in place while the user scrolls.

**CSS Code**:

.header {

position: fixed;

top: 0;

left: 0;

right: 0;

z-index: 1000;

}

.subnav {

position: fixed;

top: 60px;

left: 0;

right: 0;

z-index: 999;

}

.left-sidebar {

position: fixed;

top: 110px;

bottom: 0;

z-index: 200;

}

**Purpose**: Ensures important navigation elements stay visible at all times.

**2. Flexbox for Alignment**

**Functionality**: Helps align and distribute space between elements in a flexible way.

**CSS Code**:

.header {

display: flex;

justify-content: space-between;

align-items: center;

}

.main-content {

display: flex;

}

**Purpose**: Makes it easy to arrange elements horizontally or vertically.

**3. Spacing with Margin and Padding**

**Functionality**: Adds space around and inside elements to improve readability and avoid clutter.

**CSS Code**:

.header {

padding: 10px 20px;

}

.content {

margin-left: 270px;

padding: 20px;

}

p {

margin-bottom: 15px;

}

**Purpose**: Ensures elements don’t overlap and have enough breathing room.

**4. Handling Overflowing Content**

**Functionality**: Manages content that doesn’t fit within its container by adding scrollbars.

**CSS Code**:

.subnav {

overflow-x: auto;

white-space: nowrap;

}

.left-sidebar {

overflow-y: auto;

}

**Purpose**: Ensures all content is accessible, even if it overflows.

**5. Controlling Element Stacking with Z-Index**

**Functionality**: Determines the order in which elements are stacked on top of each other.

**CSS Code**:

.header {

z-index: 1000;

}

.subnav {

z-index: 999;

}

.left-sidebar {

z-index: 200;

}

**Purpose**: Prevents elements from overlapping incorrectly.

**6. Defining Sizes with the Box Model**

**Functionality**: Sets the width, height, padding, and margins of elements.

**CSS Code**:

.left-sidebar {

width: 230px;

padding: 20px;

}

.content {

padding: 20px;

}

.search-bar {

padding: 5px;

}

**Purpose**: Ensures consistent sizing and spacing across the page.

**7. Adding Depth with Box Shadows**

**Functionality**: Creates a subtle shadow effect to make elements stand out.

**CSS Code**:

.content {

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

**Purpose**: Adds visual depth and separation between elements.

**8. Rounding Corners with Border Radius**

**Functionality**: Softens the edges of elements for a modern look.

**CSS Code**:

.content {

border-radius: 8px;

}

.search-bar {

border-radius: 5px;

}

.search-button {

border-radius: 5px;

}

**Purpose**: Makes elements like buttons and containers look more polished.

**9. Hiding Scrollbars for a Clean Look**

**Functionality**: Removes scrollbars while still allowing scrolling.

**CSS Code**:

.subnav {

scrollbar-width: none;

-ms-overflow-style: none;

}

.subnav::-webkit-scrollbar {

display: none;

}

**Purpose**: Keeps the design clean and minimal.

**10. Making the Layout Responsive**

**Functionality**: Ensures the layout works well on different screen sizes.

**CSS Code**:

.left-sidebar {

height: calc(100vh - 110px);

}

.content {

margin-left: 270px;

}

**Purpose**: Ensures the page looks good on both desktop and mobile devices.

**11. Adding Interactivity with Hover Effects**

**Functionality**: Changes the appearance of elements when the user hovers over them.

**CSS Code**:

.topnav a:hover {

text-decoration: underline;

}

.subnav a:hover {

background-color: #1c211feb;

}

.content button:hover {

background-color: #059862;

}

**Purpose**: Provides visual feedback to improve user experience.

**12. Separating Structure and Style**

**Functionality**: Uses semantic HTML for structure and CSS for styling.

**CSS Code**:

.header {

background-color: #ffffff;

}

.left-sidebar {

background-color: #e7e9eb;

}

.content {

background-color: white;

}

**Purpose**: Keeps the code clean, maintainable, and accessible.