Lab Exercise: Building a Web Layout with CSS Float Property

Objective:

In this lab, you will explore how to create a simple web layout using HTML and CSS. You will learn how to apply the float property for structuring the layout and how different elements (such as the header, navigation, main content, aside, and footer) fit into the layout. Additionally, you will understand how to manage floating elements and clear floats to ensure proper layout behavior.

Instructions:

Step 1: Create the HTML Structure

1. **Create an HTML File:**

Open your code editor and create a new file named float-layout-lab.html. Set up the basic HTML structure by copying and pasting the following code:

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

<head>
    <link rel="stylesheet" href="styles1.css">
    <link rel="stylesheet" href="styles2.css">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <header><img src="banner.png" alt="">
    <nav class="horizontalNavigation">
        <ul>
            <li><img src="menu.png" alt=""></li>
            <li><img src="menu.png" alt=""></li>
            <li><img src="menu.png" alt=""></li>
            <li><img src="menu.png" alt=""></li>
        </ul>
    </nav>
</header>
    <section>
        <aside>
            <img src="section.png" alt="">
        </aside>
        <main><img src="main.png" alt=""></main>
    </section>
    <footer>
        <nav id="locations">
            <h3>Locations</h3>
            <ul>
```

```
                <li>1</li>
                <li>2</li>
                <li>3</li>
            </ul>
        </nav>
        <nav id="company">
            <h3>Services</h3>
            <ul>
                <li>1</li>
                <li>2</li>
                <li>3</li>
            </ul>
        </nav>
        <nav id="company">
            <h3>Contact</h3>
            <ul>
                <li>1</li>
                <li>2</li>
                <li>3</li>
                <li>4</li>
            </ul>
        </nav>
    </footer>
</body>

</html>
```

This HTML structure includes:

- A header with an image.
- A horizontal navigation menu.
- A main section with an aside and a main content area.
- A footer with three sections for locations, services, and contact.

The styles1.css contains the basic page setups. The box-sizing attribute is very important to maintain the sizing.

Step 2: Page Layout

1. In the styles2.css, add the following rules for basic layout settings:

```
html {
   background-color: rgb(186, 136, 81);
}

body {
   margin: auto;
   width: 95%;
```

```
   max-width: 960px;
   min-width: 640px;
   font-family: Verdana, Geneva, Arial, sans-serif;
}

img {
   width: 100%;
   display: block;
}

aside>img {
   height: 630px;
}

main>img {
   height: 630px;
}

footer {
   background-color: rgb(94, 214, 154);
}
```

This defines the background color and sets the margins and width of the body to ensure the layout adjusts between 640px and 960px.

Step 3: Horizontal Navigation with Float

1. **Float the navigation items** to align them horizontally:

Add the following CSS for your navigation items:

```
av.horizontalNavigation>ul>li {
   float: left;
   border: 1px white solid;
   background-color: blue;
   width: 25%;
}

nav ul {
   list-style: none;
 }
```

- **Explanation:**
  - float: left; makes each <li> (list item) align horizontally next to each other.
  - width: 25%; ensures that four items fit across the page width (since 4 × 25% = 100%).

2. **Experiment:**
    - Change float: left; to float: right; and observe how the navigation moves to the right side.

Step 4: Creating a Sidebar (Aside) and Main Content Layout

1. **Float the Aside and Main Content** to create a two-column layout:

Add the following CSS for the aside and main content sections:

```
aside {
   float: left;
   width: 25%;
   height: 630px;
}

main {
   float: left;
   width: 75%;
   height: 630px;
}
```

- **Explanation:**
    - The aside section is floated to the left with a width of 25%, and the main section is also floated to the left with a width of 75%. This creates a two-column layout with the aside occupying 25% and the main content 75%.
2. **Experiment:**
    - Swap the widths of aside and main (make aside 75% and main 25%) and see how the layout adjusts.

Step 5: Clearing Floats

1. **Fix any layout issues caused by floating elements.**

Add a clearfix for sections to ensure that the layout of floated elements doesn't affect other parts of the page:

```
footer::after {
   clear: both;
   content: '';
   display: block;
}
```

- **Explanation:**
    - Without the clearfix, floating elements might cause content below them (like the footer) to be misaligned or pushed up.

Step 6: Footer with Multiple Navigation Sections

1. **Float footer sections** to align different sections in the footer:

```css
footer>nav#locations {
   float: left;
   width: 40%;
}

footer>nav#company {
   float: left;
   width: 30%;
   /* border: 2px red dashed; */
}
```

- **Explanation:**
  - Each navigation section in the footer is floated left and given a specific width. The clearfix is applied to ensure proper layout below the footer.

Step 7: Experiment and Explore

1. **Test different float behaviors**:
   - Change the float: left; to float: right; for the aside or main sections and observe how the layout shifts.
   - Remove the float from any section and notice how the layout collapses or changes.

Final Layout Overview:

- **Navigation bar:** Horizontal layout using float: left.
- **Main content:** Two-column layout (aside and main) using float: left.
- **Footer:** Multiple sections floated horizontally with clearfix to manage the layout.

By the end of this lab, you'll have a solid understanding of how to use CSS float to structure web page layouts and how to manage issues like float clearing using the clear property.