1. **Describe the main differences between the CSS Flex layout model and the CSS Grid layout model. When would you choose to use one over the other?**

**Solution:**

CSS Flexbox and CSS Grid are two different layout models in CSS, each with its own strengths and use cases. Here are the main differences between them and when you might choose one over the other:

1. **CSS Flexbox (Flexible Box Layout):**

* **One-Dimensional Layout:** Flexbox is primarily used for one-dimensional layouts, either horizontally (in a row) or vertically (in a column).
* **Content-Focused:** It's ideal for arranging and aligning items within a container, where the size of the items is variable and you want them to fill the available space efficiently.
* **No Grid Structure:** Flexbox does not rely on a grid structure. It's best for laying out elements without specifying explicit rows and columns.
* **Auto-Sizing:** Flex items can automatically adjust their size based on content and the available space, making it suitable for responsive design.

1. **CSS Grid Layout:**

* Two-Dimensional Layout: Grid is designed for two-dimensional layouts, allowing you to create both rows and columns. This is especially useful for creating complex layouts.
* **Grid Structure:** Grid layout is based on a grid system where you define rows and columns explicitly, and you can place items at specific grid locations.
* **Alignment Control:** It provides powerful alignment and placement control for items within the grid, allowing you to create complex, symmetrical designs.
* **Ideal for Layouts:** Grid is great for creating webpage layouts, including headers, footers, and content areas with consistent grid-based structures.

**When to Choose Flexbox:**

1. When working with one-dimensional layouts such as navigation menus, lists, or arranging elements in a single row or column.
2. When you need to distribute space within a container or align items along a single axis.
3. For auto-sizing and content-driven layouts, like flexible card grids or evenly spaced buttons in a container.

**When to Choose Grid:**

1. For complex, two-dimensional layouts like webpage structure, with headers, footers, sidebars, and content areas that require specific column and row definitions.
2. When you need precise control over item placement within a grid, including overlapping items or spanning items across multiple rows and columns.
3. For creating responsive layouts that adapt to different screen sizes and orientations.
4. In practice, it's common to use both Flexbox and Grid together within a web layout.
5. **Explain the role of the following key properties in the flexbox layout model**
6. **Justify-content**
7. **Align-items**
8. **Gap**
9. **Flex-direction**
10. **Flex-wrap**

**Solution:**

1. **Justify-content** -: the justify-content is a CSS property used in the Flexbox layout model to control the horizontal alignment of flex items within a flex container. It determines how the available space is distributed between and around the flex items along the main axis of the flex container.

The justify-content property accepts several values:

1. **Flex-start (default**): This value aligns the flex items at the start of the main axis. In a horizontal layout, it aligns items to the left, and in a vertical layout, it aligns items at the top.
2. **Flex-end:** It aligns the flex items at the end of the main axis. In a horizontal layout, it aligns items to the right, and in a vertical layout, it aligns items at the bottom.
3. **Centre:** This value canters the flex items along the main axis, distributing the available space equally on both sides.
4. **Space-between:** It evenly distributes the space between the flex items, with the first item aligned to the start and the last item aligned to the end. The space between the items is also equal.
5. **Align-items: -** The align-items property is a CSS property used in the Flexbox layout model to control the vertical alignment of flex items within a flex container. It determines how the items are positioned along the cross-axis, which is perpendicular to the main axis. The cross-axis alignment is especially useful when dealing with varying heights of items within a flex container.

The align-items property accepts several values:

1. **Stretch (default):** This is the default value, and it stretches the flex items along the cross-axis to fill the full height of the container. If you don't specify a fixed height for the flex items, they will stretch to match the tallest item's height.
2. **Flex-start:** This value aligns the flex items at the start of the cross-axis. In a horizontal layout, it aligns items at the top, and in a vertical layout, it aligns items at the left.
3. **Flex-end:** It aligns the flex items at the end of the cross-axis. In a horizontal layout, it aligns items at the bottom, and in a vertical layout, it aligns items at the right.
4. **Centre:** This value canters the flex items along the cross-axis, so they are vertically cantered within the container.
5. **Gap: -** The Flex Gap property in CSS is a shorthand property that sets the spacing between flex items within a flex container. It defines the size of the gap between the rows and columns in flexbox, grid, or multi-column layout. The Flex Gap property sets the values for the row-gap and column-gap properties. The row-gap is applied horizontally between the two flex lines, and the column-gap is applied vertically between items. If the column-gap is omitted, it's set to the same value as the row-gap.
6. **Flex-direction: -** The flex-direction property is a fundamental CSS property used in the Flexbox layout model to control the direction in which flex items are placed within a flex container. It defines the main axis and the direction in which flex items are laid out. This property is applied to the flex container.
7. **Flex-wrap: -** In CSS flexbox, the flex-wrap property controls how flex items are positioned within a flex container when they cannot fit on a single line.

The flex-wrap property:

* Specifies whether flex items are forced into a single line or wrapped onto multiple lines
* Sets the direction that lines are stacked

By default, flex items will try to fit in a single line. When there is not enough space, they will overflow the container or be shrunk to fit.

1. Write the code to centre a div using CSS Flexbox.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Document</title>

<style>

\*{

margin: 0px;

padding: 0px;

}

body{

height: 100vh;

width: 100vw;

display: flex;

justify-content: center;

align-items: center;

}

.container{

background-color: blue;

width: 300px;

height: 300px;

}

</style>

</head>

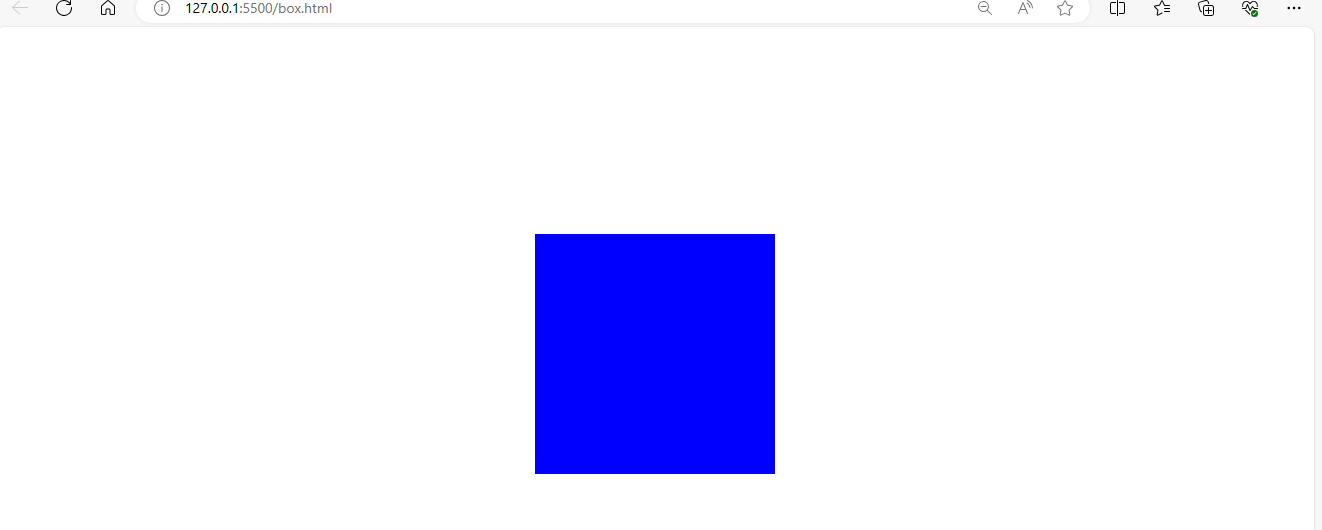
<body>

<div class="container"></div>

</body>

</html>

Output:



1. A client of yours wants to add a pricing section on their website to showcase their newly introduced premium plans.

You have to build the pricing section for their business. They have provided you with the figma design for the same.

Solution:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Pricing Section</title>

<style>

body, p,h2{

align-items: center;

padding: auto;

margin: auto;

font-family: sans-serif;

}

.pricing-section{

min-height:100vh;

padding-top: 100px;

display: flex;

justify-content: center;

gap: 20px;

background-color: #F4F2FD;

box-sizing: border-box;

}

.card{

width: 250px;

height: fit-content;

background-color: white;

border-radius: 7px;

padding-top: 50px;

padding-left: 35px;

}

.plan-name{

font-size: 35px;

}

.pricing{

font-size: 36px;

font-weight: 100;

margin: 12px 0;

}

.features{

display: flex;

align-items: center;

}

.subscribe-now-button-container{

justify-content: center;

margin-left: -35px;

}

.subscribe-now-button{

background-color: #202842;

color: white;

border-radius: 24px;

font-size: 18px;

padding: 10px 35px;

margin: 30px auto;

border-color: transparent;

}

.dark{

background-color: #202842;

color: white;

}

.dark-btn{

background-color: white;

color: black;

}

</style>

</head>

<body>

<section class="pricing-section">

<div class="card">

<h2 class="plan-name">Free Plan</h2>

<p class="plan-for">For personal</p>

<h2 class="pricing">$0</h2>

<p class="features"><span> <img src="/light\_tick.png" alt=""></span>1 Users</p>

<p class="features"><span> <img src="/light\_tick.png" alt=""></span>4 Web mails</p>

<p class="features"><span> <img src="/light\_tick.png" alt=""></span>Responsive Website</p>

<p class="features"><span> <img src="/light\_tick.png" alt=""></span>Free

SSL</p>

<div class="subscribe-now-button-container">

<button class="subscribe-now-button">Subscribe now</button>

</div>

</div>

<div class="card dark">

<h2 class="plan-name">Basic Plan</h2>

<p class="plan-for">For small business</p>

<h2 class="pricing">$29</h2>

<p class="features"><span> <img src="/dark\_tick.png" alt="tick"></span>4 Users</p>

<p class="features"><span> <img src="/dark\_tick.png" alt="tick"></span>10 Web mails</p>

<p class="features"><span> <img src="/dark\_tick.png" alt="tick"></span>Responsive Website</p>

<p class="features"><span> <img src="/dark\_tick.png" alt="tick"></span>Free SSL</p>

<div class="subscribe-now-button-container">

<button class="subscribe-now-button dark-btn">Subscribe now</button>

</div>

</div>

<div class="card">

<h2 class="plan-name">Pro Plan</h2>

<p class="plan-for">For enterprise</p>

<h2 class="pricing">$49</h2>

<p class="features"><span> <img src="/light\_tick.png" alt="tick"></span>Unlimited users</p>

<p class="features"><span> <img src="/light\_tick.png" alt="tick"></span>Unlimited Web mails</p>

<p class="features"><span> <img src="/light\_tick.png" alt="tick"></span>Responsive Website</p>

<p class="features"><span> <img src="/light\_tick.png" alt="tick"></span>Free SSL + SEO</p>

<div class="subscribe-now-button-container">

<button class="subscribe-now-button">Subscribe now</button>

</div>

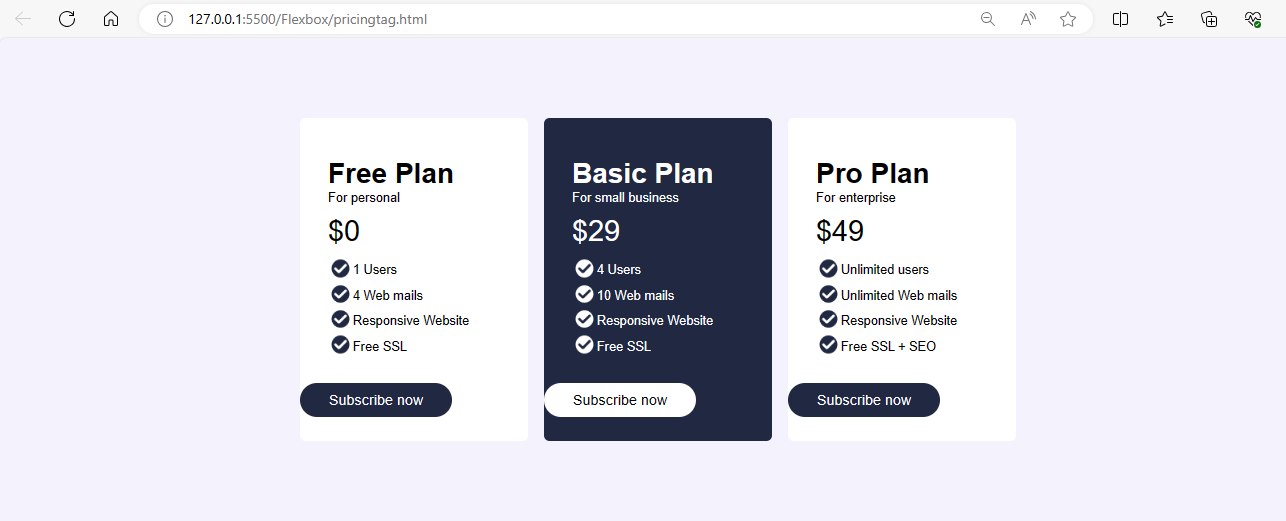
</div>

</section>

</body>

</html>

Output:



1. build a clone of the IRCTC Ticket booking page.

Code: <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>IRCTC duplicate page</title>

<link rel="stylesheet" href="./index.css">

</head>

<body>

<div class="full-container">

<section class="left">

<nav>

<div class="logos-container">

<img src="/Flexbox/Irctc image/IRCTCLogo.png" alt="IRCTC">

<img src="/Flexbox/Irctc image/indianRailwaysLogo.png" alt="indian Railways">

</div>

<div class="navigation-container">

<ul>

<li>Ask Disha</li>

<li>Contact us</li>

<li>Alerts</li>

</ul>

<ul>

<li>

<button class="selected">Login</button>

</li>

<li>

<button>Register</button>

</li>

<li>

<button>Agent Login</button>

</li>

</ul>

</div>

</nav>

<main>

<div>

<h2 class="main-title">Indian Railways</h2>

<p class="sub-main-title">

<span class="green-text">Safety</span>

<span class="orange-text">Security</span>

<span class="blue-text">Punctuality</span>

</p>

</div>

<div class="choose-container">

<button class="selected"><img src="/Flexbox/Irctc image/ticket\_icon.png" alt=""> BOOK Ticket</button>

<button><img src="/Flexbox/Irctc image/tick\_icon.png" alt=""> PNR Status</button>

<button><img src="/Flexbox/Irctc image/chart\_icon.png" alt=""> Charts / Vacancy</button>

</div>

<div class="journey-details-container">

<div class="journey-details-section-container">

<div class="card-element">

<div class="card-element-image-container">

<img src="/Flexbox/Irctc image/departure\_train\_icon.png" alt="">

</div>

<div class="select-option">

<div class="dropdown">

<span>From</span>

<img src="/Flexbox/Irctc image/expand\_more\_icon.png" alt="">

</div>

<div class="choose-location">

<h2>Bangalore</h2>

<span>KSR - Bangalore</span>

</div>

</div>

</div>

<div class="center-from-to-arrow">

<img src="/Flexbox/Irctc image/to\_from\_arrow.png" alt="">

</div>

<div class="card-element">

<div class="card-element-image-container">

<img src="/Flexbox/Irctc image/destination\_train\_icon.png" alt="">

</div>

<div class="select-option">

<div class="dropdown">

<span>To</span>

<img src="/Flexbox/Irctc image/expand\_more\_icon.png" alt="">

</div>

<div class="choose-location">

<h2>Kasaragod</h2>

<span>KSD - Kasaragod</span>

</div>

</div>

</div>

</div>

<div class="journey-details-section-container">

<div class="card-element">

<div class="card-element-image-container">

<img src="/Flexbox/Irctc image/calander-icon.png" alt="">

</div>

<div class="select-option">

<div class="dropdown">

<span>Date</span>

<img src="/Flexbox/Irctc image/expand\_more\_icon.png" alt="">

</div>

<div class="choose-location">

<h2>01 Aug 23</h2>

<span>Tuesday</span>

</div>

</div>

</div>

<div class="card-element ">

<div class="card-element-image-container">

<img src="/Flexbox/Irctc image/threedot-icon.png" alt="">

</div>

<div class="select-option">

<div class="dropdown">

<span>Class</span>

<img src="/Flexbox/Irctc image/expand\_more\_icon.png" alt="">

</div>

<div class="choose-location">

<h2>3A</h2>

<span>AC 3 tier</span>

</div>

</div>

</div>

<div class="card-element">

<div class="card-element-image-container">

<img src="/Flexbox/Irctc image/widgets\_icon.png" alt="">

</div>

<div class="select-option">

<div class="dropdown">

<span>Seat Type</span>

<img src="/Flexbox/Irctc image/expand\_more\_icon.png" alt="">

</div>

<div class="choose-location">

<h2>TATKAL</h2>

<span>AC 3 Tier</span>

</div>

</div>

</div>

</div>

</div>

<button class="search-button">Search Train <img src="/Flexbox/Irctc image/right-arrow-icon.png" alt=""></button>

</main>

</section>

<section class="right">

<img src="/Flexbox/Irctc image/heroImage.png" alt="">

</section>

</div>

</body>

</html>

Output:

