Question 1: **What is CSS Flexbox, and how is it useful for layout design? Explain the terms flex-container and flex-item.**

* **CSS Flexbox** (Flexible Box Layout) is a powerful layout model in CSS designed to help distribute space and align items within a container, even when their sizes are unknown or dynamic. It is especially useful for creating one-dimensional layouts (either in a **row** or **column**) with consistent spacing, alignment, and responsiveness.

**How Flexbox is Useful for Layout Design**

* **Easier alignment**: Items can be easily centered or aligned vertically/horizontally without using complex hacks.
* **Flexible sizing**: Items can grow or shrink automatically to fill available space.
* **Responsive design**: Flexbox adapts well to different screen sizes and resolutions.
* **Ordering**: Items can be rearranged visually (without changing HTML structure) using the order property.
* **Spacing and distribution**: Provides powerful properties to control gaps and spacing between elements.

**Key Terms**

**1. Flex Container**

A **flex-container** is the parent element that holds flex items and has the display: flex; or display: inline-flex; CSS property applied.

**Example:**

css

*.container {*

*display: flex;*

*}*

**Behavior:**

* All direct child elements become **flex items**.
* You can control layout with properties like flex-direction, justify-content, and align-items.

**2. Flex Item**

A **flex-item** is a direct child of a flex container. These items are laid out according to the rules of the Flexbox model.

**Example:**

html

*<div class="container">*

*<div class="item1">Item 1</div>*

*<div class="item2">Item 2</div>*

*</div>*

**CSS:**

css

*.container {*

*display: flex;*

*}*

*.item1, .item2 {*

*flex: 1; /\* each item takes equal space \*/*

*}*

Question 2: **Describe the properties justify-content, align-items, and flex-direction used in Flexbox.**

* In **CSS Flexbox**, the properties justify-content, align-items, and flex-direction are key to controlling the **alignment, direction, and spacing** of flex items inside a flex container.

**1. flex-direction**

This property defines the **main axis** along which the flex items are laid out.

**Values:**

* row *(default)*: Items are placed from left to right.
* row-reverse: Items are placed from right to left.
* column: Items are placed from top to bottom.
* column-reverse: Items are placed from bottom to top.

**Example:**

css

*.container {*

*display: flex;*

*flex-direction: row; /\* horizontal layout \*/*

*}*

**2. justify-content**

This property controls the **alignment of flex items along the main axis** (which is defined by flex-direction).

**Values:**

* flex-start: Items are aligned at the start of the main axis.
* flex-end: Items are aligned at the end of the main axis.
* center: Items are centered along the main axis.
* space-between: Items are evenly distributed with no space at the ends.
* space-around: Items are evenly distributed with space on both sides.
* space-evenly: Items are evenly distributed with equal space between all.

**Example:**

css

*.container {*

*display: flex;*

*justify-content: center; /\* centers items horizontally if flex-direction is row \*/*

*}*

**3. align-items**

This property aligns flex items along the **cross axis** (perpendicular to the main axis).

**Values:**

* stretch *(default)*: Items stretch to fill the container.
* flex-start: Items align at the start of the cross axis.
* flex-end: Items align at the end of the cross axis.
* center: Items are centered on the cross axis.
* baseline: Items align based on their text baseline.

**Example:**

css

*.container {*

*display: flex;*

*align-items: center; /\* vertically centers items if flex-direction is row \*/*

*}*

**Combined Example:**

css

*.container {*

*display: flex;*

*flex-direction: row;*

*justify-content: space-between;*

*align-items: center;*

*}*

This would create a horizontal layout (row) where:

* Items are spaced evenly with space between (space-between)
* Items are vertically centered (center on the cross axis)