Flexbox in CSS

Assignment.

Qn1. Describe the main differences between the CSS Flexbox layout model and the CSS Grid layout model. When would you choose to use one over the other?

Ans.

Main Differences Between Flexbox and Grid

Feature	Flexbox	Grid
Layout Type	One-dimensional (either row or column)	Two-dimensional (both rows and columns)
Alignment	Great for distributing space along a single axis	Ideal for precise control over both horizontal and vertical alignment
Item Placement	Items flow along a single axis, adjusting based on content size	Items can be explicitly placed in defined rows and columns
Responsiveness	Easier for flexible, content-driven layouts	More structured, making it easier to create complex layouts with fixed or responsive tracks
Explicit vs Implicit	Items are arranged dynamically based on available space	Items can be placed explicitly in specific grid cells

When to Use Flexbox vs. Grid

- Use Flexbox when:
    - You need to align elements along a single axis (e.g., navigation bars, form elements, buttons).
    - Items need to dynamically adjust in size (e.g., flexible lists or cards).
    - You require efficient space distribution, including wrapping and aligning items.
- Use Grid when:
    - You need a well-defined, two-dimensional layout (e.g., page templates, dashboards).
    - Items need to be positioned with precision in both rows and columns.
    - You want more control over alignment and spacing across multiple axes.

Can They Be Used Together?

Yes! You can use Grid for overall page layout and Flexbox for smaller components within the grid. For example, a grid-based layout can structure a webpage, while flexbox can be used inside a grid cell to align buttons within a card.

Qn2. Explain the role of the following key properties in the Flexbox layout mode

1.justify-content

2. align-items

3. gap

4. flex-direction

5. Flex-wrap

Ans.

Key Properties in the Flexbox Layout Model

1. `justify-content` (Main Axis Alignment)

- Controls how flex items are distributed along the main axis (horizontally in `row`, vertically in `column`).
- Common values:
    - `flex-start` (default) → Items align at the start.
    - `flex-end` → Items align at the end.
    - `center` → Items are centered.
    - `space-between` → Items are evenly spaced with no gaps at the edges.
    - `space-around` → Items have equal space around them.
    - `space-evenly` → Items have equal space between and at the edges.

2. `align-items` (Cross Axis Alignment)

- Aligns items along the cross axis (perpendicular to `flex-direction`).
- Common values:
    - `stretch` (default) → Items stretch to fill the container.
    - `flex-start` → Items align at the start.
    - `flex-end` → Items align at the end.
    - `center` → Items align in the middle.
    - `baseline` → Items align based on their text baseline.

3. `gap` (Spacing Between Flex Items)

- Adds space between flex items without affecting margins or padding.

Example:
display: flex;
gap: 20px;

- This creates a 20px gap between flex items.

4. `flex-direction` (Main Axis Control)

- Defines the direction of the main axis.
- Values:

    - `row` (default) → Left to right (or right to left in RTL languages).
    - `row-reverse` → Right to left.
    - `column` → Top to bottom.
    - `column-reverse` → Bottom to top.

5. `flex-wrap` (Handling Overflow)

- Controls whether flex items wrap onto multiple lines when they exceed the container width.
- Values:
    - `no wrap` (default) → Items stay in a single line, possibly overflowing.
    - `wrap` → Items wrap to the next line if needed.
    - `wrap-reverse` → Items wrap but in reverse order.

Example Combining These Properties

```
.container {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
  flex-wrap: wrap;
  gap: 15px;
}
```

This layout:

- Aligns items in a horizontal row (`flex-direction: row`).
- Spaces them evenly with no gap at the edges (`justify-content: space-between`).
- Centers items vertically (`align-items: center`).
- Allows items to wrap to the next line if needed (`flex-wrap: wrap`).
- Adds 15px of space between items (`gap: 15px`).

Qn3. Write the code to center a div using CSS Flexbox.

Ans. given by code Q3.ans.html .

Qn4. A client of yours wants to add a pricing section on their website to showcase their newly introduced premium plans.

Ans. Ans by code.