Flexbox Layout Activity: Understanding Flexbox Properties and Hands-On Practice

Objective:
This activity is designed to help learners solidify their understanding of flexbox properties, memorise key properties and their values, and then apply them in a practical implementation. The activity is divided into two parts: a conceptual part to reinforce understanding and a hands-on part for practising flexbox layout.

Additional Resources If Needed:

- HTML Display Property Explained: Block vs Inline vs Inline-Block with Examples
- CSS Flexbox Layout Explained: Easy Visual Examples and Flex Tutorial
- Flexbox Playground Codepen
- Flexbox Navbar Code Demo
- CSS Flexbox Layout Guide Written Lesson

Part 1: Conceptual Understanding and Memorisation

Task 1: Flexbox Property Definitions

Fill in the table below with definitions and explanations for each property related to flexbox. If you're unsure, refer to W3Schools for help.

Property	Definition/Explanation
`display: flex`	The display property is a CSS property that defines a flex container. When flex is specified for an element such as a , the Flexible Box Layout, more commonly known as Flexbox, is activated for the element and its direct children. An example of what Flexbox would do for an element is to display an unordered list in a horizontal layout rather than a vertical layout.
`justify-content: center`	This is another CSS property for the Flex container. It is used to control the alignment of items on a horizontal or main axis

	when the default direction is horizontal. The “center” specifies that all the items will be centered in the parent container with no gaps between items.
`flex-direction: row-reverse`	This is another CSS property for the Flex Container. When used, row-reverse reverses the order of items that display in a row. So, instead of 1,2,3,4,5 the order would be 5,4,3,2,1.
`flex-wrap: wrap`	Yet another CSS property for the Flex Container. By default, the flex container will try to keep all the items on one line. By specifying ‘wrap’ for the flex-wrap property, the items will wrap to the next line.
`align-items: center`	The Flexbox container property aligns its items within itself. When center is specified, the items are centered within the parent container.
`flex-grow`	This is a property for flex items. When used, the flex item will grow if necessary and take up available space.
`flex-shrink`	This is another property for flex items. This property determines how much an item can shrink relative to other items if there isn’t enough space.
`align-content`	This is a property for the flex container. It is used to control the alignment of items on a vertical axis or the cross axis. The various options align items to the flex-start, flex-end, center, stretch, space-between, and space-around.

Task 2: Flexbox Property Matching

Match the flexbox properties with their correct function or description:

Flexbox Property	Function/Description
`justify-content: space-between`	e) Space is distributed evenly between and around items in a flex container.

`align-items: flex-end`	a) Defines how items are aligned along the main axis.
`flex-direction: column`	b) Defines whether the flex container is a row or a column.
`flex-wrap: nowrap`	d) Items will not wrap, and overflow the container.
`flex-grow: 1`	c) Allows items to shrink or grow to fit the container.

Task 3: Reflection Questions

Answer the following questions to reflect on your understanding:

1. **What is the primary advantage of using flexbox for layouts compared to traditional methods like floats?**
    - **Answer:** Flexbox is a simpler and more powerful responsive method for providing flexible layouts. Using traditional methods like floats would require complex calculations and hacks. It is much easier to design UIs that can be viewed on a large screen but can be viewed equally well on a mobile phone screen.
2. **What does the `flex-direction` property control in a flex container, and how does it affect item alignment?**
    - **Answer:** This property establishes the main axis of the flexible item. The two main choices for this property are row and column. Using "row" will display items in a row, while "column" will display items in a column. Other choices for flex-direction include row-reverse, which displays items in a row but in a reverse order, and column-reverse, which displays items in a column but in a reverse order.
3. **What happens when you set `flex-wrap` to `wrap` in a flex container with many items?**
    - **Answer:** The items in a flex container with flex-wrap set to wrap will wrap items to the next line when the container shrinks and the items don’t have the space to stay on one line.

Part 2: Hands-On Flexbox Implementation

Scenario:

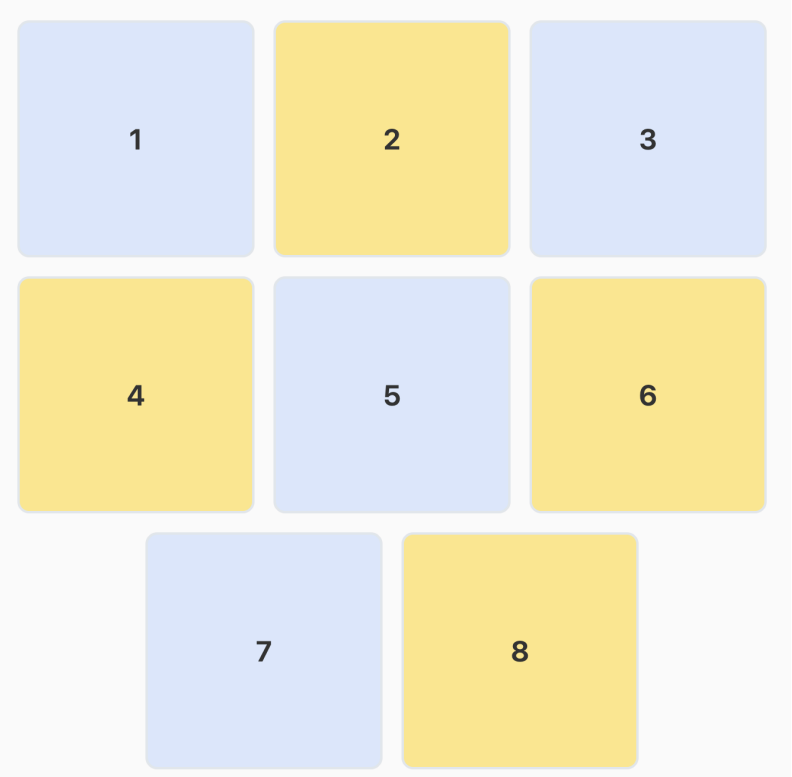
You have been tasked to create a **responsive image gallery** for a client. The client wants the images to be aligned horizontally on larger screens and wrap to the next line on smaller screens. They also want the items to be centered on the page.

Instructions:

1. **Create a container:**
    - Create a `div` with a class name of `container`. Set its display property to `flex` to turn it into a flex container.
2. **Add multiple items:**
    - Inside the container, add multiple `div` elements with the class `box`. Each `box` will represent an image (for now, you can just use background colors to differentiate them).
3. **Apply flexbox properties:**
    - Use flexbox properties to achieve the following layout:
        - The boxes should be horizontally aligned on larger screens.
        - The boxes should wrap to the next line if the screen size is too small.
        - The boxes should be centered within the container both horizontally and vertically.
        - Add margins and padding for spacing.
4. **Customize the box properties:**
    - Set a width and height for each box, and add some styles like background color, padding, and a border.
5. **Test responsiveness:**
    - Resize the browser window to see the flexbox wrapping behavior in action.

Output in full screen view:

Output when browser window shrinks and there is no enough space to display the whole line of boxes on the same row



Hints: Example Structure (No Example Code):

- **Container:** A div with the `display: flex` property.
- **Boxes:** Several div elements representing images (use background colors for now).
- **Flex Properties:**

- Use `justify-content`, `flex-wrap`, `flex-direction`, and `align-items` to achieve the desired layout.

Reflection:

1. **Check Your Work:**
   - Open the developer tools and inspect your container to see how the flexbox properties are applied. Use the **flexbox inspector** in the dev tools to visualize the alignment.
2. **Challenges:**
   - Did you encounter any difficulties when items started wrapping? How did you resolve them?
     i. I didn't encounter any difficulties when items started wrapping. I just specified 'flex-wrap: wrap;' and everything worked as required.
3. **Final Layout:**
   - Ensure that the boxes align correctly when the screen is resized. Check and everything works.

Completion:

Once you've completed the activity, review the layout, and experiment with different flexbox properties to see how they affect the layout. This activity will help solidify your understanding of flexbox before moving on to more complex layouts.