### **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](https://codepen.io/Mycloud055/pen/MYwwVmV)
* [CSS Flexbox Layout Explained: Easy Visual Examples and Flex Tutorial](https://codepen.io/Mycloud055/pen/OPVVvgR)
* [Flexbox Playground Codepen](https://codepen.io/enxaneta/full/adLPwv/)
* [Flexbox Navbar Code Demo](https://codepen.io/Mycloud055/pen/YPXPYOL)
* [CSS Flexbox Layout Guide Written Lesson](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)

### **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](https://www.w3schools.com) for help.

| **Property** | **Definition/Explanation** |
| --- | --- |
| display: flex | This changes the display type to a Flexbox instead of a Block. |
| justify-content: center | This constrains the information inside the box to align in the center of the horizontal axis. |
| flex-direction: row-reverse | This flips the display of the information so that the first piece starts on the right and the last piece is on the left. |
| flex-wrap: wrap | This allows the information to wrap around when the screen is made smaller |
| align-items: center | This aligns the boxes in the center of the cross axis. |
| flex-grow | This specifies how much the item will grow relative to the rest of the flexible items inside the same container. |
| flex-shrink | This specifies how much the item will shrink relative to the rest of the flexible items inside the same container. |
| align-content | This specifies how the flex lines are distributed along the cross axis in a flexbox container. |

#### Task 2: Flexbox Property Matching

Match the flexbox properties with their correct function or description:

| **Flexbox Property** | **Function/Description** |
| --- | --- |
| justify-content: space-between | a) Defines how items are aligned along the main axis. |
| align-items: flex-end | b) Defines whether the flex container is a row or a column. |
| flex-direction: column | c) Allows items to shrink or grow to fit the container. |
| flex-wrap: nowrap | d) Items will not wrap, and overflow the container. |
| flex-grow: 1 | e) Space is distributed evenly between and around items in a flex 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 handles alignment and spaces distribution in both horizontal and vertical directions, it also has powerful built in properties.**
2. **What does the flex-direction property control in a flex container, and how does it affect item alignment?**
   * **Answer: The flex-direction property specifies the direction of the flexible items, it can rearrange the ordering or turn a row into a colum or a colum into a row.**
3. **What happens when you set flex-wrap to wrap in a flex container with many items?**
   * **Answer: The items that do not fit on the first line will wrap around to the next line. If the window size is made smaller more items will wrap and if it is made bigger less items will wrap.**

### **Part 2: Hands-On Flexbox Implementation**

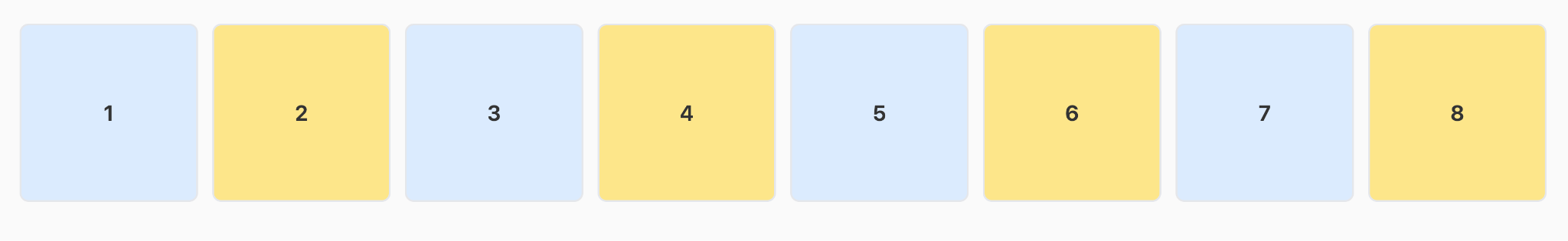
#### Scenario:

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.

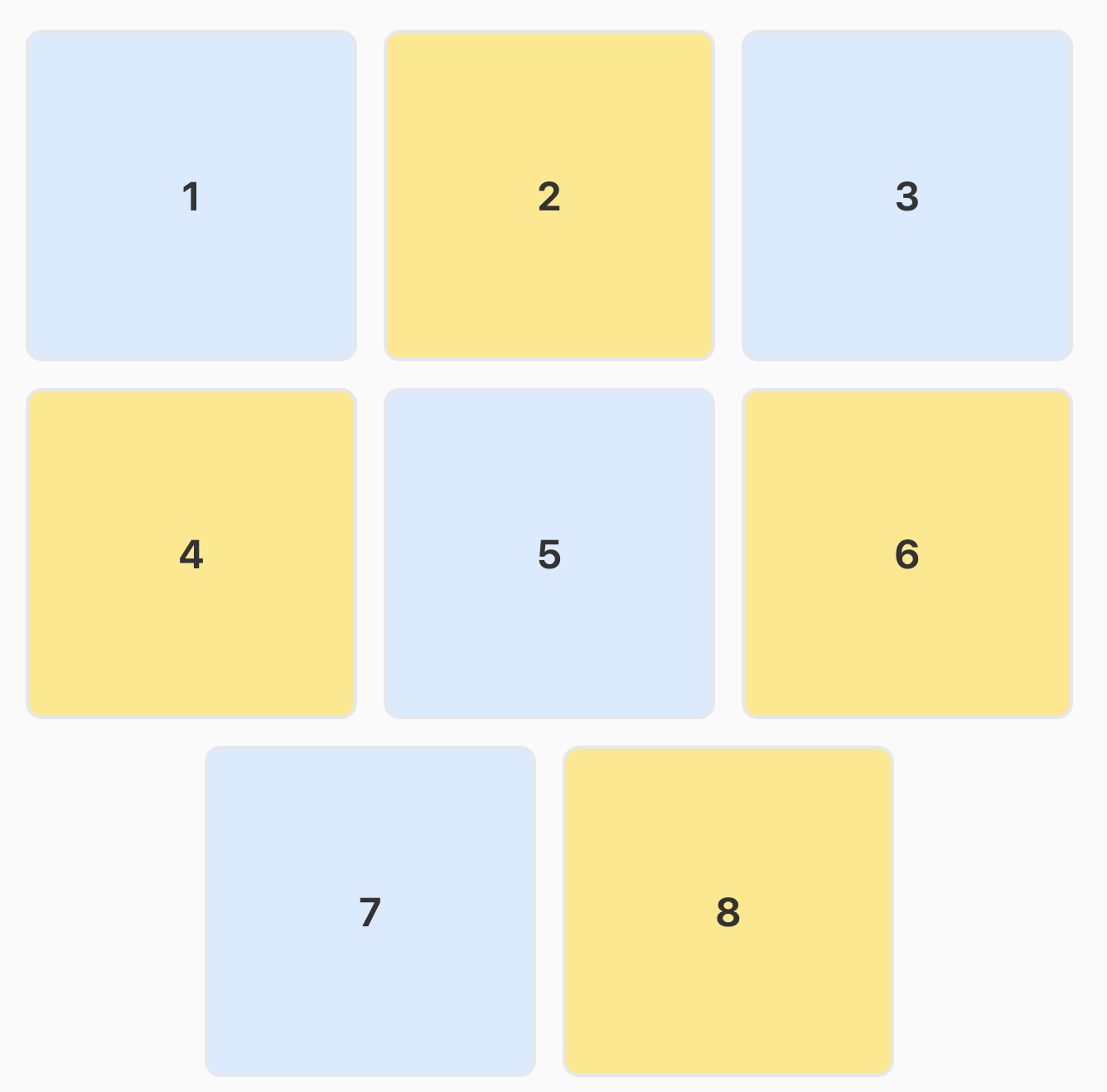
#### Instructions:

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? When they started wrapping I had to adjust my top and bottom margins so that they were equally spaced left/right and top/bottom.
3. **Final Layout:**
   * Ensure that the boxes align correctly when the screen is resized.

### **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.