### **Task 1: Control Flex Child Growth**

**Objective**: Use the flex-grow property to control how much space a flex child should take when compared to other children. - Create a div container with three child div elements. - Apply Flexbox to the container. - Set flex-grow: 1 for the first child, flex-grow: 2 for the second child, and flex-grow: 0 for the third child. - Observe how the second child grows twice as fast as the first one, and the third child does not grow.

<div class="flex-container">  
 <div class="flex-child">Child 1</div>  
 <div class="flex-child">Child 2</div>  
 <div class="flex-child">Child 3</div>  
</div>

### **Task 2: Control Flex Basis**

**Objective**: Explore the effect of flex-basis on determining the initial size of flex children. - Create a div container with three child div elements. - Assign different flex-basis values (e.g., 100px, 200px, auto) to the children and observe their starting sizes relative to each other.

<div class="flex-container">  
 <div class="flex-child">Child 1</div>  
 <div class="flex-child">Child 2</div>  
 <div class="flex-child">Child 3</div>  
</div>

### **Task 3: Flex Shrink Effect**

**Objective**: Investigate how flex children shrink when the flex container’s space is limited. - Create a div container with four child div elements of various widths. - Apply flex-shrink to see which elements shrink more when the container size is reduced.

<div class="flex-container">  
 <div class="flex-child">Child 1 (wider)</div>  
 <div class="flex-child">Child 2</div>  
 <div class="flex-child">Child 3</div>  
 <div class="flex-child">Child 4</div>  
</div>

### **Task 4: Change Flex Order**

**Objective**: Experiment with the order property to change the visual order of flex children. - Create a div container with five child div elements. - Change the visual order of the children by assigning different order values to each child.

<div class="flex-container">  
 <div class="flex-child">Child 1</div>  
 <div class="flex-child">Child 2</div>  
 <div class="flex-child">Child 3</div>  
 <div class="flex-child">Child 4</div>  
 <div class="flex-child">Child 5</div>  
</div>

### **Task 5: Align Self for Individual Flex Children**

**Objective**: Use align-self to override the container’s alignment for individual children. - Create a div container with three child div elements. - Apply align-self to one child to center it along the cross-axis while keeping others in the default alignment.

<div class="flex-container">  
 <div class="flex-child">Child 1</div>  
 <div class="flex-child">Child 2 (aligned center)</div>  
 <div class="flex-child">Child 3</div>  
</div>

### **Task 6: Set Different Flex Values (Shorthand)**

**Objective**: Use the flex shorthand property to set flex-grow, flex-shrink, and flex-basis in one declaration. - Create a div container with three child div elements. - Use the flex shorthand to assign different combinations of flex-grow, flex-shrink, and flex-basis values for each child (e.g., flex: 2 1 150px, flex: 1 0 100px). - Observe how the elements behave based on these properties when the container size changes.

<div class="flex-container">  
 <div class="flex-child">Child 1</div>  
 <div class="flex-child">Child 2</div>  
 <div class="flex-child">Child 3</div>  
</div>

### **Task 7: Test Flex Wrap Behavior**

**Objective**: Explore how flex items wrap onto multiple lines when space is constrained. - Create a div container with six child div elements. - Set the container’s width to a fixed value and apply flex-wrap to allow children to wrap onto new lines when there’s not enough space. - Experiment with different child widths to see how they rearrange when the container size is smaller.

<div class="flex-container">  
 <div class="flex-child">Child 1</div>  
 <div class="flex-child">Child 2</div>  
 <div class="flex-child">Child 3</div>  
 <div class="flex-child">Child 4</div>  
 <div class="flex-child">Child 5</div>  
 <div class="flex-child">Child 6</div>  
</div>

### **Task 8: Use align-items and align-content Together**

**Objective**: Understand the interaction between align-items (applies to flex children) and align-content (applies to the entire flex container). - Create a div container with five child div elements. - Apply different values for align-items (e.g., flex-start, center, stretch) to the container. - Experiment with align-content (e.g., space-between, center) to see how the entire group of children is aligned within the container, especially when the container height is greater than needed for the children.

<div class="flex-container">  
 <div class="flex-child">Child 1</div>  
 <div class="flex-child">Child 2</div>  
 <div class="flex-child">Child 3</div>  
 <div class="flex-child">Child 4</div>  
 <div class="flex-child">Child 5</div>  
</div>