1. Introduction to CSS Grid

CSS Grid is a powerful layout system that allows you to create complex and responsive web designs. It provides a two-dimensional grid system that can handle both rows and columns. With CSS Grid, you can control the positioning and sizing of elements within the grid, making it easier to create responsive designs.

2. Creating a Grid Container

To start using CSS Grid, you need to define a grid container by setting the display property to grid on an HTML element:

HTML:

```html
<div class="grid-container">
  <!-- Grid items go here -->
</div>
```

CSS:

```css
.grid-container {
  display: grid;
}
```

3. Defining Grid Columns and Rows

To define the columns and rows of your grid, you can use the grid-template-columns and grid-template-rows properties:

CSS:

```css
.grid-container {
  display: grid;
  grid-template-columns: 200px 200px 200px;
  grid-template-rows: 100px 100px 100px;
}
```

This code creates a grid with three columns and three rows, each with fixed sizes.

4. Adding Grid Items

To add items to the grid, simply place them inside the grid container:

HTML:

```html
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <!-- and so on -->
</div>
```

5. Positioning Grid Items

To position items within the grid, you can use the grid-column and grid-row properties:

CSS:

```css
.grid-item {
  grid-column: 1 / 3;
  grid-row: 1 / 3;
}
```

This code positions the first grid item to span two columns and two rows.

6. **Exercise 1: Basic Grid**

Create a simple 3x3 grid with equal-sized columns and rows. Place nine items in the grid, numbered 1 through 9.

7. **Using Fractional Units (fr)**

The fr unit represents a fraction of the available space in the grid container. This unit allows you to create responsive designs that adjust to the container's size:
CSS:

```css
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 1fr 1fr 1fr;
}
```

8. **Exercise 2: Responsive Grid**

Modify the grid from Exercise 1 to use the fr unit instead of fixed sizes. Test the responsiveness by resizing the browser window.

9. **Using Grid Gap**

To add space between grid items, you can use the grid-gap property (or its individual properties grid-row-gap and grid-column-gap):

CSS:

```css
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 1fr);
  grid-gap: 10px;
}
```

10. **Exercise 3: Grid with Gaps**

Add gaps to the grid from Exercise 2, and experiment with different gap sizes.

Once you've completed these exercises, you'll understand the basics of CSS Grid. To create more complex layouts, you can now explore more advanced grid features like auto-placement, grid-template-areas, and minmax() function.

More exercises:

Create the following pages from Figma:

https://www.figma.com/proto/gPibJxL1JLvL3qK0wQfxok/CSS-Grid?node-id=1-2&viewport=551%2C-133%2C1.2567064762115479&scaling=min-zoom

Helpful links:

https://css-tricks.com/snippets/css/complete-guide-grid/

https://codepen.io/tag/grid