CSS Grid is a powerful layout system for creating responsive and structured web designs. It allows developers to define rows, columns, and areas for precise control over layout and alignment. This guide introduces CSS Grid concepts, syntax, and practical examples.

What is CSS Grid?

CSS Grid Layout is a two-dimensional system for managing both rows and columns in a container. It's ideal for complex layouts, allowing for grid-based alignment of elements without the need for additional markup or complex calculations.

CSS Grid Syntax

The parent container (grid container) is defined with the property `display: grid;`. Inside the container, child elements (grid items) are aligned into rows and columns.

Example:

```
<div class="grid-container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
</div>
<style>
  .grid-container {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 10px;
  }
  .item {
```

```
    background-color: lightblue;
    border: 1px solid #ccc;
    text-align: center;
    padding: 20px;
    font-size: 18px;
  }
</style>
```

Explanation:

- `grid-template-columns: repeat(2, 1fr);`: Creates two equal-width columns.
- `gap: 10px;`: Adds spacing between grid items.

CSS Grid Properties

1. Grid Container Properties

- `grid-template-columns` and `grid-template-rows`: Define column and row sizes.
- `gap` or `grid-gap`: Adds spacing between rows and columns.
- `justify-items`: Aligns items horizontally within their grid area.
- `align-items`: Aligns items vertically within their grid area.

2. Grid Item Properties

- `grid-column-start` and `grid-column-end`: Define where an item starts and ends horizontally.
- `grid-row-start` and `grid-row-end`: Define where an item starts and ends vertically.
- `grid-area`: Assigns items to specific grid areas.

Examples

Example 1: Simple Grid Layout

```
<div class="grid-container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
</div>
<style>
  .grid-container {
```

```
    display: grid;
    grid-template-columns: 100px 100px 100px;
    gap: 5px;
  }
  .item {
    background: coral;
    text-align: center;
    padding: 10px;
  }
</style>
```

Example 2: Responsive Grid Using `auto-fit`

```
<div class="grid-container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
</div>
<style>
  .grid-container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
    gap: 10px;
  }
  .item {
    background: teal;
    text-align: center;
    color: white;
    padding: 20px;
  }
</style>
```

Exercise 1: Define a Grid Layout

1. Create a grid container with 3 rows and 4 columns.
2. Add a gap of 15px between items.
3. Center-align all items both horizontally and vertically.

Solution:

```
<div class="grid-container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
</div>
<style>
  .grid-container {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-template-rows: repeat(3, 100px);
    gap: 15px;
    justify-items: center;
    align-items: center;
  }
  .item {
    background: gray;
    color: white;
    padding: 10px;
    font-size: 20px;
  }
</style>
```

Multiple-Choice Questions

Question 1:

Which property is used to define the spacing between rows in a CSS Grid?

1. `gap`
2. `grid-gap`
3. `row-gap`
4. All of the above

Answer: 4. All of the above
Explanation: All the options (`gap`, `grid-gap`, and `row-gap`) can be used to define spacing between rows, though `grid-gap` is shorthand and less commonly used in modern syntax.

Question 2:

What does `grid-template-columns: repeat(3, 1fr);` mean?

1. Creates a grid with 1 fixed column.
2. Creates a grid with 3 columns of equal width.
3. Creates 3 rows of equal height.
4. None of the above.

Answer: 2. Creates a grid with 3 columns of equal width.
Explanation: `repeat(3, 1fr)` divides the available space into 3 columns of equal width.

Advanced Example: Grid Areas

```
<div class="grid-container">
  <div class="header">Header</div>
  <div class="sidebar">Sidebar</div>
  <div class="main">Main Content</div>
  <div class="footer">Footer</div>
</div>
<style>
  .grid-container {
    display: grid;
    grid-template-areas:
      "header header"
      "sidebar main"
      "footer footer";
    grid-template-columns: 1fr 3fr;
    gap: 10px;
  }
  .header {
    grid-area: header;
    background: lightblue;
  }
  .sidebar {
    grid-area: sidebar;
    background: lightgray;
  }
  .main {
    grid-area: main;
    background: white;
```

```
  }
  .footer {
    grid-area: footer;
    background: darkgray;
  }
</style>
```

Exercise 2: Recreate the Layout

Using `grid-template-areas`, create a layout with:

1. A header spanning 2 columns.
2. Two rows with a sidebar and main content.
3. A footer spanning 2 columns.

This guide introduces fundamental concepts, practical examples, and exercises to help learners build a solid foundation in CSS Grid. Let me know if you'd like further examples or a different focus!