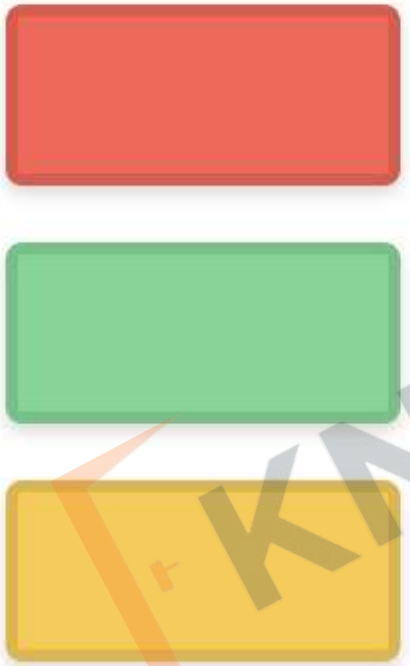
Grid

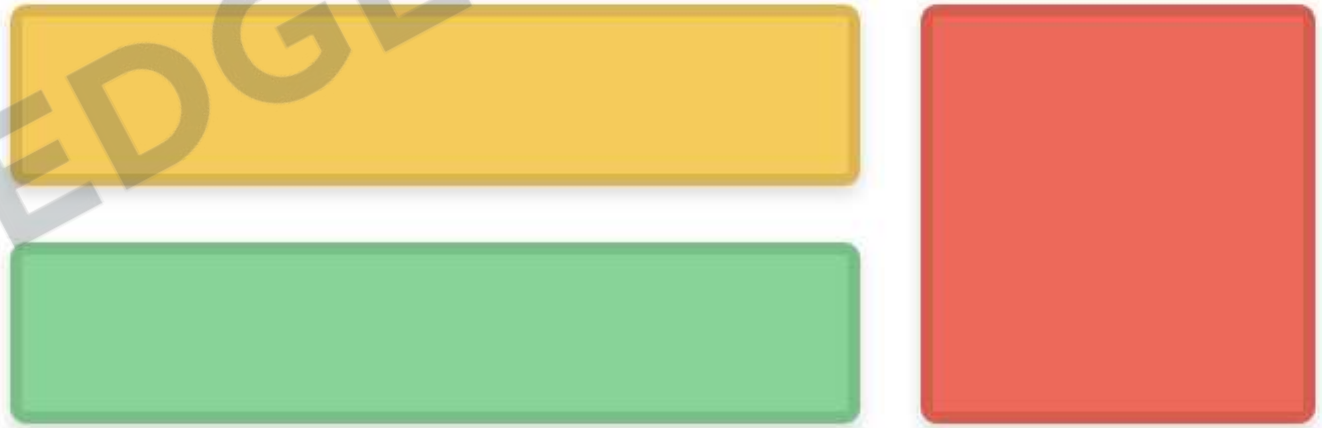
Flexbox

One-dimensional layout



Grid

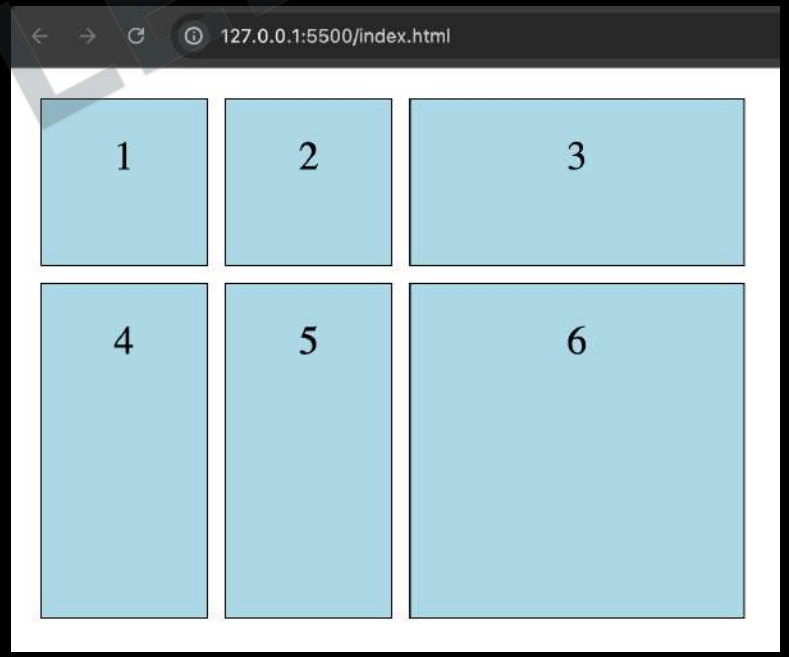
Multi-dimensional layout



Grid template rows & Columns

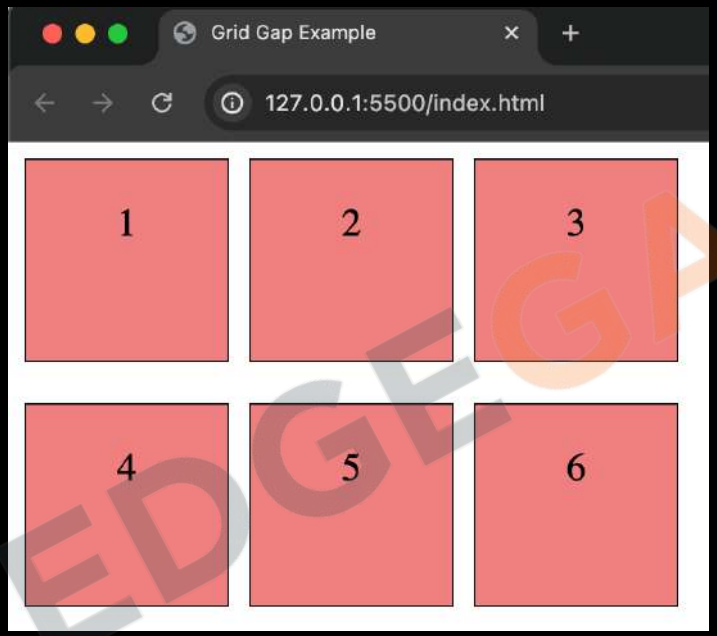
```html
<head>
    <title>Grid Template Example</title>
    <style>
        .grid-container {
            display: grid;
            grid-template-rows: 100px 200px;
            grid-template-columns: 100px 100px 200px;
            gap: 10px;
            padding: 10px;
        }
        .grid-item {
            background-color: lightblue;
            border: 1px solid #000;
            text-align: center;
            padding: 20px;
            font-size: 1.5em;
        }
    </style>
</head>
<body>
    <div class="grid-container">
        <div class="grid-item">1</div>
        <div class="grid-item">2</div>
        <div class="grid-item">3</div>
        <div class="grid-item">4</div>
        <div class="grid-item">5</div>
        <div class="grid-item">6</div>
    </div>
</body>
```

- CSS Grid Template Rows:
  - Defines the rows of the grid with specific sizes.
  - Accepts values such as pixels (px), percentages (%), fractions (fr), or the auto keyword.
  - Can create fixed or flexible grid layouts.
- Grid Template Columns:
  - Defines the columns of the grid with specific sizes.
  - Similar to rows, it accepts pixels, percentages, fractions, or the auto keyword.
  - Helps in designing the layout structure by specifying the width of each column.



Grid-Gap

```html
<head>
  <title>Grid Gap Example</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-rows: 100px 100px 100px;
      grid-template-columns: 100px 100px 100px;
      gap: 20px 10px; /* 20px row gap, 10px column gap */
    }
    .grid-item {
      background-color: lightcoral;
      border: 1px solid #000;
      text-align: center;
      padding: 20px;
      font-size: 1.2em;
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
  </div>
</body>
```



Definition:

- **grid-gap:** A shorthand property for setting the gaps (spaces) between rows and columns in a grid layout.
- **gap:** The modern shorthand for setting both row and column gaps, replacing grid-gap.

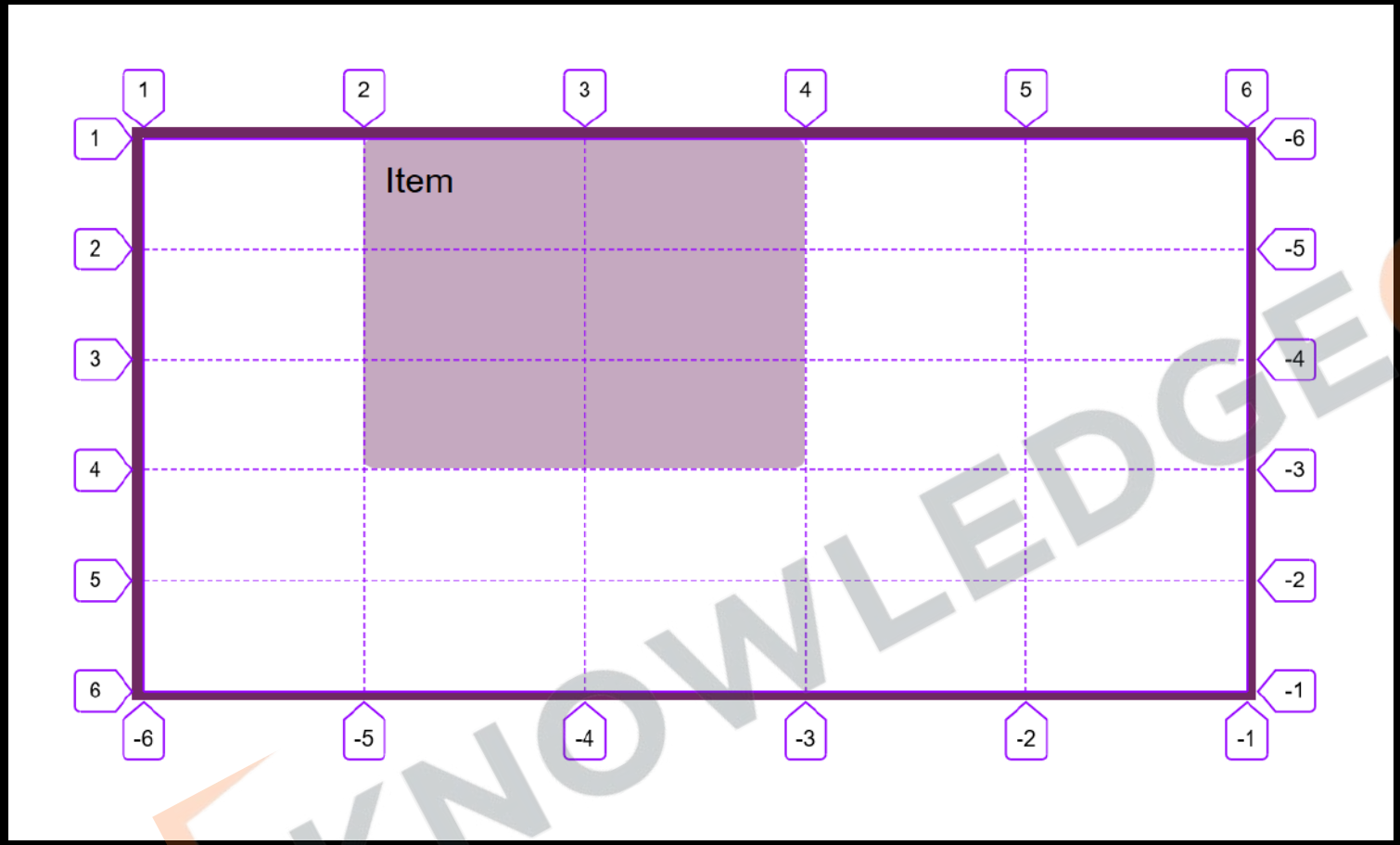
Components:

- **row-gap:** Specifies the size of the gap between grid rows.
- **column-gap:** Specifies the size of the gap between grid columns.

Syntax:

- gap: <row-gap> <column-gap>;
- If only one value is provided, it applies to both row and column gaps.

Grid Lines

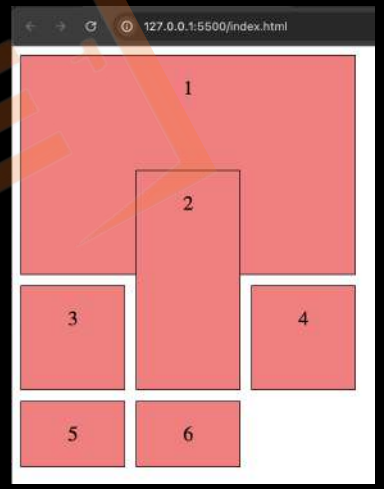


- Grid lines are the horizontal and vertical lines that divide the grid into cells, defined by the grid rows and columns.
- **Numbering**: Grid lines are numbered starting from 1, with the first line before the first row/column and the last line after the last row/column.
- **Usage**: Grid lines are used to place and span grid items precisely within the grid layout.
- Negative numbers can be used to reference grid lines, starting from the end of the grid. -1 refers to the last grid line, -2 to the second-last, and so on.

Grid row/column start/end

```css
.grid-container {
    display: grid;
    grid-template-rows: 100px 100px 100px;
    grid-template-columns: 100px 100px 100px;
    gap: 10px;
}
.grid-item {
    background-color: lightcoral;
    border: 1px solid #000;
    text-align: center;
    padding: 20px;
    font-size: 1.2em;
}
.item1 {
    grid-row-start: 1;
    /* Spans from row line 1 to row line 3 */
    grid-row-end: 3;
    grid-column-start: 1;
    /* Spans from column line 1 to column line 4 */
    grid-column-end: 4;
}
.item2 {
    grid-row-start: 2;
    /* Spans from row line 2 to row line 4 */
    grid-row-end: 4;
    grid-column-start: 2;
    /* Spans from column line 2 to column line 3 */
    grid-column-end: 3;
}
```

```html
<div class="grid-container">
    <div class="grid-item item1">1</div>
    <div class="grid-item item2">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
</div>
```

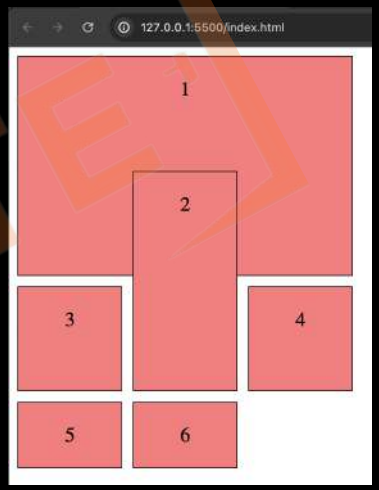


- Grid Row Start/End:
  - grid-row-start: Specifies the starting grid line for a grid item on the row axis.
  - grid-row-end: Specifies the ending grid line for a grid item on the row axis.
  - Usage: Defines the vertical position and span of a grid item.
- Grid Column Start/End:
  - grid-column-start: Specifies the starting grid line for a grid item on the column axis.
  - grid-column-end: Specifies the ending grid line for a grid item on the column axis.
  - Usage: Defines the horizontal position and span of a grid item.

Grid-row & Grid-column

```css
.grid-container {
    display: grid;
    grid-template-rows: 100px 100px 100px;
    grid-template-columns: 100px 100px 100px;
    gap: 10px;
}
.grid-item {
    background-color: lightcoral;
    border: 1px solid #000;
    text-align: center;
    padding: 20px;
    font-size: 1.2em;
}
.item1 {
  /* Equivalent to grid-row-start: 1;
   grid-row-end: 3; */
    grid-row: 1 / 3;
    /* Equivalent to grid-column-start:
     1; grid-column-end: 4; */
    grid-column: 1 / 4;
}
.item2 {
  /* Equivalent to grid-row-start: 2;
   grid-row-end: 4; */
    grid-row: 2 / 4;
    /* Equivalent to grid-column-start: 2;
     grid-column-end: 3; */
    grid-column: 2 / 3;
}
```

```html
<div class="grid-container">
    <div class="grid-item item1">1</div>
    <div class="grid-item item2">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
</div>
```

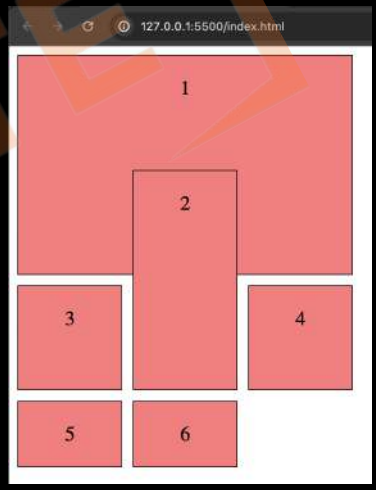


- **Grid Row:**
  - grid-row: A shorthand property for setting both grid-row-start and grid-row-end.
  - Syntax: grid-row: <start-line> / <end-line>;
  - Usage: Simplifies the declaration by combining the start and end lines of a grid item on the row axis.
- **Grid Column:**
  - grid-column: A shorthand property for setting both grid-column-start and grid-column-end.
  - Syntax: grid-column: <start-line> / <end-line>;
  - Usage: Simplifies the declaration by combining the start and end lines of a grid item on the column axis.

Span Keyword

```css
.grid-container {
    display: grid;
    grid-template-rows: 100px 100px 100px;
    grid-template-columns: 100px 100px 100px;
    gap: 10px;
}
.grid-item {
    background-color: lightcoral;
    border: 1px solid #000;
    text-align: center;
    padding: 20px;
    font-size: 1.2em;
}
.item1 {
/* Spans 2 rows starting from row line 1 */
    grid-row: 1 / span 2;
/* Spans 3 columns starting from column line 1 */
    grid-column: 1 / span 3;
}
.item2 {
/* Spans 2 rows starting from row line 2 */
    grid-row: 2 / span 2;
/* Spans 1 column starting from column line 2 */
    grid-column: 2 / span 1;
}
```

```html
<div class="grid-container">
    <div class="grid-item item1">1</div>
    <div class="grid-item item2">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
</div>
```

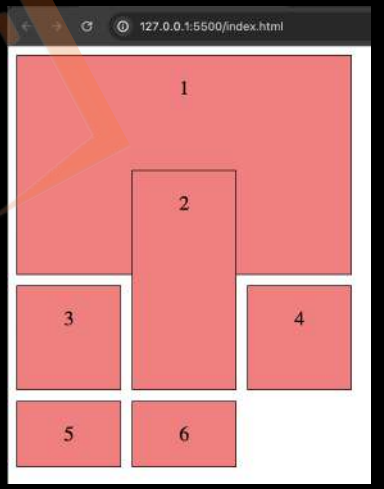


- Grid Row and Column with span:
  - span: A keyword used with grid-row and grid-column to span a grid item across multiple rows or columns.
  - Syntax: grid-row: <start-line> / span <number-of-rows>; and grid-column: <start-line> / span <number-of-columns>;
  - Usage: Provides a more intuitive way to specify the number of rows or columns a grid item should span, starting from a given line.

Grid-Area

```css
.grid-container {
    display: grid;
    grid-template-rows: 100px 100px 100px;
    grid-template-columns: 100px 100px 100px;
    gap: 10px;
}
.grid-item {
    background-color: lightcoral;
    border: 1px solid #000;
    text-align: center;
    padding: 20px;
    font-size: 1.2em;
}
.item1 {
/* Spans from row 1 to 3 and column 1 to 4 */
    grid-area: 1 / 1 / 3 / 4;
}
.item2 {
/* Spans from row 2 to last row and
column 2 to second last column */
    grid-area: 2 / 2 / -1 / -2;
}
```

```html
<div class="grid-container">
    <div class="grid-item item1">1</div>
    <div class="grid-item item2">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
</div>
```

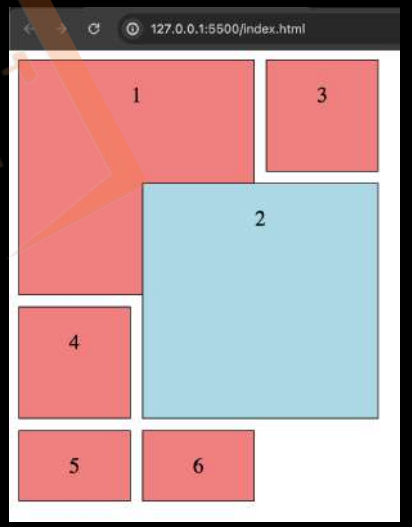


- Grid Area:
  - Definition: The grid-area property is used to define a grid item's size and location within a grid by specifying the start and end lines for both rows and columns.
  - Syntax: grid-area: <row-start> / <column-start> / <row-end> / <column-end>;
  - Usage: Combines grid-row-start, grid-row-end, grid-column-start, and grid-column-end into a single property for more concise code.

Layering with Z-index

```css
.grid-container {
    display: grid;
    grid-template-rows: 100px 100px 100px;
    grid-template-columns: 100px 100px 100px;
    gap: 10px;
}
.grid-item {
    background-color: lightcoral;
    border: 1px solid #000;
    text-align: center;
    padding: 20px;
    font-size: 1.2em;
}
.item1 {
/* Spans from row 1 to 3 and column 1 to 3 */
    grid-area: 1 / 1 / 3 / 3;
/* Lower z-index */
    z-index: 1;
}
.item2 {
/* Spans from row 2 to 4 and column 2 to 4 */
    grid-area: 2 / 2 / 4 / 4;
    z-index: 2; /* Higher z-index */
    background-color: lightblue;
}
```

```html
<div class="grid-container">
    <div class="grid-item item1">1</div>
    <div class="grid-item item2">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
</div>
```

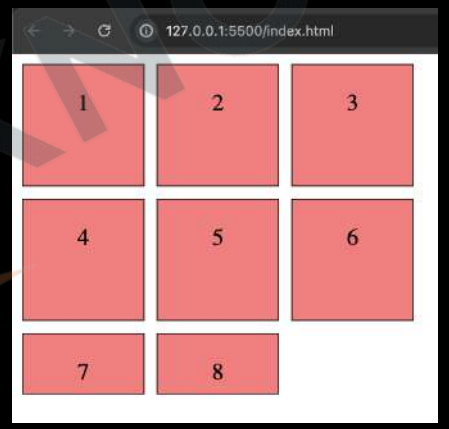


- Using z-index with CSS Grid
  - Definition: A CSS property that specifies the stack order of elements. Elements with higher z-index values are rendered on top of elements with lower values.
  - Usage in Grid: When working with CSS Grid, you can use z-index to control the layering of grid items. This is particularly useful when grid items overlap, either due to negative margins or explicitly overlapping grid areas.

Grid Overflow & Grid-Auto-Rows

```css
.grid-container {
  display: grid;
  grid-template-rows: 100px 100px;
  grid-template-columns: 100px 100px 100px;
  /* Automatically added rows will be 50px high */
  grid-auto-rows: 50px;
  /* Adds scrollbars if content overflows */
  overflow: auto;
  gap: 10px;
  /* Limits the height of the grid container */
  max-height: 350px;
}
.grid-item {
  background-color: lightcoral;
  border: 1px solid #000;
  text-align: center;
  padding: 20px;
  font-size: 1.2em;
}
```

```html
<body>
  <div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
    <div class="grid-item">7</div>
    <!-- Implicit row -->
    <div class="grid-item">8</div>
    <!-- Implicit row -->
  </div>
</body>
```



Grid Overflow:

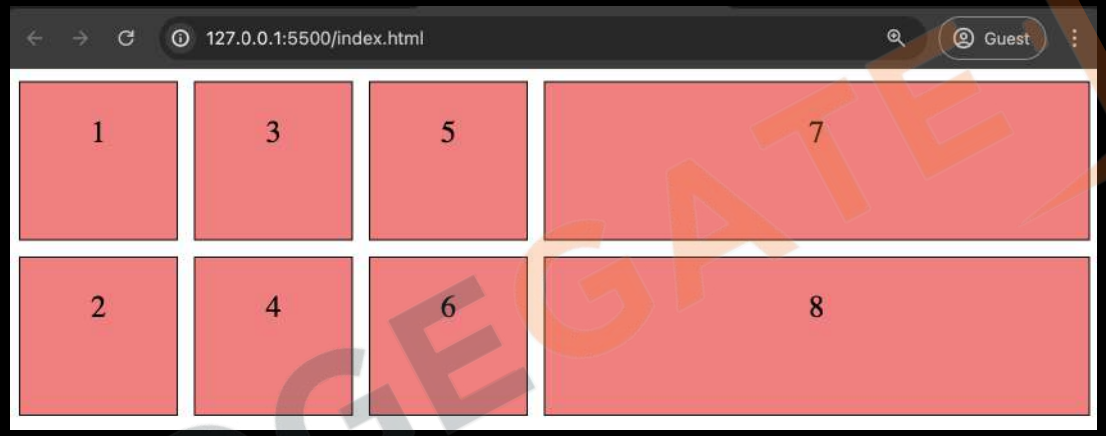
- Definition: Determines how content that overflows the boundaries
- Properties:
  - overflow: Controls both horizontal and vertical overflow.
  - overflow-x: Controls horizontal overflow.
  - overflow-y: Controls vertical overflow.
- Values:
  - visible: Default. Content is not clipped and may be rendered outside the container.
  - hidden: Content is clipped and not visible outside the container.
  - scroll: Content is clipped, but scrollbars are provided to view the hidden content.
  - auto: Similar to scroll, but scrollbars are only provided when necessary.

Grid-Auto-Rows:

- Definition: Defines the size of implicitly created rows in a grid layout when the number of items exceeds the explicitly defined grid structure.
- Usage: Ensures that additional rows created dynamically have a consistent size.

Grid-Auto-Flow

```html
<head>
  <title>Grid Auto Flow Example</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-rows: 100px 100px;
      grid-template-columns: 100px 100px 100px 100px;
      grid-auto-flow: column; /* Items will flow in columns */
      gap: 10px;
    }
    .grid-item {
      background-color: lightcoral;
      border: 1px solid #000;
      text-align: center;
      padding: 20px;
      font-size: 1.2em;
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
    <div class="grid-item">7</div>
    <div class="grid-item">8</div>
  </div>
</body>
```



grid-auto-flow: Controls how auto-placed items are inserted into the grid.

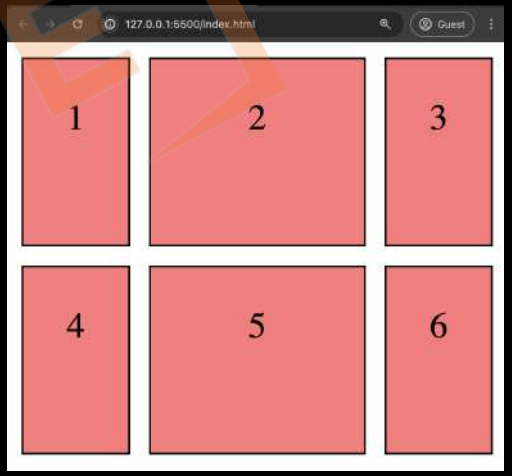
Values:

- **row:** Places items in rows, filling each row before moving to the next.
- **column:** Places items in columns, filling each column before moving to the next.
- **dense:** Tries to fill in gaps in the grid, ensuring items are placed in the smallest available spot.
- **row dense:** Combines row and dense, placing items in rows and trying to fill gaps.
- **column dense:** Combines column and dense, placing items in columns and trying to fill gaps.

Fractional Units

```css
<style>
  .grid-container {
    display: grid;
    grid-template-rows: 100px 100px;
    /* Columns will take 1/4, 1/2, and 1/4
     of the space respectively */
    grid-template-columns: 1fr 2fr 1fr;
    gap: 10px;
  }
  .grid-item {
    background-color: lightcoral;
    border: 1px solid #000;
    text-align: center;
    padding: 20px;
    font-size: 1.2em;
  }
</style>
```

```html
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
</div>
```



Fractional Units (fr): A flexible unit that represents a fraction of the available space in the grid container.

Usage: Allows for responsive and flexible grid layouts by dividing the remaining space among grid tracks.

min-max Function

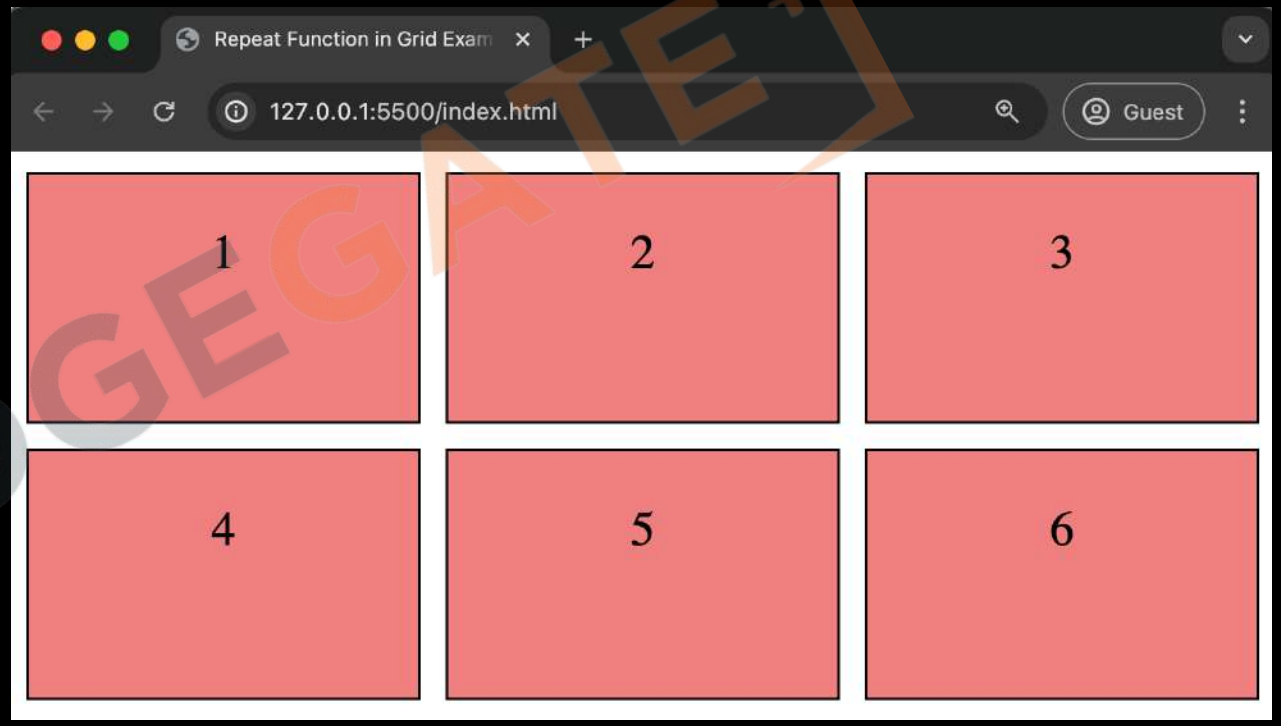
```
<style>
    .grid-container {
        display: grid;
        grid-template-rows: 100px 100px;
        grid-template-columns: 100px minmax(150px, 1fr) 100px;
        /* Second column has a minimum of 150px and a maximum of 1fr */
        gap: 10px;
    }
    .grid-item {
        background-color: lightcoral;
        border: 1px solid #000;
        text-align: center;
        padding: 20px;
        font-size: 1.2em;
    }
```

minmax(min, max): A CSS Grid function that defines a track size range, setting a minimum and maximum size for grid tracks (rows or columns).

Usage: Ensures grid tracks are flexible within specified constraints, improving layout responsiveness.

repeat Function

```
<style>
  .grid-container {
    display: grid;
    grid-template-rows: repeat(2, 100px);
    /* Creates 2 rows, each 100px high */
    grid-template-columns: repeat(3, 1fr);
    /* Creates 3 columns, each 1 fraction of the available space */
    gap: 10px;
  }
  .grid-item {
    background-color: lightcoral;
    border: 1px solid #000;
    text-align: center;
    padding: 20px;
    font-size: 1.2em;
  }
```

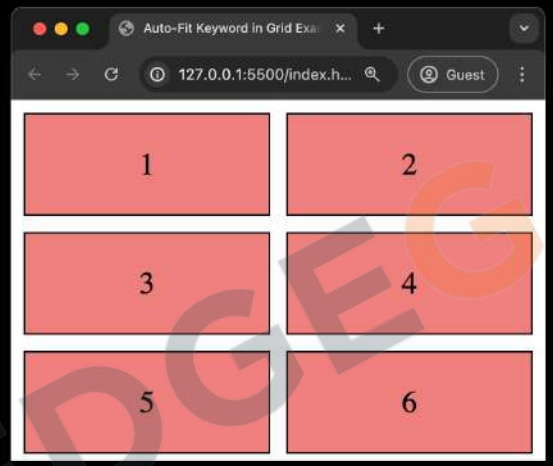


repeat(count, size): A CSS Grid function that allows you to repeat grid track definitions (rows or columns) a specified number of times.

- **Usage:** Simplifies the definition of repetitive grid structures, reducing code redundancy and improving readability.

auto-fit Keyword

```html
<head>
  <title>Auto-Fit Keyword in Grid Example</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
      /* Auto-fit as many 150px columns as possible */
      gap: 10px;
    }
    .grid-item {
      background-color: lightcoral;
      border: 1px solid #000;
      text-align: center;
      padding: 20px;
      font-size: 1.2em;
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
  </div>
</body>
```



auto-fit: A keyword used in the repeat() function to automatically fit as many columns (or rows) as possible into the available space. The columns (or rows) will expand to fill the space.

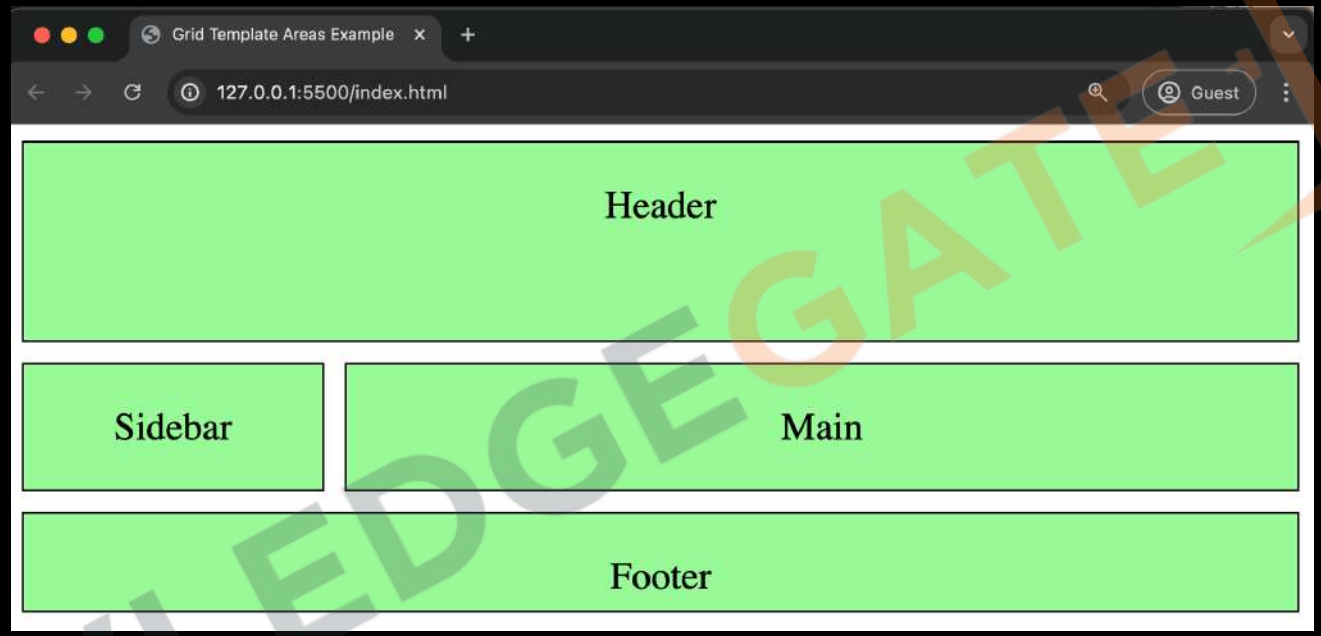
Advantages:

- Creates responsive layouts that adapt to different screen sizes.
- Automatically adjusts the number of columns (or rows) based on the available space, ensuring optimal use of space.

Grid-Template-Areas

```css
.grid-container {
    display: grid;
    grid-template-areas:
        "header header header"
        "sidebar main main"
        "footer footer footer";
    grid-template-rows: 100px 1fr 50px;
    grid-template-columns: 150px 1fr 1fr;
    gap: 10px;
}
.grid-item {
    background-color: palegreen;
    border: 1px solid #000;
    text-align: center;
    padding: 20px;
    font-size: 1.2em;
}
.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.main { grid-area: main; }
.footer { grid-area: footer; }
```

```html
<div class="grid-container">
  <div class="grid-item header">Header</div>
  <div class="grid-item sidebar">Sidebar</div>
  <div class="grid-item main">Main</div>
  <div class="grid-item footer">Footer</div>
</div>
```



grid-template-areas: A CSS Grid property that defines a grid template by specifying named grid areas. These named areas can then be used to place grid items within the grid layout.

Advantages:

- Simplifies the layout design by using named areas.
- Makes the grid layout more readable and maintainable.

Syntax:

- Define areas using strings, with each string representing a row in the grid.
- Use the same name for grid cells that should be merged into a single area.
- Use a period (.) to represent an empty cell.

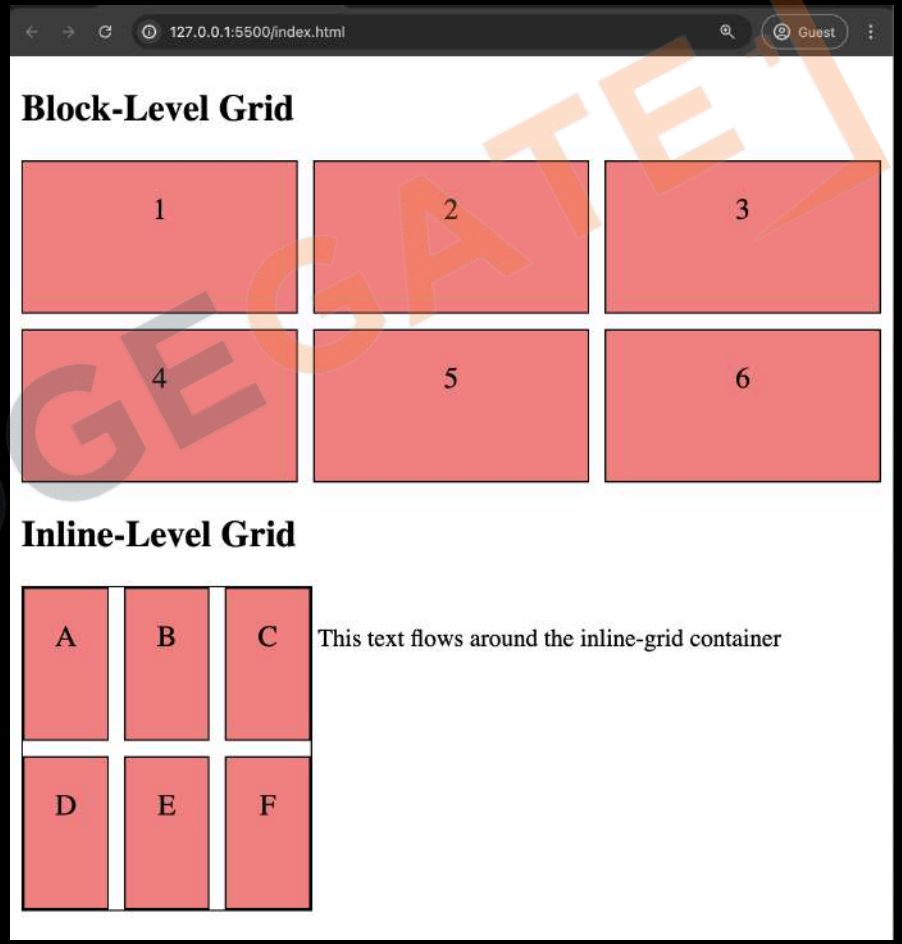
Grid vs Inline-Grid

```css
<style>
  .grid-container {
    display: grid;
    grid-template-rows: 100px 100px;
    grid-template-columns: 1fr 1fr 1fr;
    gap: 10px;
    margin-bottom: 20px;
  }
  .inline-grid-container {
    display: inline-grid;
    grid-template-rows: 100px 100px;
    grid-template-columns: 1fr 1fr 1fr;
    gap: 10px;
    border: 1px solid #000;
  }
  .grid-item {
    background-color: lightcoral;
    border: 1px solid #000;
    text-align: center;
    padding: 20px;
    font-size: 1.2em;
  }
</style>
```

```html
<body>
  <h2>Block-Level Grid</h2>
  <div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
  </div>

  <h2>Inline-Level Grid</h2>
  <div class="inline-grid-container">
    <div class="grid-item">A</div>
    <div class="grid-item">B</div>
    <div class="grid-item">C</div>
    <div class="grid-item">D</div>
    <div class="grid-item">E</div>
    <div class="grid-item">F</div>
  </div>

  <span>
    This text flows around the
    inline-grid container
  </span>
</body>
```

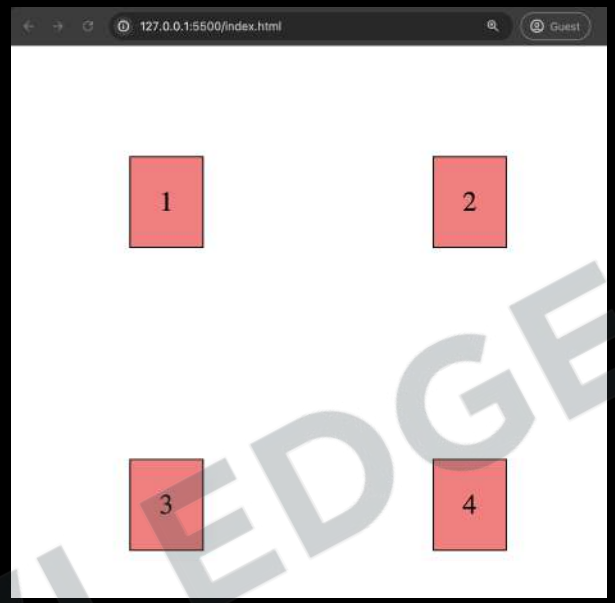


display: grid;: Establishes a block-level grid container.

display: inline-grid;: Establishes an inline-level grid container, making the grid container behave like an inline element while retaining grid layout capabilities.

Justify-Items & Align-Items

```html
<head>
  <title>Justify-Items and Align-Items Example</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-rows: 200px 200px;
      grid-template-columns: 200px 200px;
      gap: 10px;
      /* Centers items horizontally */
      justify-items: center;
      /* Centers items vertically */
      align-items: center;
    }
    .grid-item {
      background-color: lightcoral;
      border: 1px solid #000;
      text-align: center;
      padding: 20px;
      font-size: 1.2em;
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
  </div>
</body>
```



justify-items and align-items in CSS Grid

Definition:

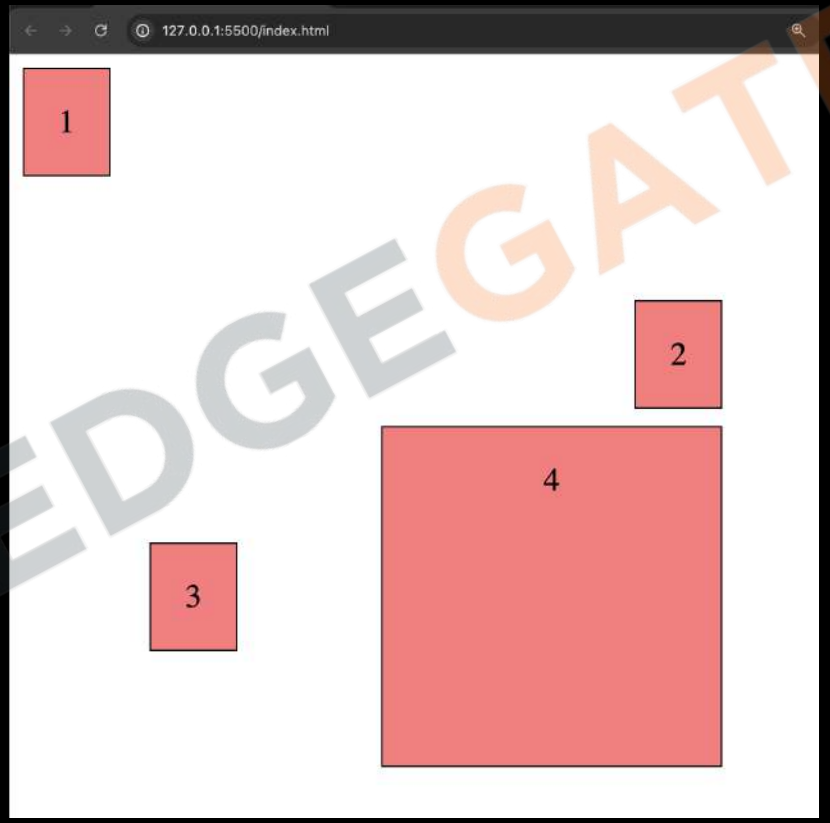
- **justify-items:** Aligns grid items along the inline (row) axis.
- **align-items:** Aligns grid items along the block (column) axis.

Values:

- **start:** Aligns items at the start of the grid area.
- **end:** Aligns items at the end of the grid area.
- **center:** Aligns items in the center of the grid area.
- **stretch:** Stretches items to fill the grid area (default).

Justify-Self & Align-Self

```css
.grid-container {
  display: grid;
  grid-template-rows: 200px 200px;
  grid-template-columns: 200px 200px;
  gap: 10px;
  justify-items: center; /* Centers all items horizontally by default */
  align-items: center; /* Centers all items vertically by default */
}
.grid-item {
  background-color: lightcoral;
  border: 1px solid #000;
  text-align: center;
  padding: 20px;
  font-size: 1.2em;
}
.item1 {
  justify-self: start; /* Aligns item 1 to the start horizontally */
  align-self: start; /* Aligns item 1 to the start vertically */
}
.item2 {
  justify-self: end; /* Aligns item 2 to the end horizontally */
  align-self: end; /* Aligns item 2 to the end vertically */
}
.item3 {
  justify-self: center; /* Aligns item 3 to the center horizontally */
  align-self: center; /* Aligns item 3 to the center vertically */
}
.item4 {
  justify-self: stretch; /* Stretches item 4 to fill horizontally */
  align-self: stretch; /* Stretches item 4 to fill vertically */
}
```



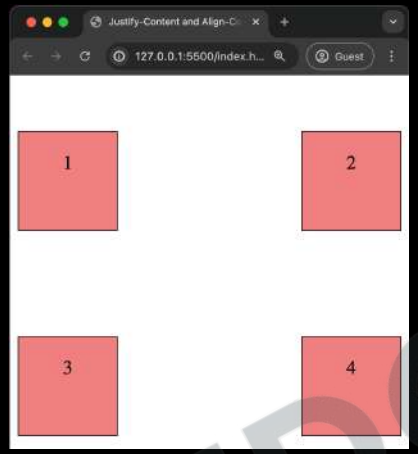

```html
<div class="grid-container">
  <div class="grid-item item1">1</div>
  <div class="grid-item item2">2</div>
  <div class="grid-item item3">3</div>
  <div class="grid-item item4">4</div>
</div>
```

justify-self and align-self in CSS Grid

- **justify-self:** Aligns a single grid item along the inline (row) axis.
- **align-self:** Aligns a single grid item along the block (column) axis.

Justify-Content & Align-Content

```html
<head>
  <title>Justify-Content and Align-Content Example</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-rows: 100px 100px;
      grid-template-columns: 100px 100px;
      gap: 10px;
      /* Distributes space between columns */
      justify-content: space-between;
      /* Distributes space around rows */
      align-content: space-around;
      /* Added height to demonstrate align-content */
      height: 400px;
    }
    .grid-item {
      background-color: lightcoral;
      border: 1px solid #000;
      text-align: center;
      padding: 20px;
      font-size: 1.2em;
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
  </div>
</body>
```



grid-template-areas: A CSS justify-content and align-content in CSS Grid

- justify-content: Aligns the entire grid along the inline (row) axis.
- align-content: Aligns the entire grid along the block (column) axis.

Values:

- start: Aligns the grid at the start of the container.
- end: Aligns the grid at the end of the container.
- center: Aligns the grid in the center of the container.
- stretch: Stretches the grid to fill the container.
- space-between: Distributes the grid items with space between them.
- space-around: Distributes the grid items with space around them.
- space-evenly: Distributes the grid items with equal space around them.

Practice Set

Grid

Instructions:

Below is an HTML structure for a simple grid layout. Using the specified CSS properties, create a responsive grid layout and apply basic transitions.

```
<!DOCTYPE html>
<html lang="en">
<head><title>Easy Grid Layout</title>
<style>
/* Your CSS code here */
</style>
</head>
<body>
<div class="container">
<div class="item item1">Item 1</div>
<div class="item item2">Item 2</div>
<div class="item item3">Item 3</div>
<div class="item item4">Item 4</div>
</div>
</body>
</html>
```

CSS Tasks:

- **Grid Template Rows & Columns:** Define the grid container with 2 rows and 2 columns, each taking up 1fr of space.
- **Grid Gap:** Set a gap of 10px between rows and columns.
- **Grid Item Placement:** Place each item in a separate cell.
- **Transitions:** Apply a transition to change the background color of .item when hovered.

Practice Set (Solution)

Grid

```
/* Grid Template Rows & Columns
*/
.container {
display: grid;
grid-template-rows: 1fr 1fr;
grid-template-columns: 1fr 1fr;
gap: 10px;
}

/* Grid Item Placement */
.item1 {
grid-row: 1 / 2;
grid-column: 1 / 2;
}

.item2 {
grid-row: 1 / 2;
grid-column: 2 / 3;
}

.item3 {
grid-row: 2 / 3;
grid-column: 1 / 2;
}

.item4 {
grid-row: 2 / 3;
grid-column: 2 / 3;
}

/* Transitions */
.item {
background-color: lightblue;
transition: background-color
0.5s ease-in-out;
}

.item:hover {
background-color: lightcoral;
}
```

Advanced Practice Set

Grid

Instructions:

Below is an HTML structure for a more complex grid layout with additional elements for using advanced grid properties. Using the specified CSS properties, create an advanced responsive grid layout.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Advanced Grid Layout</title>
<style>
/* Your CSS code here */
</style>
</head>
<body>
<div class="grid-container">
<div class="grid-item item1">Item 1</div>
<div class="grid-item item2">Item 2</div>
<div class="grid-item item3">Item 3</div>
<div class="grid-item item4">Item 4</div>
<div class="grid-item item5">Item 5</div>
<div class="grid-item item6">Item 6</div>
</div>
</body>
</html>
```

Grid Template Areas: Define a grid layout with the following template areas:

"item1 item1 item2”

"item3 item4 item4”

"item5 item5 item6"

Grid Gap: Set a gap of 15px between rows and columns.

Grid Item Placement: Place each item in its respective grid area.

Grid Span: Use the grid-column and grid-row properties with the span keyword to make items span multiple rows or columns.

Grid Overflow: Set the overflow property to handle content that overflows the boundaries of the grid items.

Advanced Practice Set (Solution)

Grid

```css
/* Grid Template Areas */
.grid-container {
  display: grid;
  grid-template-areas:
  "item1 item1 item2"
  "item3 item4 item4"
  "item5 item5 item6";
  gap: 15px;
  width: 100%;
  height: 100vh;
}
```

```css
/* Grid Item Placement */
.item1 {
  grid-area: item1;
}

.item2 {
  grid-area: item2;
}

.item3 {
  grid-area: item3;
}

.item4 {
  grid-area: item4;
}

.item5 {
  grid-area: item5;
}

.item6 {
  grid-area: item6;
}
```

```css
/* Grid Span */
.item1 {
  grid-column: 1 / span 2;
}
.item2 {
  grid-column: 3 / span 1;
  grid-row: 1 / span 2;
}
.item3 {
  grid-row: 2 / span 1;
}
.item4 {
  grid-column: 2 / span 2;
  grid-row: 2 / span 2;
}
.item5 {
  grid-column: 1 / span 2;
  grid-row: 3 / span 1;
}
.item6 {
  grid-column: 3 / span 1;
  grid-row: 3 / span 1;
}
```

```css
/* Grid Overflow */
.grid-item {
  background-color:
  lightblue;
  padding: 20px;
  text-align: center;
  overflow: auto;
}
```