AIMERZ.ai
Aim.Act.Achieve

NOTES

Flexbox & Grid

Flex Item Properties

Introduction

A flex container's immediate child is called flex items. A flex container (the large yellow area in the image) is an element in HTML whose display property's value is flex or inline-flex. The smaller boxes inside the yellow container are called flex items, and they are a direct offspring of a flex container.

2. Order

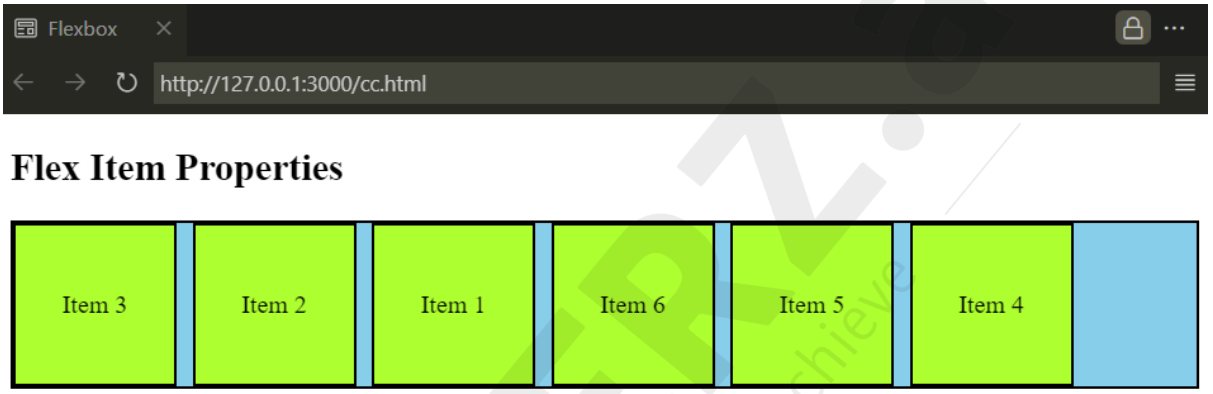
Order property in HTML is used to customise the order of items in a container, the syntax to write it is shown below.
Example:

Unset

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
            background-color:skyblue;
            gap: 10px;

        }
        .item {
            border:2px solid black;
            width: 100px;
            height: 100px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }
    </style>
</head>
<body>

    <h2>Flex Item Properties</h2>
    <div class="container" >
```

```
        <div class="item" style="order: 3">Item 1</div>
        <div class="item" style="order: 2">Item 2</div>
        <div class="item" style="order: 1">Item 3</div>
        <div class="item" style="order: 6">Item 4</div>
        <div class="item" style="order: 5">Item 5</div>
        <div class="item" style="order: 4">Item 6</div>
    </div>
</body>
</html>
```

OUTPUT:



3. Flex Grow

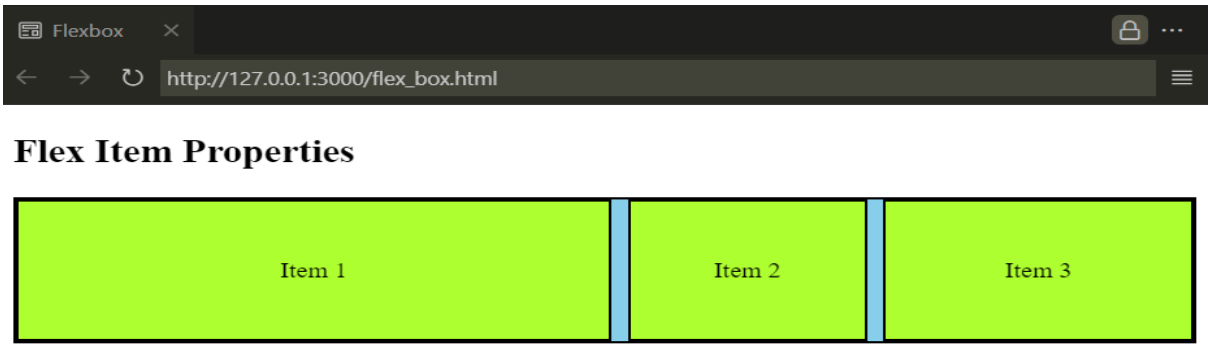
The **flex-grow** property in CSS is used within the Flexbox layout to control how much a flex item should grow relative to the other flex items in the same flex container. It defines the ability of a flex item to occupy the extra available space in the flex container.

Example:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
          <meta   name="viewport"   content="width=device-width,
initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
```

```
            display: flex;
            background-color:skyblue;
            gap: 10px;

        }
        .item {
            border:2px solid black;
            width: 100px;
            height: 100px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }
    </style>
</head>
<body>

    <h2>Flex Item Properties</h2>
    <div class="container" >
        <div class="item" style="flex-grow: 6">Item 1</div>
        <div class="item" style="flex-grow: 1">Item 2</div>
        <div class="item" style="flex-grow: 2">Item 3</div>
    </div>
</body>
</html>
```

OUTPUT:



3.1 Working of `flex-grow`

- If you have multiple flex items in a flex container, and one or more of them has a `flex-grow` value greater than `0`, they will stretch to fill the available space in proportion to their `flex-grow` values.

  Flow for previous example:

  - **`item1 has flex-grow:`** 6, which means it will grow to take up some extra space which is 6 units in total.
  - **`item2 has flex-grow:`** 1, meaning it will grow as much as 1 unit then `item1`.
  - **`item3 has flex-grow:`** 2 , so it will not grow at 2x units; it will only occupy the twice width necessary for content.

4. Flex Shrink

The **`flex-shrink`** property in CSS controls how much a flex item will shrink relative to the other flex items inside a flex container when there is not enough space for all items. It is part of the **flexbox layout** system and works when the items overflow the flex container.

4.1 How `flex-shrink` Works

Example:

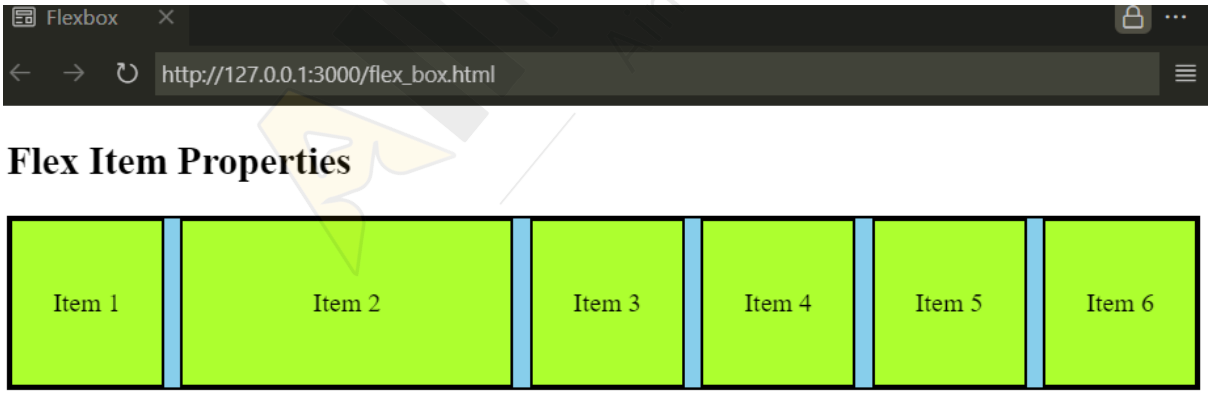
In the following example, the container is smaller than the total width of the items, and we use `flex-shrink` to control how each item shrinks.

HTML:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
            background-color:skyblue;
            gap: 10px;

        }
```

```
        .item {
            border:2px solid black;
            width: 200px;
            height: 100px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }
    </style>
</head>
<body>

    <h2>Flex Item Properties</h2>
    <div class="container" >
        <div class="item" style="flex-shrink: 1;">Item 1</div>
        <div class="item"style="flex-shrink: 0;">Item 2</div>
        <div class="item">Item 3</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
    </div>
</body>
</html>
```

OUTPUT:



4.2 Explanation of Flex Shrink

- **Item 1 has `flex-shrink`:** 1, so it will shrink at the default rate.
- **Item 2 has `flex-shrink`:** `0`, meaning it will not shrink at all, even if there is not enough space in the container.

5. Flex Basis

The **`flex-basis`** property in CSS is one of the three components of the `flex` shorthand property (the other two being `flex-grow` and `flex-shrink`). It defines the **initial size** of a flex item before any remaining space in the flex container is distributed according to the `flex-grow` or `flex-shrink` values. In simpler terms, `flex-basis` tells the flex item how much space it should take up *before* the flex-grow or flex-shrink rules are applied.

5.1 Features of `flex-basis` properties

- **Default value**: `auto` (the flex item sizes itself based on its content or any set width/height).
- **Can be set in any valid CSS size unit**, such as `px`, `%`, `em`, `rem`, or `auto`.
- **Takes precedence over `width/height`** in a flex context when determining the item's initial size, but only if both `flex-grow` and `flex-shrink` are zero.

Syntax:

```
Unset
flex-basis: <length> | auto;
```

- `<length>`: You can specify a size (e.g., `200px`, `30%`), and the flex item will be that size initially.
- `auto`: The flex item size will be based on its content or any applied width/height properties.

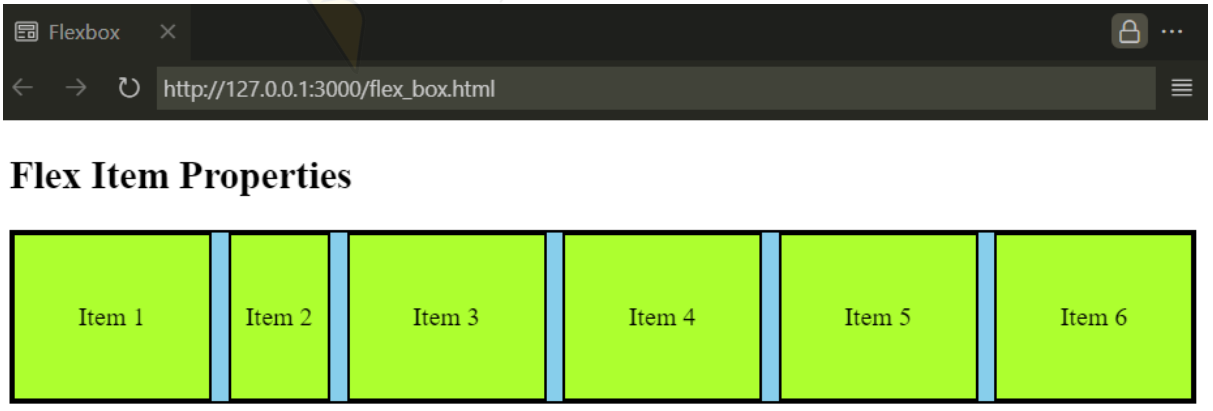
5.2 Using `flex-basis`

HTML :

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
```

```
            background-color:skyblue;
            gap: 10px;

        }
        .item {
            border:2px solid black;
            width: 200px;
            height: 100px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }
    </style>
</head>
<body>

    <h2>Flex Item Properties</h2>
    <div class="container" >
        <div class="item">Item 1</div>
        <div class="item"style="flex-basis: 100px;">Item 2</div>
        <div class="item">Item 3</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
    </div>
</body>
</html>
```

OUTPUT:



Here in above example the flex-basis is set to 100px which is very low w.r.t. Other elements, and are being reflected in output as shown above.

6. Flex

The flex property is a shorthand property for the flex-grow, flex-shrink, and flex-basis properties.

- It takes three style parameters in order:
    a. Flex Grow
    b. Flex Shrink
    c. Flex Basis

Example: flex: 0 0 200px , which means flex-grow:0 , flex-shrink:0 , flex-basis:200px

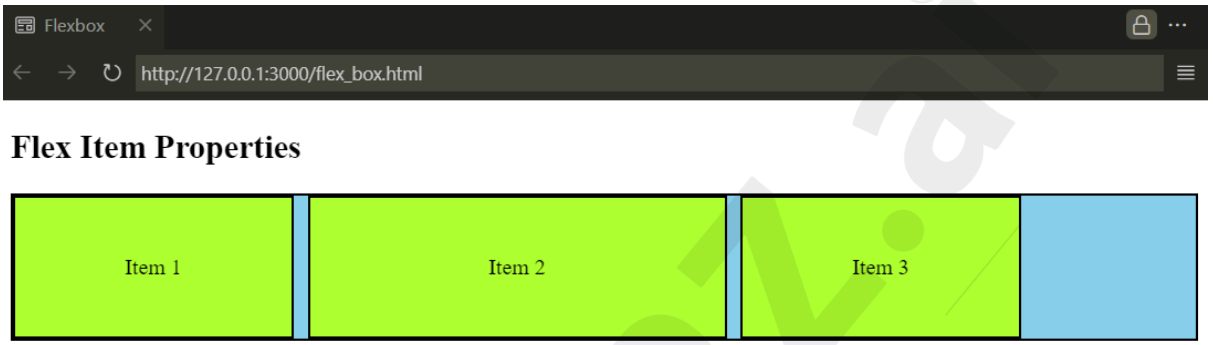
Lets, apply the same, with an example:

Unset

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
            background-color:skyblue;
            gap: 10px;

        }
        .item {
            border:2px solid black;
            width: 200px;
            height: 100px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }
    </style>
</head>
<body>

    <h2>Flex Item Properties</h2>
    <div class="container" >
        <div class="item">Item 1</div>
```

```
        <6div class="item"style="flex: 0 0 300px">Item 2</div>
        <div class="item">Item 3</div>

    </div>
</body>
</html>
```

OUTPUT:



In the above output you can easily observe that the property of flex-basis is applied as previously but this time using flex: property.

7. Align Self

The **`align-self`** property in CSS is used to override the default alignment for a specific flex item. It allows you to control how a single flex item is aligned along the **cross axis** (perpendicular to the flex direction) within the flex container, overriding the `align-items` property set on the container.

7.1 Possible Values for `align-self`:

- **`auto`**: Default. The item inherits the `align-items` value from the flex container.
- **`flex-start`**: Aligns the item to the start of the cross axis.
- **`flex-end`**: Aligns the item to the end of the cross axis.
- **`center`**: Centres the item along the cross axis.
- **`baseline`**: Aligns the item along the baseline of the content.
- **`stretch`**: Stretches the item to fill the container (if no height/width is set).

7.1.1 Align Self: auto

Align-self:auto is a default property here the element inherits its parent container's align-items property, or "stretch" if it has no parent container

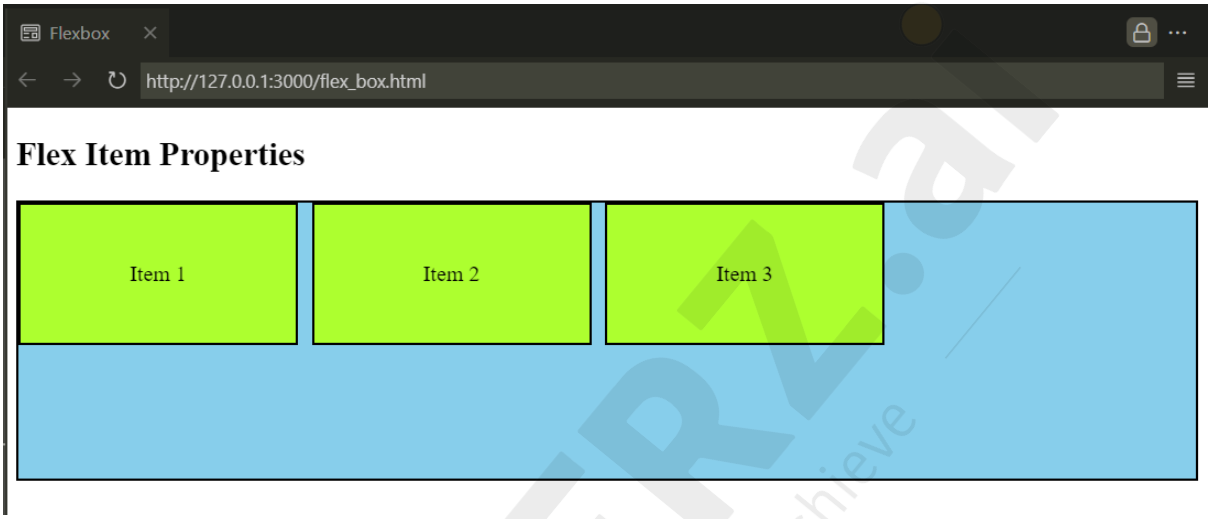
Example:

Unset

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
            background-color:skyblue;
            gap: 10px;
            height:200px;
        }
        .item {
            border:2px solid black;
            width: 200px;
            height: 100px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }
    </style>
</head>
<body>

    <h2>Flex Item Properties</h2>
    <div class="container" >
        <div class="item">Item 1</div>
        <div class="item"style="align-self:auto">Item 2</div>
```

```
        <div class="item">Item 3</div>

    </div>
</body>
</html>
```

OUTPUT:



7.1.2 align-self: center

The element is positioned at the center of the container. Its example demonstration is shown below.

Example:

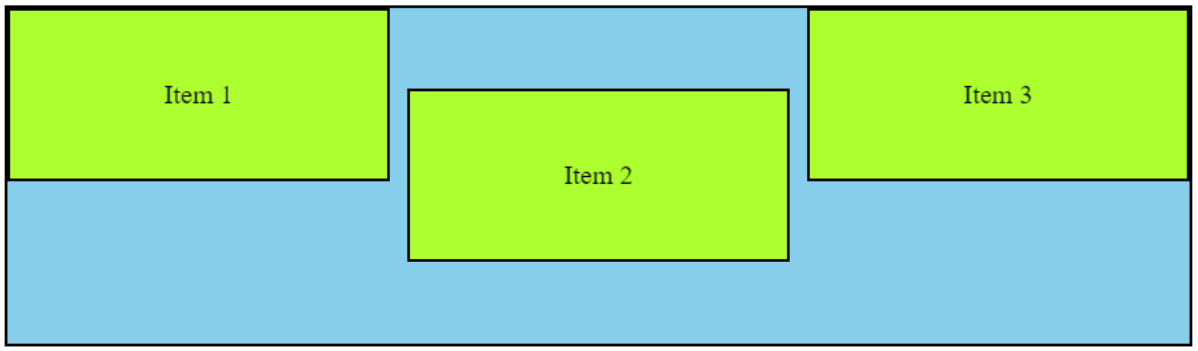
Unset

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
            background-color:skyblue;
            gap: 10px;
```

```
            height:200px;
        }
        .container div {
            flex: 1;
        }
        .item {
            border:2px solid black;
            width: 200px;
            height: 100px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }

    </style>
</head>
<body>
    <h2>Flex Properties</h2>
    <div class="container" >
        <div class="item">Item 1</div>
        <div class="item" style="align-self:center">Item 2</div>
        <div class="item">Item 3</div>

    </div>
</body>
</html>
```

OUTPUT:

Flex Properties



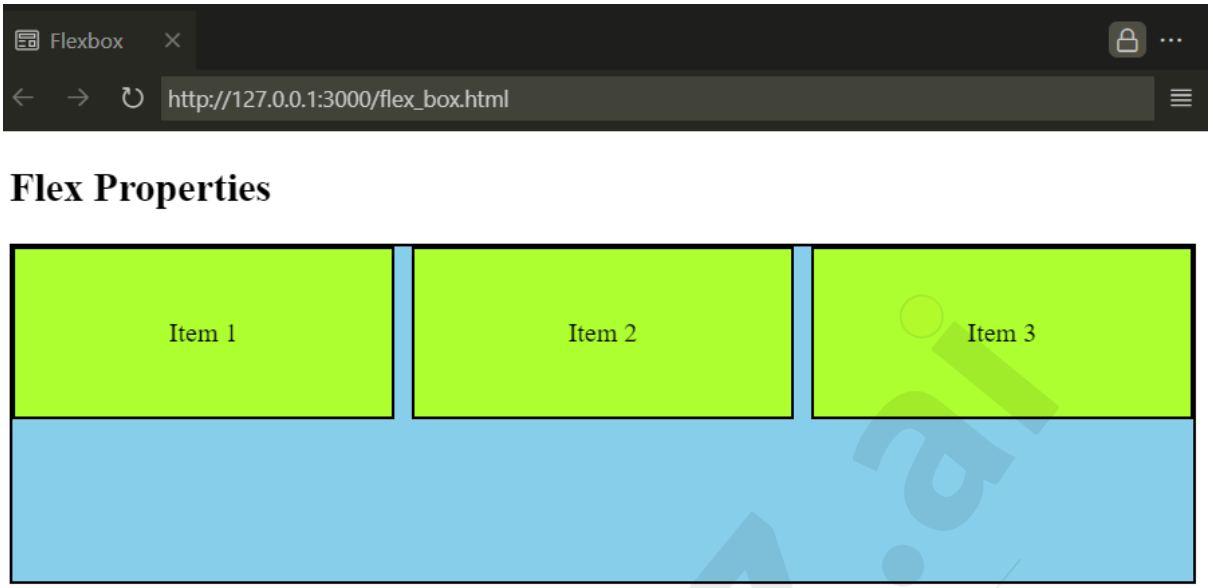
7.1.3 align-self: flex-start

The element is positioned at the start of the flex container. Its example demonstration is shown below.

Example:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
            background-color:skyblue;
            gap: 10px;
            height:200px;
        }
        .container div {
            flex: 1;
        }
        .item {
            border:2px solid black;
            width: 200px;
            height: 100px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }

    </style>
</head>
<body>
    <h2>Flex Properties</h2>
    <div class="container" >
        <div class="item">Item 1</div>
        <div class="item" style="align-self:flex-start">Item 2</div>
        <div class="item">Item 3</div>

    </div>
</body>
</html>
```

OUTPUT:



7.1.4 align-self: flex-end

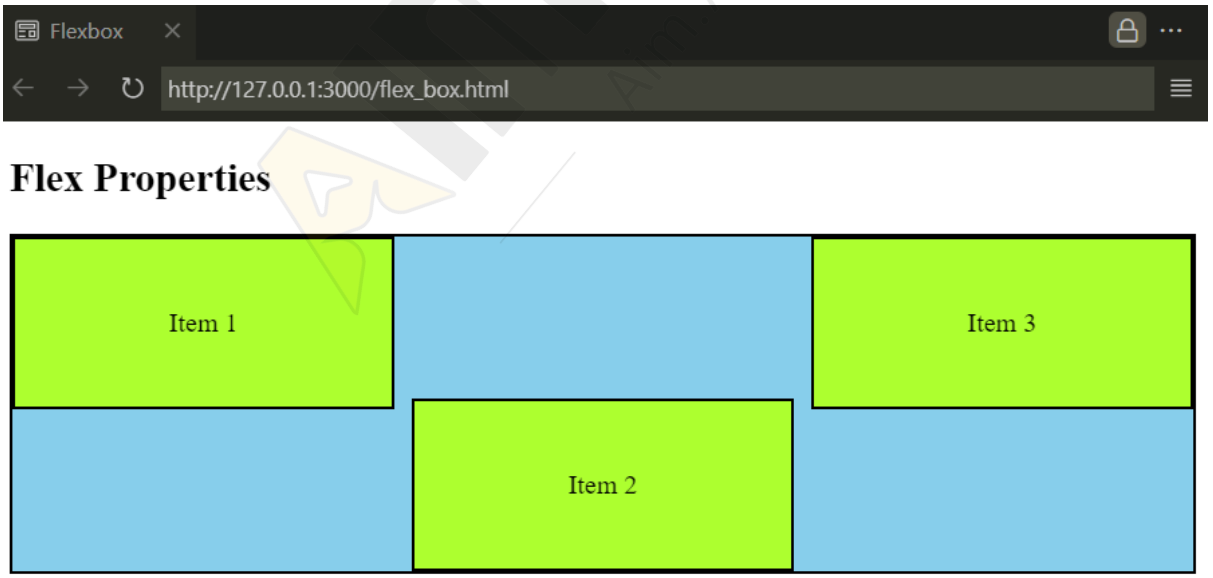
The element is positioned at the end of the flex container. Its example demonstration is shown below.

Example:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
            background-color:skyblue;
            gap: 10px;
            height:200px;
        }
        .container div {
            flex: 1;
```

```
        }
        .item {
            border:2px solid black;
            width: 200px;
            height: 100px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }

    </style>
</head>
<body>
    <h2>Flex Properties</h2>
    <div class="container" >
        <div class="item">Item 1</div>
        <div class="item" style="align-self:flex-end">Item 2</div>
        <div class="item">Item 3</div>

    </div>
</body>
</html>
```

OUTPUT:



7.1.5 align-self: baseline

The element is positioned at the baseline of the flex container. Its example demonstration is shown below.

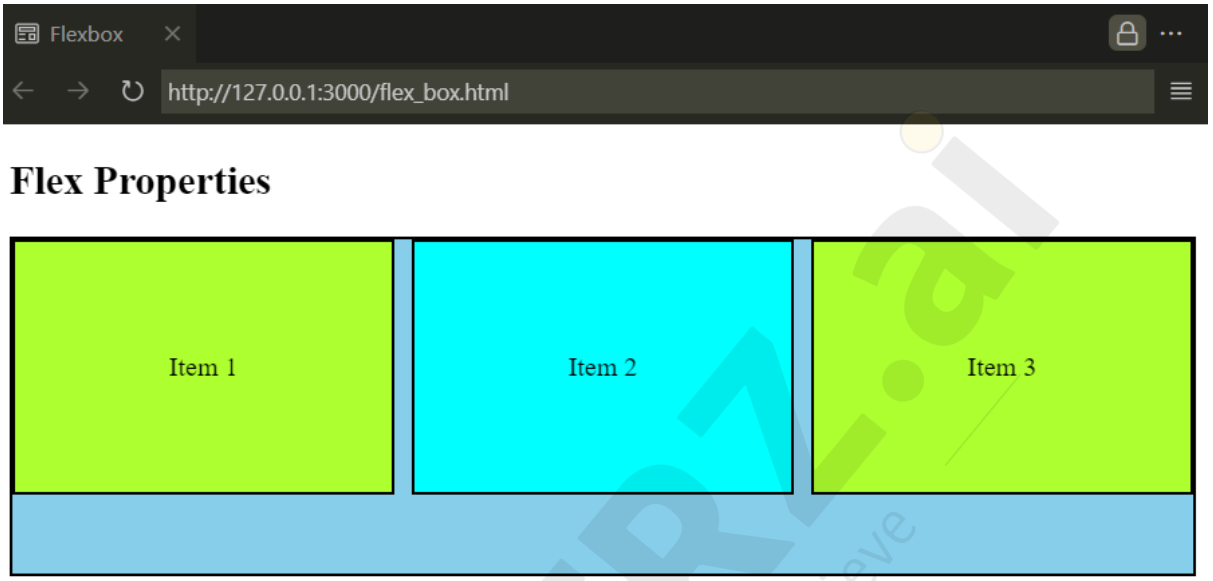
Example:

Unset

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
            background-color:skyblue;
            gap: 10px;
            height:200px;
        }
        .container div {
            flex: 1;
        }
        .item {
            border:2px solid black;
            width: 200px;
            height: 150px;
            background-color: greenyellow;
            display: flex;
            justify-content: center;
            align-items: center;
        }

    </style>
</head>
<body>

    <h2>Flex Properties</h2>
    <div class="container" >
        <div class="item">Item 1</div>
        <div class="item" style="align-self:baseline; background-color:
aqua;">Item 2</div>
        <div class="item">Item 3</div>

    </div>
```

```
</body>
</html>
```

OUTPUT:



7.1.6 align-self: stretch

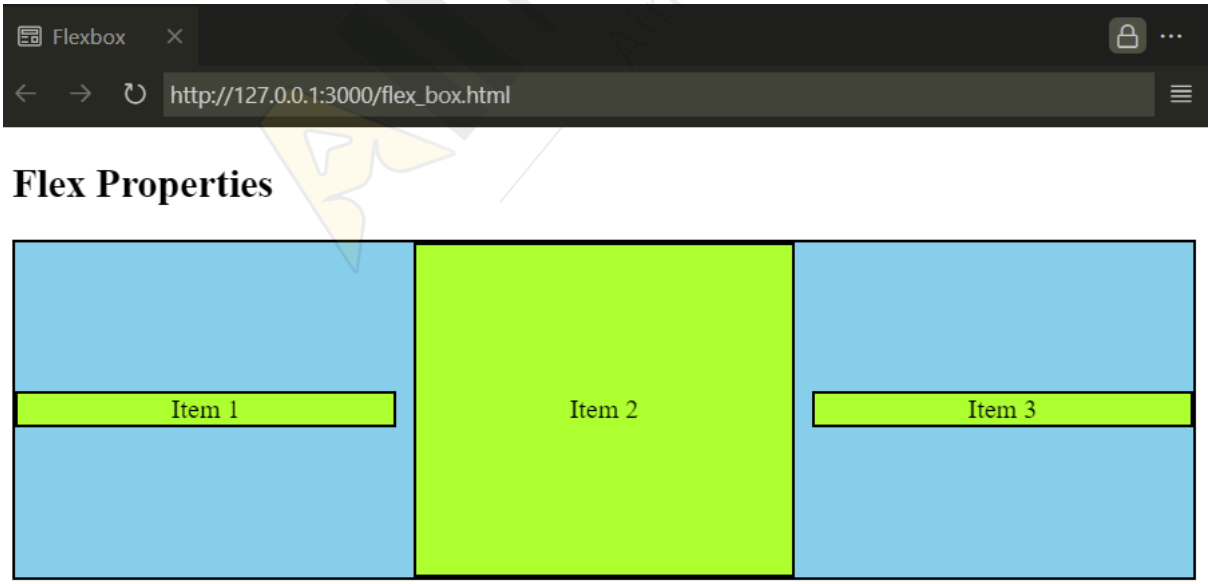
The element is positioned to fit the container. As you can see in the example given below.

Example:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Flexbox </title>
    <style>
        .container {
            border: 2px solid black;
            display: flex;
            background-color:skyblue;
            gap: 10px;
            height:200px;
            align-items: center;
        }
```

```
        .container div {
            flex: 1;
        }
        .item {
            border:2px solid black;
            background-color: greenyellow
            display: flex;
            justify-content: center;
            align-items: center;
        }

    </style>
</head>
<body>

    <h2>Flex Properties</h2>
    <div class="container" >
        <div class="item">Item 1</div>
        <div class="item" style="align-self:stretch">Item 2</div>
        <div class="item">Item 3</div>

    </div>
</body>
</html>
```

OUTPUT:



CSS Grid

Introduction

CSS grid layout or CSS grid creates complex responsive web design grid layouts more easily and consistently across browsers. Historically, there have been other methods for controlling web page layout methods, such as tables, floats, and more recently, CSS Flexible Box Layout.

In other words, CSS Grid is a powerful layout system in CSS that allows you to design complex, two-dimensional layouts on the web. Unlike Flexbox, which is primarily one-dimensional (either row-based or column-based), Grid allows for layouts in both rows and columns at the same time.

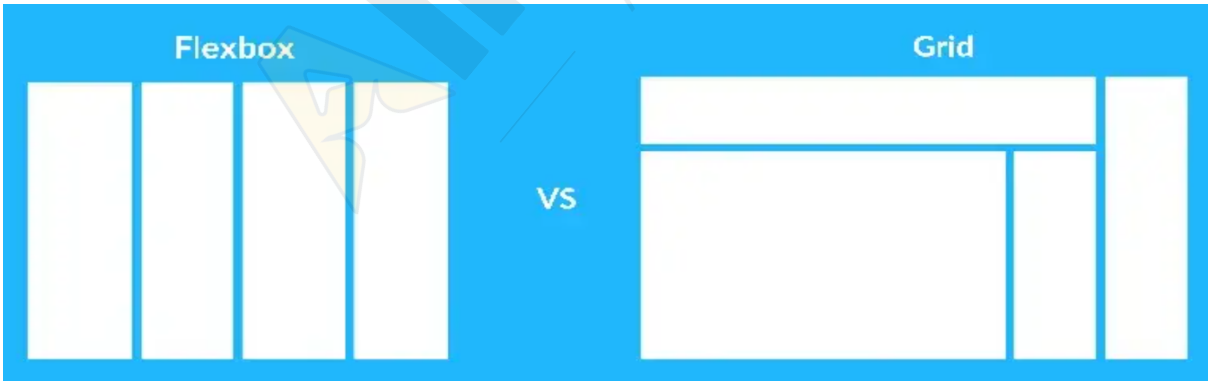
When to Use CSS Grid:

1. **Complex Layouts**:
    - If your layout requires precise control over both rows and columns, Grid is ideal. You can easily create multi-column, multi-row layouts, such as a web page layout with a header, sidebar, content area, and footer.
    - **Example**: Dashboards, product galleries, multi-column layouts.
2. **Fixed and Flexible Layouts**:
    - CSS Grid allows you to create layouts that can be both fixed (using fixed sizes like `px`) and flexible (using `fr` units). This makes it useful for layouts that need to adapt to different screen sizes while maintaining a structure.
    - **Example**: Creating responsive layouts that rearrange based on screen size.
3. **Overlapping Elements**:
    - Unlike Flexbox, CSS Grid allows you to position items to overlap each other easily by positioning them over the same grid cells.
    - **Example**: Artistic layouts where elements need to overlap for visual design purposes, like magazine-style designs.

4. **Aligning Items in Rows and Columns**:
    - If you need to align items both horizontally and vertically with ease, CSS Grid provides properties like `justify-items`, `align-items`, and `place-items` to manage the alignment across both axes.
    - **Example**: Aligning content inside a complex card layout where images, text, and buttons need precise alignment in both dimensions.

Benefits of using Grid over Flexbox:

- **Two-dimensional layouts**: Control both rows and columns simultaneously.
- **Precise item placement**: Explicitly position elements on specific grid lines.
- **Grid template areas:** Intuitively define areas for easy layout management.
- **Responsive design:** Simplify responsive layouts with minimal code.
- **Overlapping elements:** Easily layer elements.



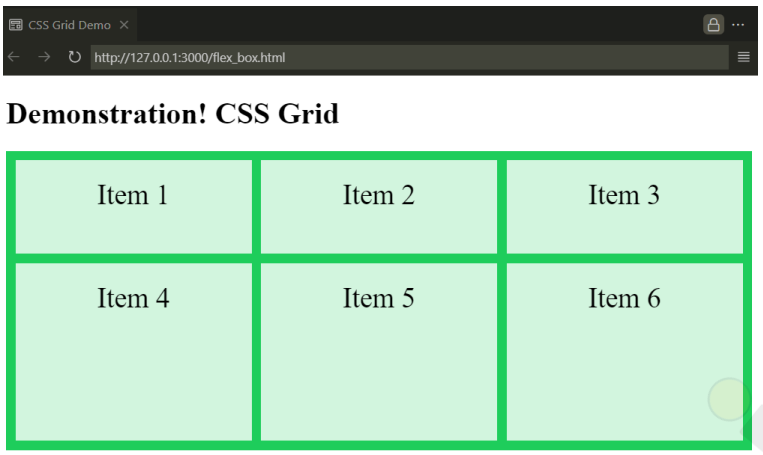
Let's take an example to demonstrate how we can apply grid property in CSS.

Example:

Unset

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Grid Demo</title>
    <style>
        .container {
  height:300px;
  display: grid;
  gap: 10px;
  background-color: #1ecd5b;
  padding: 10px;
  grid: 100px / auto auto auto;
}
 div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
    </style>
</head>
<body>
    <h1>Demonstration! CSS Grid</h1>
    <div class="container">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
    </div>
</body>
</html>
```

OUTPUT:



Properties of CSS Grid

There are several important properties of Grid in CSS, let's discuss them one by one.

1. grid-template-column

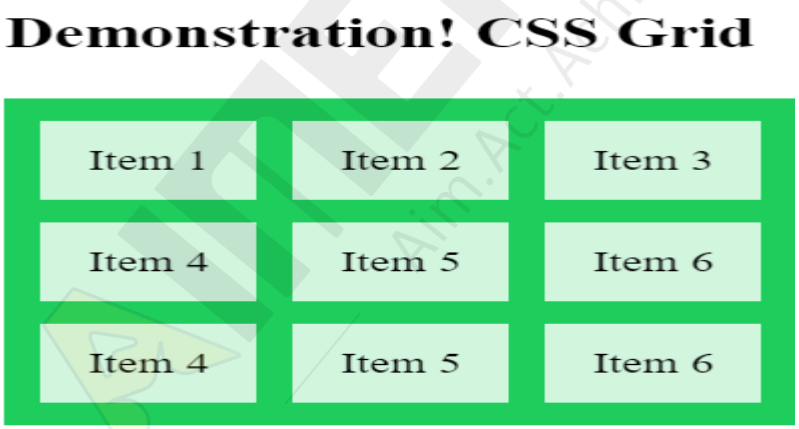
The grid-template-columns CSS property defines the line names and track sizing functions of the grid columns.

Example:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Grid Demo</title>
    <style>
        .container {
  display: grid;
  gap: 10px;
  background-color: #1ecd5b;
  padding: 10px;
  grid-template-columns: 60px 60px 60px;

}
 div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 10px 0;
  font-size: 30px;
}
.item{
    font-size: small;
```

```
}
    </style>
</head>
<body>
    <h3>Demonstration! CSS Grid</h3>
    <div class="container">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
    </div>
</body>
</html>
```

OUTPUT:



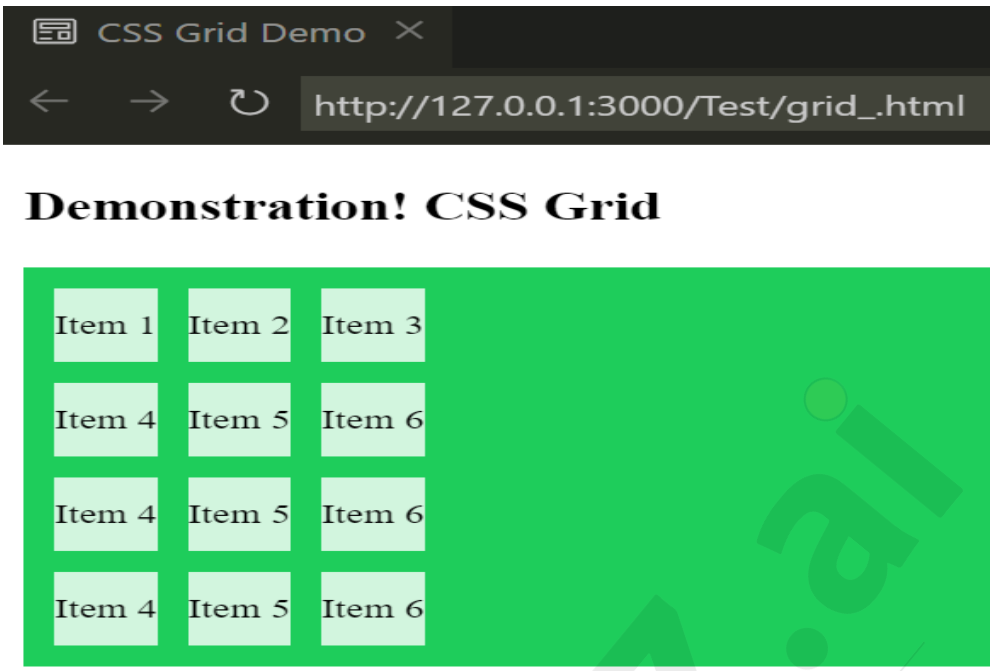
1.1 Other Important properties in Grid Template Column

We can apply various properties using grid-template-columns as described below:

- **max-content** : Sets the size of each column to depend on the largest item in the column.
  Example:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Grid Demo</title>
    <style>
        .container {
  display: grid;
  gap: 10px;
  background-color: #1ecd5b;
  padding: 10px;
  grid-template-columns: max-content max-content max-content;
}
 div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 10px 0;
  font-size: 30px;
}
.item{
    font-size: small;
}
    </style>
</head>
<body>
    <h3>Demonstration! CSS Grid</h3>
    <div class="container">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
    </div>
</body>
</html>
```

OUTPUT:



- **min-content** : Sets the size of each column to depend on the smallest item in the column length Sets the size of the columns, by using a legal length value.

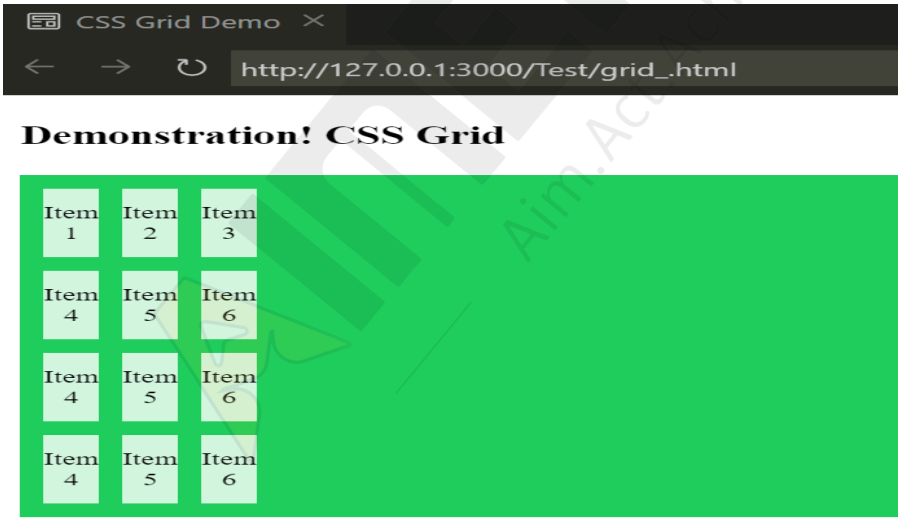
Example:

Unset

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Grid Demo</title>
    <style>
        .container {
  display: grid;
  gap: 10px;
  background-color: #1ecd5b;
  padding: 10px;
  grid-template-columns:min-content min-content min-content;
}
 div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 10px 0;
  font-size: 30px;
}
.item{
    font-size: small;
}
```

```
    </style>
</head>
<body>
    <h3>Demonstration! CSS Grid</h3>
    <div class="container">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
    </div>
</body>
</html>
```

OUTPUT:



2. grid auto column

The grid-auto-columns property in CSS Grid defines the size of implicit columns that are created when grid items are placed outside the explicitly defined grid.

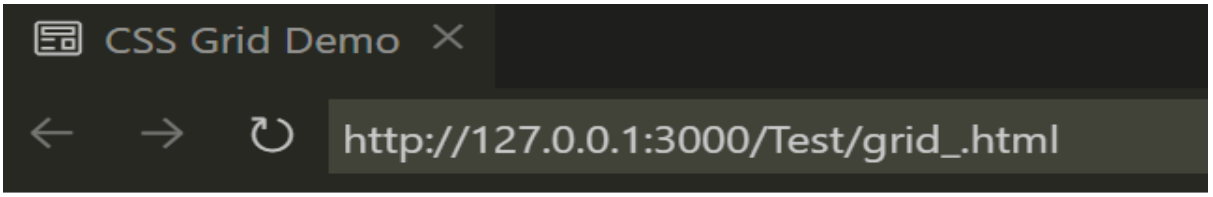
> **Note: When using grid-template-columns property it overrides auto column property.**

Example:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Grid Demo</title>
    <style>
.container {
  display: grid;
  gap: 5px;
  background-color: #1ecd5b;
  padding: 5px;
  grid-auto-columns:80px;
  align-items: center;
}
 div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 10px 0;
  font-size: 30px;
}
/* grid-area: 3/2/4/3; */
.item{
    font-size: small;

}
.item1{
    font-size: small;

}
.item2{
    font-size: small;

}
.item3{
    font-size: small;

}
.item4{
    font-size: small;

}
.item5{
    font-size: small;
```

```
}
.item6{
    font-size: small;

}
.item7{
    font-size: small;

}
.item8{
    font-size: small;

}
.item9{
    font-size: small;

}
.item10{
    font-size: small;
}
.item11{
  font-size: small;
  grid-area: 3/4; //It clarifies position of class at 3rd row and 4th column
}
    </style>
</head>
<body>
    <h3>Demonstration! CSS Grid</h3>
    <div class="container">
        <div class="item">Item 1</div>
        <div class="item1">Item 2</div>
        <div class="item2">Item 3</div>
        <div class="item3">Item 4</div>
        <div class="item4">Item 5</div>
        <div class="item5">Item 6</div>
        <div class="item6">Item 7</div>
        <div class="item7">Item 8</div>
        <div class="item8">Item 9</div>
        <div class="item9">Item 10</div>
        <div class="item10">Item 11</div>
        <div class="item11">Item 12</div>

    </div>
</body>
</html>
```

OUTPUT:



Demonstration! CSS Grid

Item 1	Item 2	Item 3	Item 4
Item 5	Item 6	Item 7	Item 8
Item 9	Item 10	Item 11	Item 12

3. grid template row

The grid-template-rows property in CSS Grid defines the number and size of rows in a grid container. It allows you to explicitly set the height of each row in the grid.

Example:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Grid Demo</title>
    <style>
        .container {
  display: grid;
  gap: 10px;
  background-color: #1ecd5b;
  padding: 10px;
  grid-template-columns: auto auto auto ;
  grid-template-rows:40px 40px 40px;
}
 .container div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
```

```
  padding: 10px 0;
  font-size:15px;

}
.item{
    font-size: small;
}
    </style>
</head>
<body>
    <h3>Demonstration! CSS Grid</h3>
    <div class="container">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
        <div class="item">Item 6</div>
        <div class="item">Item 7</div>
        <div class="item">Item 8</div>
        <div class="item">Item 9</div>
    </div>
</body>
</html>
```

OUTPUT:

Demonstration! CSS Grid

Item 1	Item 2	Item 3
Item 4	Item 5	Item 6
Item 4	Item 5	Item 6
Item 4	Item 5	Item 6

4. grid auto row

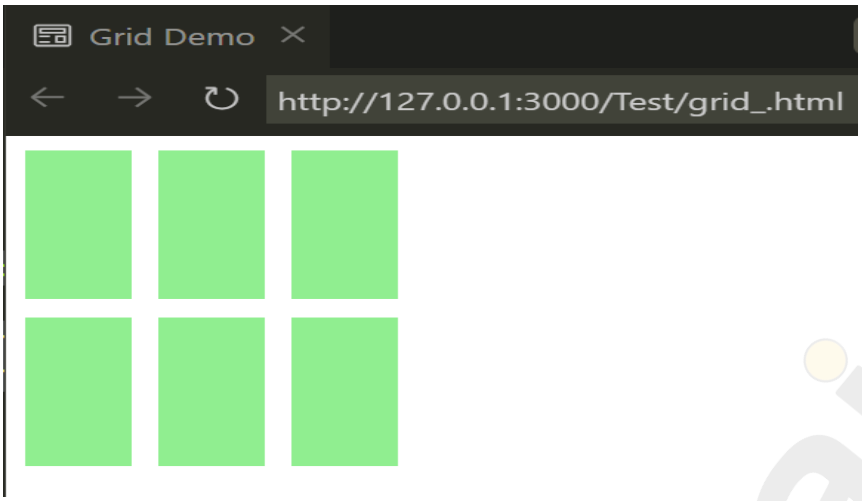
The `grid-template-areas` property in CSS Grid allows you to define specific areas of the grid by naming them, making it easier to position items within those areas. This is done by using a visual, grid-like pattern made up of strings that represent each named area.

Example:

Unset

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Grid Demo</title>
        <meta name="description" content="Grid Demo">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <style>
       #grid {
  display: grid;
  grid-template-columns:40px 40px 40px;
  gap: 10px;
  grid-auto-rows: 80px;
}

#grid div{
  background-color:lightgreen;
}
        </style>
    </head>
    <body>
    <div id="grid">
  <div id="item1"></div>
  <div id="item2"></div>
  <div id="item3"></div>
  <div id="item4"></div>
  <div id="item5"></div>
  <div id="item6"></div>
</div>

    </body>
</html>
```

OUTPUT:



5. column gap

The column-gap CSS property sets the size of the gap (gutter) between an element's columns.
Initially a part of Multi-column Layout, the definition of column-gap has been broadened to include multiple layout methods. Now specified in CSS box alignment, it may be used in multi-column, flexible box, and grid layouts.

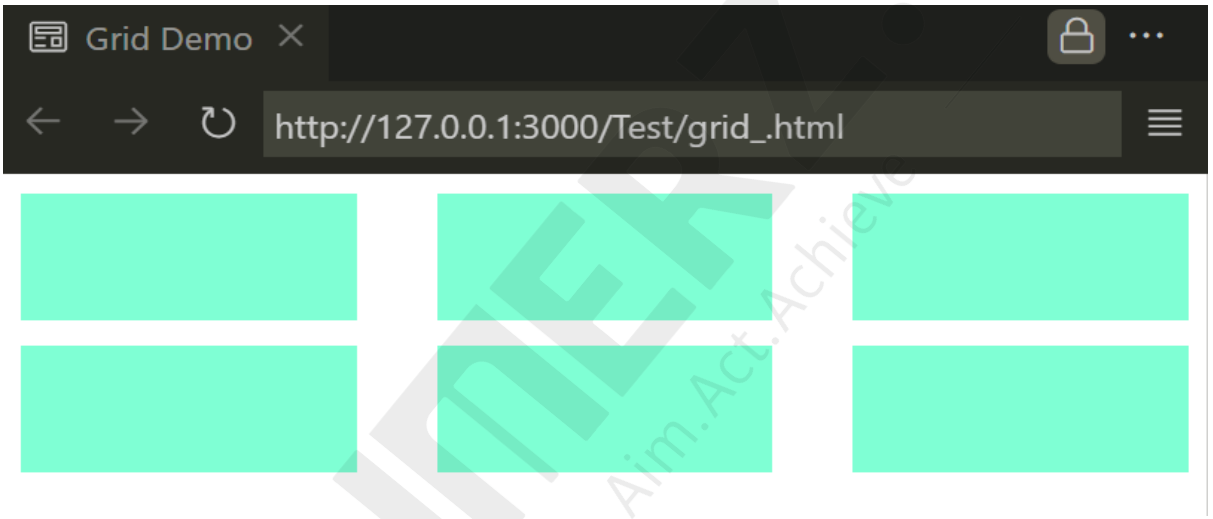
Example:

Unset

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Grid Demo</title>
        <meta name="description" content="Grid Demo">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <style>
       #grid {
  display: grid;
  grid-template-areas: "a a a";
  gap: 10px;
  grid-auto-rows: 50px;
  column-gap: 30px;
}

#grid > div {
  background-color:aquamarine;
}
        </style>
```

```
    </head>
    <body>
    <div id="grid">
  <div id="item1"></div>
  <div id="item2"></div>
  <div id="item3"></div>
  <div id="item4"></div>
  <div id="item5"></div>
  <div id="item6"></div>
</div>

    </body>
</html>
```

OUTPUT:



6. row gap

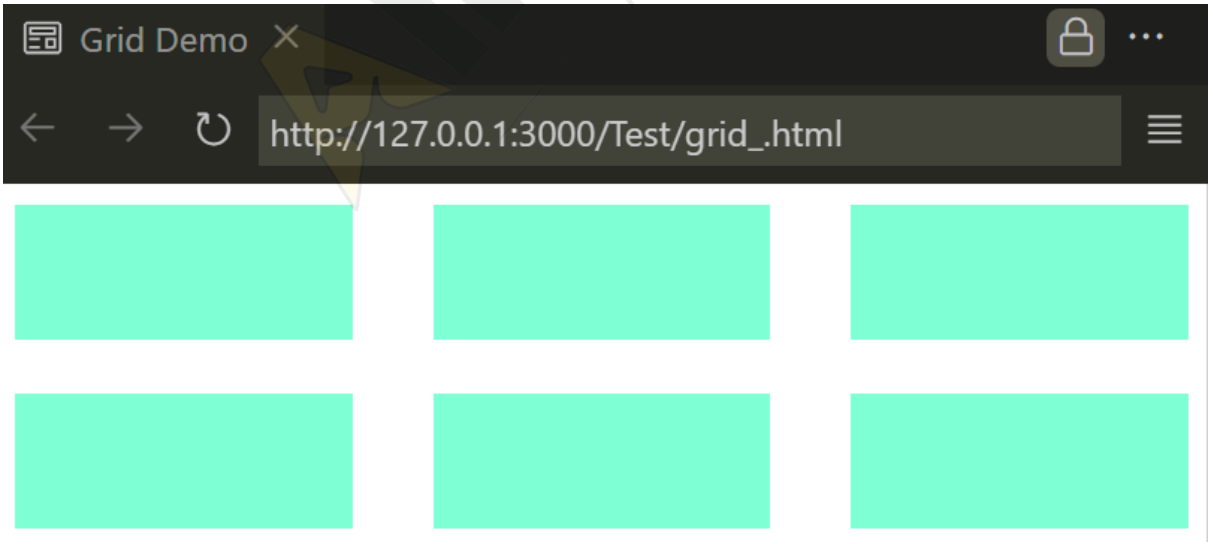
The spacing (gap) between rows in a grid layout is defined by the row-gap property in CSS Grid (formerly called grid-row-gap in previous versions). It facilitates the creation of separation between items along the rows of the grid without the need for margins by controlling the vertical spacing between rows.

Example:

```
Unset
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Grid Demo</title>
        <meta name="description" content="Grid Demo">
```

```
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <style>
       #grid {
  display: grid;
  grid-template-areas: "a a a";
  gap: 10px;
  grid-auto-rows: 50px;
  column-gap: 30px;
  row-gap: 20px;
}

#grid > div {
  background-color:aquamarine;
}
        </style>
    </head>
    <body>
    <div id="grid">
  <div id="item1"></div>
  <div id="item2"></div>
  <div id="item3"></div>
  <div id="item4"></div>
  <div id="item5"></div>
  <div id="item6"></div>
</div>
    </body>
</html>
```

OUTPUT:



7. grid gap

The `grid-gap` property in CSS Grid is a shorthand property that specifies both the row and column spacing between grid items. It controls the space between both rows and columns in the grid.

Example:

```
Unset
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Grid Demo</title>
        <meta name="description" content="Grid Demo">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <style>
       #grid {
  display: grid;
  grid-template-areas: "a a a";
  gap: 10px;
  grid-auto-rows: 70px;
    grid-gap: 30px;
}

#grid > div {
  background-color:aquamarine;
}
        </style>
    </head>
    <body>
    <div id="grid">
  <div id="item1"></div>
  <div id="item2"></div>
  <div id="item3"></div>
  <div id="item4"></div>
  <div id="item5"></div>
  <div id="item6"></div>
</div>

    </body>
</html>
```

OUTPUT:

8. place-items

The place-items property in CSS is a shorthand that sets both align-items (vertical alignment) and justify-items (horizontal alignment) at once. It controls how grid or flexbox items are aligned both vertically and horizontally within their container.

8.1 Key Values:

- **start**: Aligns items to the start of the grid cell (top for vertical, left for horizontal).
- **end**: Aligns items to the end of the grid cell (bottom for vertical, right for horizontal).
- **center**: Centers items inside their grid cells both vertically and horizontally.
- **stretch** (default): Stretches items to fill the grid cell both vertically and horizontally.

Example:

```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
            #grid {
                display: grid;
                height: 400px;
                grid-template-columns: auto auto auto;
                gap: 10px;
                background-color: #2196F3;
                padding: 10px;
                place-items:start; /* start | end | center */

            }

            #grid > div {
                background-color: lightgreen;
                text-align: center;
                padding: 10px;
                font-size: 10px;
            }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT:

9. align content

The align-content property in CSS Grid controls how the grid content is aligned vertically when there is extra space in the container. It applies when the grid doesn't fill the entire container after rows and columns are set.

9.1 `align-content` values:

- **`start`**: Aligns the content to the top of the container.

Example :

Example:

```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
```

```
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        place-items:start;
        align-content:start;

      }

      #grid > div {
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>'
  </body>
</html>
```

OUTPUT:

- **`end`**: Aligns the content to the bottom of the container.
- **`center`**: Centers the content vertically in the container.
- **`stretch`**: Stretches the content to fill the container (this is the default).
- **`space-between`**: Distributes the content evenly, with the first item at the top and the last at the bottom, leaving space between the rows.

- **`space-around`**: Distributes the content with even space around each row, including the first and last items.
- **`space-evenly`**: Distributes the content so that the space between all rows (including before the first and after the last) is equal.

Example:

Unset

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        place-items:start;
        align-content: end;

/* Other Values which could be also used as per the requiurement of the usecase:  start | end | center | stretch | space-around space-between | space-evenly */

      }

      #grid > div {
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 3</div>
```

```
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>'
  </body>
</html>
```

OUTPUT:

10. align items

The `align-items` property in CSS Grid controls how grid items are aligned vertically within their grid cells along the block axis (typically the vertical axis).

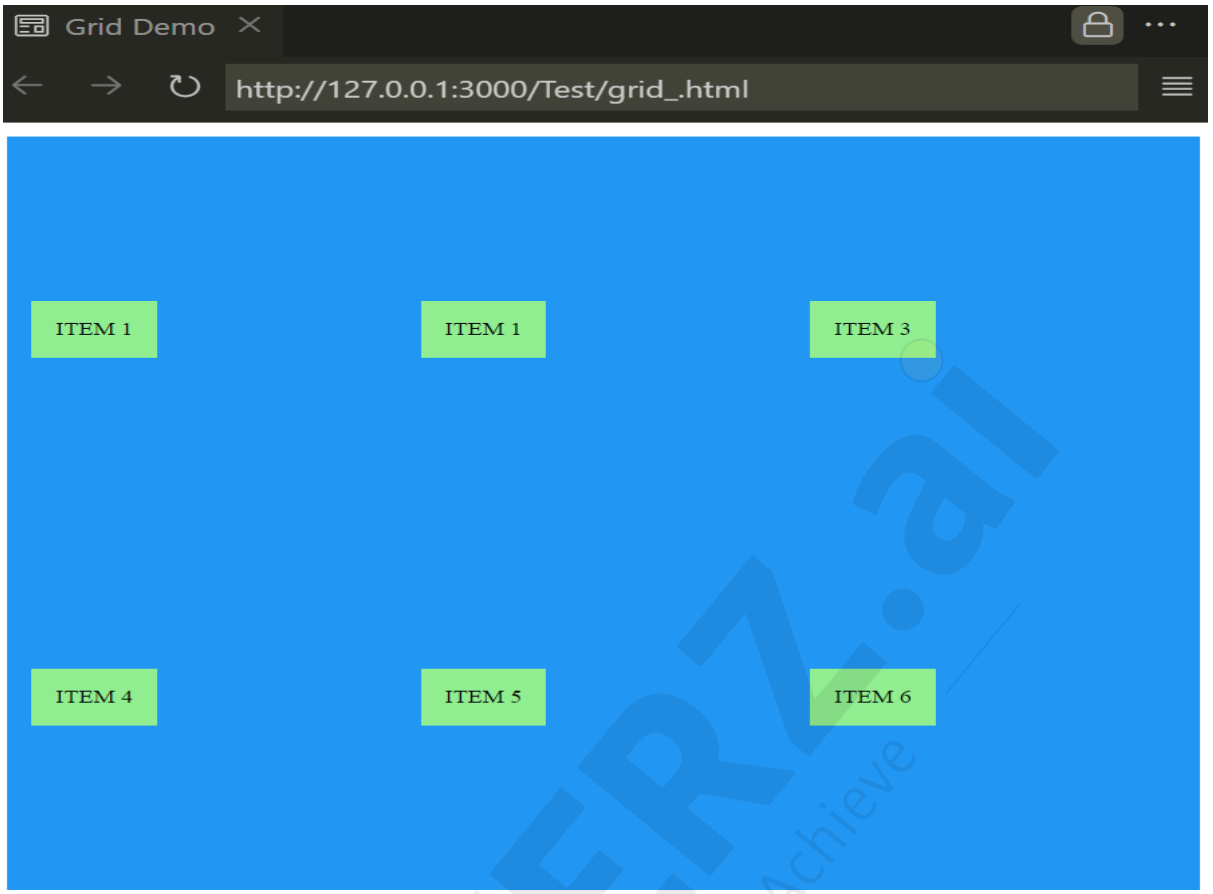
10.1 Key Values for `align-items`:

- **`start`**: Aligns items to the top of their grid cells.
- **`end`**: Aligns items to the bottom of their grid cells.
- **`center`**: Aligns items in the center vertically within their grid cells.
- **`stretch`** (default): Stretches items to fill the height of their grid cells.

Example:

```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        place-items:start;
        /* align-content: end; */
        align-items: center;

       /* Other values for align-items , end | start | stretch*/
      }

      #grid > div {
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT:



11. justify content

The `justify-content` property in CSS Grid controls the horizontal alignment of the entire grid within the grid container when there is extra space available along the row (inline) axis.

11.1 Key Values for `justify-content`:

- **`start`**: Aligns the grid items to the left of the container.
- **`end`**: Aligns the grid items to the right of the container.
- **`center`**: Centers the grid horizontally in the container.
- **`stretch`** (default): Stretches the grid items to fill the width of the container.
- **`space-between`**: Distributes grid items evenly, with the first item at the start and the last at the end.
- **`space-around`**: Distributes grid items evenly with space around them, including at the edges.
- **`space-evenly`**: Distributes grid items with equal space between and around them.

Example:

```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        align-items: center;
        justify-content:start;
      }

      #grid > div {
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT:

12. justify items

The justify-items property in CSS Grid controls how individual grid items are aligned horizontally within their grid cells along the inline axis (typically the horizontal axis).

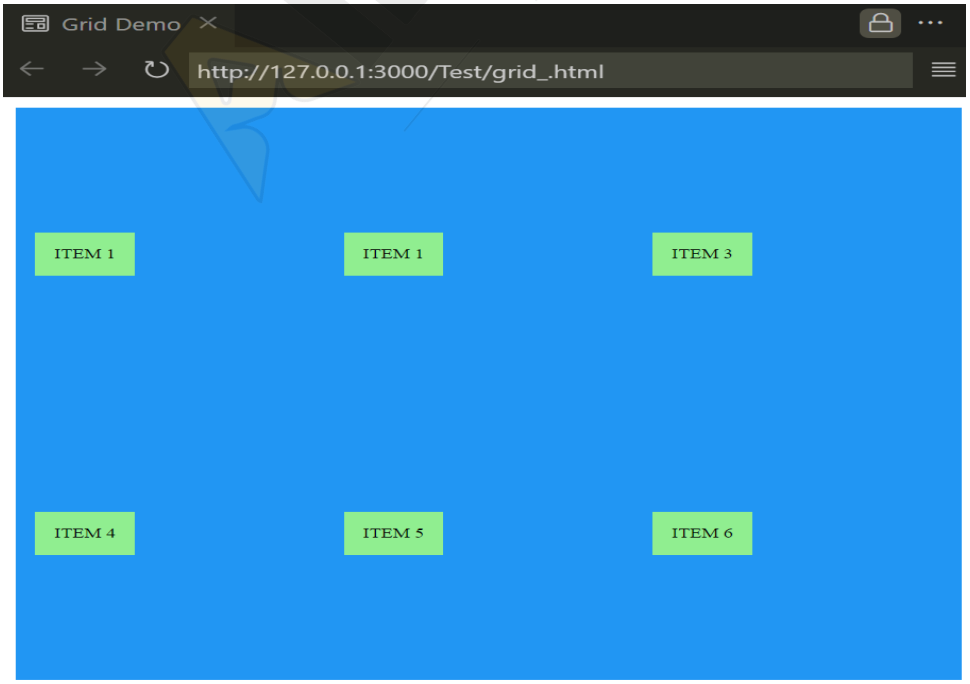
12.1 Key Values for justify-items:

- **start**: Aligns grid items to the left edge of their grid cells.
- **end**: Aligns grid items to the right edge of their grid cells.
- **center**: Horizontally centers grid items within their grid cells.
- **stretch** (default): Stretches grid items to fill the width of their grid cells.

```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
```

```
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        align-items: center;
        justify-items:start;
/* Other Justify content Prop, start ,end, centre ,stretch*/
      }

      #grid > div {
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 1</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT:



Grid Container Properties

1. Grid Column

The `grid-column` property in CSS specifies how many columns an element spans and where it starts and ends in a CSS Grid Layout. It's a shorthand for `grid-column-start` and `grid-column-end`.

1.1 grid-column-start

- **Integer**: Specifies the index of the grid line where the item starts (e.g., 1, 2, etc.).

  Example:

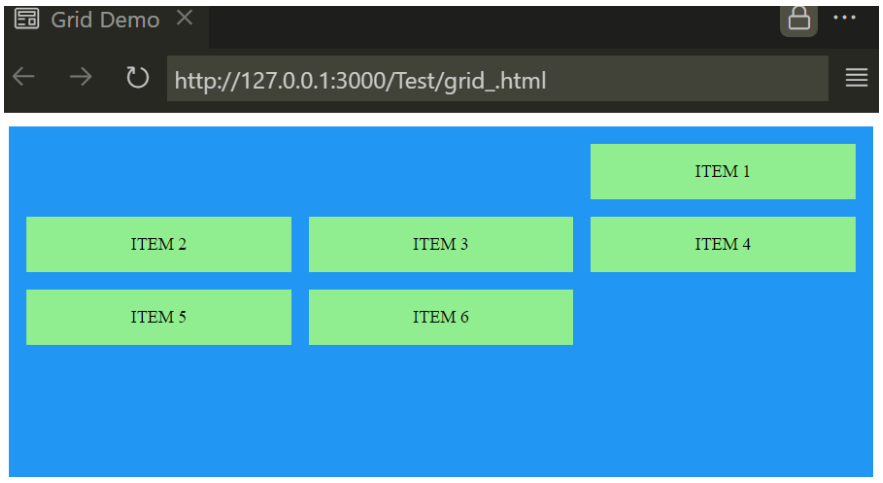
```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        align-content:start;
        justify-items:stretch;
      }
      #item1{
        grid-column-start:2;
      }

      #grid div {
```

```
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item1">ITEM 1</div>
      <div id="item">ITEM 2</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT :

Without grid-column-start:3:

After Applying grid-column-start property:



Note: It is clearly visible in the output that our grid is starting from the last column of line one, hence the subsequent items appear to be shifted.

- **Span keyword**: You can also use the span keyword to start the item at a specified line and span a certain number of columns.

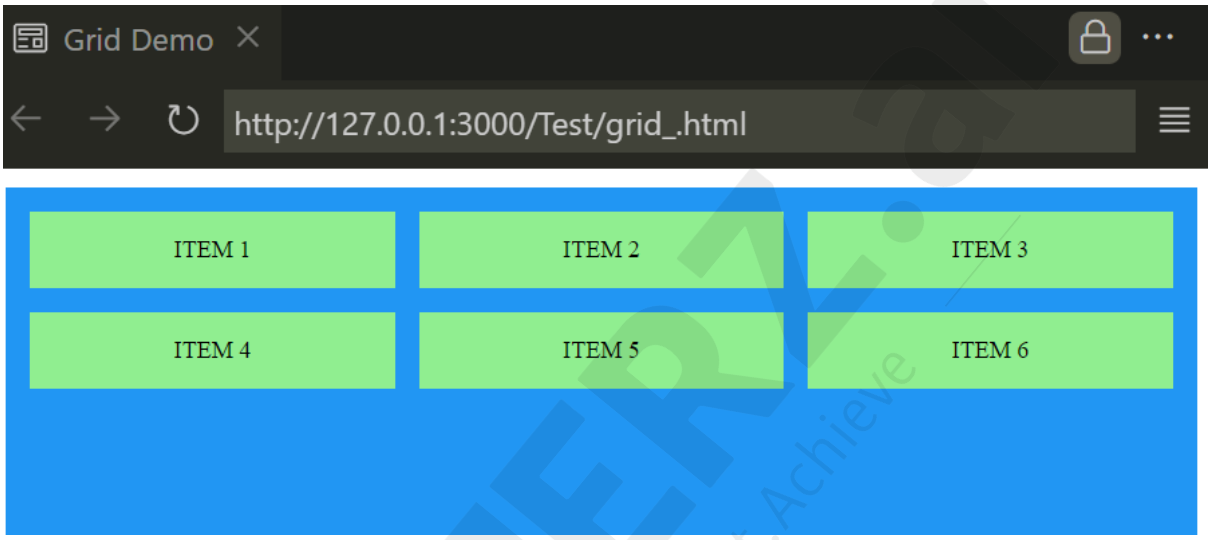
  Example:

```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        align-content:start;
        justify-items:stretch;

      }
      #item1{
        grid-column-start:span 2;
      }

      #grid div {
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item1">ITEM 1</div>
      <div id="item">ITEM 2</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
```

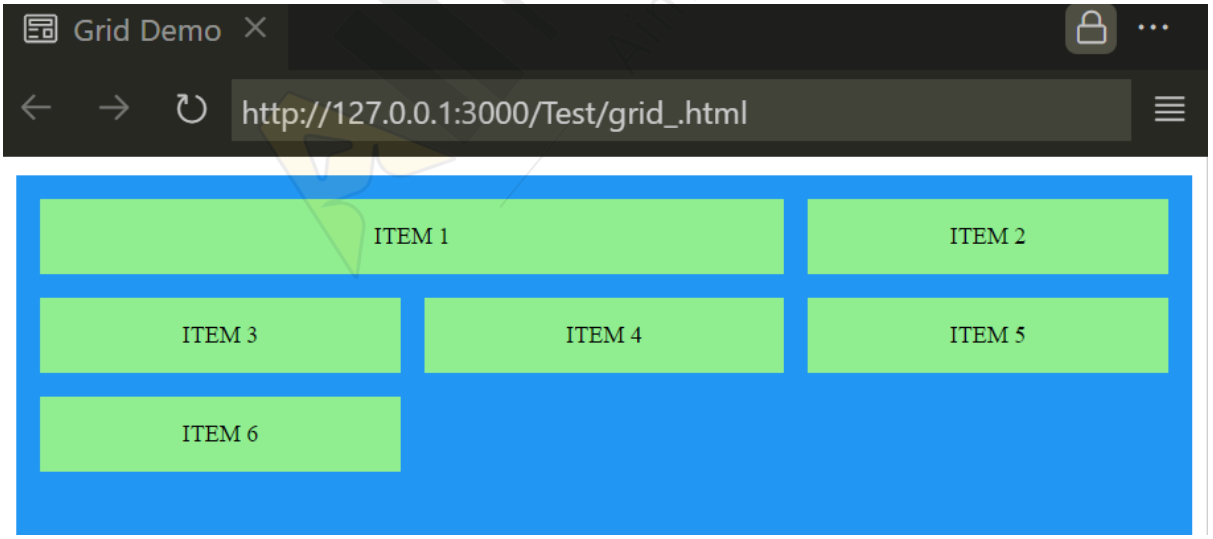
```
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT:

Without applying grid-column-start: span 2;



After Applying grid-column-start:span 2;



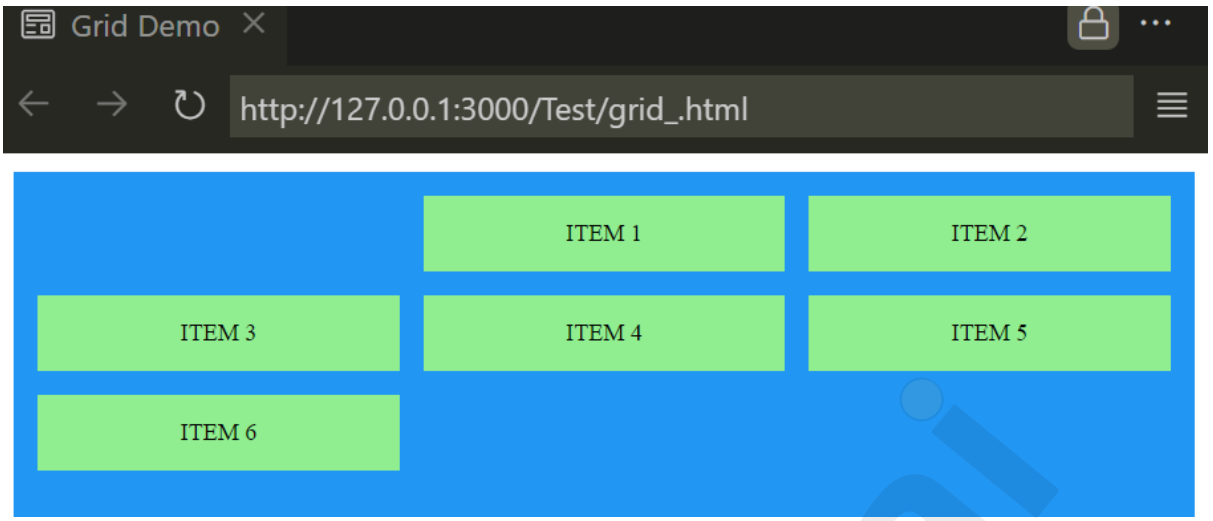
1.2 grid-column-end

The grid-column-end property in CSS defines where a grid item ends horizontally by specifying the column line it stops at. It controls how many columns the item spans.

Example:

```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        align-content:start;
        justify-items:stretch;

      }
      #item1{
        grid-column-end:3;
      }

      #grid div {
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item1">ITEM 1</div>
      <div id="item">ITEM 2</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT:



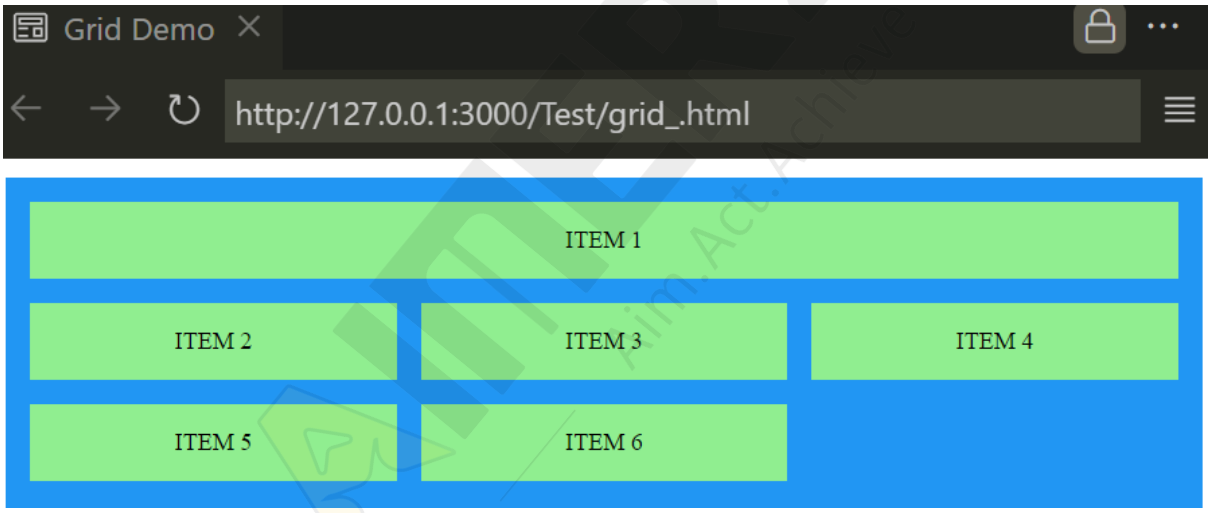
Note: Other values of grid-column-start are also applicable with grid-column-end in the similar fashion.

Example: grid-column-end:span 3;

```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        align-content:start;
        justify-items:stretch;

      }
      #item1{
        grid-column-end:span 3;
      }

      #grid div {
        background-color: lightgreen;
        text-align: center;
```

```
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item1">ITEM 1</div>
      <div id="item">ITEM 2</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT:



2. Grid Row

The grid-row property in CSS defines where a grid item ends vertically by specifying the column line it stops at. It controls how many columns the item spans.

Example:

```
Unset
<!DOCTYPE html>
<html>
  <head>
```

```
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        grid-template-columns: auto auto auto;
        gap: 10px;
        background-color: #2196F3;
        padding: 10px;
        align-content:start;
        justify-items:stretch;

      }
      #item1{
        grid-row:span 2;  // this will assign a span of 2 rows for #item1
      }

      #grid div {
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item1">ITEM 1</div>
      <div id="item">ITEM 2</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT:

3. Grid Area

The grid-area CSS shorthand property specifies a grid item's size and location within a grid by contributing a line, a span, or nothing (automatic) to its grid placement, thereby specifying the edges of its grid area.

3.1 Item Names

Specifies a name for the grid item as specified in the declaration of grid-area with customised names.

Example:

```
Unset
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Grid Demo</title>
    <meta name="description" content="Grid Demo" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      #grid {
        display: grid;
        height: 400px;
        gap: 10px;
```

```
        background-color: #2196F3;
        padding: 10px;
        align-content:start;
        justify-items:stretch;
        grid-template-areas: 'myArea myArea myArea myArea myArea';
      }
      #item1{
        grid-area: "myArea" ;   /* alterantive values, start | end  */
      }

      #grid div {
        background-color: lightgreen;
        text-align: center;
        padding: 10px;
        font-size: 10px;
      }
    </style>
  </head>
  <body>
    <div id="grid">
      <div id="item1">ITEM 1</div>
      <div id="item">ITEM 2</div>
      <div id="item">ITEM 3</div>
      <div id="item">ITEM 4</div>
      <div id="item">ITEM 5</div>
      <div id="item">ITEM 6</div>
    </div>
  </body>
</html>
```

OUTPUT:

THANK YOU

AIMERZ.ai
Aim.Act.Achieve