Flex properties (IMPORTANT):

**Flex-grow:**

The flex-grow CSS property sets the flex grow factor, which specifies how much of the flex container's remaining space should be assigned to the flex item's main size.

When the flex-container's main size is larger than the combined main size's of the flex items, the extra space is distributed among the flex items, with each item growth being their growth factor value as a proportion of the sum total of all the container's items' flex grow factors.

This property will take all the remaining space of the flex container after the items

Syntax:

Flex-grow: number/initial/inherit;

**Property values:**

1. **number**: A number that defines how the item will grow compare to other flexible items.
2. **initial**: It sets the value to it’s default value (0).
3. **inherit**: It inherit the property from it’s parent elements.

<!DOCTYPE html**>**

**<html>**

**<head>**

**<style>**

.container {

display: -webkit-flex;

display: flex;

background-color: green;

}

.flex-item {

background-color: lightgreen;

text-align: center;

font-size: 25px;

width: 100px;

height: 100px;

padding-top: 20px;

margin: 5px;

}

**</style>**

**</head>**

**<body>**

**<h1>** flex-grow: 0; **</h1>**

**<div** class="container"**>**

**<div** class="flex-item" style = "flex-grow: 0;"**>** flex-item 1 **</div>**

**<div** class="flex-item" style = "flex-grow: 0;"**>** flex-item 2 **</div>**

**<div** class="flex-item" style = "flex-grow: 0;"**>** flex-item 3 **</div>**

**</div>**

**<h1>** flex-grow: 1; **</h1>**

**<div** class="container"**>**

**<div** class="flex-item" style = "flex-grow: 1;"**>** flex-item 1 **</div>**

**<div** class="flex-item" style = "flex-grow: 1;"**>** flex-item 2 **</div>**

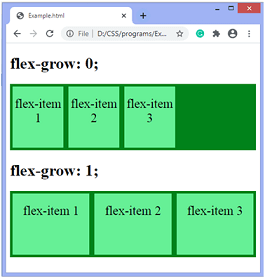
**<div** class="flex-item" style = "flex-grow: 1;"**>** flex-item 3 **</div>**

**</div>**

**</div>**

**</body>**

**</html>**



**Flex-gap:**

The gap property in CSS is a shorthand for row-gap and column-gap, specifying the size of gutters, which is the space between rows and columns within grid, flex, and multi-column layouts.

The gap property defines the size of the gap between the rows and between the columns in flexbox, grid or multi-column layout. It is a shorthand for the following properties:

1. row-gap
2. column-gap

in css gird formaly know as grid-gap

Syntax:

gap: rows column;

[**Syntax**](https://developer.mozilla.org/en-US/docs/Web/CSS/gap#syntax)

CSSCopy to Clipboard

/\* One <length> value \*/

gap: 20px;

gap: 1em;

gap: 3vmin;

gap: 0.5cm;

/\* One <percentage> value \*/

gap: 16%;

gap: 100%;

/\* Two <length> values \*/

gap: 20px 10px;

gap: 1em 0.5em;

gap: 3vmin 2vmax;

gap: 0.5cm 2mm;

/\* One or two <percentage> values \*/

gap: 16% 100%;

gap: 21px 82%;

/\* calc() values \*/

gap: calc(10% + 20px);

gap: calc(20px + 10%) calc(10% - 5px);

/\* Global values \*/

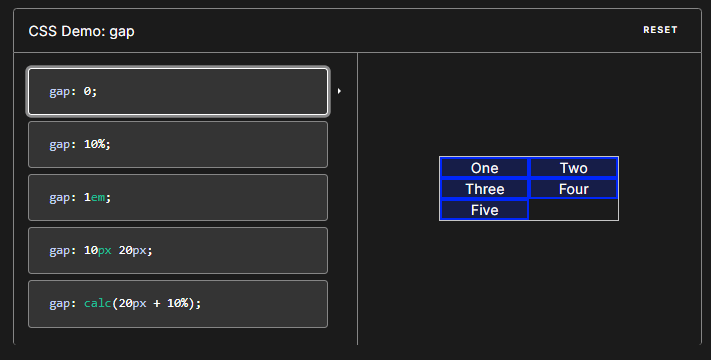
gap: inherit;

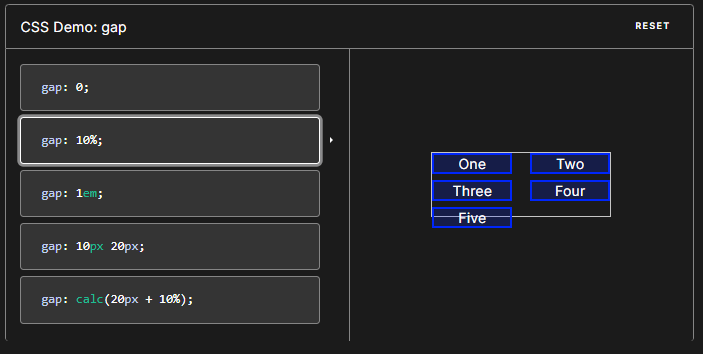
gap: initial;

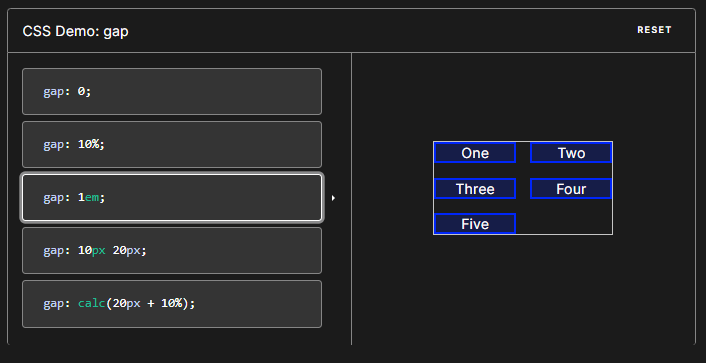
gap: revert;

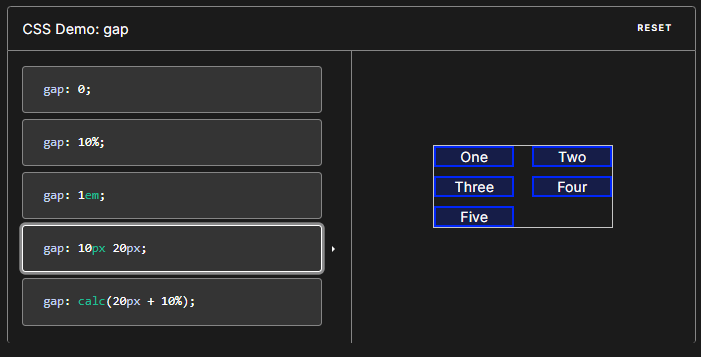
gap: revert-layer;

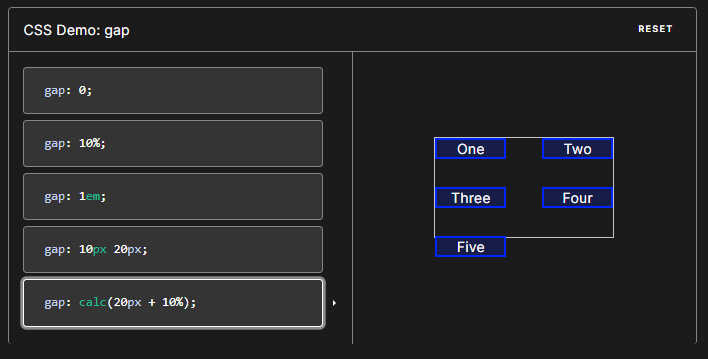
gap: unset;











/\* Grid layout \*/

.container {

display: grid;

grid-template-columns: repeat(3, 1fr);

grid-template-rows: 1fr 2fr 1fr;

gap: 30px 20px;

}

/\* Flex layout \*/

.container {

display: flex;

gap: 10%;

}

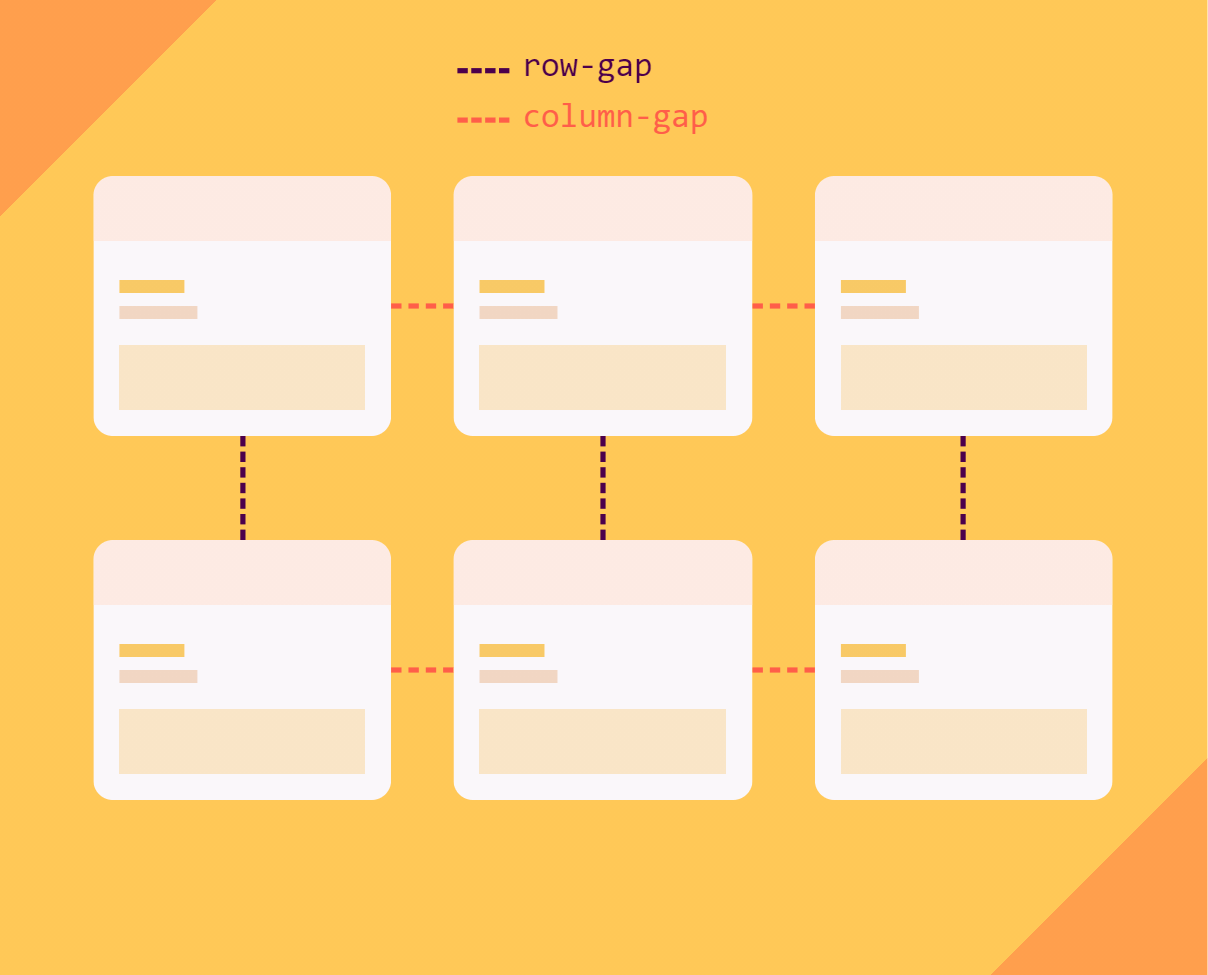
/\* Multi-column layout \*/

.container {

column-count: 5;

gap: 20px;

}



A single value sets both row-gap and column-gap by the same value.

When two values are used, the first sets the row-gap and the second sets the column-gap.

.container {

gap: 1rem;

*/\* Is equivalent to:*

*\* row-gap: 1rem;*

*\* column-gap: 1rem*

*\*/*

gap: 10px 15%;

*/\* Is equivalent to:*

*\* row-gap: 10px;*

*\* column-gap: 15%;*

*\*/*

}

Percentages in gap properties



When the size of a container in the gap dimension is definite, gap resolves percentages against the size of the container’s content box in any layout types.

In other words, when the browser knows the size of the container, it can calculate the percentage value of the gap. For example, when the container’s height is 100px and the gap is set to 10%, browser knows that 10% of 100px is 10px.

But when the browser doesn’t know the size, it will wonder, “10% of what?” In these cases, gap behaves differently based on the layout type.

In a grid layout, percentages resolve against zero for determining intrinsic size contributions, but resolve against the element’s content box when laying out the box’s contents, meaning it will put space between items but the space doesn’t affect the container’s size.

In this demo, the container’s height is not definite. See what happens when you increase the gap size. Then set the gap in pixel units and try again:

**Using the calc() function with gap**

You can use calc() function to specify the size of the gap but, at the time of this writing, there is no support for it on Safari and iOS.

.flex-layout {

display: flex;

gap: calc(5vh + 5px) calc(5vw + 5px);

}

.grid-layout {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: calc(5vmin + 5px);

}

**Flex-flow:**

The flex-flow CSS shorthand property specifies the direction of a flex container, as well as its wrapping behavior.

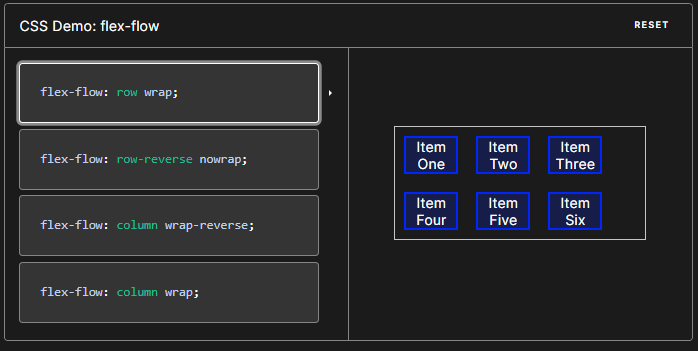
The flex-flow property is a shorthand property for:

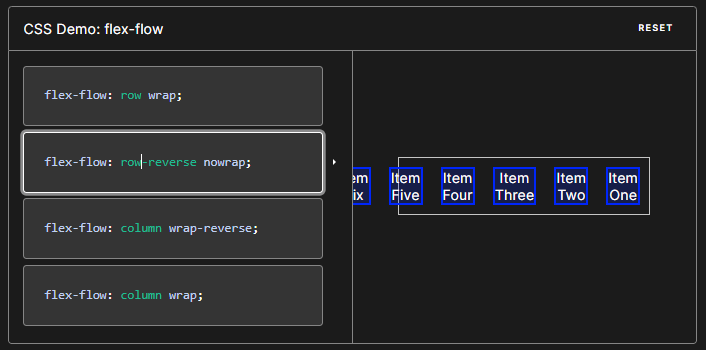
* flex-direction
* flex-wrap

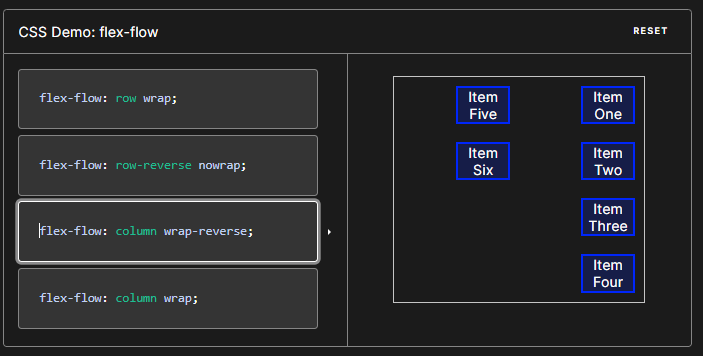
Note: If the elements are not flexible items, the flex-flow property has no effect.

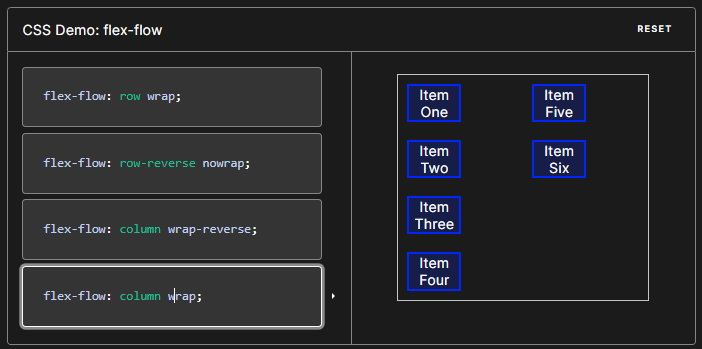
Syntax

flex-flow: <‘flex-direction’> || <‘flex-wrap>;









**Syntax**

CSS

/\* flex-flow: <'flex-direction'> \*/

flex-flow: row;

flex-flow: row-reverse;

flex-flow: column;

flex-flow: column-reverse;

/\* flex-flow: <'flex-wrap'> \*/

flex-flow: nowrap;

flex-flow: wrap;

flex-flow: wrap-reverse;

/\* flex-flow: <'flex-direction'> and <'flex-wrap'> \*/

flex-flow: row nowrap;

flex-flow: column wrap;

flex-flow: column-reverse wrap-reverse;

/\* Global values \*/

flex-flow: inherit;

flex-flow: initial;

flex-flow: revert;

flex-flow: revert-layer;

flex-flow: unset;

Example:

HTML

<h1>flex-wrap: wrap; flex-direction: row;</h1>

<ul class="flex-container longhand">

<li class="flex-item">1</li>

<li class="flex-item">2</li>

<li class="flex-item">3</li>

<li class="flex-item">4</li>

</ul>

<h1>flex-flow: row wrap;</h1>

<ul class="flex-container shorthand">

<li class="flex-item">1</li>

<li class="flex-item">2</li>

<li class="flex-item">3</li>

<li class="flex-item">4</li>

</ul>

CSS

.flex-container {

padding: 0;

margin: 0;

list-style: none;

-ms-box-orient: horizontal;

display: -webkit-box;

display: -moz-box;

display: -ms-flexbox;

display: -moz-flex;

display: -webkit-flex;

display: flex;

}

h1 {

padding-left: .5em;

}

.shorthand {

-webkit-flex-wrap: wrap;

flex-wrap: wrap;

-webkit-flex-direction: row;

flex-direction: row;

}

.longhand {

-webkit-flex-flow: wrap row;

flex-flow: wrap row;

}

.longhand li {

background: deepskyblue;

}

.flex-item {

background: tomato;

padding: 5px;

width: 100px;

height: 100px;

margin: 10px;

line-height: 100px;

color: white;

font-weight: bold;

font-size: 2em;

text-align: center;

}

