gap

Mojtaba Seyedi on Aug 13, 2020 (Updated on Apr 29, 2021)

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 (https://css-tricks.com/snippets/css/complete-guide-grid/) , flex (https://css-tricks.com/almanac/properties/f/flex/) , and multi-column (https://css-tricks.com/almanac/properties/c/columns/) layouts.

```css
/* 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;
}
```

Use the slider in the demo below to see the `gap` property in action:

Embedded Pen Here

(#syntax) Syntax

`gap` accepts one or two values:

- 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`.

```css
.container {
  gap: 1rem;
  /* Is equivalent to:
  *  row-gap: 1rem;
  *  column-gap: 1rem
  */

  gap: 10px 15%;
  /* Is equivalent to:
  *  row-gap: 10px;
  *  column-gap: 15%;
  */
}
```

> ***Hey!*** The specification for the CSS Grid Layout Module defined the space between grid tracks using the `grid-gap` property. `gap` is intended to replace it (https://css-tricks.com/chromium-lands-flexbox-gap/) so that gaps can be defined in multiple CSS layout methods, like flexbox, but `grid-gap` still needs to be used in instances where a browser may have implemented `grid-gap` but has yet to start supporting the newer `gap` property.

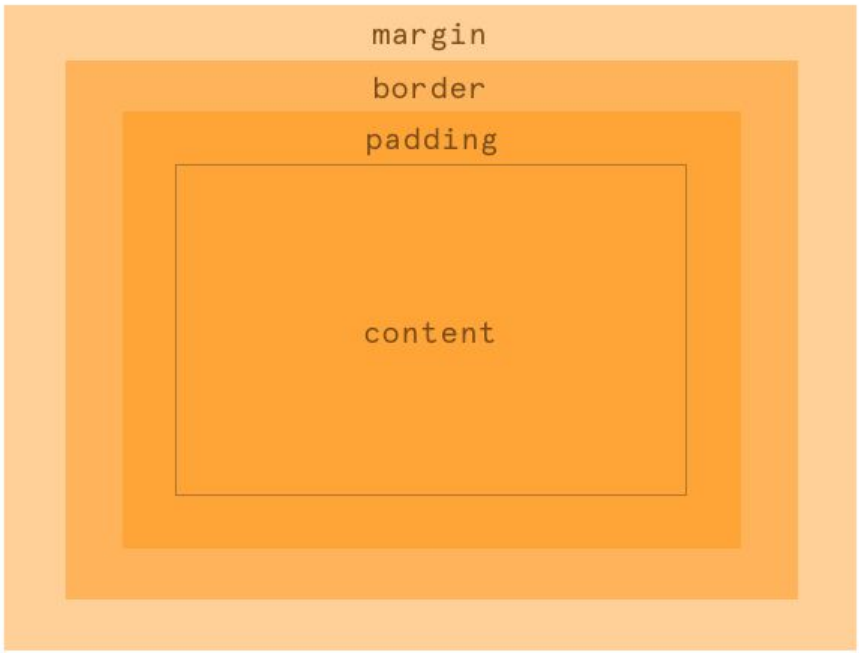
(#values) Values

`gap` accepts the following values:

- **`normal`:** (Default) The browser will specify a used value of `1em` for multi-column layout and `0px` for all other layout contexts (i.e. grid and flex).
- **`<length>`:** Any valid and non-negative CSS length, such as `px`, `em`, `rem` and `%`, among others.
- **`<percentage>`:** The size of the gap as a non-negative percentage value relative to the dimension of the element. (See below for details.)
- **`initial`:** Applies the property's default setting, which is `normal`.
- **`inherit`:** Adopts the gap value of the parent.
- **`unset`:** Removes the current `gap` from the element.

(#percentages-in-gap-properties) 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:

Embedded Pen Here

In a flex layout, percentages resolve against zero in all cases, meaning that gaps will not apply when the size of the container is not known to the browser:

Embedded Pen Here

(#using-the-calc-function-with-gap) Using the calc() function with gap

You can use `calc()` function (https://css-tricks.com/a-complete-guide-to-calc-in-css/) to specify the size of the `gap` but, at the time of this writing, there is **no support for it** (https://caniuse.com/%23feat=mdn-css_properties_gap_grid_context_calc_values) **on Safari and iOS.**

```css
.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);
}
```

(#examples) Examples

The `gap` property is designed for use in grid, flex and multi-column layouts. Let's check out some examples.

(#grid-layout) Grid layout

In the following demo, you can see `gap` being used to specify the `row-gap` and `column-gap` properties on a grid container, defining the gutters between grid rows and grid columns, respectively:

Embedded Pen Here

(#flex-layout) Flex layout

Applying gap to the **main axis** of a flex container indicates spacing **between flex items** in a single line of the flex layout.

Here's `column-gap` used in a row direction:

Embedded Pen Here

Here's `row-gap` used in a column direction:

Embedded Pen Here

Applying `gap` to the **cross axis** of a flex container indicates spacing **between flex lines** of the flex layout.

Here's `row-gap` in a row direction:

Embedded Pen Here

Here's `column-gap` in a column direction:

Embedded Pen Here

(#multi-column-layout) Multi-column layout

`column-gap` appears in multi-column (https://css-tricks.com/almanac/properties/c/columns/) layouts to create gaps between column boxes, but note that `row-gap` has no effect since we're only working in columns. `gap` can still be used in this context, but only the `column-gap` will be applied.

As you can see in the next demo, although the `gap` property has a value of 2rem, it's only separating items horizontally instead of both directions since we're working in columns:

Embedded Pen Here

(#the-more-you-know) The more you know...

There are a couple of things worth noting about working with the `gap` property.

(#its-a-nice-way-to-prevent-unwanted-spacing) It's a nice way to prevent unwanted spacing

Have you ever used margins to create spacing between elements? If we're not careful, we can end up with extra spacing before and after the group of items.

Embedded Pen Here

Solving that usually requires adding negative margins or resorting to pseudo-selectors to remove margin from specific items. But the nice thing about using `gap` in more modern layout methods is that you only have space *between* items. The extra cruft at the start and end is never an issue!

Embedded Pen Here

But, hey, if you *want* to have space around the items while using `gap`, put `padding` around the container like this:

```css
.container {
  display: flex;
  gap: 1rem;
  padding: 1rem;
}
```

(#gutter-size-is-not-always-equal-to-the-gap-value) Gutter size is not always equal to the gap value

The `gap` property is not the only thing that can put space between items. Margins, paddings, `justify-content` and `align-content` can also increase the size of the gutter and affect the actual `gap` value.

In the following example, we're setting a `1em` `gap` but, as you can see, there is much more space between the items, caused by the use of distributed alignments, like `justify-content` and `align-content`:

Embedded Pen Here

(#browser-support) Browser support

Feature queries are usually a nice way to check if a browser supports a specific property, but in this case, if you check for the `gap` property in flexbox, you may get a false positive because a feature query won't distinguish between layout modes. In other words, it might be supported in a flex layout which results in a positive result, but it is actually not supported if it's used in a grid layout.

(#grid-layout) Grid layout

IE	Edge	Firefox	Chrome	Safari	Opera
No	16+	61+	66+	12+	53+

iOS Safari	Opera Mobile	Android Browser	Chrome for Android	Firefox for Android
12+	No	81+	84+	68+

(#grid-layout-with-calc-values) Grid layout with calc() values

IE	Edge	Firefox	Chrome	Safari	Opera
No	84+	79+	84+	No	69+

iOS Safari	Opera Mobile	Android Browser	Chrome for Android	Firefox for Android
No	No	81+	84+	68+

(#grid-layout-with-percentage-value) Grid layout with percentage value

IE	Edge	Firefox	Chrome	Safari	Opera
No	84+	79+	84+	No	69+

iOS Safari	Opera Mobile	Android Browser	Chrome for Android	Firefox for Android

iOS Safari	Opera Mobile	Android Browser	Chrome for Android	Firefox for Android
No	No	81+	84+	68+

(#flex-layout) Flex layout

The specification (https://www.w3.org/TR/css-align-3/%23gaps) for using `gap` with flexbox is currently in Working Draft status.

This browser support data is from Caniuse (https://caniuse.com/#feat=”flexbox-gap”) , which has more detail. A number indicates that browser supports the feature at that version and up.

Desktop

Chrome: 84 **Firefox:** 63 **IE:** No **Edge:** 84 **Safari:** 14.1

Mobile / Tablet

Android Chrome: 90 **Android Firefox:** 87 **Android:** 90 **iOS Safari:** 14.5-14.6

(#multi-column-layout) Multi-column layout

IE	Edge	Firefox	Chrome	Safari	Opera
No	84+	79+	84+	No	69+

iOS Safari	Opera Mobile	Android Browser	Chrome for Android	Firefox for Android
No	No	81+	84+	68+

(#more-information) More information

- CSS Box Alignment Module Level 3 (https://drafts.csswg.org/css-align-3/%23gaps)
- Chromium lands Flexbox gap (https://web.dev/flexbox-gap/) (Ticket #761904 (https://bugs.chromium.org/p/chromium/issues/detail?id=761904))
- WebKit CSS Feature Status (https://webkit.org/css-status/#property-gap)

(#related) Related

- Grid Layout (https://css-tricks.com/snippets/css/complete-guide-grid)
- Flexbox Layout (https://css-tricks.com/snippets/css/a-guide-to-flexbox)
- Multi-Column Layout (https://css-tricks.com/almanac/properties/c/column-gap)

PROPERTIES

> A
> B
> C
> D
> E
> F
> G
> H
> I
> J
> L
> M
> O
> P
> Q
> R
> S
> T
> U
> V
> W
> Z

SELECTORS

> A
> B
> C
> D
> E
> F
> G
> H
> I
> L
> M
> N
> O
> P
> R
> S
> T
> U
> V
> W