medium.com

The CSS Grid Enchiridion

Stephen Koo

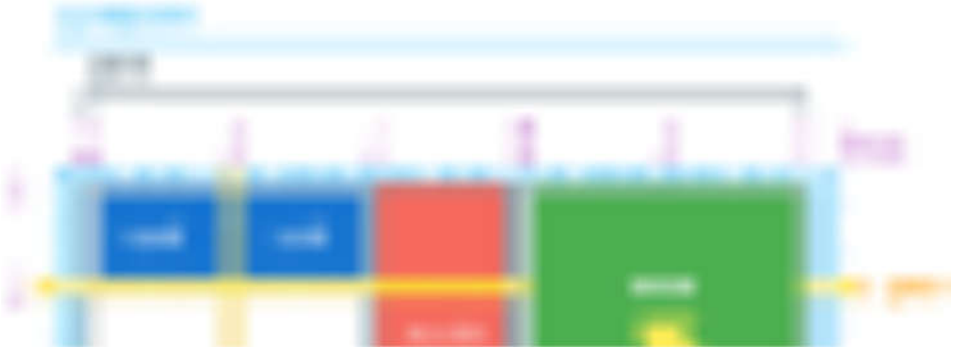
4-5 minutos

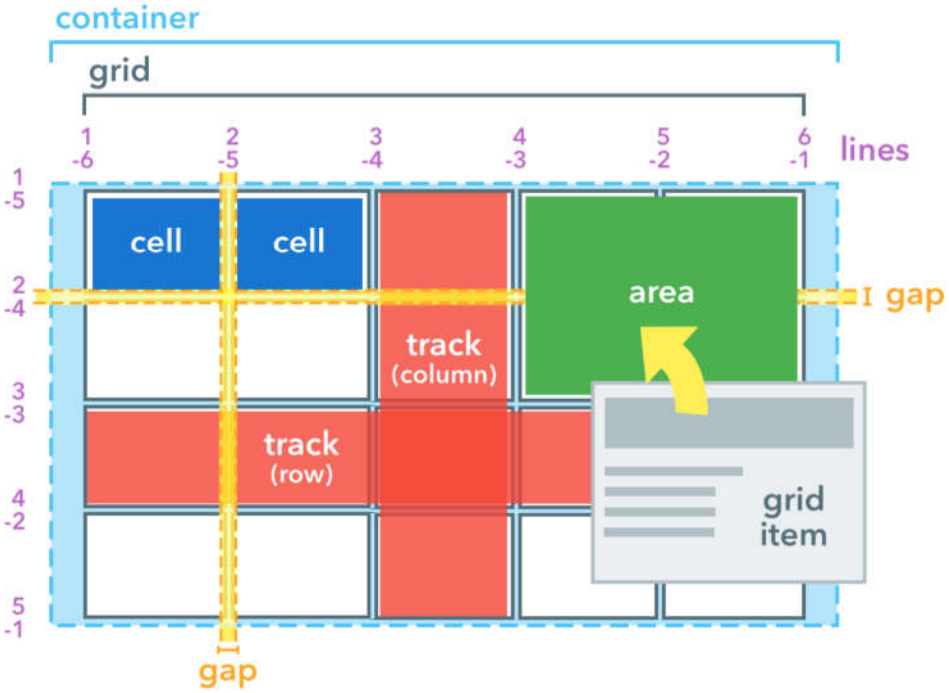
Developer Handbook to CSS Grid

A handy reference guide on how to use the CSS Grid.

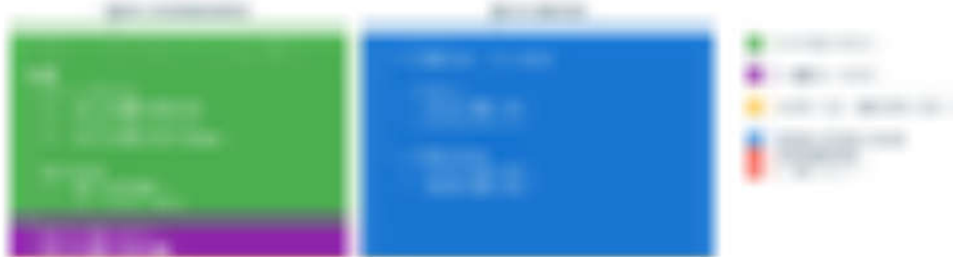
- MDN | grid
- CSS-Tricks |A Complete Guide to CSS Grid
- Grid by Example | Grab & Go Patterns
- Grid by Example | Examples

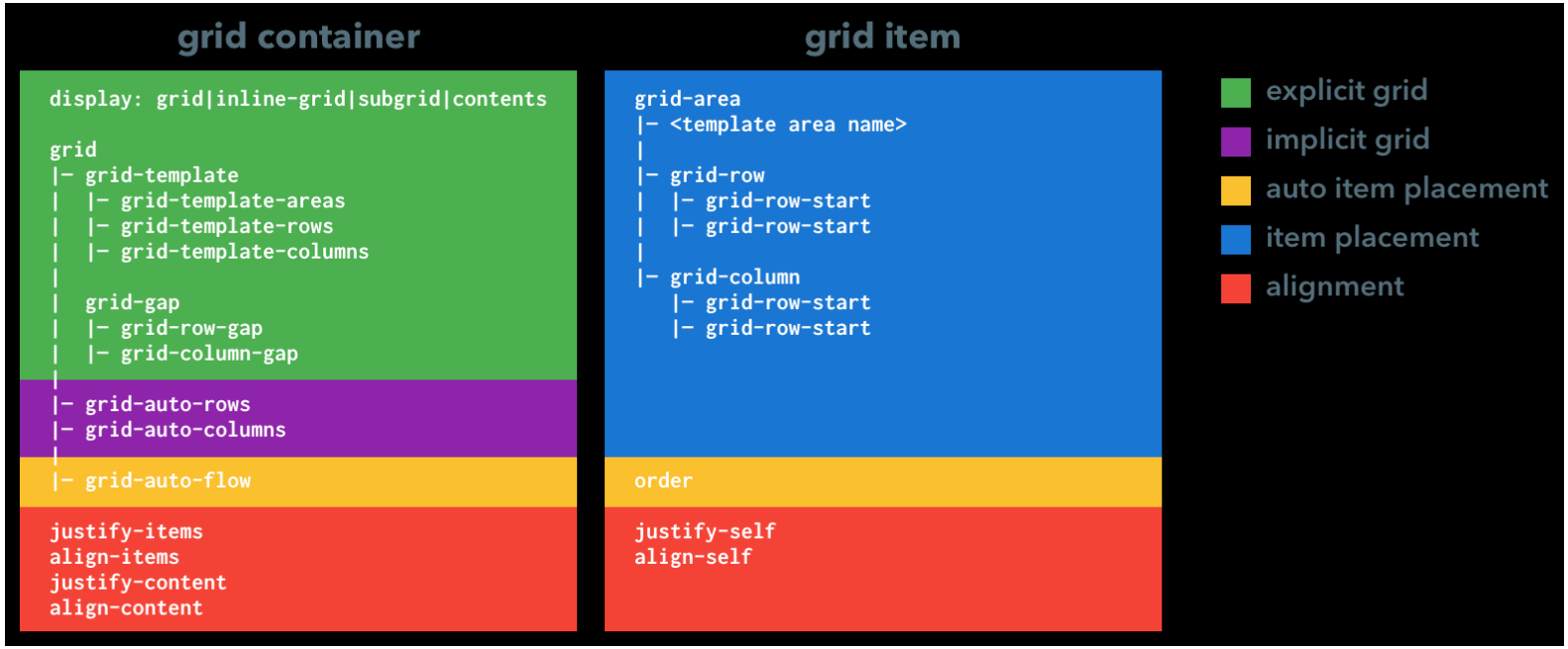
Grid elements





Grid properties



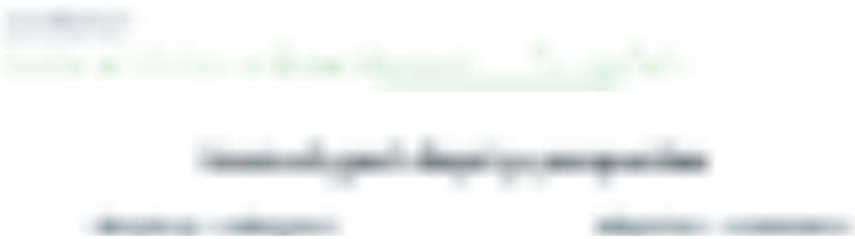


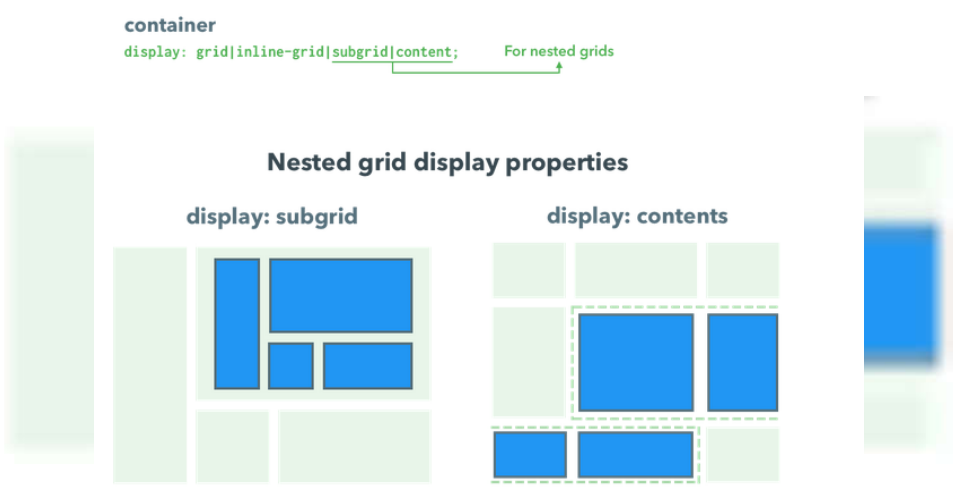
Ensure browser grid support

Prevent IE (which uses an older Grid spec) from using CSS Grid by checking if a browser supports any of these new Grid spec properties:

```
@supports (grid-area: auto)                 { ... }
@supports (grid-column-gap: 1px)            { ... }
@supports (grid-auto-flow: dense)           { ... }
@supports (grid-auto-rows: max-content)     { ... }
@supports (grid-auto-columns: min-content)  { ... }
```

1.1. Make an item a grid





1.2. Choose whether grid is explicit, implicit, or both

If you want:

- A **fixed amount** of columns and rows — explicit grid
- **Automatically created** columns or rows based on the amount of items — implicit grid
- A **fixed amount** of columns and rows **+ automatically created** columns or rows if there are more items than your explicit grid can contain (or when you place items outside your explicit grid) — explicit + implicit grid

1A. Create an explicit grid

container

```
grid
|- grid-template: grid-template-areas grid-template-rows / grid-template-columns;
|- grid-gap:      grid-row-gap / grid-column-gap;
```

```
/* === GRID TRACKS === */
grid-template-rows:    100px 30vmin 20vmax 5em  6rem;
grid-template-columns: 1fr   20%    25vw   30vh;
                       Line 1 2     3      4    5    6
grid-template: 100px 30vmin 20vmax 5em 6rem / 1fr 20% 25vw 30vh; // Shorthand

/* === GRID GAPS === */
grid-gap: 20px 20px;
grid-gap: 20px; // Shorthand
```

- `grid-template` sets the number and size of the tracks
- `grid-gap` sets the size of the gaps

1A.1 Track sizes: Simple units

> fr, %, px, rem, em, vh, vw, vmin, vmax, none

- `fr` takes a fraction of the available container's free space
- `fr` factors in gaps, `%` doesn't — potentially causing container overflow

  % can cause containers to be larger than screen width, fr won't

1A.2 Track sizes: Content-based units

> max-content, min-content

- `min-content`: Size of item when wrapped
- `max-content`: Size of item when unwrapped

1A.3 Track sizes: Flexible values

> minmax(), auto, fit-content()

- `minmax()` sets a minimum and maximum size for a track
- `auto` acts like:
  (a) `minmax(min-content, max-content)` if used alongside `fr`.
  Max size == unwrapped content size
  (b) `minmax(min-content, 1fr)` otherwise. Max size == free

space

- In short, `auto` takes up space unless `fr` exists

  auto in the presence & absence of fr tracks

- `fit-content(argument)` acts like `minmax(auto, the smaller of max-content or argument)`

- Remember: As `auto` acts differently in the presence of `fr`, so too will `fit-content()`.

  fit-content() in the presence & absence of fr tracks

1A.4 Repeated tracks

repeat(int|auto-fit|auto-fill, track-size)

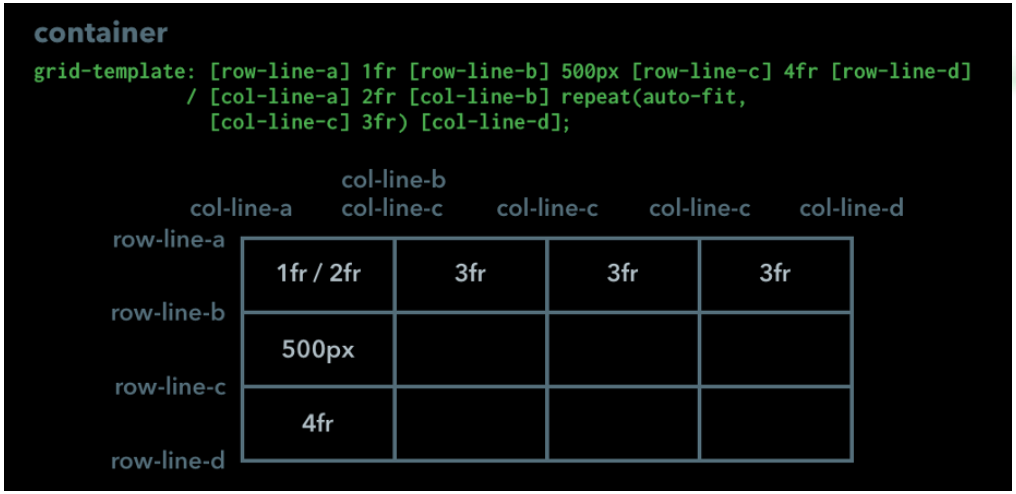
- Instead of `grid-template-columns: 200px 400px 400px 400px`, we can use `grid-template-columns: 200px repeat(3, 400px)`

- To repeat tracks until the container is full, use `auto-fit` or `auto-fill`

- The only difference between `auto-fit` and `auto-fill` occurs when:
  (A) Using `repeat(auto-fit|auto-fill, minmax(_, _fr))`
  &&
  (B) The grid is wide enough to fit all items into a single row plus more
  ...`auto-fit` will spread out existing tracks;
  ...`auto-fill` will try to squeeze in empty tracks

1A.5 Labelling grid lines

- You can name grid lines to make it easy to position items by referring to a line's name
- Note: Line names are currently not Sass-friendly, so don't use them in sass/scss files just yet



1B. Create an implicit grid

- Extra tracks are created for item overflow (excess items that don't fit in your explicit grid, or if you didn't make an explicit grid)

`grid-auto-rows` automatically creates rows when:

(A) item overflow + `grid-auto-flow: row` (default) OR

(B) you manually place an item on a row beyond the explicit grid

`grid-auto-columns` automatically creates columns when:

(A) item overflow + `grid-auto-flow: column` (see: *2A. Auto item placement* below) OR

(B) you manually place an item on a column beyond the explicit grid

- You can use all units available in the explicit grid, except `repeat()`:

  fr, %, px, rem, em, vh, vw, vmin, vmax, minmax(), auto, fit-content()

- The sequence of track sizes set in `grid-auto-rows` and `grid-auto-columns` will be looped through until all necessary tracks are created.

1C. Create an explicit + implicit grid

You can use an explicit and implicit grid together:

```
container
.grid {
    grid-template: repeat(2, 200px) / repeat(3, 200px);
    grid-auto-rows: 100px;
}
// After the first two 200px rows, 100px rows will be created for additonal items
```

2A. Auto-place items on the grid

- `grid-auto-flow: column` auto-places items in grid by column (from top to bottom, then to the right column).
- `grid-auto-flow: row` (the default) auto-places items in grid by row (from left to right, then to the row below).
- `order` takes items out of their original order and places them in front or behind the rest of the items. Items have `order: 0`by default, so assigning an order <0 puts them in front, >0 puts them behind.

2.1. Position item by cell name

- You can place items in grid areas by name.

```
container
grid
|- grid-template: grid-template-areas grid-template-rows / grid-template-columns;

item
grid-area: <template-area-name>;
```

2.2 Position item by line

- You can also position items by line number

```
item
grid-area: grid-row-start / grid-column-start / grid-row-end / grid-column-end;
|- grid-row: grid-row-start / grid-row-end;
|- grid-column: grid-column-start / grid-column-end;
```