
CSS Gridish



CSS Gridish takes design specs of your product's grid and builds out several resources for your team to use:

- Sketch file with artboards and grid/layout settings for designers
- CSS/SCSS code using CSS Grid with a CSS Flexbox fallback for developers
- Google Chrome extension for anyone to check a webpage's alignment

This tool is not a grid system with a grid already designed for you. Instead, CSS Gridish builds all of the resources for the grid your team designed.

We hope it helps teams adapt CSS Grid sooner and enables more complex layouts. To show how versatile the tool is, we have examples of grids from Bootstrap, Carbon Design System, and Material Design.

The truth is that many enterprise projects can't afford to drop support for browsers that do not support CSS Grid Layout yet. This tool takes your grid's design specs and builds a slim CSS Grid Layout implementation and a fallback to CSS Flexbox support.

Build your grid code

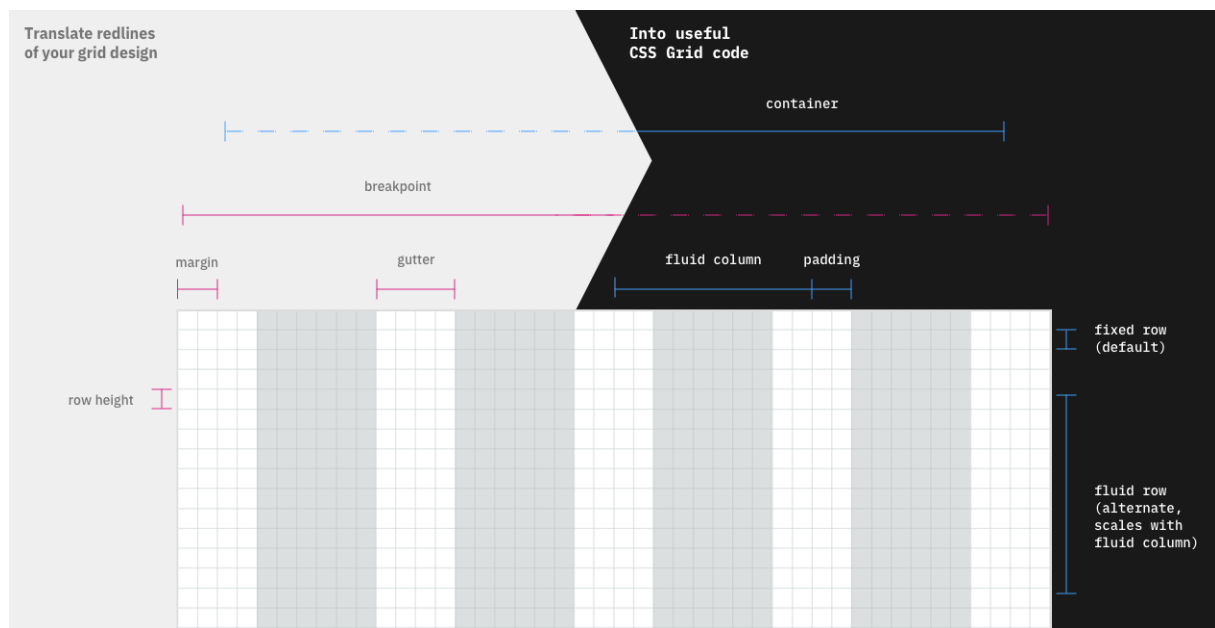
Requires Node v8.2.0 or higher, which includes npm and npx.

1. Create a file called `css-gridish.json` in your project root. See the config documentation or an example config for help.
2. Determine how you want your grid built:
 - If you want to build once, run command `npx css-gridish`.
 - If you want to add the grid building to your project's build process:

1. Run command `npm install css-gridish`.
2. Add `scripts: {build: "css-gridish"}` in your `package.json`.
3. Run command `npm run build`.

Your CSS and `README.md` with class documentation will be built into `./css-gridish/`.

Config file



The config file is where all of your design system specs live. See this example for help. Edit your `css-gridish.json` to have all generated grid content match your design system:

```
1 {
2   "prefix": "gridish", // Custom prefix for all classes and filenames
3   "breakpoints": {
4     // Class name for a breakpoint
5     "sm": {
6       "breakpoint": 20, // Min-width for media query (number in rem)
7       "columns": 4, // Quantity of columns (number)
8       "gutter": "2rem", // Space between columns (rem string, px string
9       // Horizontal margin of grid container (rem
10      "margin": "3vw" string, px string, vw string or 0)
11    },
12    ...
13    "max": {
14      "breakpoint": 100,
15      "columns": 12,
16      "gutter": "4rem",
```

```

16     "margin": "5vw"
17   }
18 },
19 "extraArtboards": {
20   "xlg": 100 // Additional breakpoint for the Sketch file (number in
              rem)
21 },
22 "rem": 16, // Base rem unit for all measurements (number in px)
23 "rowHeight": 0.5, // Height of a fixed row (number in rem)
24 "rows": 30, // Quantity of row variables (number)
25 "paths": {
26   "route": "css-gridish", // Route that files save in from project
                          root (optional, use `""` for project root, `css-gridish` is
                          default)
27   "intro": "intro.md" // Path to any markdown you want inserted at
                          the top of your README.md documentation (optional)
28 }
29 }

```

Tip: For the best results in Sketch, we recommend you make your grid breakpoints, margin, and gutter divisible by the row height.

Required: Even if your design specs do not change between breakpoints, you need to list the max-width breakpoint in the `breakpoints` object.

The first breakpoint min-width media query is not used to save kilobytes, but we recommend stating it anyways for future artboard-making tools.

Legacy support

If you are supporting browsers that lack CSS Grid Layout support, you can use `css-gridish/yourPrefix-legacy.min.css` and the legacy classes detailed in the `README.md`. With the legacy file and classes, the browsers that do not support the final CSS Grid Legacy spec will fallback to a CSS Flexbox alternative. The CSS Flexbox alternative supports embedded subgrids that still reflect the overall grid system's column structure.

User-defined breakpoints

One of the best parts about CSS Grid Layout is that your users can rearrange the layout at any width in their own media query. Your grid will also support rearranging layout at custom breakpoints for the legacy implementation when the user compiles their own Sass. Just have them define the following map of rem widths before they import in your Sass file:

```

1 $extraBreakpoints: (

```

```
2   xsm: 10,  
3   whatever: 78,  
4   superxlarge: 1000,  
5   ...  
6 );  
7 @import './css-gridish/scss/yourPrefix-legacy.scss';
```

Transitioning from Legacy

Once your experience can drop support for browsers like IE 11 and Edge <15, you can simply remove all legacy classes and switch over to the non-legacy files. This is a great progressive-enhancement for your performance when it happens.

Future Updates

- ☐ Once Edge and Safari gain `display: subgrid` support, we can remove our dependence on `vw` units.
- ☐ Once a solution in the CSS Grid spec is given for one item to ignore `grid-gap`, we can utilize `grid-gap` by default instead of opting in to padding classes.

FAQs

Why does none of the CSS Grid code use `grid-gap` for gutters?

A lot of times, you will want an item to break out of the gutters for background color, to extend media, or for another reason. Until the CSS Grid spec has a way to ignore that gutter, we use the padding classes (`.yourGrid-padding`) to opt-in to respecting the gutter. The padding classes are always half the size of a gutter for alignment.

Why are columns using `vw` units and sometimes the `calc` function?

Until Edge and Safari support `display: subgrid`, it will be difficult for you to write semantic HTML with CSS Grid Layout. We are able to take advantage of `vw` units and the `calc` function so you can embed your `.yourPrefix-grid` class inside of itself as much as is needed for you.

Why are there no row classes for the legacy implementation?

Thanks to flexbox's wrapping functionality, nodes that specify rows are not necessary. Only create a node for a row if it has semantic or accessibility significance. You can keep embedding `.yourPrefix-grid` as necessary to accomplish this.

What happens in the legacy implementation if I specify the column width for one breakpoint, but not the next larger breakpoint?

To maintain a mobile-first opinion, column widths will scale to the next breakpoint if not specified. This means that a `.yourPrefix__col--sm--3` (with 6 total columns) would automatically grow into a `.yourPrefix__col--md--6` (with 12 total columns) if no `md` class was declared to maintain half of the screen size.