Overcoming Grid Reluctance

By Chen Hui Jing / @hj_chen
Chen Hui Jing

@hj_chen
Screens, screens, screens

Image source: Inch Calculator
```css
.wrapper {
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  display: flex;
}

MDN: Backwards Compatibility of Flexbox
```
Browser configuration pages

about:config

chrome://flags
# Grid release dates

## March 2017

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>1</strong></td>
<td><strong>2</strong></td>
<td><strong>3</strong></td>
<td><strong>4</strong></td>
</tr>
<tr>
<td><strong>5</strong></td>
<td><strong>6</strong></td>
<td>![Browser Icon]</td>
<td><strong>8</strong></td>
<td><strong>9</strong></td>
<td><strong>10</strong></td>
<td><strong>11</strong></td>
</tr>
<tr>
<td><strong>12</strong></td>
<td><strong>13</strong></td>
<td>![Browser Icon]</td>
<td><strong>15</strong></td>
<td><strong>16</strong></td>
<td><strong>17</strong></td>
<td><strong>18</strong></td>
</tr>
<tr>
<td><strong>19</strong></td>
<td><strong>20</strong></td>
<td>![Browser Icon]</td>
<td><strong>22</strong></td>
<td><strong>23</strong></td>
<td><strong>24</strong></td>
<td><strong>25</strong></td>
</tr>
<tr>
<td><strong>26</strong></td>
<td>![Browser Icon]</td>
<td><strong>28</strong></td>
<td><strong>29</strong></td>
<td><strong>30</strong></td>
<td><strong>31</strong></td>
<td></td>
</tr>
</tbody>
</table>

## October 2017

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>1</strong></td>
<td><strong>2</strong></td>
<td><strong>3</strong></td>
<td><strong>4</strong></td>
</tr>
<tr>
<td><strong>5</strong></td>
<td><strong>6</strong></td>
<td><strong>7</strong></td>
<td><strong>8</strong></td>
<td><strong>9</strong></td>
<td><strong>10</strong></td>
<td><strong>11</strong></td>
</tr>
<tr>
<td><strong>12</strong></td>
<td><strong>13</strong></td>
<td><strong>14</strong></td>
<td><strong>15</strong></td>
<td><strong>16</strong></td>
<td>![Browser Icon]</td>
<td><strong>18</strong></td>
</tr>
<tr>
<td><strong>19</strong></td>
<td><strong>20</strong></td>
<td><strong>21</strong></td>
<td><strong>22</strong></td>
<td><strong>23</strong></td>
<td><strong>24</strong></td>
<td><strong>25</strong></td>
</tr>
<tr>
<td><strong>26</strong></td>
<td>![Browser Icon]</td>
<td><strong>29</strong></td>
<td><strong>30</strong></td>
<td>![Browser Icon]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### CSS Flexible Box Layout Module

Method of positioning elements in horizontal or vertical stacks. Support includes all properties prefixed with 'flex', as well as 'display: flex', 'display: inline-flex', 'align-content', 'align-items', 'align-self', 'justify-content' and 'order'.

<table>
<thead>
<tr>
<th></th>
<th>Edge</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>iOS Safari</th>
<th>Opera Mini</th>
<th>Chrome for Android</th>
<th>Android Browser</th>
<th>Samsung Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>105</td>
<td>104</td>
<td>105</td>
<td>15.6</td>
<td>15.6</td>
<td>4.4</td>
<td>4.4</td>
<td>16.0</td>
<td>16.0</td>
</tr>
<tr>
<td>10</td>
<td>106</td>
<td>105</td>
<td>106</td>
<td>16.0</td>
<td>19.0</td>
<td>4.4</td>
<td>4.4</td>
<td>17.0</td>
<td>18.0</td>
</tr>
<tr>
<td>11</td>
<td>107</td>
<td>106</td>
<td>107</td>
<td>16.1</td>
<td>all</td>
<td>107</td>
<td>107</td>
<td>18.0</td>
<td></td>
</tr>
</tbody>
</table>

Data from caniuse.com | Link from caniuse.bitfox.de

### CSS Grid Layout (Level 1)

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all ‘grid-*’ properties and the ‘fr’ unit.

<table>
<thead>
<tr>
<th></th>
<th>Edge</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>iOS Safari</th>
<th>Opera Mini</th>
<th>Chrome for Android</th>
<th>Android Browser</th>
<th>Samsung Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>105</td>
<td>104</td>
<td>105</td>
<td>15.6</td>
<td>15.6</td>
<td>4.4</td>
<td>4.4</td>
<td>16.0</td>
<td>16.0</td>
</tr>
<tr>
<td>10</td>
<td>106</td>
<td>105</td>
<td>106</td>
<td>16.0</td>
<td>16.0</td>
<td>4.4</td>
<td>4.4</td>
<td>17.0</td>
<td>18.0</td>
</tr>
<tr>
<td>11</td>
<td>107</td>
<td>106</td>
<td>107</td>
<td>16.1</td>
<td>all</td>
<td>107</td>
<td>107</td>
<td>18.0</td>
<td></td>
</tr>
</tbody>
</table>

Data from caniuse.com | Link from caniuse.bitfox.de
State of CSS 2021 survey

Flexbox: 98.9%

Grid: 83.5%

Disclaimer

The following theory may or may not oppose your view on the matter, and that is PERFECTLY FINE. I am not here to tell you what to think, merely here to share a theory based on my personal observations and experiences. You are absolutely free to agree, disagree or not care at all.

Source: The History of the Dotcom Bubble
Welcome to Michel's Video Games page!

Last update: 16 September 1994

These pages have been consulted 1088 times September 15th 1994!

Fast jumps to sections: What's New, 3DO, Doom, Jrpgame, SNES, Pinball, Netrek, GameBoy, Shockwave(Old), Way Of The Warrior(Old), the Video Game FAQ Page

This page is updated daily!! Check the What's New page to see the recent additions.

Introduction: Who am I?

I removed my digitized picture but if you really want to see my ugly face, check my ugly face page! Other pictures and info are available in my personal home page.

Well, I'm a poor French lost in the United States for a year. I had to leave my SNES in France so I bought a 3DO as soon as I could when I arrived in Pittsburgh. I don't regret it. See the 3DO and SNES related stuff further in this page. Check also The Alien Jaguar links. Netrek and Doom fans may find some interesting links too. Pinball players will find the best infos for their favorite pinballs too! Arcade Game players can't miss my Video Game FAQ Page.

Please send me suggestions and interesting links by email at:

<Michel.Buffa@cmu.edu>

First, some interesting links for video game addicts:

Check these sites!

- The Games Domain home page (PC, XBox, games, others...) Contains tons of FAQs in html format, lots of links to other games-related WWW pages. Tons of links there!
- The official Game Bytes WWW page! Brand New on the WWW! Game Bytes is an electronic fanzine that talks mainly about the PC and computer games but also has columns for the consoles (SNES, Genesis, Sega CD, 3DO).
- Video Games FAQs ftp server. Mortal Combat I and II, NBA Jam, Art of Fighting, lots of Infos about real arcade games and console/computer games. Some of the files available there in ASCII have been translated by me in mosaic format. Check my Video Game FAQ Page.
- Check out the video game mosaic pages from Cardiff. This site contains lots of FAQs, pictures, info about video games for all type of machines.
- ftp.dipnet.net. Almost anything you can imagine for most platforms. New site.
- samite.unc.edu. Sega stuff only, including FAQs, electronic publications etc.
- ftp.an.psu.edu. 3DO material, mostly lots of game screenshots in JPEG format.
- rfm.unl.edu. For video game news/organ archives.

Source: Michel Buffa's Video Games Page in 1994
Parent-child relationship

Flex/Grid container

Flex/Grid item  Flex/Grid item  Flex/Grid item
Basic grid syntax

```
.basic-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 1em;
}
```
# Named grid areas

<table>
<thead>
<tr>
<th>Links and stuff?</th>
<th>Banner</th>
<th>Your main content</th>
</tr>
</thead>
</table>

Footer, for copyright and moar links?

```css
.named-grid {
  display: grid;
  grid-template-columns: 200px 1fr;
  grid-template-rows: auto 1fr auto;
  grid-template-areas: 'header header'
                        'sidebar main'
                        'footer footer';
}

.h { grid-area: header }
.s { grid-area: sidebar }
.m { grid-area: main }
.f { grid-area: footer }
```
Placing grid items

Not my cat

He just naps in my house sometimes. But isn't he super cute?
Container-led sizing

Item-led sizing
Common example of grid system CSS

“Bootstrap’s grid system uses a series of containers, rows, and columns to layout and align content.”

```css
.col-sm {
    flex: 1 0 0%;
}
.row-cols-sm-auto > * {
    flex: 0 0 auto;
    width: auto;
}
.row-cols-sm-1 > * {
    flex: 0 0 auto;
    width: 100%;
}
.row-cols-sm-2 > * {
    flex: 0 0 auto;
    width: 50%;
}
.row-cols-sm-3 > * {
    flex: 0 0 auto;
    width: 33.33%;
}
```

```html
<div class="container">
    <div class="row">
        <div class="col-sm-4">
            One of three columns
        </div>
        <div class="col-sm-4">
            One of three columns
        </div>
        <div class="col-sm-4">
            One of three columns
        </div>
    </div>
</div>
```
Yet another spicy opinion

\_(ツ)_\_/¯

“*It uses CSS Flexbox (rather than CSS Grid) for high flexibility.*”

– Material UI (Grid version 2)
Alignment only along cross-axis
Extra row wrapper required

```
<div class="row">
  <div class="columns small-2 large-4">4</div>
  <div class="columns small-4 large-4">4</div>
  <div class="columns small-6 large-4">4</div>
</div>

<div class="row">
  <div class="columns large-3">3</div>
  <div class="columns large-6">6</div>
  <div class="columns large-3">3</div>
</div>

<div class="container text-center">
  <div class="row">
    <div class="col align-self-start">
      One of three columns
    </div>
    <div class="col align-self-center">
      One of three columns
    </div>
    <div class="col align-self-end">
      One of three columns
    </div>
  </div>
</div>
```
Flexbox-powered grid markup

```html
<div class="flex-grid">
  <div class="row">
    <div class="col").col</div>
    <div class="col").col</div>
    <div class="col").col</div>
    <div class="col").col</div>
  </div>
  <div class="row">
    <div class="col col-md-8").col-md-8"></div>
    <div class="col col-md-4").col-md-4"></div>
  </div>
  <div class="row">
    <div class="col col-md-4").col-md-4"></div>
    <div class="col col-md-4 offset-md-4").col-md-4 .offset-md-4"></div>
  </div>
  <div class="row">
```

Flexbox-powered grid

```
.row {
  display: flex;
  flex-wrap: wrap;
}

.col {
  flex: 0 0 0%;
}

@media screen and (min-width: 768px) {
  .col-md-8 {
    flex: 0 0 auto;
    width: 66.66666667%;
  }

  .col-md-6 {
    flex: 0 0 auto;
    width: 50%;
  }
```
Grid-powered grid

```css
.grid {
  display: grid;
  grid-template-columns: repeat(12, 1fr);
}

[class*='item'] {
  grid-column-end: span 6;
}

.item-3 {
  grid-column-end: span 3;
}

@media screen and (min-width: 768px) {
  .item-md-8 {
    grid-column-end: span 8;
  }
}
```
Column breaks

Breaking columns to a new line in flexbox requires a small hack: add an element with `width: 100%` wherever you want to wrap your columns to a new line. Normally this is accomplished with multiple `.row`, but not every implementation method can account for this.

```html
<div class="container text-center">
  <div class="row">
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  </div>
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>

  <!-- Force next columns to break to new line -->
  <div class="w-100"></div>

  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
</div>
</div>
```

Source: Bootstrap v5.2 documentation
But is it a grid? 😐

"a network of lines that cross each other to form a series of squares or rectangles."

Google's dictionary provided by Oxford languages
Designers after repeatedly being told their design cannot be built

This seems fine.

The system works fine.
Bootstrap uses this “separate grid system” approach.

How it works

With Bootstrap 5, we’ve added the option to enable a separate grid system that’s built on CSS Grid, but with a Bootstrap twist. You still get classes you can apply on a whim to build responsive layouts, but with a different approach under the hood.

- **CSS Grid is opt-in.** Disable the default grid system by setting `enable-grid-classes: false` and enable the CSS Grid by setting `enable-cssgrid: true`. Then, recompile your Sass.

- **Replace instances of .row with .grid.** The .grid class sets `display: grid` and creates a grid-template that you build on with your HTML.

- **Replace .col-* classes with .g-col-* classes.** This is because our CSS Grid columns use the `grid-column` property instead of `width`.

- **Columns and gutter sizes are set via CSS variables.** Set these on the parent .grid and customize however you want, inline or in a stylesheet, with `--bs-columns` and `--bs-gap`.

In the future, Bootstrap will likely shift to a hybrid solution as the gap property has achieved nearly full browser support for flexbox.
# A pretty standard grid

<table>
<thead>
<tr>
<th>Size</th>
<th>Min</th>
<th>Max</th>
<th>Cols</th>
<th>Margin</th>
<th>Gutter</th>
</tr>
</thead>
<tbody>
<tr>
<td>xs</td>
<td>320px</td>
<td>639px</td>
<td>4</td>
<td>16px</td>
<td>16px</td>
</tr>
<tr>
<td>sm</td>
<td>640px</td>
<td>899px</td>
<td>8</td>
<td>30px</td>
<td>16px</td>
</tr>
<tr>
<td>md</td>
<td>900px</td>
<td>1199px</td>
<td>12</td>
<td>50px</td>
<td>16px</td>
</tr>
<tr>
<td>lg</td>
<td>1200px</td>
<td>1599px</td>
<td>12</td>
<td>90px</td>
<td>24px</td>
</tr>
<tr>
<td>xl</td>
<td>1600px</td>
<td>-</td>
<td>12</td>
<td>&gt;180px</td>
<td>24px</td>
</tr>
</tbody>
</table>
```javascript
import { ReactNode, createElement } from "react";
import styles from "./Grid.module.scss";

interface GridProps extends React.HTMLProps<HTMLDivElement> {
  className?: string;
  children: ReactNode;
  tag?: keyof JSX.IntrinsicElements,
  props: GridProps;
}

export default function Grid({
  className = "",
  children,
  tag = "div",
  props
}) {
  const Wrapper = tag;
  return createElement(
    Wrapper,
    {
      className: `${styles.grid} ${className}`,
      ...props,
      children,
    };
  );
}
```
Option 1: vanilla CSS (or SCSS)

<table>
<thead>
<tr>
<th>Size</th>
<th>Min</th>
<th>Max</th>
<th>Cols</th>
<th>Margin</th>
<th>Gutter</th>
</tr>
</thead>
<tbody>
<tr>
<td>xs</td>
<td>320px</td>
<td>639px</td>
<td>4</td>
<td>16px</td>
<td>16px</td>
</tr>
<tr>
<td>sm</td>
<td>640px</td>
<td>899px</td>
<td>8</td>
<td>30px</td>
<td>16px</td>
</tr>
<tr>
<td>md</td>
<td>900px</td>
<td>1199px</td>
<td>12</td>
<td>50px</td>
<td>16px</td>
</tr>
<tr>
<td>lg</td>
<td>1200px</td>
<td>1599px</td>
<td>12</td>
<td>90px</td>
<td>24px</td>
</tr>
<tr>
<td>xl</td>
<td>1600px</td>
<td>-</td>
<td>12</td>
<td>&gt;180px</td>
<td>24px</td>
</tr>
</tbody>
</table>

```css
.grid {
  min-width: 320px;
  max-width: 1600px;
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 1em;
  margin-left: 16px;
  margin-right: 16px;
}
@media screen and (min-width: 640px) {
  .grid {
    grid-template-columns: repeat(8, 1fr);
    margin-left: 30px;
    margin-right: 30px;
  }
```

Option 1: vanilla CSS (or SCSS)

```css
.grid__item--full,
.grid__item--half,
.grid__item--third,
.grid__item--quarter {
  grid-column: 1 / -1;
}

@media screen and (min-width: 640px) {
  .grid__item--quarter {
    grid-column: span 4;
  }
}

@media screen and (min-width: 900px) {
  .grid__item--half {
    grid-column: span 6;
  }
}
```
Option 1: vanilla CSS (or SCSS)

```css
.custom-thingy {
  grid-column: 1 / -1;
  font-size: var(--step-1);
}

@media screen and (min-width: 640px) {
  .custom-thingy {
    grid-column: 1 / 6;
    padding-top: 2em;
    padding-bottom: 1em;
  }
}

@media screen and (min-width: 900px) {
  .custom-thingy {
    grid-column: 1 / 7;
  }
}
```
Option 2: Container and Item components
import { ReactNode, createElement } from "react";
import styles from "./Grid.module.scss";

interface GridProps extends React.HTMLProps<htmlelement> {
  className?: string;
  children: ReactNode;
  tag?: keyof JSX.IntrinsicElements;
}

export default function Grid({
  className = "",
  children,
  tag = "div",
  ...props
}: GridProps) {
  const Wrapper = tag;
}

import { ReactNode,.createElement } from "react";
import cn from "classnames";
import styles from "./Col.module.scss";

interface ColProps extends React.HTMLProps<htmlelement> {
  className?: string;
  children: ReactNode;
  colWidth?: "full" | "half" | "third" | "quarter";
  tag?: keyof JSX.IntrinsicElements;
}

export default function Col({
  className = "",
  children,
  colWidth,
  tag = "div",
  ...props
}: ColProps) {
  const Wrapper = tag;
}
Col.module.css

.full,
.half,
.third,
.quarter {
  grid-column: 1 / -1;
}

@media screen and (min-width: 640px) {
  .quarter {
    grid-column: span 4;
  }
}

@media screen and (min-width: 900px) {
  .half {
    grid-column: span 6;
  }
}
CustomThingy.module.scss

```css
p.customThingy {
  grid-column: 1 / -1;
  font-size: var(--step-1);
}

@media screen and (min-width: 640px) {
  p.customThingy {
    grid-column: 1 / 6;
    padding-top: 2em;
    padding-bottom: 1em;
  }
}

@media screen and (min-width: 900px) {
  p.customThingy {
    grid-column: 1 / 7;
  }
}
```
Option 3: Using Tailwind classes

⚠️ Yet Another Disclaimer 🌶️

The following opinion may or may not oppose your view on the matter, and that is PERFECTLY FINE. You are absolutely free to agree, disagree or not care at all.
module.exports = {
  theme: {
    screens: {
      xs: "320px",
      sm: "640px",
      md: "900px",
      lg: "1200px",
      xl: "1600px",
      maxSm: { max: "639px" },
      maxMd: { max: "899px" },
      btwSmMd: { min: "640px", max: "899px" }
    },
    prefix: "tw-"
  }
}
export default function TailwindThingy {
  return (
      <p class="tw-col-span-full">Option 3: Use Tailwind classes</p>
      <p class="tw-col-span-full md:tw-col-span-6">Well, this is spicy</p>
      <p class="tw-col-span-full md:tw-col-span-4">FWIW, Tailwind has managed to support grid fairly well in this latest version</p>
      <p class="tw-col-span-full md:tw-col-span-4">You will have to learn the tailwind classes to use them correctly</p>
      <p class="tw-col-span-full md:tw-col-span-4">This basic example is able to match the previous 2 options</p>
    </section>
  )
}
But does it work?

CodeSandbox demo
SO MANY CHOICES...
Source: Manu Cornet, 2011-06-27 edition of Bonkers World (modified to fit slide)
So you want to introduce Grid to your application?

- Are there preferred technologies used within the organisation?
- How big is your application and how is it structured?
- How flexible does the design system need to be?
- Are there cases where code is contributed by new developers often?
- What is the documentation culture like in your organisation?
So you want to introduce Grid to your application?

• Who is responsible for the maintenance and development of new components or pages on the application?
  ▪ Is it a small team of full-time developers overseeing the entire project?
  ▪ Is it numerous teams responsible for their own respective set of components and pages?
  ▪ What is the overall CSS skill level of the developers contributing to the codebase?
  ▪ Are the contributing developers very familiar with the frameworks and libraries used in the codebase?
Document the “Why”
“One size does not fit all”

–Frank Zappa
Thank you

https://chenhuijing.com

@hj_chen

@huijing

@huijing

Font is Fraunces by Undercase Type.