JEN KRAMER • HARVARD UNIVERSITY EXTENSION SCHOOL
@JEN4WEB

RESPONSIVE WEB DESIGN WITH FLEXBOX & CSS GRID

RESPONSIVE DESIGN

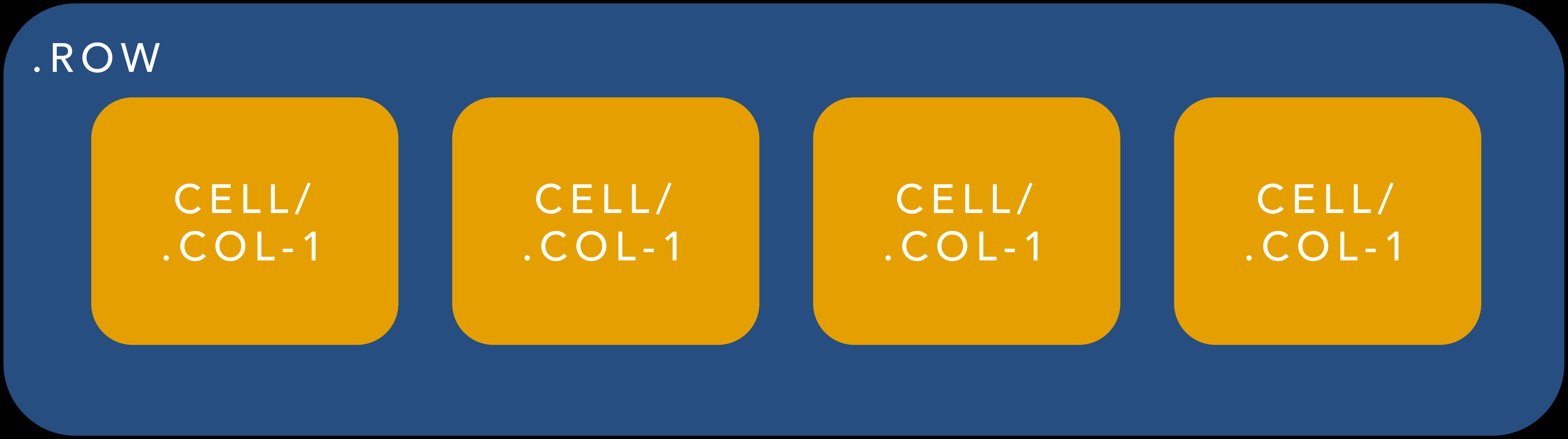
- Defined by three characteristics
  - Flexible grid-based layout
  - Media queries (CSS3)
  - Images that resize
- www.alistapart.com/articles/responsive-web-design/

PART 1

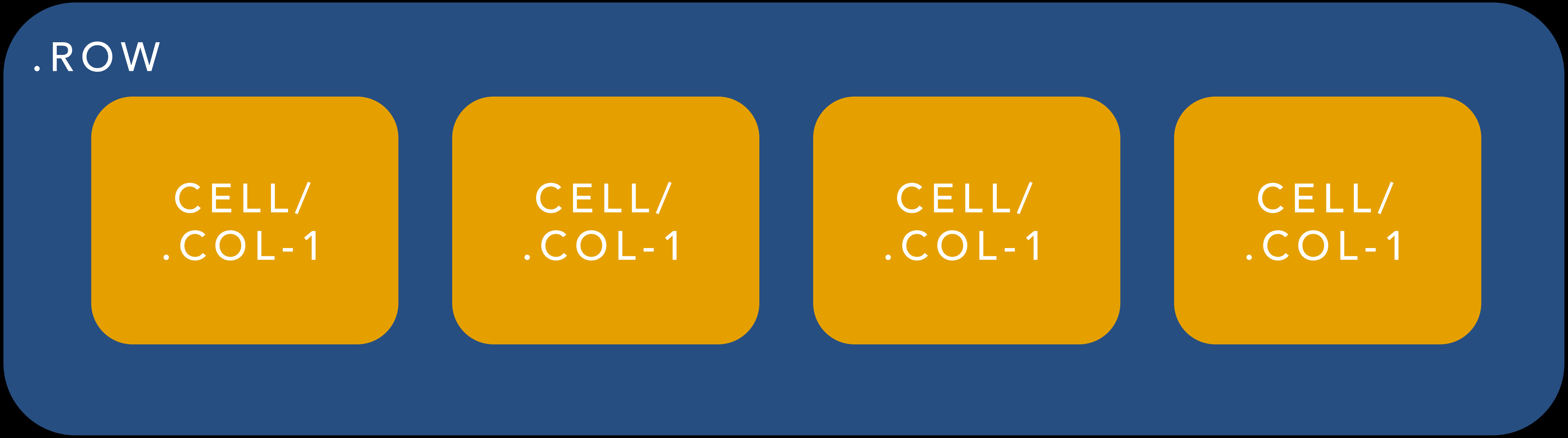
FLOATS

FLOATS

- A hack from the start, right after table-based layout!
- Features rows and cells.
- Rows clear the floats on the cells.
- Source ordering determines display, though some (minor) rearrangement is possible.
- Major disadvantage: equal column heights

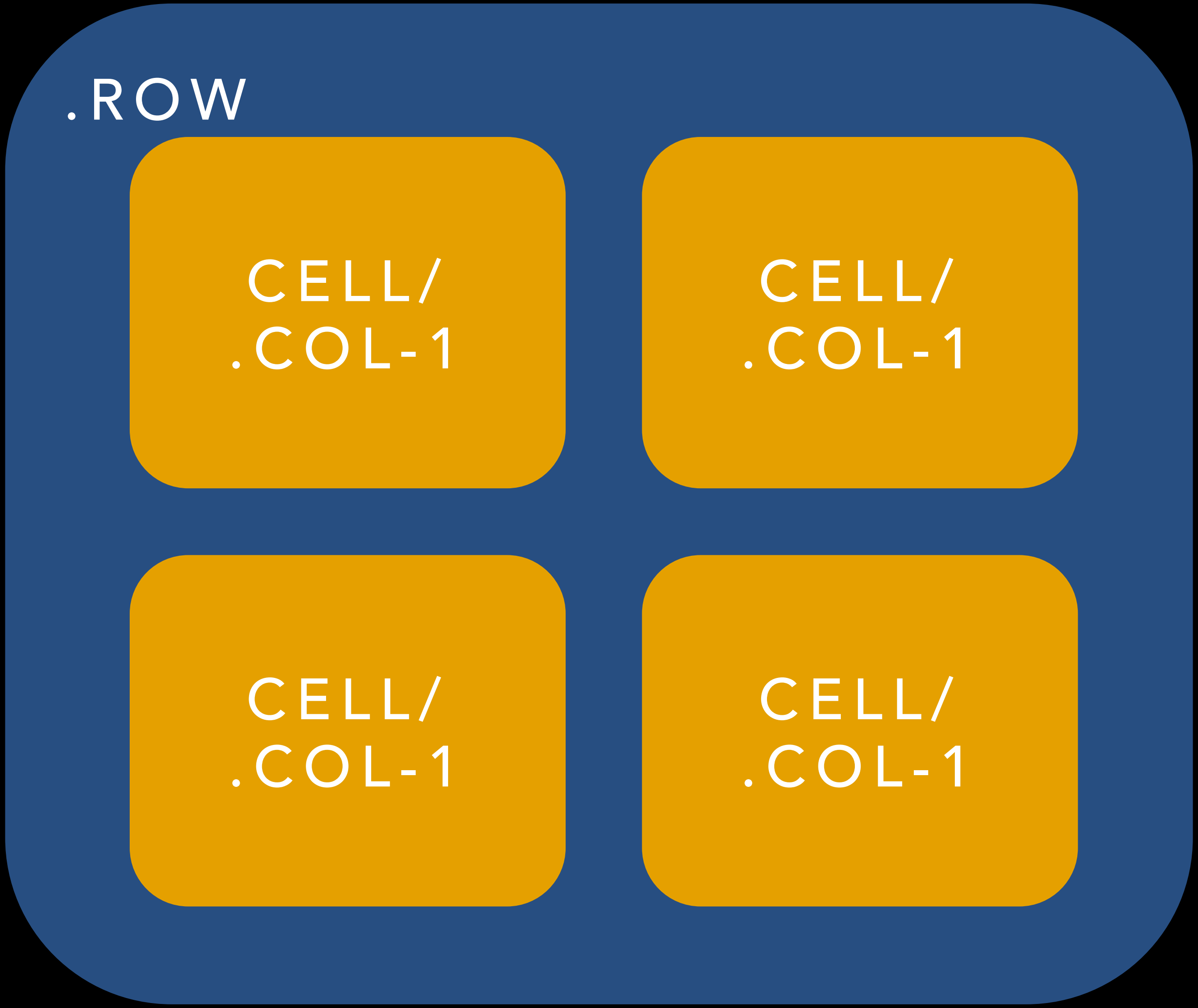

.ROW
CELL/ .COL-1
CELL/ .COL-1
CELL/ .COL-1
CELL/ .COL-1

```
<div class="row">
     <div class="col-1"></div>
     <div class="col-1"></div>
     <div class="col-1"></div>
     <div class="col-1"></div>
</div>
```

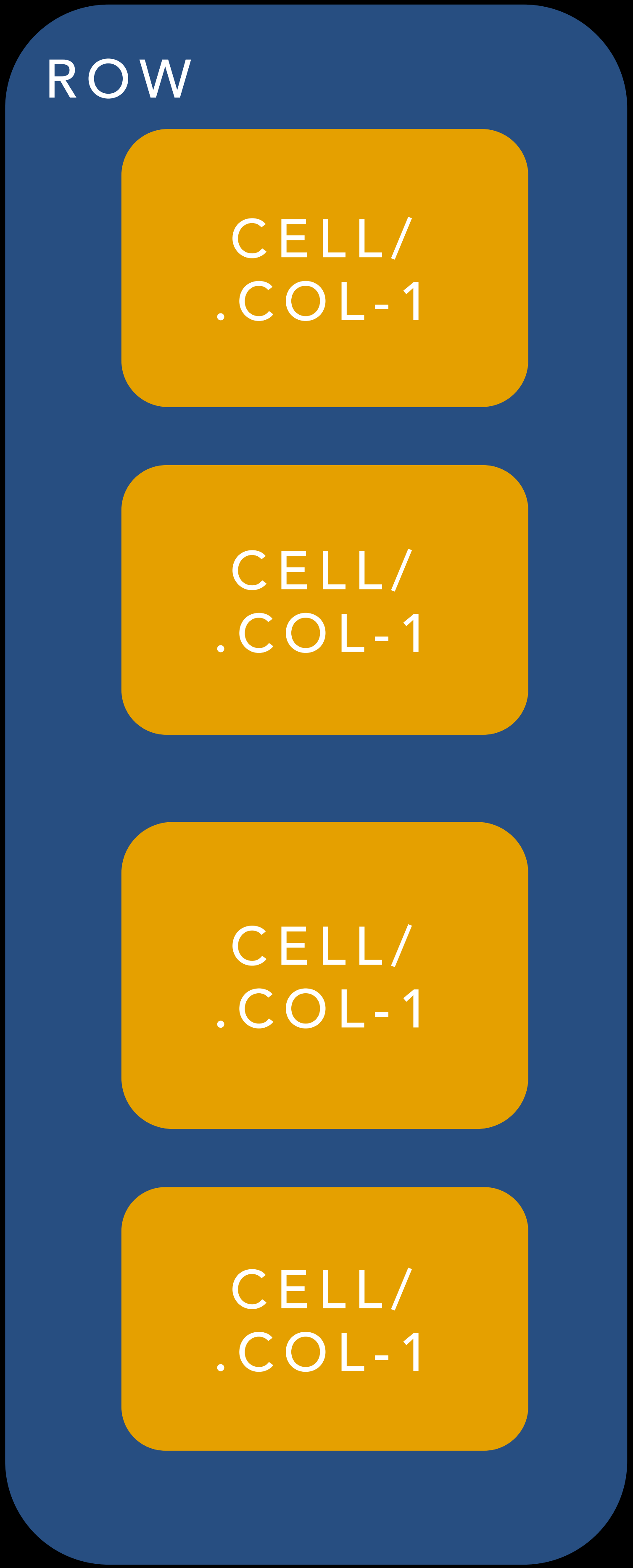


```
.row::after {
  content: "";
  display: table;
  clear: both;
}
```

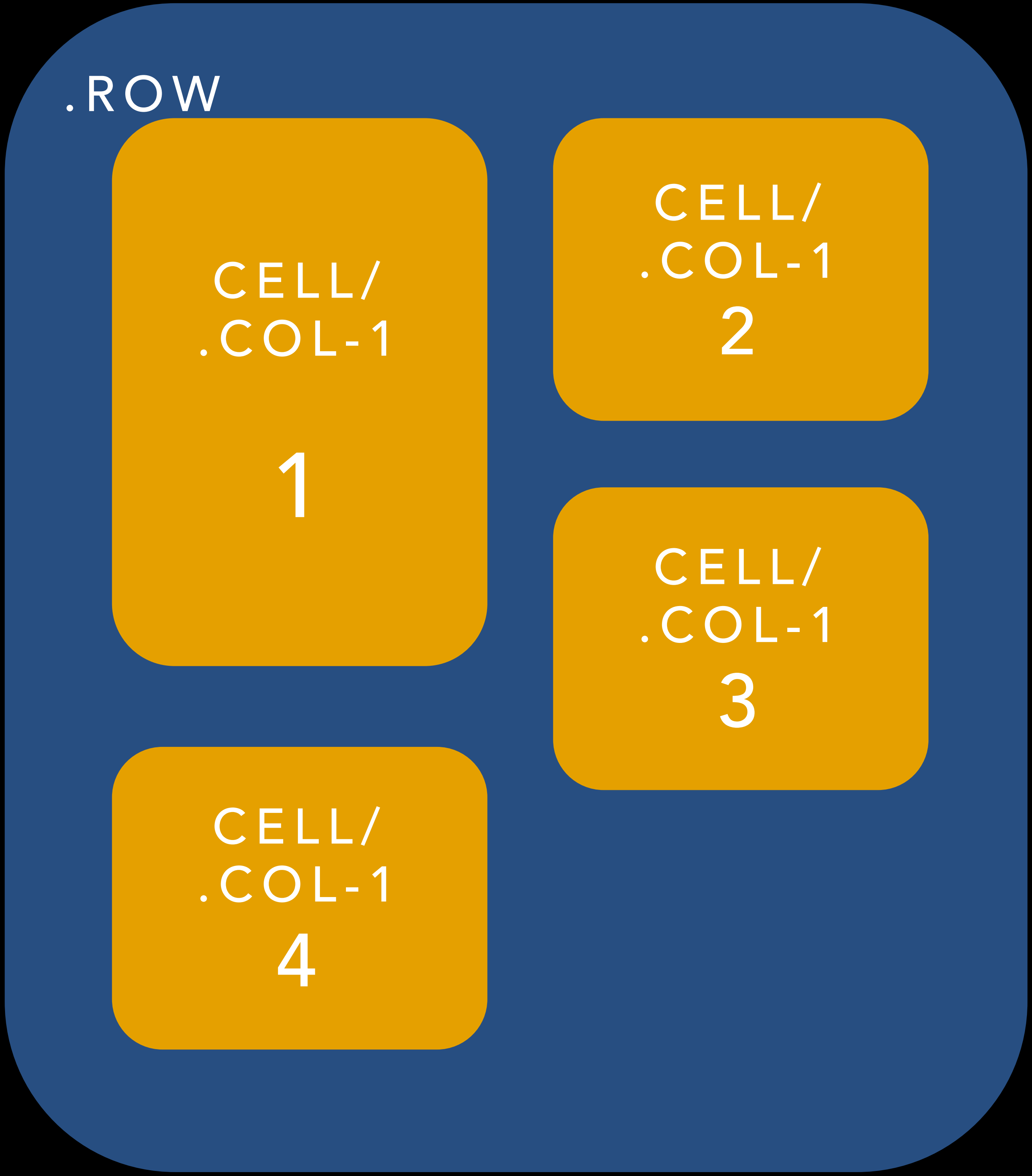
```
.col-1 {
   float: left;
   margin-left: 4%;
   width: 20%;
}
```



```
@media only screen and (min-width: 480px)
                   and (max-width: 767px) {
    .col-1 {
       width: 44%;
    }
}
```



```
@media only screen and
       (max-width: 479px) {

    .col-1 {
        width: 98%;
        margin: 1%;
        float: none;
    }
}
```



There can be layout problems with floats.

This can be resolved with JavaScript, with a column equalizer script.

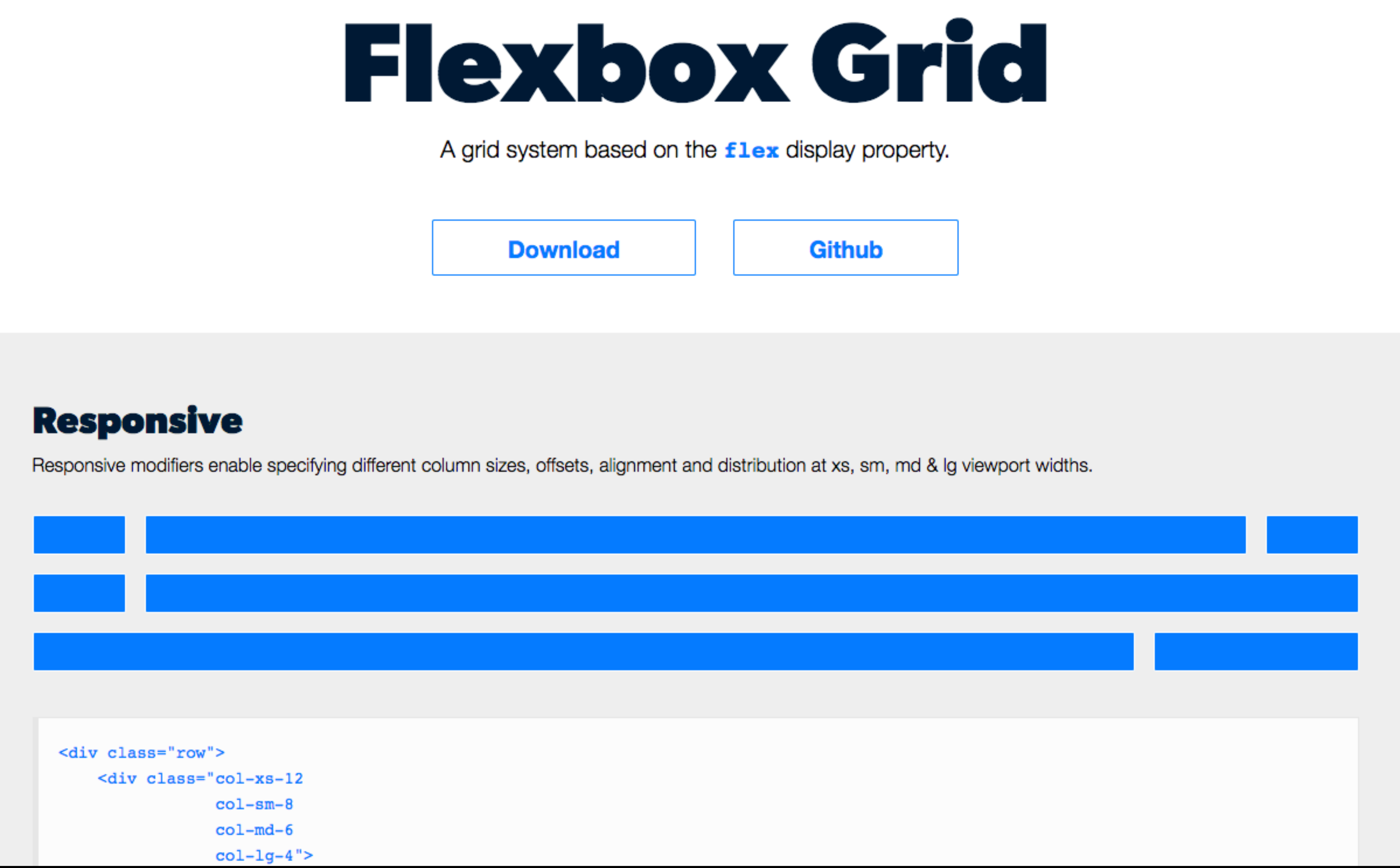
```css
/* rearranging the columns */

[class*="col-"] {
    position: relative;
}
.col-push-1 {
    left: 26%;
}
.col-pull-3 {
    left: -74%;
}
```

PART 2

FLEXBOX

FLEXBOX

- The first layout elements – but not designed to lay out whole web pages
- Features flex-containers (row) and flex-items (cells). Both are required to work.
- Excels at vertical centering and equal heights
- Very easy to reorder boxes
- Major disadvantages:
    - Wasn’t designed to be locked down for layouts! Works in 1 dimension only.
    - Browser support and syntax is challenging.



Flexbox Grid

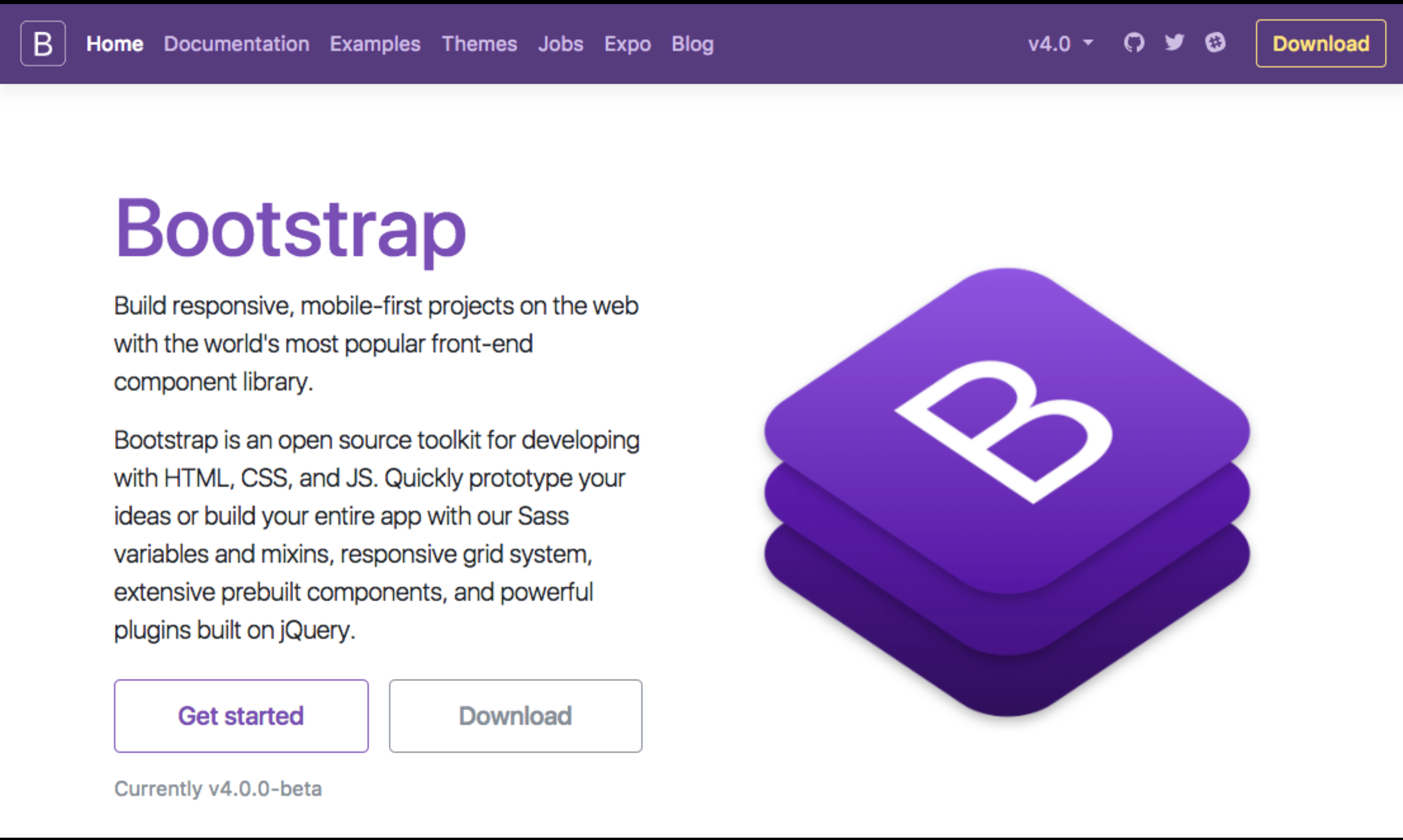
A grid system based on the `flex` display property.

Responsive

Responsive modifiers enable specifying different column sizes, offsets, alignment and distribution at xs, sm, md & lg viewport widths.

```
<div class="row">
    <div class="col-xs-12
                col-sm-8
                col-md-6
                col-lg-4">
```

flexboxgrid.com



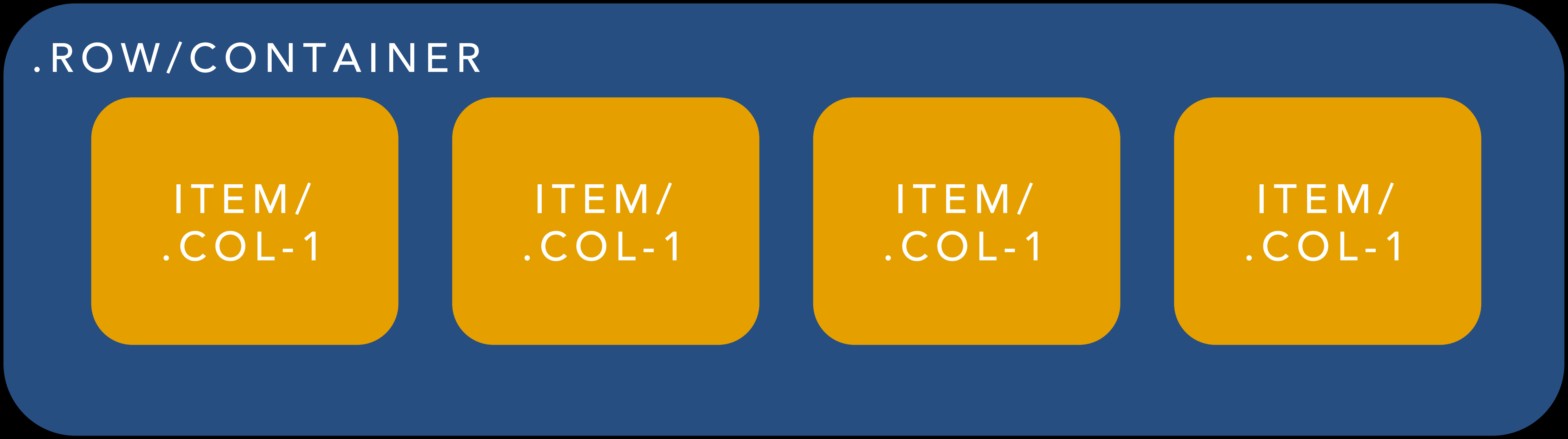
Bootstrap

Build responsive, mobile-first projects on the web with the world's most popular front-end component library.

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

Currently v4.0.0-beta

getbootstrap.com


.ROW/CONTAINER
ITEM/ .COL-1
ITEM/ .COL-1
ITEM/ .COL-1
ITEM/ .COL-1

```
<div class=“row”>
     <div class=“col-1”></div>
     <div class=“col-1”></div>
     <div class=“col-1”></div>
     <div class=“col-1”></div>
</div>
```

```css
.row {
  display: flex;
  flex-flow: row wrap;
  justify-content: center;
}
```

Change flex-flow to other values to change direction of rows – row reverse, column reverse, no wrap

```
.col-1 includes:

  flex: 0 0 24%; /* desktop */

  flex: 0 0 48%; /* tablet */

  flex: 0 0 98%; /* phone */
```

To change widths on .col-1, change the flex-basis property. This is more flexible than width.

```
/* rearranging the columns */

.col-push-1 {
  order: 2;
}
.col-pull-3 {
  order: 1;
}
```

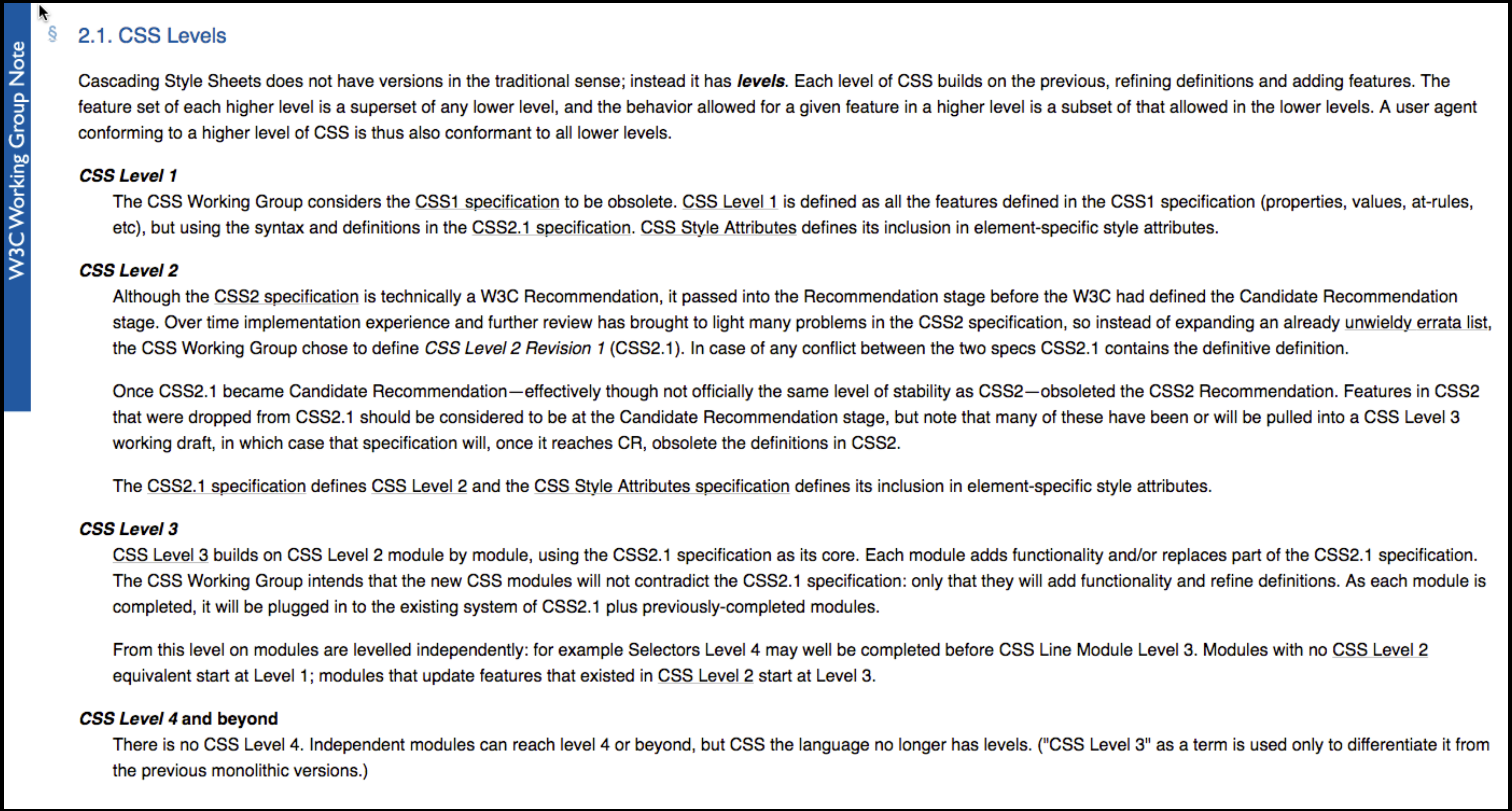
- Flexbox grid is a hack!
- Flexbox = "flexible boxes" – grid reduces their flexibility
- How can we use Flexbox properly?

JEN KRAMER • HARVARD UNIVERSITY EXTENSION SCHOOL
@JEN4WEB

CSS4 GRID

JEN KRAMER • HARVARD UNIVERSITY EXTENSION SCHOOL
@JEN4WEB

CSS~~4~~ GRID


W3C Working Group Note

§ 2.1. CSS Levels

Cascading Style Sheets does not have versions in the traditional sense; instead it has ***levels***. Each level of CSS builds on the previous, refining definitions and adding features. The feature set of each higher level is a superset of any lower level, and the behavior allowed for a given feature in a higher level is a subset of that allowed in the lower levels. A user agent conforming to a higher level of CSS is thus also conformant to all lower levels.

CSS Level 1

The CSS Working Group considers the CSS1 specification to be obsolete. CSS Level 1 is defined as all the features defined in the CSS1 specification (properties, values, at-rules, etc), but using the syntax and definitions in the CSS2.1 specification. CSS Style Attributes defines its inclusion in element-specific style attributes.

CSS Level 2

Although the CSS2 specification is technically a W3C Recommendation, it passed into the Recommendation stage before the W3C had defined the Candidate Recommendation stage. Over time implementation experience and further review has brought to light many problems in the CSS2 specification, so instead of expanding an already unwieldy errata list, the CSS Working Group chose to define *CSS Level 2 Revision 1* (CSS2.1). In case of any conflict between the two specs CSS2.1 contains the definitive definition.

Once CSS2.1 became Candidate Recommendation—effectively though not officially the same level of stability as CSS2—obsoleted the CSS2 Recommendation. Features in CSS2 that were dropped from CSS2.1 should be considered to be at the Candidate Recommendation stage, but note that many of these have been or will be pulled into a CSS Level 3 working draft, in which case that specification will, once it reaches CR, obsolete the definitions in CSS2.

The CSS2.1 specification defines CSS Level 2 and the CSS Style Attributes specification defines its inclusion in element-specific style attributes.

CSS Level 3

CSS Level 3 builds on CSS Level 2 module by module, using the CSS2.1 specification as its core. Each module adds functionality and/or replaces part of the CSS2.1 specification. The CSS Working Group intends that the new CSS modules will not contradict the CSS2.1 specification: only that they will add functionality and refine definitions. As each module is completed, it will be plugged in to the existing system of CSS2.1 plus previously-completed modules.

From this level on modules are levelled independently: for example Selectors Level 4 may well be completed before CSS Line Module Level 3. Modules with no CSS Level 2 equivalent start at Level 1; modules that update features that existed in CSS Level 2 start at Level 3.

***CSS Level 4* and beyond**

There is no CSS Level 4. Independent modules can reach level 4 or beyond, but CSS the language no longer has levels. ("CSS Level 3" as a term is used only to differentiate it from the previous monolithic versions.)

“There is no CSS Level 4… CSS the language no longer has levels.”

https://www.w3.org/TR/css-2015/#css-levels

WHY CSS GRID?

- Built into CSS specification (now a recommendation).
- No “row” markup required.
- Grid is designed to work in 2 dimensions.
- Use Flexbox for UI elements, but use Grid for major layout.

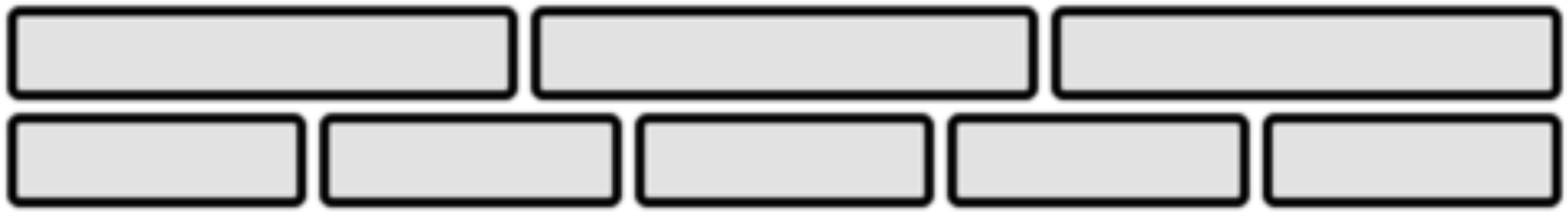


Figure 1 *Exemplary Flex Layout Example*

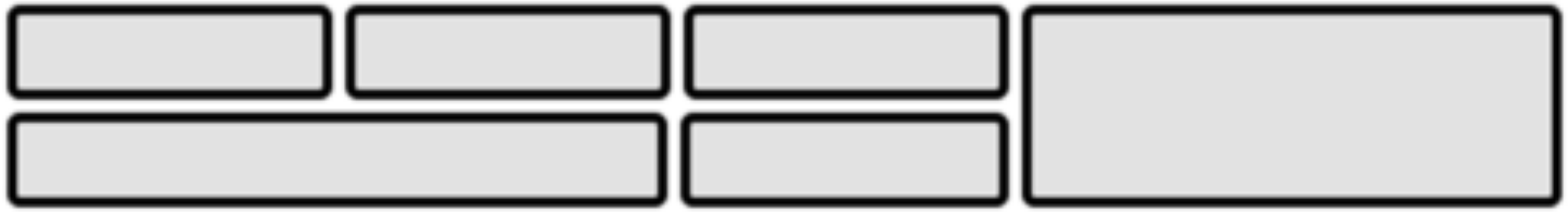


Figure 2 *Exemplary Grid Layout Example*

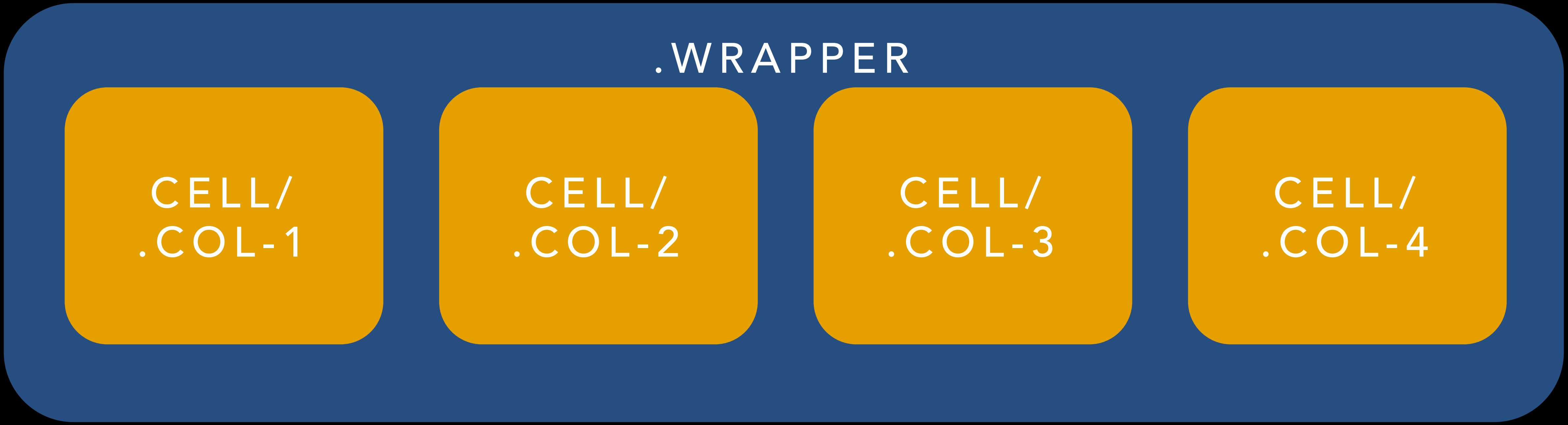
https://drafts.csswg.org/css-grid/

BROWSER SUPPORT

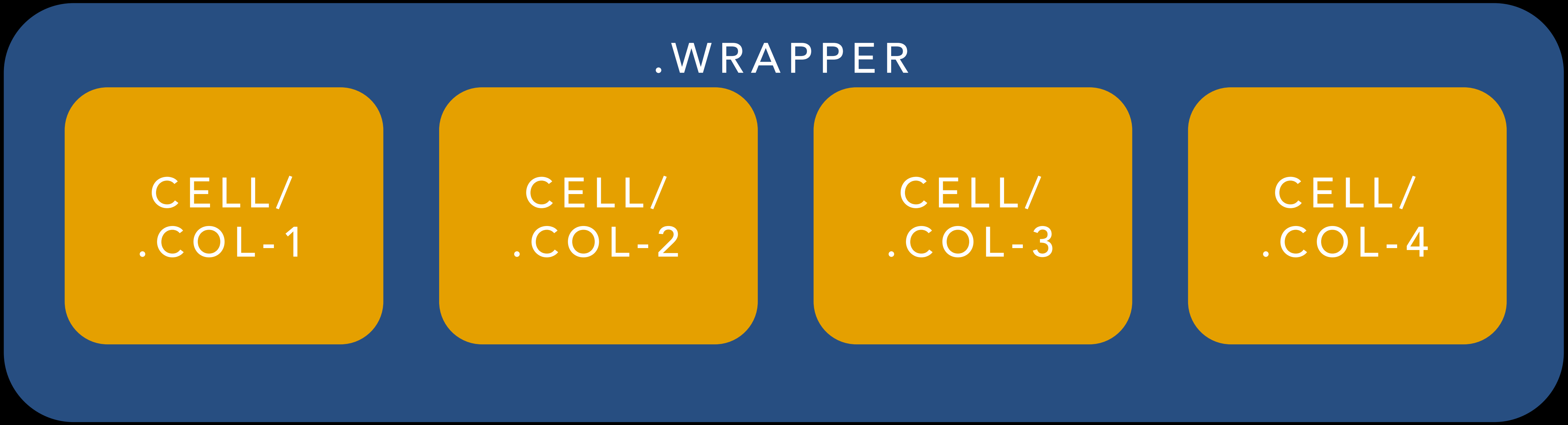
- Full support: FF 52+, Chrome 57+, Safari 10.1+, Opera 44+ , iOS Safari 10.3+, Chrome for Android 59+, Firefox for Android 55+, Edge 16, Android Browser 56+, iOS Safari 10.3+
- Partial support: IE 10, IE 11, Edge 12-15, IE Mobile 10+ (these based on an older spec).
- No support: Mostly mobile browsers not listed above (Opera Mini, Blackberry, QQ, Baidu)
- http://caniuse.com/#search=grid

POLYFILLS & FALLBACKS

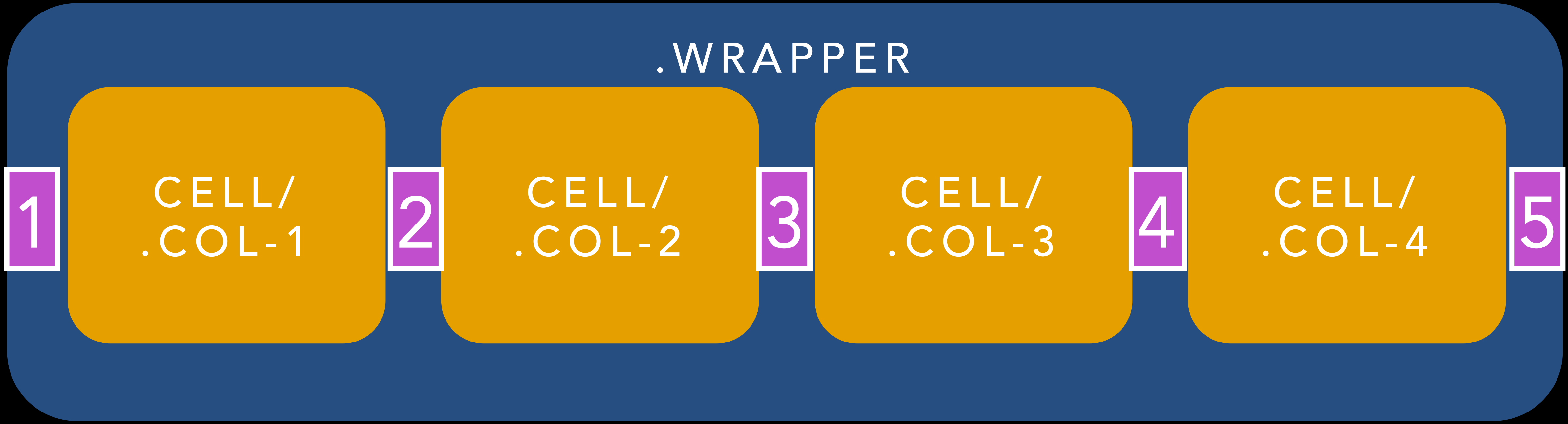
- Old spec (old MS browsers): https://github.com/codler/Grid-Layout-Polyfill
- @supports may help with all but IE browsers: https://developer.mozilla.org/en-US/docs/Web/CSS/@supports
- **Best approach**: Rachel Andrew: Grid "fallbacks" and overrides https://rachelandrew.co.uk/css/cheatsheets/grid-fallbacks


.WRAPPER
CELL/ .COL-1
CELL/ .COL-2
CELL/ .COL-3
CELL/ .COL-4

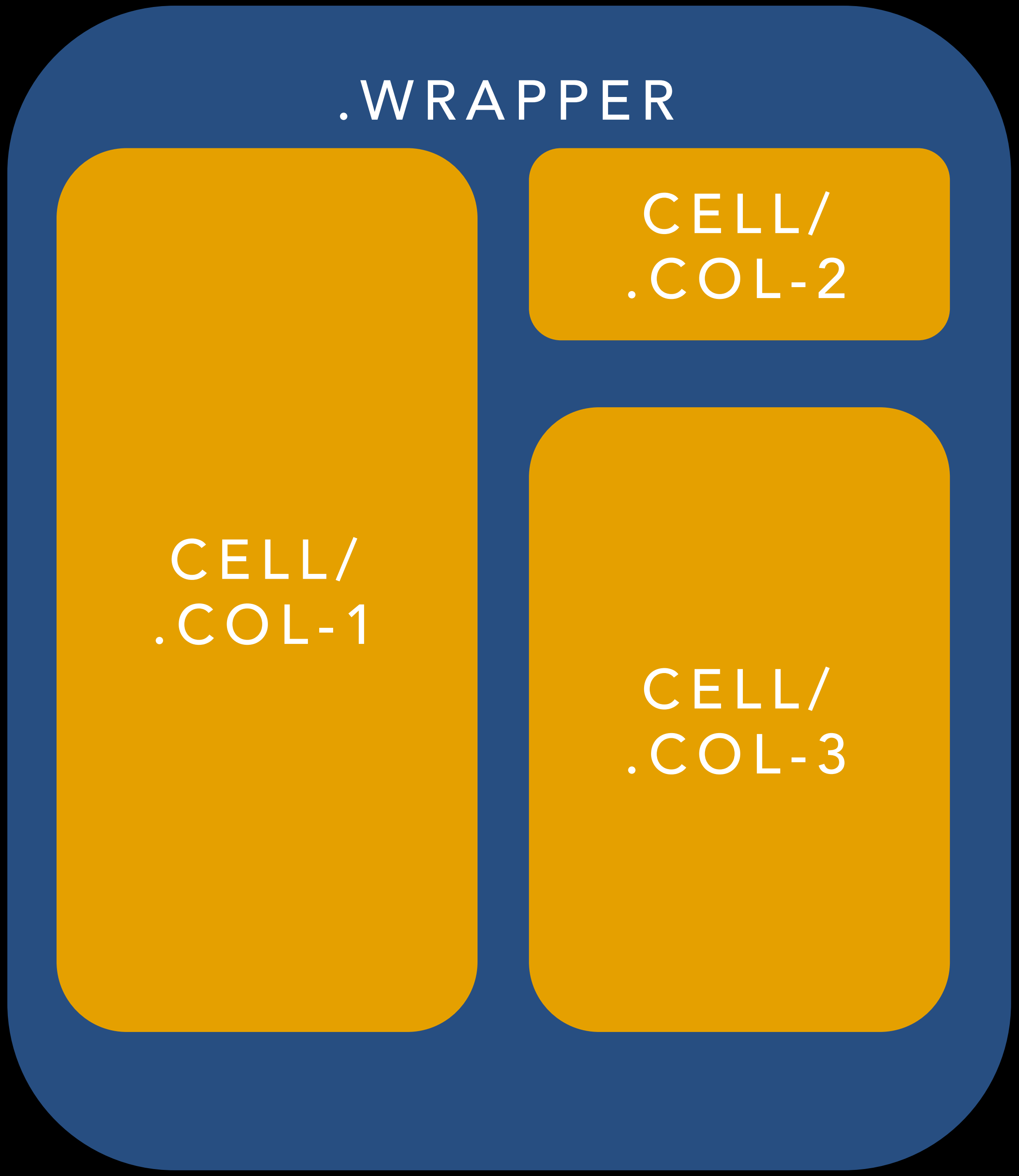
```
 <div class=“wrapper”>
<div class=“col-1”></div>
<div class=“col-2”></div>
<div class=“col-3”></div>
<div class=“col-4”></div>
        </div>
```



```
.wrapper {
  display: grid;
  grid-gap: 1em;
}
```



```
.col-1 {
  grid-column: 1 / 2;
}
.col-2 {
  grid-column: 2 / 3;
}
.col-3 {
  grid-column: 3 / 4;
}
```

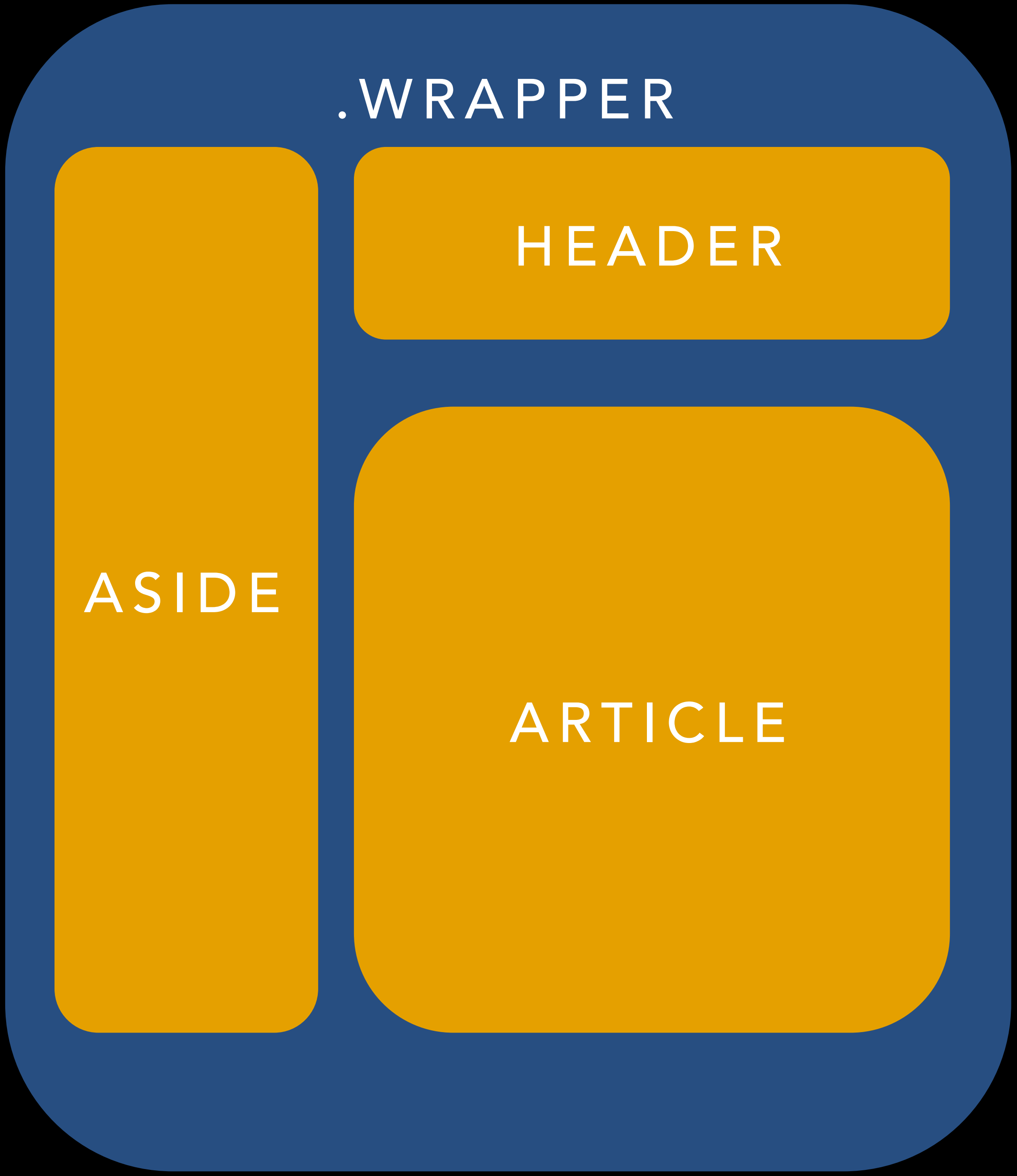


```
.col-1 {
  grid-column: 1 / 2;
     grid-row: 1 / 3;
}
.col-2 {
  grid-column: 2 / 3;
     grid-row: 1 / 2;
}
.col-3 {
  grid-column: 2 / 3;
     grid-row: 2 / 3;
}
```

ALTERNATE SYNTAX

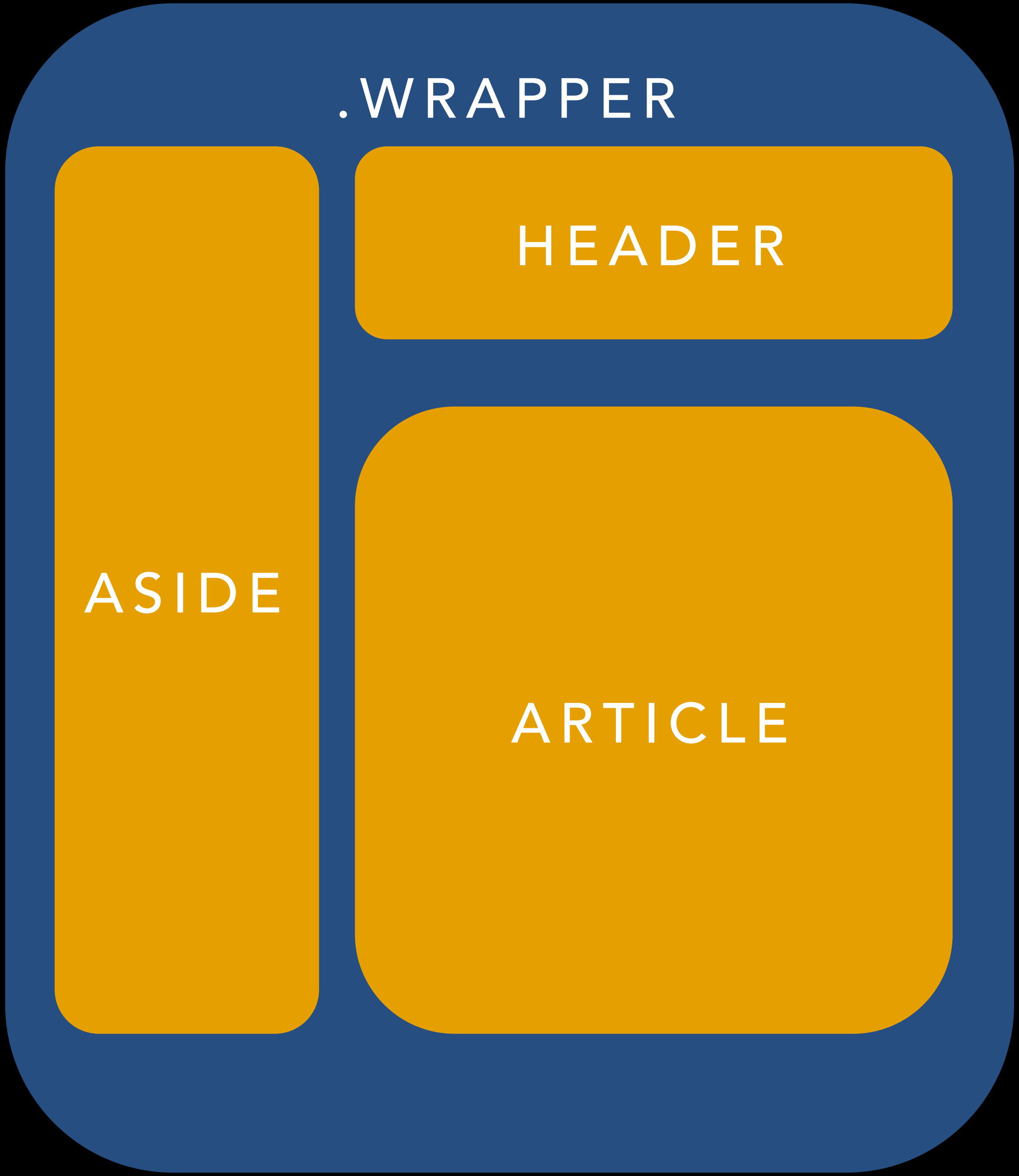
- Named grid template areas (header, footer, etc): http://gridbyexample.com/examples/example11
- You can also call out patterns (some number of rows and/or columns)
- We will cover this later in the day after we master this first type of syntax

REORDERING



```
<div class="wrapper">
   <header></header>
   <article></article>
   <aside></aside>
</div>
```

REORDERING

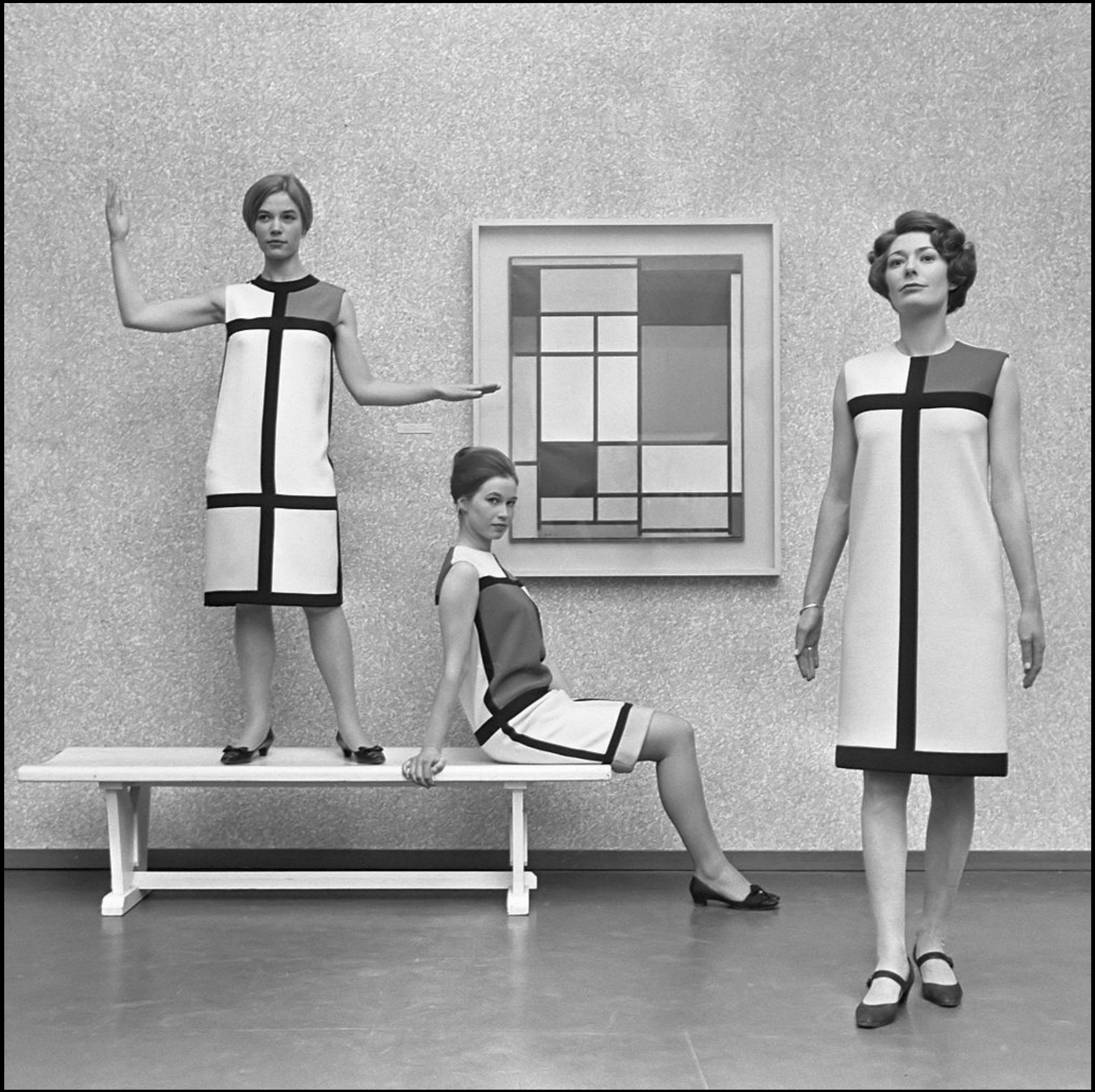


Show code in GitHub!

reordering.html

reordering.css

https://github.com/jen4web/cssgrid



LET'S CODE

QUESTIONS?

Jen Kramer

Watertown, MA, USA

Phone: 802-257-2657

jen@jenkramer.org

www.jenkramer.org

Twitter: @jen4web

Facebook: facebook.com/webdesignjen

Code available at
www.github.com/jen4web/cssgrid

Slides available at
www.slideshare.net/jen4web