Bootstrap

LAYOUT

IN A ROCKET

Learn front-end development at rocket speed

in a ROCKET

CSS GRID: GETTING STARTED

~ ~ ~

STEP 1

GRID GARDEN

Level 1 of 28

Welcome to Grid Garden, where you write CSS code to grow your carrot garden!
Water only the areas that have carrots by using the `grid-column-start` property.

For example, `grid-column-start: 3;` will water the area starting at the 3rd vertical grid line, which is another way of saying the 3rd vertical border from the left in the grid.

```
1  #garden {
2    display: grid;
3    grid-template-columns: 20% 20% 20% 20% 20%;
4    grid-template-rows: 20% 20% 20% 20% 20%;
5  }
6
7  #water {
8
9  }
10
11
12
13
14
```

Next

Grid Garden is created by Codepip • GitHub • Twitter • English

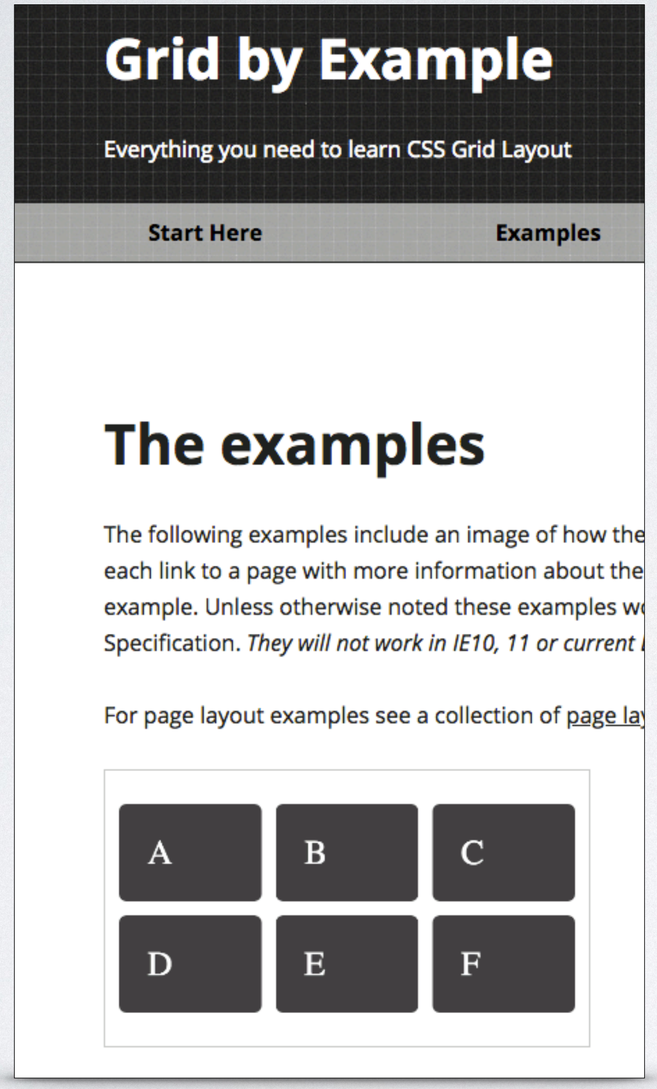
Want to learn CSS flexbox? Play Flexbox Froggy.

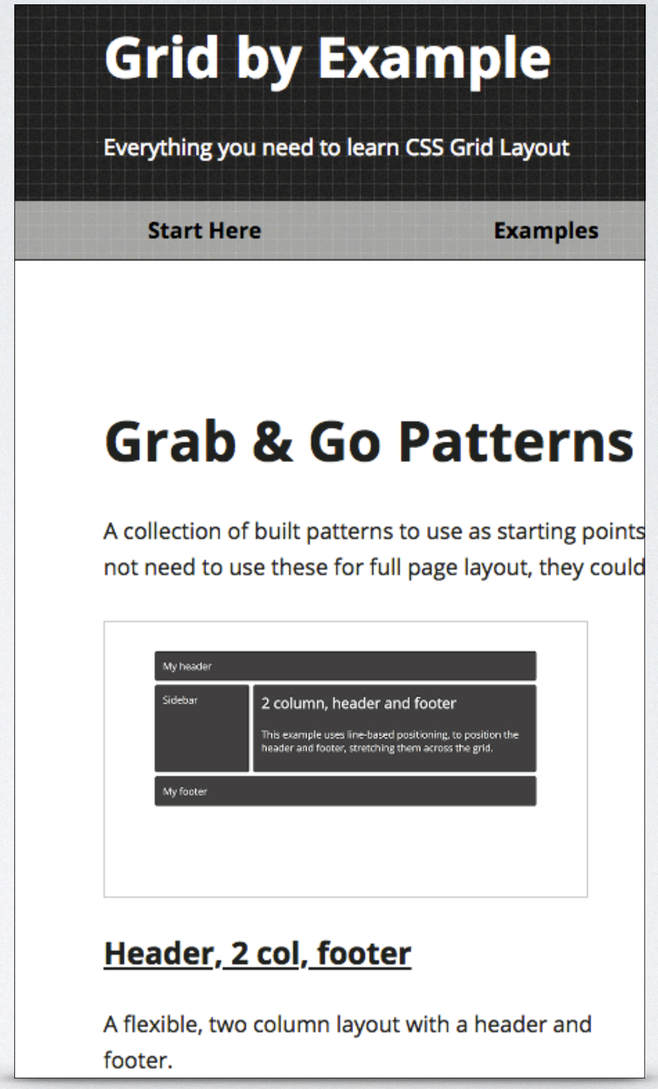
A guide to learning CSS grid by @jonsuh

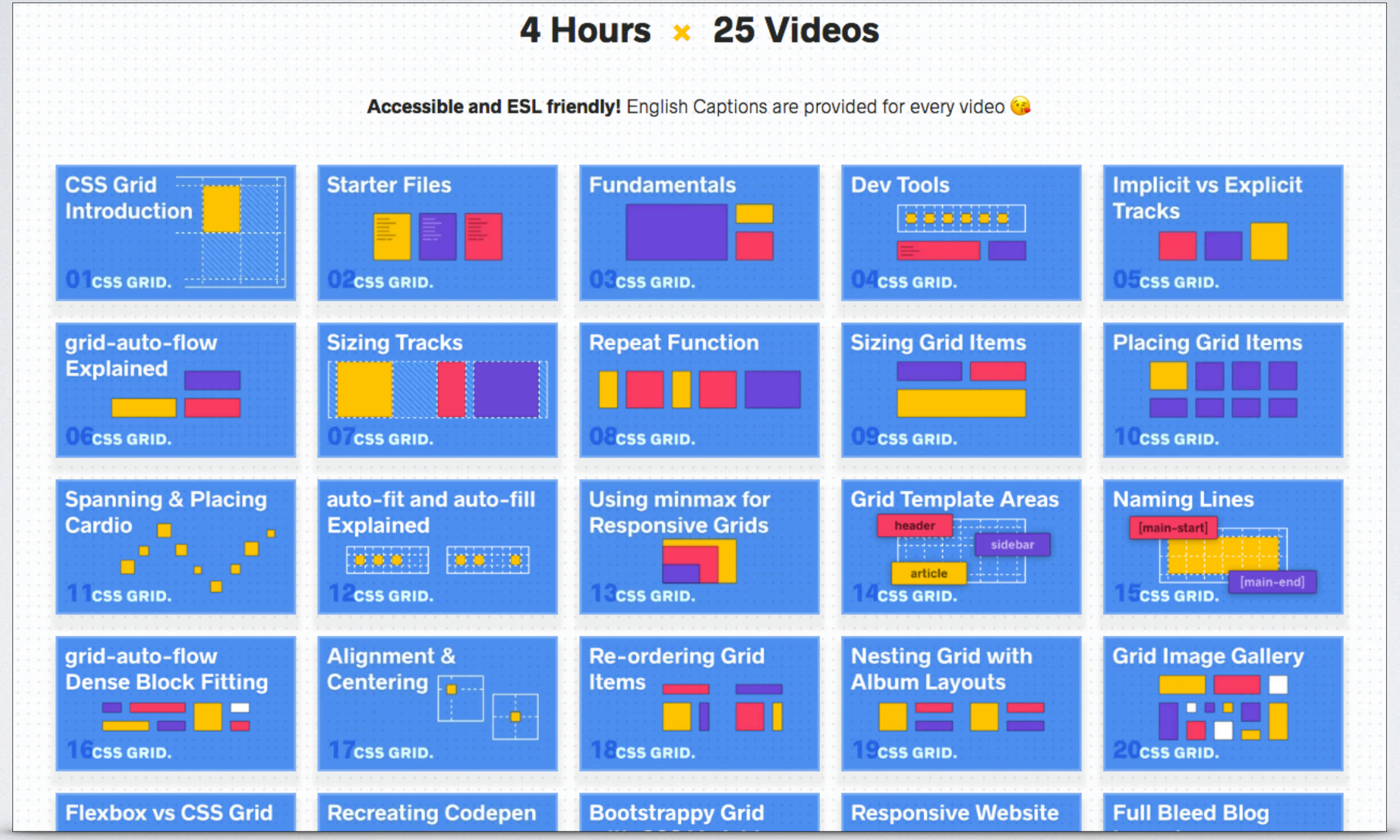
CSS Grid is a powerful tool that allows for two-dimensional layouts to be created on the web. This guide was created as a resource to help you better understand and learn Grid, and was organized in a way I thought made the most sense when learning it.

Table of Contents

STEP 3


Grid by Example
Everything you need to learn CSS Grid Layout
Start Here
Examples
The examples
The following examples include an image of how the
each link to a page with more information about the
example. Unless otherwise noted these examples w
Specification. They will not work in IE10, 11 or current
For page layout examples see a collection of page la
A
B
C
D
E
F


Grid by Example
Everything you need to learn CSS Grid Layout
Start Here
Examples
Grab & Go Patterns
A collection of built patterns to use as starting points
not need to use these for full page layout, they could
My header
Sidebar
2 column, header and footer
This example uses line-based positioning, to position the header and footer, stretching them across the grid.
My footer
Header, 2 col, footer
A flexible, two column layout with a header and
footer.


4 Hours × 25 Videos
Accessible and ESL friendly! English Captions are provided for every video
CSS Grid Introduction
01 CSS GRID.
Starter Files
02 CSS GRID.
Fundamentals
03 CSS GRID.
Dev Tools
04 CSS GRID.
Implicit vs Explicit Tracks
05 CSS GRID.
grid-auto-flow Explained
06 CSS GRID.
Sizing Tracks
07 CSS GRID.
Repeat Function
08 CSS GRID.
Sizing Grid Items
09 CSS GRID.
Placing Grid Items
10 CSS GRID.
Spanning & Placing Cardio
11 CSS GRID.
auto-fit and auto-fill Explained
12 CSS GRID.
Using minmax for Responsive Grids
13 CSS GRID.
Grid Template Areas
header
sidebar
article
14 CSS GRID.
Naming Lines
[main-start]
[main-end]
15 CSS GRID.
grid-auto-flow Dense Block Fitting
16 CSS GRID.
Alignment & Centering
17 CSS GRID.
Re-ordering Grid Items
18 CSS GRID.
Nesting Grid with Album Layouts
19 CSS GRID.
Grid Image Gallery
20 CSS GRID.
Flexbox vs CSS Grid
Recreating Codepen
Bootstrappy Grid
Responsive Website
Full Bleed Blog

RECOMMENDED TOOLS: FIREFOX CSS GRID INSPECTOR

MDN web docs moz://a Technologies References & Guides Feedback Sign in

Firefox Developer Tools | **CSS Grid Inspector: Examine grid layouts**

Languages Edit

Jump to: Grid icons in the Rules view | The Grid Layout panel | See also

Firefox Developer Tools › Page Inspector › How to › CSS Grid Inspector: Examine grid layouts

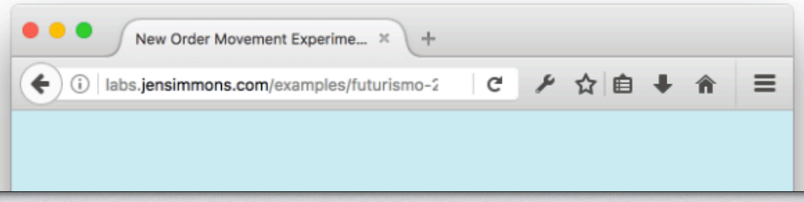
- CORE TOOLS
- MORE TOOLS
- CONNECTING THE DEVTOOLS
- DEBUGGING THE BROWSER
- EXTENDING THE DEVTOOLS

Grid icons in the Rules view

> New in Firefox 52

Starting in Firefox 52 DevTools, you can ask the Inspector to overlay a representation of the grid when inspecting grid layouts.

When an element in the Rules view has a `display: grid` declaration, then it gets a grid icon next to it: #. Click the icon to display the grid overlaid on the page, including grid lines and tracks:



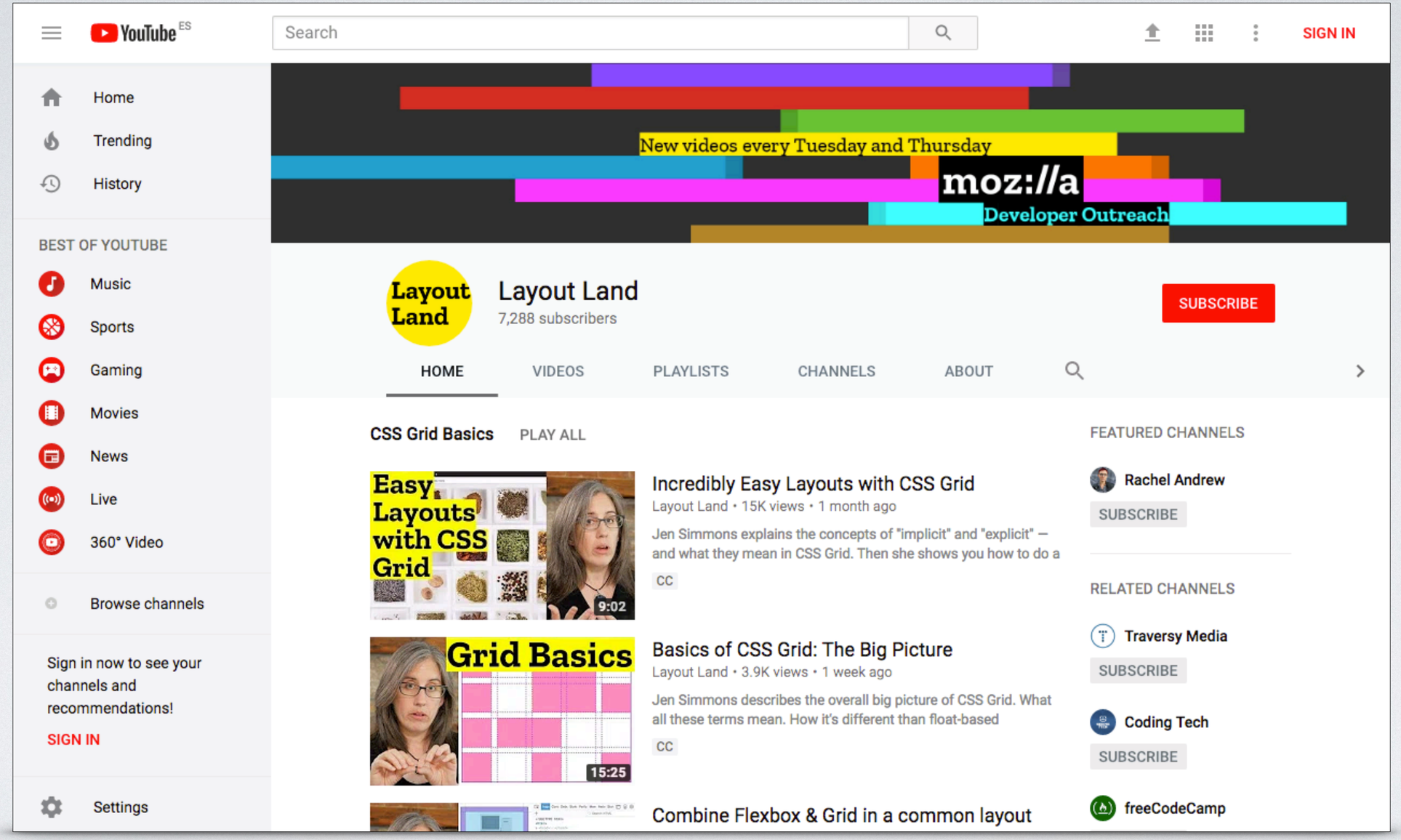
REFERENCES

Jen Simmons

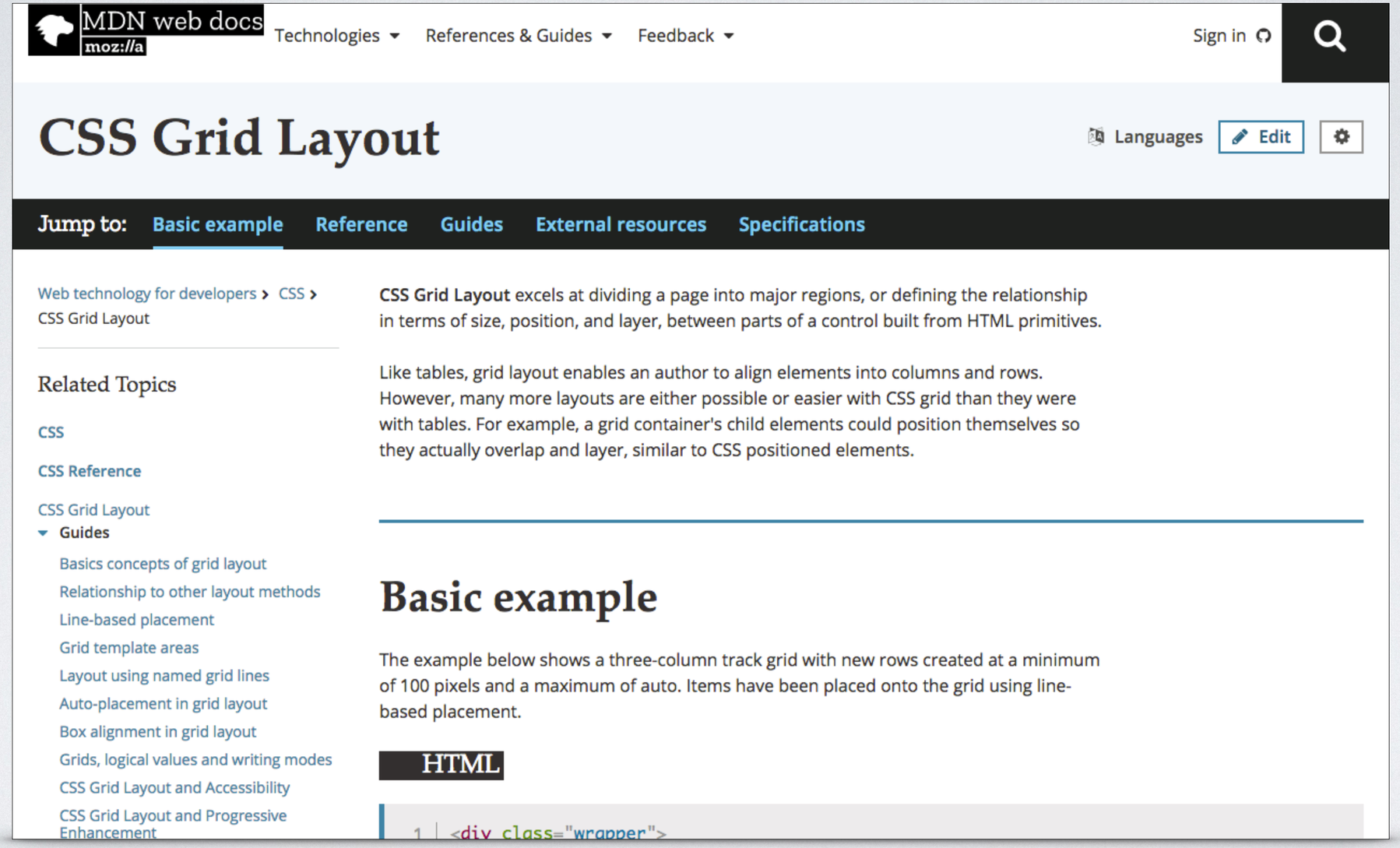
- Designer & Developer Advocate at Mozilla.
- Member of CSS Working Group.
- Speaker & writer teaching you how CSS Grid changes everything graphic design on the web.

Rachel Andrew

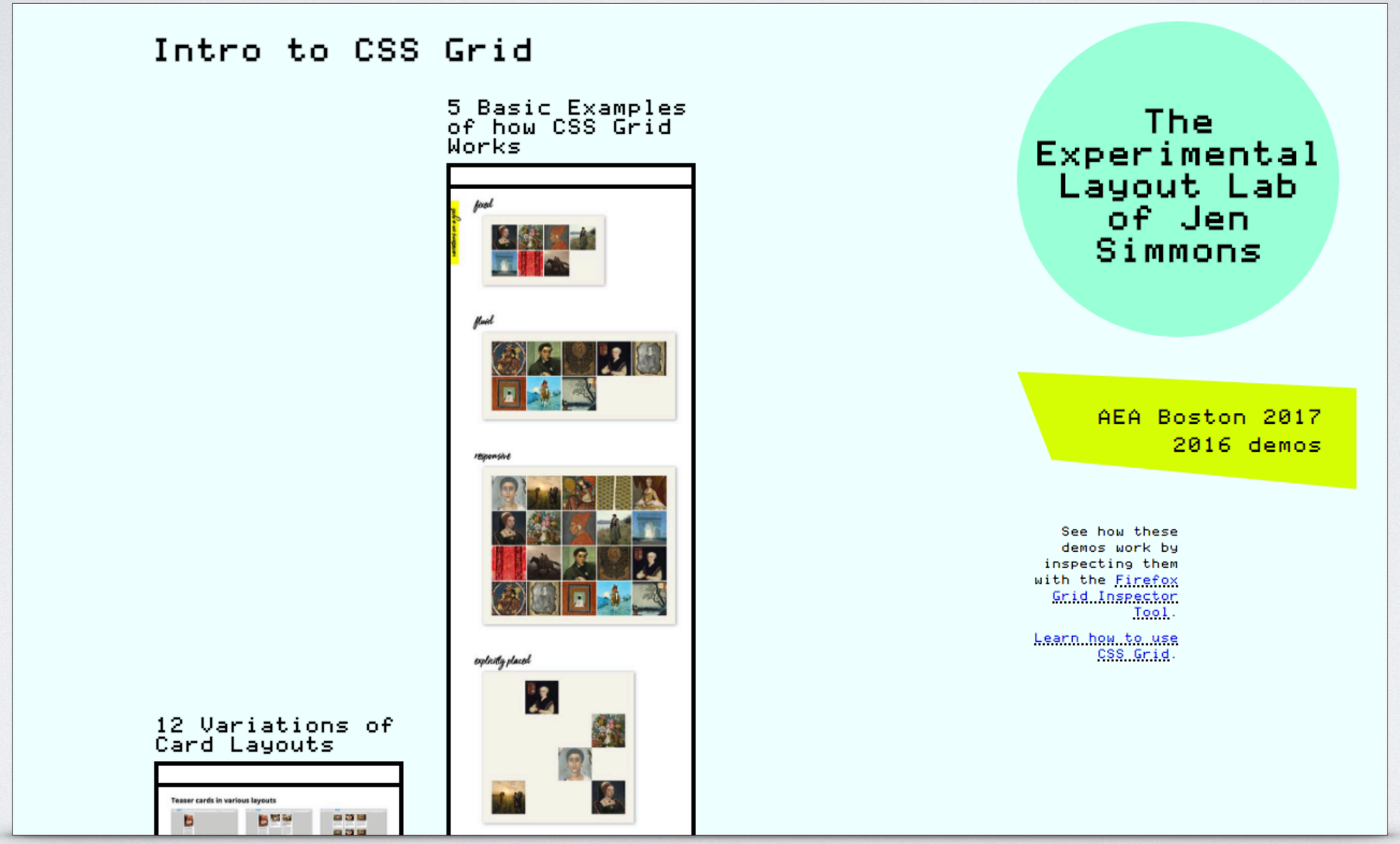
- Web Developer, half of @grabaperch CMS.
- CSS Working Group Invited Expert.
- Smashing Mag Editor in Chief.
- Student pilot.
- Runner.


YouTube ES
Search
SIGN IN
Home
Trending
History
BEST OF YOUTUBE
Music
Sports
Gaming
Movies
News
Live
360° Video
Browse channels
Sign in now to see your channels and recommendations!
SIGN IN
Settings
New videos every Tuesday and Thursday
moz://a
Developer Outreach
Layout Land
Layout Land
7,288 subscribers
SUBSCRIBE
HOME
VIDEOS
PLAYLISTS
CHANNELS
ABOUT
CSS Grid Basics
PLAY ALL
Easy Layouts with CSS Grid
9:02
Incredibly Easy Layouts with CSS Grid
Layout Land • 15K views • 1 month ago
Jen Simmons explains the concepts of "implicit" and "explicit" — and what they mean in CSS Grid. Then she shows you how to do a
CC
Grid Basics
15:25
Basics of CSS Grid: The Big Picture
Layout Land • 3.9K views • 1 week ago
Jen Simmons describes the overall big picture of CSS Grid. What all these terms mean. How it's different than float-based
CC
Combine Flexbox & Grid in a common layout
FEATURED CHANNELS
Rachel Andrew
SUBSCRIBE
RELATED CHANNELS
Traversy Media
SUBSCRIBE
Coding Tech
SUBSCRIBE
freeCodeCamp

COMPLETE REFERENCE



EXPERIMENTS



in a ROCKET

BOOTSTRAP & CSS GRID

~ ~ ~

Is **Bootstrap** still useful
now that we have **CSS Grid** 

Content
Reboot
Typography
Code
Images
...

Components
Alerts
Badge
Breadcrumb
Buttons
...

Layout
Overview
Grid
Media object
Responsive utilities

Utilities
Borders
Clearfix
Close icon
Colors
...

CONTAINER + MEDIA QUERIES + ROWS & COLS

CONTAINER + MEDIA QUERIES + ROWS & COLS

IS THE CONTAINER COMPATIBLE WITH CSS GRID?

Container

You can choose:

fixed width

Its max-width
changes at each breakpoint.

```
<div class="container">
  <!-- Content here -->
</div>
```

fluid width

It's 100% wide
all the time.

```
<div class="container-fluid">
  <!-- Content here -->
</div>
```

IS THE CONTAINER COMPATIBLE WITH CSS GRID?

```
.container {
  width: 100%;
  padding-right: 15px;
  padding-left: 15px;
  margin-right: auto;
  margin-left: auto;
}
```

Final width is determined by breakpoints

```
.container-fluid {
  width: 100%;
  padding-right: 15px;
  padding-left: 15px;
  margin-right: auto;
  margin-left: auto;
}
```

IS THE CONTAINER COMPATIBLE WITH CSS GRID?

```
.container {
  width: 100%;
  padding-right: 15px;
  padding-left: 15px;
  margin-right: auto;
  margin-left: auto;
}
```

```
.container-fluid {
  width: 100%;
  padding-right: 15px;
  padding-left: 15px;
  margin-right: auto;
  margin-left: auto;
}
```

CONTAINER

+

MEDIA QUERIES

+

ROWS & COLS

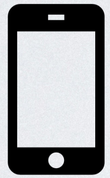
ARE THE BREAKPOINTS COMPATIBLE WITH CSS GRID?

xs

Extra small devices

576px

Viewport width.



e.g.: portrait phones.

No media query since this is the default in Bootstrap.

sm

Small devices

576px

Viewport width.

e.g.: landscape phones.

```
@media (min-width: 576px)
```

md

Medium devices

768px

Viewport width.

e.g.: tablets.

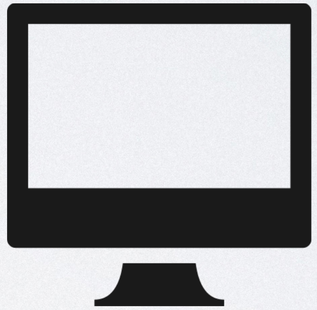
```
@media (min-width: 768px)
```

lg

Large devices

992px

Viewport width.



e.g.: desktops.

```
@media (min-width: 992px)
```

xl

Extra large devices

1200px

Viewport width.

e.g.: large desktops.

```
@media (min-width: 1200px)
```

ARE THE BREAKPOINTS COMPATIBLE WITH CSS GRID?

xs

Extra small devices

576px

Viewport width.

Max container width:

None (auto)

Class prefix:

`.col-`

sm

Small devices

576px

Viewport width.

Max container width:

540px

Class prefix:

`.col-sm-`

md

Medium devices

768px

Viewport width.

Max container width:

720px

Class prefix:

`.col-md-`

lg

Large devices

992px

Viewport width.

Max container width:

960px

Class prefix:

`.col-lg-`

xl

Extra large devices

1200px

Viewport width.

Max container width:

1140px

Class prefix:

`.col-xl-`

ARE THE BREAKPOINTS COMPATIBLE WITH CSS GRID?

```
@media (min-width: 576px) {
  .container {
    max-width: 540px;
  }
}
```

```
@media (min-width: 768px) {
  .container {
    max-width: 720px;
  }
}
```

```
@media (min-width: 992px) {
  .container {
    max-width: 960px;
  }
}
```

```
@media (min-width: 1200px) {
  .container {
    max-width: 1140px;
  }
}
```

ARE THE BREAKPOINTS COMPATIBLE WITH CSS GRID?

```
@media (min-width: 576px) {
  .container {
    max-width: 540px;
  }
}
```

```
@media (min-width: 768px) {
  .container {
    max-width: 720px;
  }
}
```

```
@media (min-width: 992px) {
  .container {
    max-width: 960px;
  }
}
```

```
@media (min-width: 1200px) {
  .container {
    max-width: 1140px;
  }
}
```

CONTAINER

+

MEDIA QUERIES

+

ROWS & COLS

FLEXBOX

Flex layout is superficially similar to block layout. It lacks many of the more complex text- or document-centric properties that can be used in block layout, such as floats and columns. In return it gains simple and powerful tools for distributing space and aligning content in ways that web apps and complex web pages often need. The contents of a flex container:

SOURCE: CSS Flexible Box Candidate Recommendation by W3C,

FLEXBOX

Proporcionar una manera más eficiente
para diseñar, alinear y distribuir
el espacio entre los elementos en un contenedor,
incluso cuando su tamaño es desconocido y / o dinámico.

- CSS grids are great for building the bigger picture. They makes it really easy to manage the layout of the page, and can even handle more unorthodox and asymmetrical designs.
- Flexbox is great at aligning the content inside elements. Use flex to position the smaller details of a design.
- Use CSS grids for 2D layouts (rows AND columns).
- Flexbox works better in one dimension only (rows OR columns).
- There is no reason to use only CSS grids or only flexbox. Learn both and use them together.

SOURCE: *CSS grid vs. flexbox* by Tutorialzine.

CSS GRID NOT INCLUDED

Version 1

2011
August

Version 2

2012
January

- 12-column responsive grid layout system
- improved and new components
- inbuilt support for Glyphicons

Version 3

2013
August

- mobile-first
- flat design

Version 3.3

2014
October

- accessibility improvements
- easier customization (LESS variables)
- bug fixes

Version 4

2018
January

SOURCE: Bootstrap, Twitter and Wikipedia.

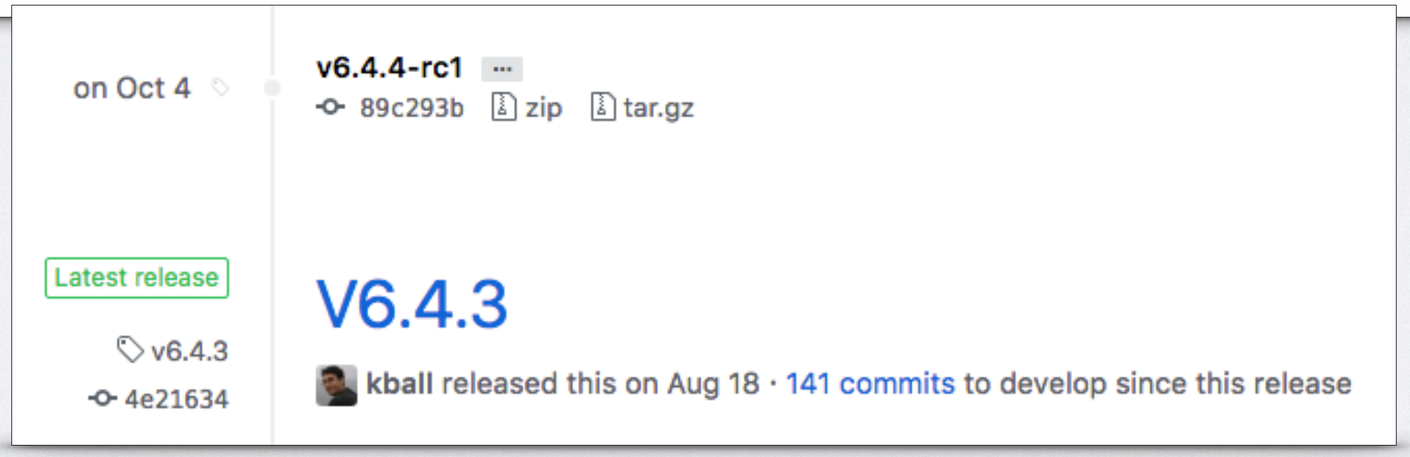
CSS GRID
NOT INCLUDED

Foundation

ZURB

XY Grid Basics

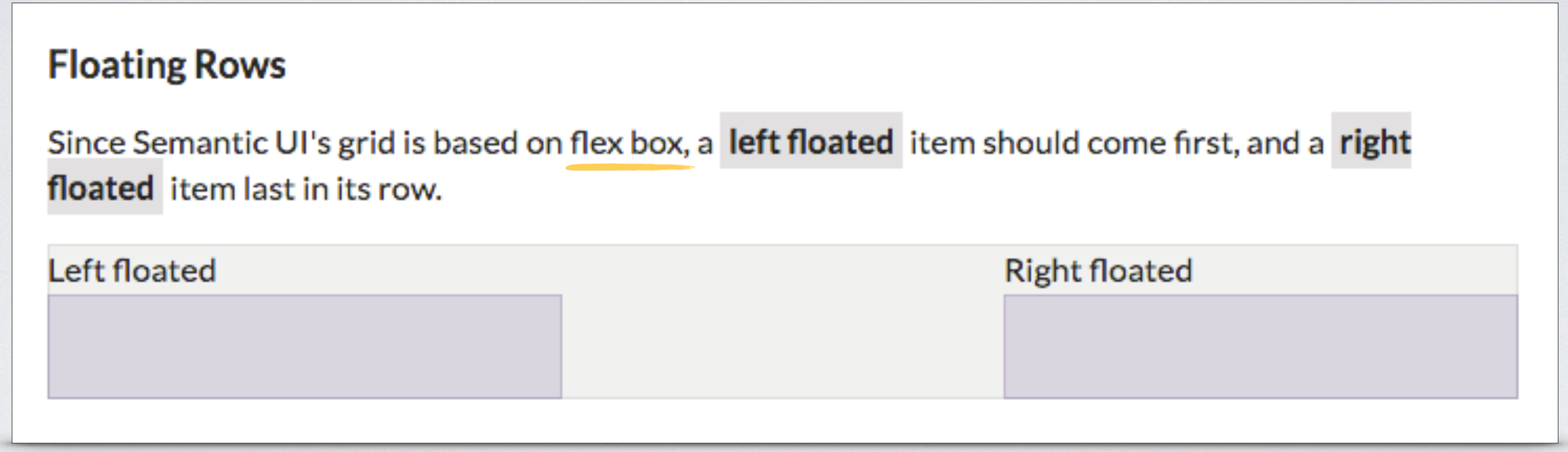
The XY grid works very similarly to the standard float grid, but includes a number of useful features only possible with Flexbox, like horizontal and vertical alignment, automatic sizing and a full vertical grid.



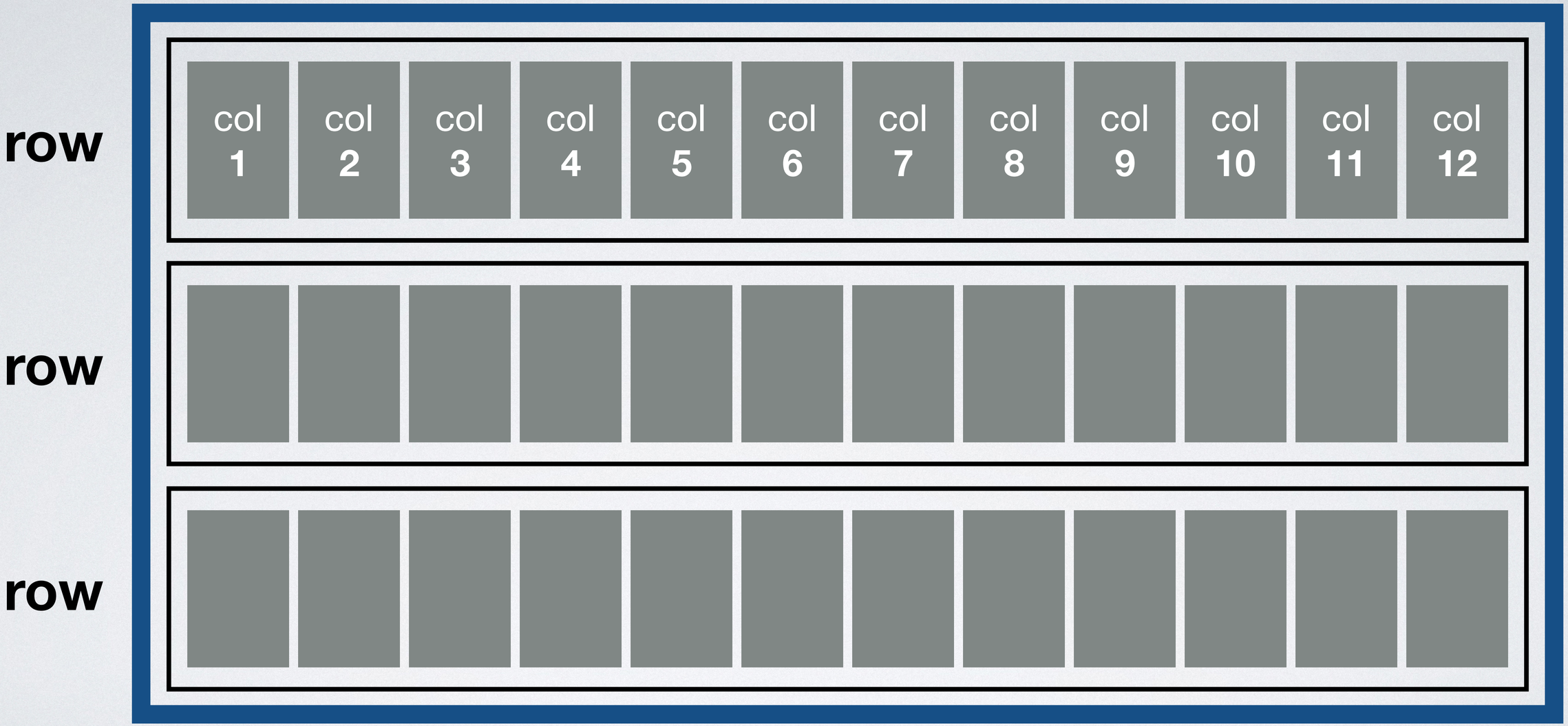
CSS GRID
NOT INCLUDED

Semantic UI

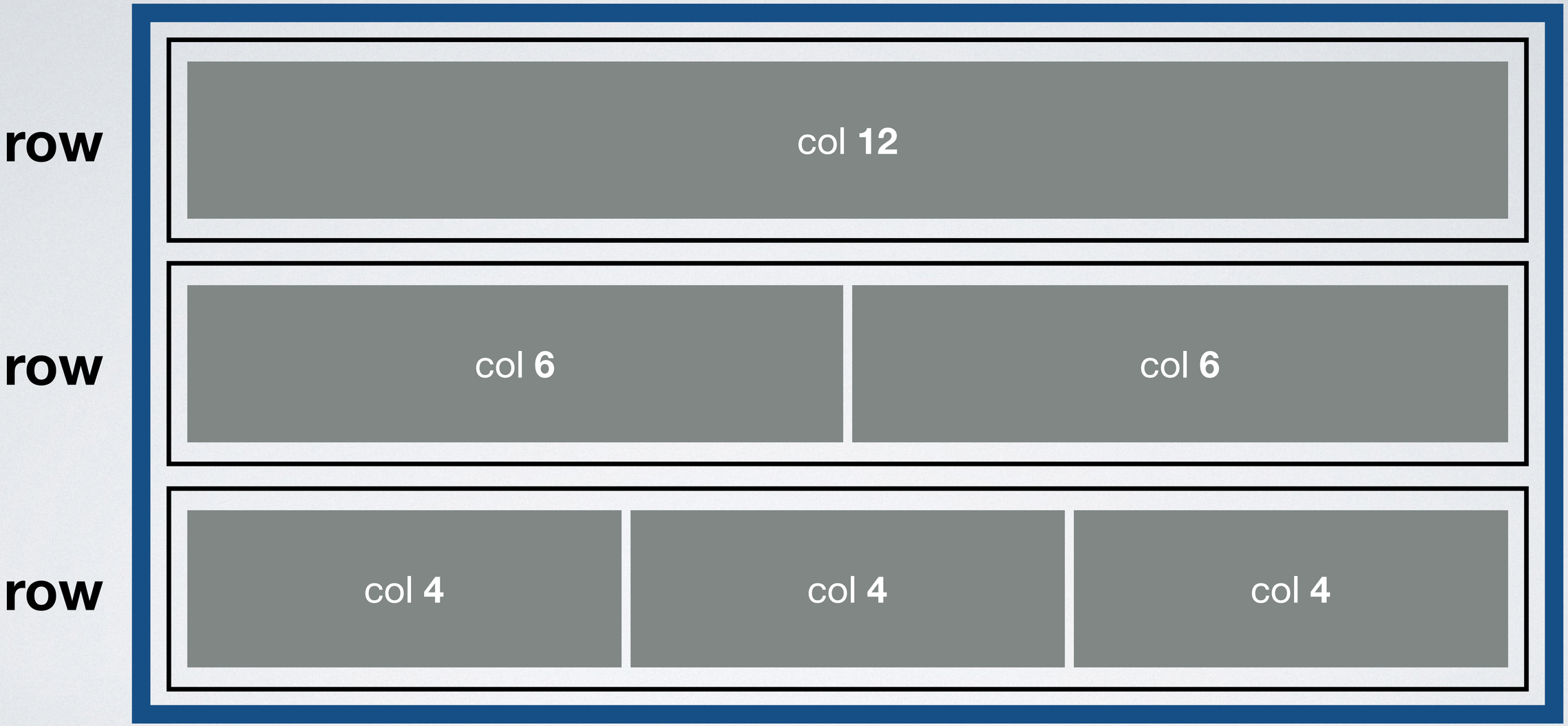
"Semantic UI 2.2 is now available (June 2016)"



container


row
col 1
col 2
col 3
col 4
col 5
col 6
col 7
col 8
col 9
col 10
col 11
col 12
row
row

container



BOOTSTRAP CODE FOR ROWS, COLUMNS AND GUTTERS

```
.row {
  display: -ms-flexbox;
  display: flex;
  -ms-flex-wrap: wrap;
      flex-wrap: wrap;
  margin-right: -15px;
  margin-left: -15px;
}

.no-gutters {
  margin-right: 0;
  margin-left: 0;
}

.no-gutters > .col,
.no-gutters > [class*="col-"] {
  padding-right: 0;
  padding-left: 0;
}

.col-1, .col-2, .col-3, .col-4, .col-5, .col-6, .col-7, .col-8, .col-9, .col-10, .col-11, .col-12, .col,
.col-auto, .col-sm-1, .col-sm-2, .col-sm-3, .col-sm-4, .col-sm-5, .col-sm-6, .col-sm-7, .col-sm-8, .col-
sm-9, .col-sm-10, .col-sm-11, .col-sm-12, .col-sm,
.col-sm-auto, .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6, .col-md-7, .col-md-8, .col-
md-9, .col-md-10, .col-md-11, .col-md-12, .col-md,
.col-md-auto, .col-lg-1, .col-lg-2, .col-lg-3, .col-lg-4, .col-lg-5, .col-lg-6, .col-lg-7, .col-lg-8, .col-
lg-9, .col-lg-10, .col-lg-11, .col-lg-12, .col-lg,
.col-lg-auto, .col-xl-1, .col-xl-2, .col-xl-3, .col-xl-4, .col-xl-5, .col-xl-6, .col-xl-7, .col-xl-8, .col-
xl-9, .col-xl-10, .col-xl-11, .col-xl-12, .col-xl,
```

BOOTSTRAP CODE FOR ROWS, COLUMNS AND GUTTERS

```
.col-xl-auto {
  position: relative;
  width: 100%;
  min-height: 1px;
  padding-right: 15px;
  padding-left: 15px;
}

.col {
  -ms-flex-preferred-size: 0;
      flex-basis: 0;
  -ms-flex-positive: 1;
      flex-grow: 1;
  max-width: 100%;
}

.col-auto {
  -ms-flex: 0 0 auto;
      flex: 0 0 auto;
  width: auto;
  max-width: none;
}

.col-1 {
  -ms-flex: 0 0 8.333333%;
      flex: 0 0 8.333333%;
  max-width: 8.333333%;
}
```

```
.col-2 {
  -ms-flex: 0 0 16.666667%;
      flex: 0 0 16.666667%;
  max-width: 16.666667%;
}

.col-3 {
  -ms-flex: 0 0 25%;
      flex: 0 0 25%;
  max-width: 25%;
}

.col-4 {
  -ms-flex: 0 0 33.333333%;
      flex: 0 0 33.333333%;
  max-width: 33.333333%;
}

.col-5 {
  -ms-flex: 0 0 41.666667%;
      flex: 0 0 41.666667%;
  max-width: 41.666667%;
}

.col-6 {
  -ms-flex: 0 0 50%;
      flex: 0 0 50%;
  max-width: 50%;
}
```

BOOTSTRAP CODE FOR ROWS, COLUMNS AND GUTTERS

```
.col-7 {
  -ms-flex: 0 0 58.333333%;
      flex: 0 0 58.333333%;
  max-width: 58.333333%;
}

.col-8 {
  -ms-flex: 0 0 66.666667%;
      flex: 0 0 66.666667%;
  max-width: 66.666667%;
}

.col-9 {
  -ms-flex: 0 0 75%;
      flex: 0 0 75%;
  max-width: 75%;
}

.col-10 {
  -ms-flex: 0 0 83.333333%;
      flex: 0 0 83.333333%;
  max-width: 83.333333%;
}

.col-11 {
  -ms-flex: 0 0 91.666667%;
      flex: 0 0 91.666667%;
  max-width: 91.666667%;
}
```

```
.col-12 {
  -ms-flex: 0 0 100%;
      flex: 0 0 100%;
  max-width: 100%;
}

.order-first {
  -ms-flex-order: -1;
      order: -1;
}

.order-1 {
  -ms-flex-order: 1;
      order: 1;
}

.order-2 {
  -ms-flex-order: 2;
      order: 2;
}

.order-3 {
  -ms-flex-order: 3;
      order: 3;
}
```

BOOTSTRAP CODE FOR ROWS, COLUMNS AND GUTTERS

```
.order-4 {
  -ms-flex-order: 4;
      order: 4;
}

.order-5 {
  -ms-flex-order: 5;
      order: 5;
}

.order-6 {
  -ms-flex-order: 6;
      order: 6;
}

.order-7 {
  -ms-flex-order: 7;
      order: 7;
}

.order-8 {
  -ms-flex-order: 8;
      order: 8;
}

.order-9 {
  -ms-flex-order: 9;
      order: 9;
}
```

```
.order-10 {
  -ms-flex-order: 10;
      order: 10;
}

.order-11 {
  -ms-flex-order: 11;
      order: 11;
}

.order-12 {
  -ms-flex-order: 12;
      order: 12;
}

.offset-1 {
  margin-left: 8.333333%;
}

.offset-2 {
  margin-left: 16.666667%;
}
```

and continues...

CSS GRID CODE FOR A 12-COLUMN GRID: ROWS

```
.row {
  display: grid;
  grid-template-columns: repeat(12, 1fr);
  grid-gap: 20px;
}
```

CSS GRID CODE FOR A 12-COLUMN GRID: COLS

```
.col-1 {
  grid-column: span 1;
}

.col-2 {
  grid-column: span 2;
}

.col-3 {
  grid-column: span 3;
}

.col-4 {
  grid-column: span 4;
}

.col-5 {
  grid-column: span 5;
}

.col-6 {
  grid-column: span 6;
}

.col-7 {
  grid-column: span 7;
}

.col-8 {
  grid-column: span 8;
}

.col-9 {
  grid-column: span 9;
}

.col-10 {
  grid-column: span 10;
}

.col-11 {
  grid-column: span 11;
}

.col-12 {
  grid-column: span 12;
}
```

CSS GRID CODE FOR A 12-COLUMN GRID: COLS WITH SASS

CSS

```
.col-1 {
  grid-column: span 1;
}

.col-2 {
  grid-column: span 2;
}

.col-3 {
  grid-column: span 3;
}

...
```

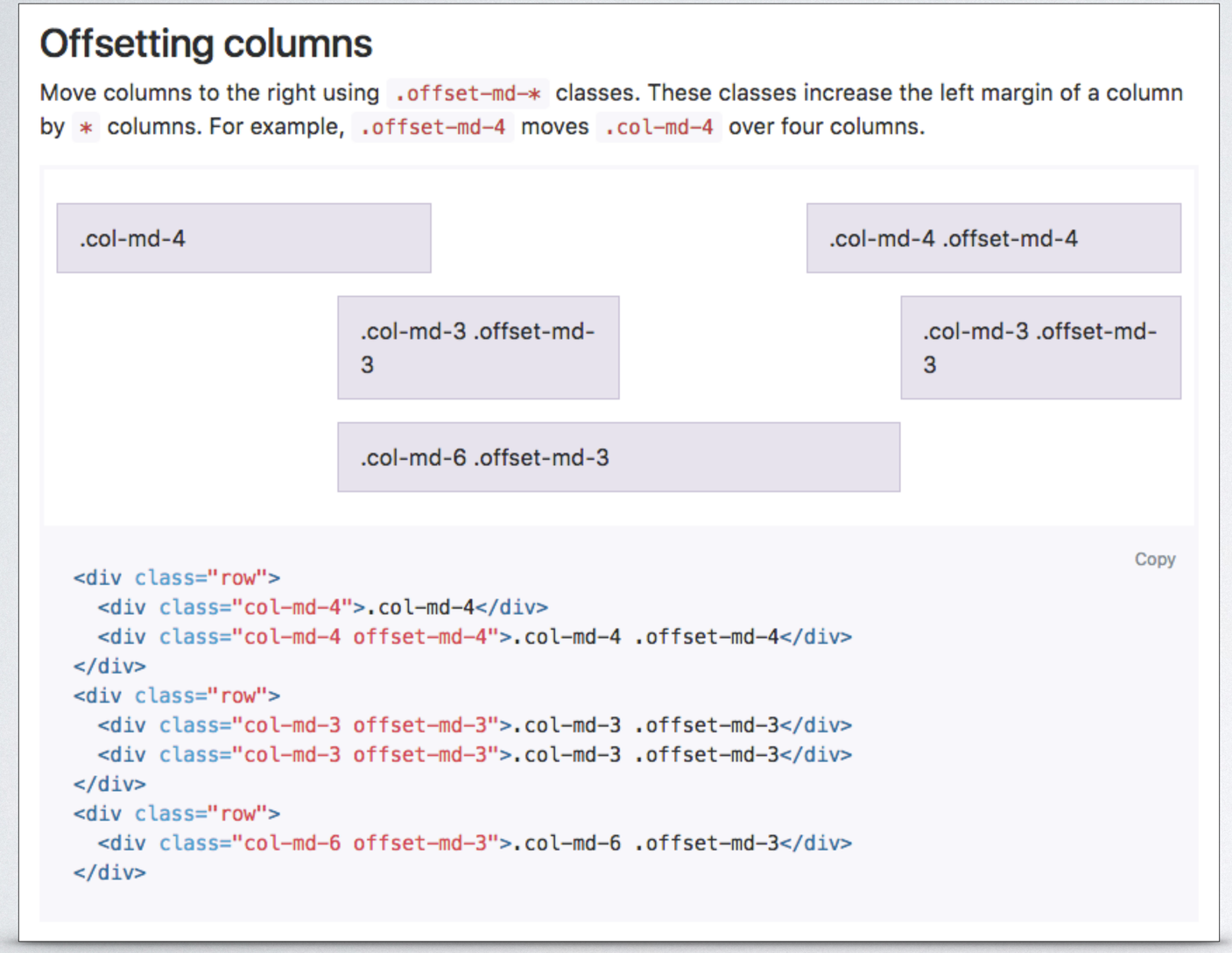
SCSS

```
$columns: 12;

@mixin col-x {
  @for $i from 1 through $columns {
    .col-#{$i} { grid-column: span $i; }
  }
}

@include col-x;
```

CSS GRID CODE FOR A 12-COLUMN GRID: USING AREAS

```
grid-template-areas: "header header header sidebar"
                     "main   main   main   sidebar"
                     "footer footer footer footer"
```

OFFSETTING COLUMNS: BOOTSTRAP VS. CSS GRID



Offsetting columns

Move columns to the right using `.offset-md-*` classes. These classes increase the left margin of a column by `*` columns. For example, `.offset-md-4` moves `.col-md-4` over four columns.

.col-md-4

.col-md-4 .offset-md-4

.col-md-3 .offset-md-3

.col-md-3 .offset-md-3

.col-md-6 .offset-md-3

Copy

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4 offset-md-4">.col-md-4 .offset-md-4</div>
</div>
<div class="row">
  <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
  <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
</div>
<div class="row">
  <div class="col-md-6 offset-md-3">.col-md-6 .offset-md-3</div>
</div>
```

```
grid-column-start: 3;
grid-column-end: 5;
```

```
grid-column: 3 / 5
```

```
grid-column: 3 / span 2
```

How would you do this with Flexbox?

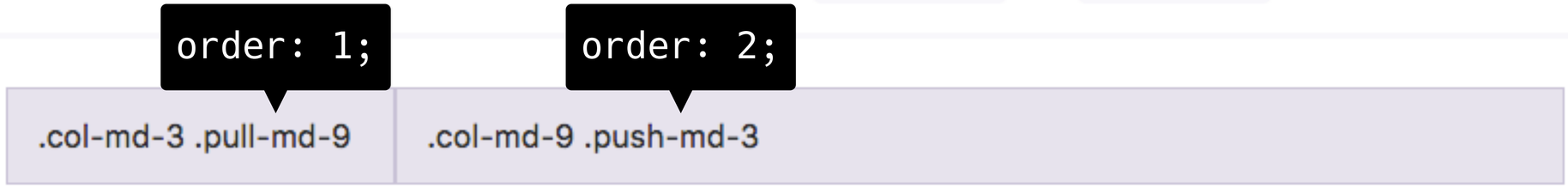
```
grid-row-start: 3;
```

```
grid-area: 1 / 2 / 4 / 6;
```

ORDERING COLUMNS

Push and pull

Easily change the order of our built-in grid columns with `.push-md-*` and `.pull-md-*` modifier classes.



Copy

```
<div class="row">
  <div class="col-md-9 push-md-3">.col-md-9 .push-md-3</div>
  <div class="col-md-3 pull-md-9">.col-md-3 .pull-md-9</div>
</div>
```

COMBINE THE POWER OF BOOTSTRAP + CSS GRID

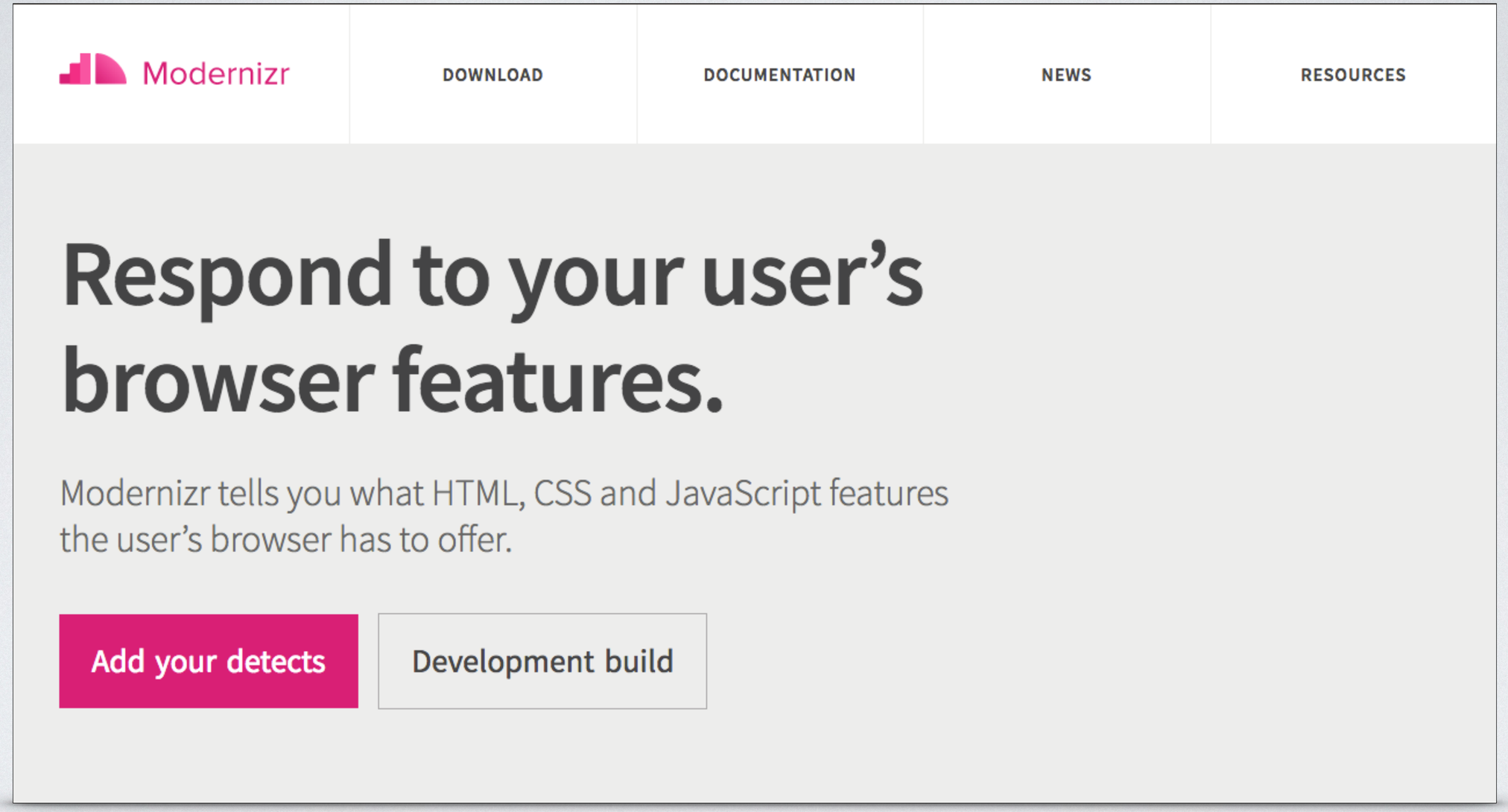
BOOTSTRAP

+

FEATURE QUERY

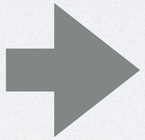
+

CSS GRID


Modernizr
DOWNLOAD
DOCUMENTATION
NEWS
RESOURCES
Respond to your user's browser features.
Modernizr tells you what HTML, CSS and JavaScript features the user's browser has to offer.
Add your detects
Development build

JS

Modernizr



CSS W3C®

Feature Queries

```
@supports
```

FEATURE QUERIES

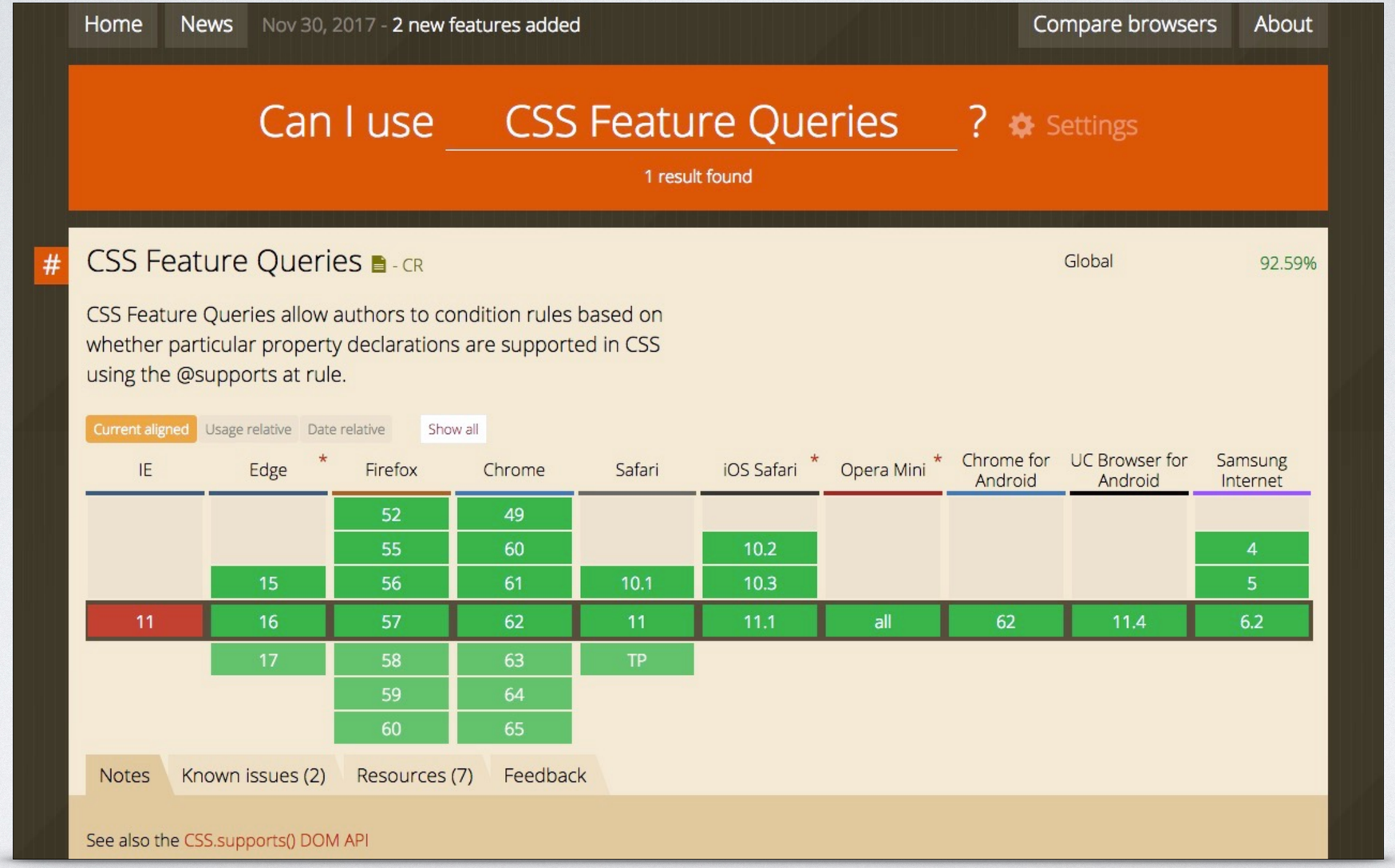
```
@supports (display: flex) {
  div { display: flex; }
}
```

```
@supports not (display: flex) {
  div { float: right; }
}
```

```
@supports (display: table-cell) and (display: list-item) {
  div { background: gold; }
}
```

```
@supports (display: table-cell) or (display: list-item) {
  div { background: gold; }
}
```

SOURCE: CSS Conditional Rules by MDN web docs.


Home
News
Nov 30, 2017 - 2 new features added
Compare browsers
About
Can I use CSS Feature Queries ?
Settings
1 result found
#
CSS Feature Queries - CR
Global
92.59%
CSS Feature Queries allow authors to condition rules based on whether particular property declarations are supported in CSS using the @supports at rule.
Current aligned
Usage relative
Date relative
Show all
IE
Edge *
Firefox
Chrome
Safari
iOS Safari *
Opera Mini *
Chrome for Android
UC Browser for Android
Samsung Internet
52
49
55
60
10.2
4
15
56
61
10.1
10.3
5
11
16
57
62
11
11.1
all
62
11.4
6.2
17
58
63
TP
59
64
60
65
Notes
Known issues (2)
Resources (7)
Feedback
See also the CSS.supports() DOM API

COMBINE THE POWER OF BOOTSTRAP + CSS GRID

Bootstrap Grid code

```
@supports (display: grid) {
   .row {
     display: grid;
     grid-template-columns: repeat(12, 1fr);
     grid-gap: 20px;
   }
   ...
}
```

CONTAINER

+

MEDIA QUERIES

+

ROWS & COLS

Is **Bootstrap** still useful
now that we have **CSS Grid** 

Content
Reboot
Typography
Code
Images
...

Components
Alerts
Badge
Breadcrumb
Buttons
...

Layout
Overview
Grid
Media object
Responsive utilities

Utilities
Borders
Clearfix
Close icon
Colors
...

Content
Reboot
Typography
Code
Images
...

Components
Alerts
Badge
Breadcrumb
Buttons
...

Layout
Overview
Grid
Media object
Responsive utilities

Utilities
Borders
Clearfix
Close icon
Colors
...

Bootstrap

LAYOUT

IN A ROCKET

Learn front-end development at rocket speed