CSS GRID LAYOUT

FINALLY!

Manuel
Matuzović
@mmatuzo

Overview

- What is Grid and why do we need it.
- Browser support: Can we use it today?
- New properties, units and keywords.
- CSS Box Alignment Module.
- CSS Intrinsic And Extrinsic Sizing.
- Accessibility.

Overview

- Common patterns.
- Stacking order.
- Responsive Webdesign.
- Progressive Enhancement.
- Debugging.
- The future of Grid (Level 2).

WHAT'S CSS GRID LAYOUT?

What's CSS Grid Layout?

- The first true layout method in CSS.
- float, display: inline-block, position, display: table were originally not intended for building layouts.
- Two-dimensional layouting.
- Grid structures are described in CSS and not in HTML.
- Kinda like table layouts, but in CSS, responsive and flexible.

History and background

- Early 90s: First ideas, but “too complex” to implement.
- 1996 “frame-based” layout model.
- 2005 Advanced Layout Module, which later turned into the Template Layout Module.
- Later Microsoft was in the planning stages of Windows 8 and were going to enable building apps with web technologies.

History and background

- Microsoft shipped a grid layout implementation behind the `-ms-` vendor prefix in Internet Explorer 10 in 2011.
- They presented a draft Grid Layout spec to the W3C in 2012.
- The spec that shipped is not the same spec we have today. Among other differences, it has no lines, auto placement or templates.
- CSS Working Group began tweaking Microsoft's proposal. The plan was to unite Mircrosofts proposal, Bos' ideas in the Advanced Layout Module and the idea of *drawing lines*.

History and background

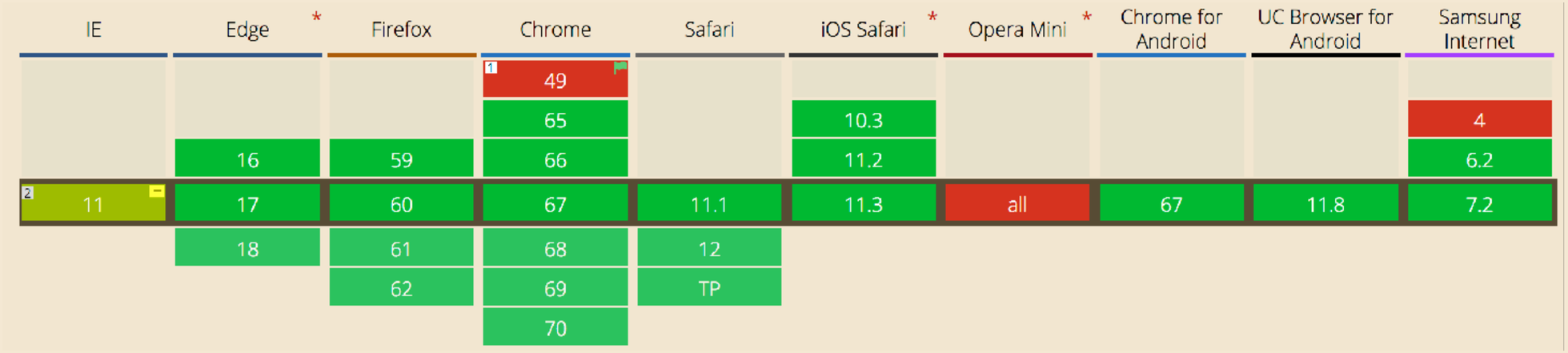
- The media company Bloomberg hired Igalia, an open source consultancy, to implement CSS Grid for both Blink and WebKit.
- Even with two ready-made implementations, some browsers were still concerned the feature wouldn’t find its footing.
- But, a good number of designers and developers were starting to get excited about Grid and began to put pressure on browser vendors to implement it.
- Especially demos and articles by Rachel Andrew and Jen Simmons got people really exited.

History and background

- Google landed an implementation of CSS Grid in Chromium 56 for Android in January of 2017.
- Chrome and Firefox implementations in early March.
- Opera and Safari by the end of March 2017.
- Ironically, the last company to ship CSS Grid was Microsoft (October 17th).
- All major browsers rolled out their implementations in a single year!

BROWSERSUPPORT?

Browsersupport



Browsersupport

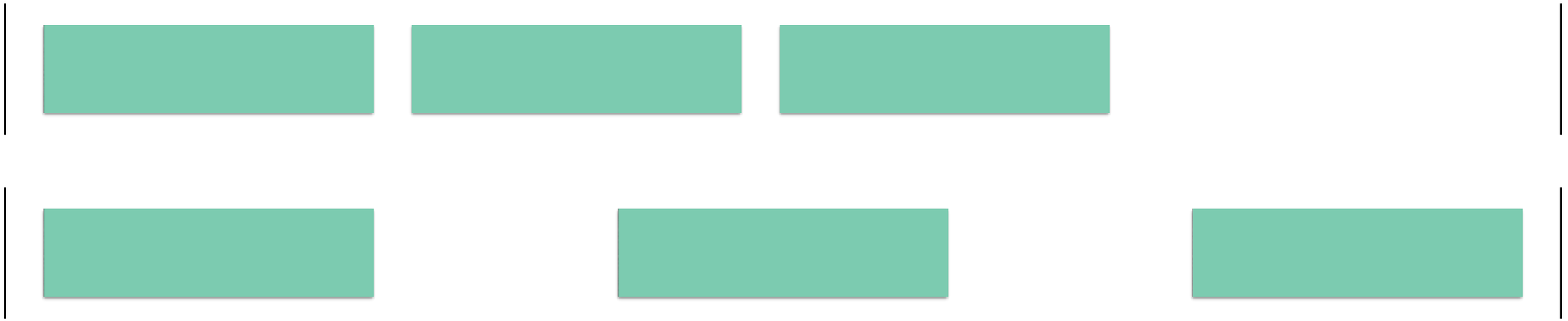
- All major desktop and mobile browsers.
- IE 10 - Edge 15: Old Grid specification.
- Edge 16+ both specs.

IS GRID A REPLACEMENT FOR FLEXBOX?

Is Grid a replacement for Flexbox?

Flexbox

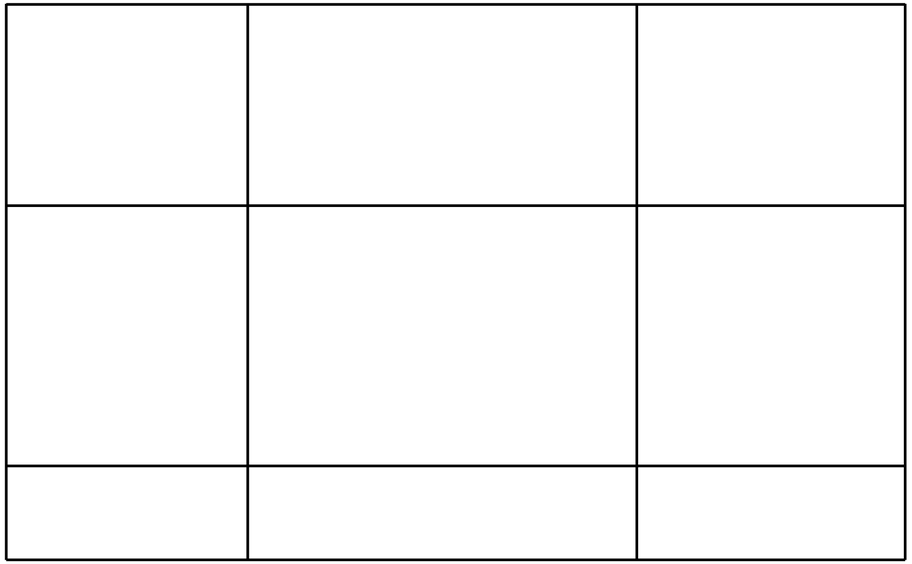
With Flebox **content dictates the layout.** For example, we use it for even distribution of items or filling available space with items.

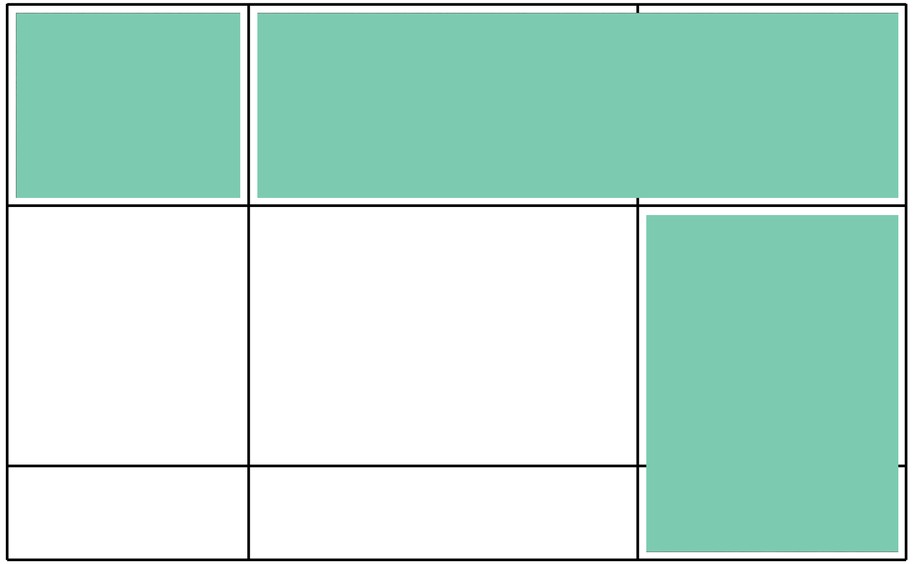


Is Grid a replacement for Flexbox?

Grid

With grid **the layout has to be defined**. We define columns and rows.
Items are explicitly or implicitly placed in those rows.





Is Grid a replacement for Flexbox?

Flexbox

...for one dimension (column or row).

...when child items dictate the layout.

...if you want to distribute the available space.

Is Grid a replacement for Flexbox?

Grid

…for two dimensions (column and row).

…when the parent item dictates the layout.

…as soon as flex items are losing their flexibility.

„If you are adding widths to all your flex items, you probably need grid “

– Rachel Andrew | Render 2017

GRID BASICS

grid, tracks, cells, gaps

Grid Basics

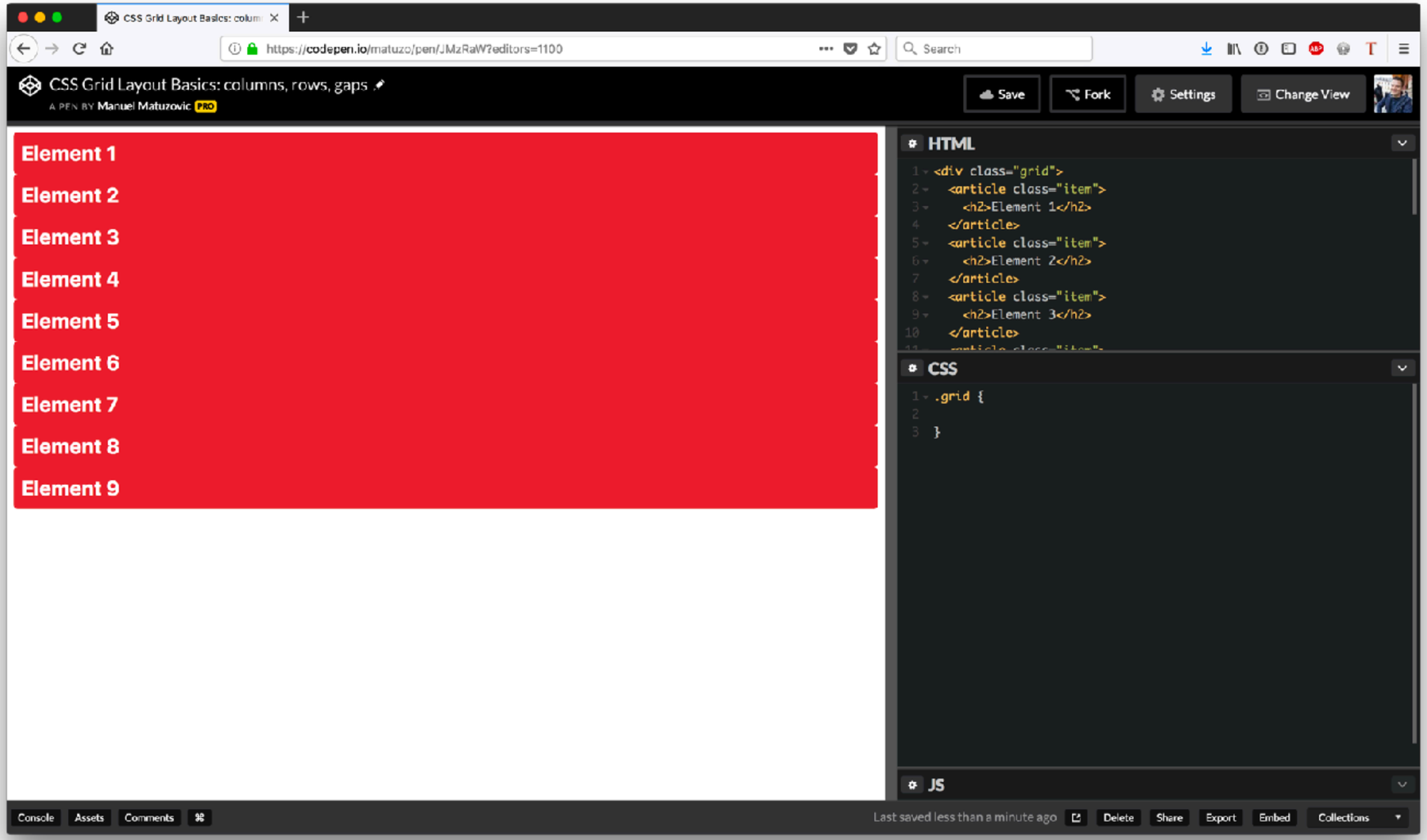
LEGEND

The element where display has been set to grid.
The grid container.

A direct child item of .container.

```
.container {
  …
}
```

```
.grid-item {
  …
}
```



COLUMNS, ROWS & GAPS

bit.ly/grid_basic1

Grid Basics

Defining grids.

Values:
grid
inline-grid

```
.container {
  display: grid;
  display: inline-grid;
}


/* Starting with level 2 of the spec */
.container {
  display: subgrid;
}
```

Grid Basics

Adding rows
(explicit grid)

```css
.container {
  grid-template-rows: 100px 200px;
}
```

Adding columns
(explicit grid)

```css
.container {
  grid-template-columns: 200px 200px 150px;
}
```

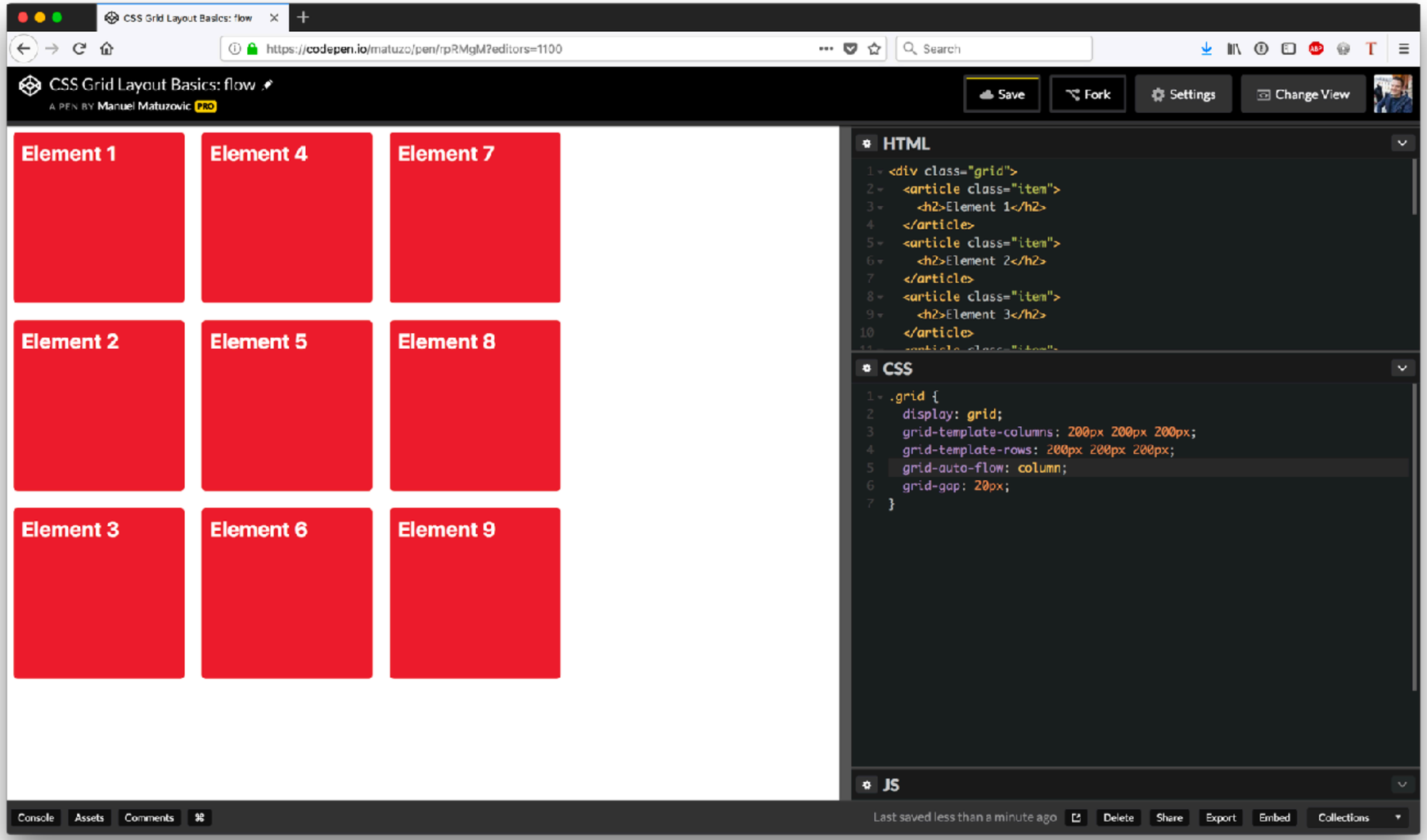
Grid Basics

Gaps (gutter)

```
.container {
  grid-gap: 20px;
}
```

Gaps for each direction

```
.container {
  grid-column-gap: 10px;
  grid-row-gap: 10px;
}
```



FLOW

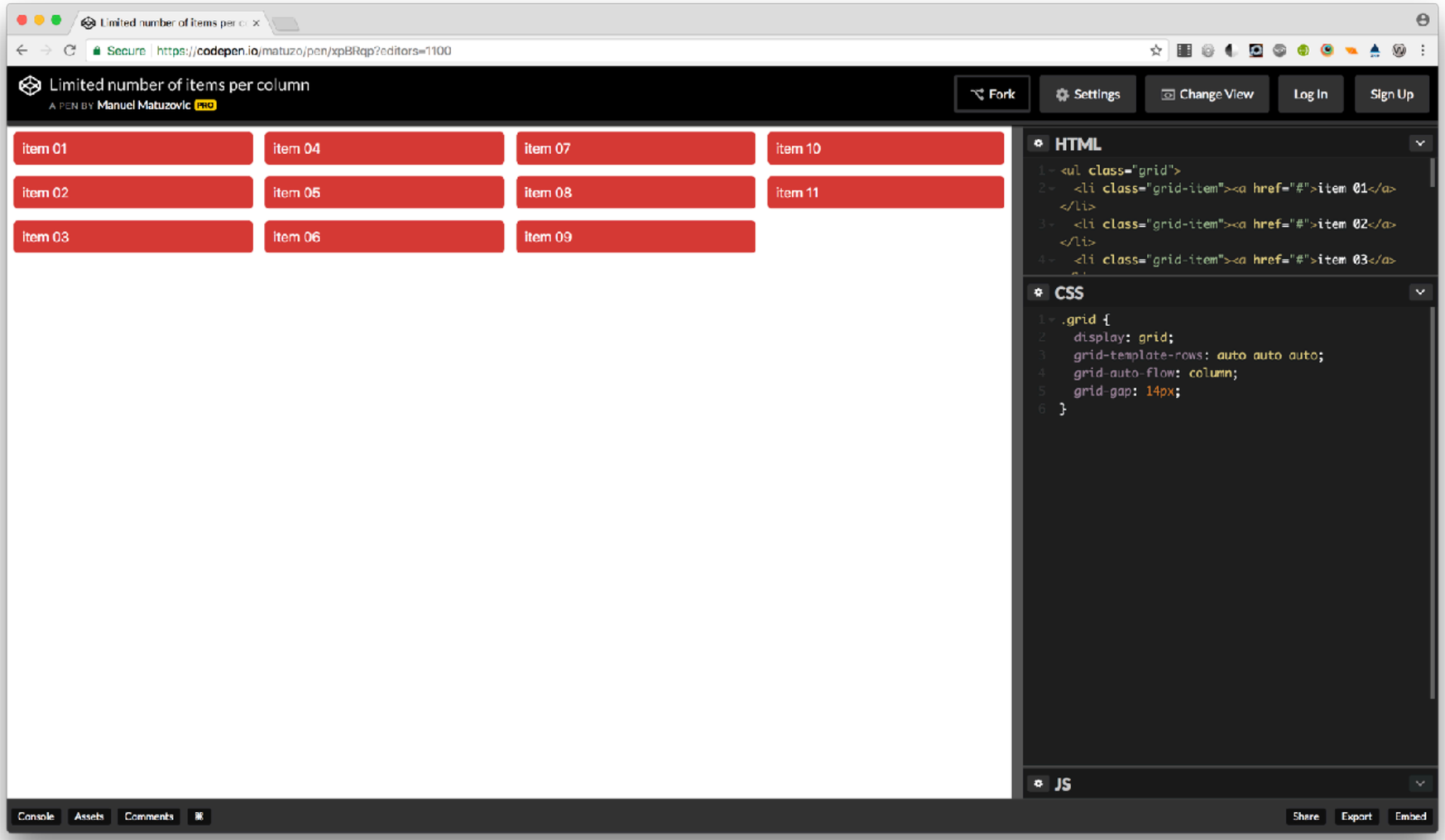
bit.ly/grid_flow

Grid Basics

Controlling
auto-placement

Values:
row (default)
column
dense

```
.grid {
  grid-auto-flow: column;
}
```



LIMITED NUMBER OF ITEMS PER COLUMN

bit.ly/grid_flow2

Grid Basics

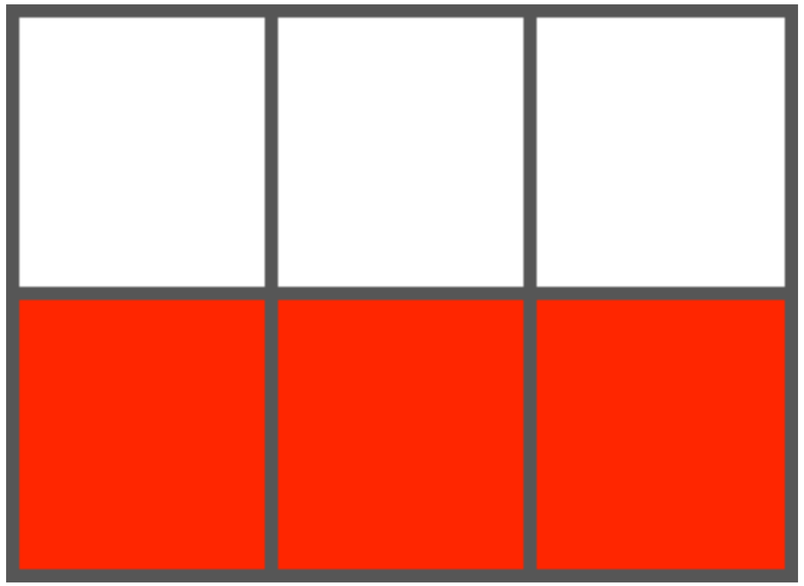
A “dense” packing algorithm, which attempts to fill in holes earlier in the grid if smaller items come up later.

May cause a disconnect between DOM and visual order. Use it deliberately.

```
.grid {
  grid-auto-flow: dense;
}
```

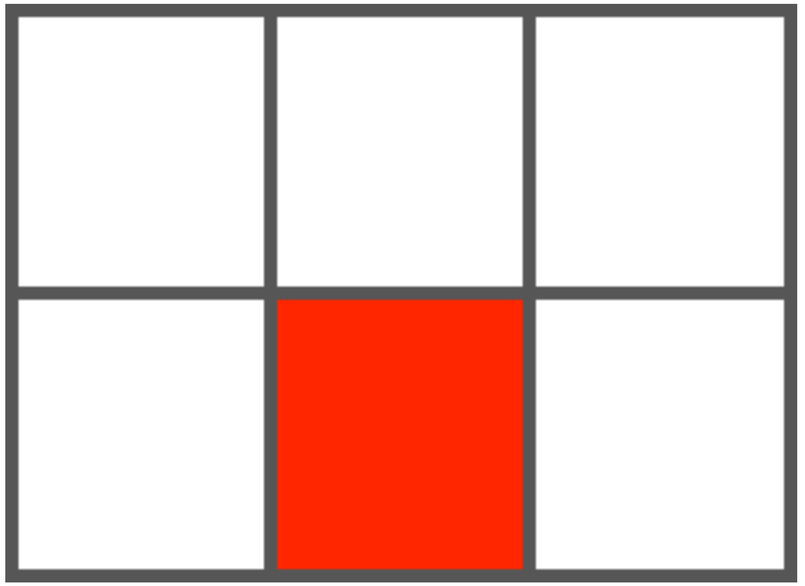
Terminology

TRACK



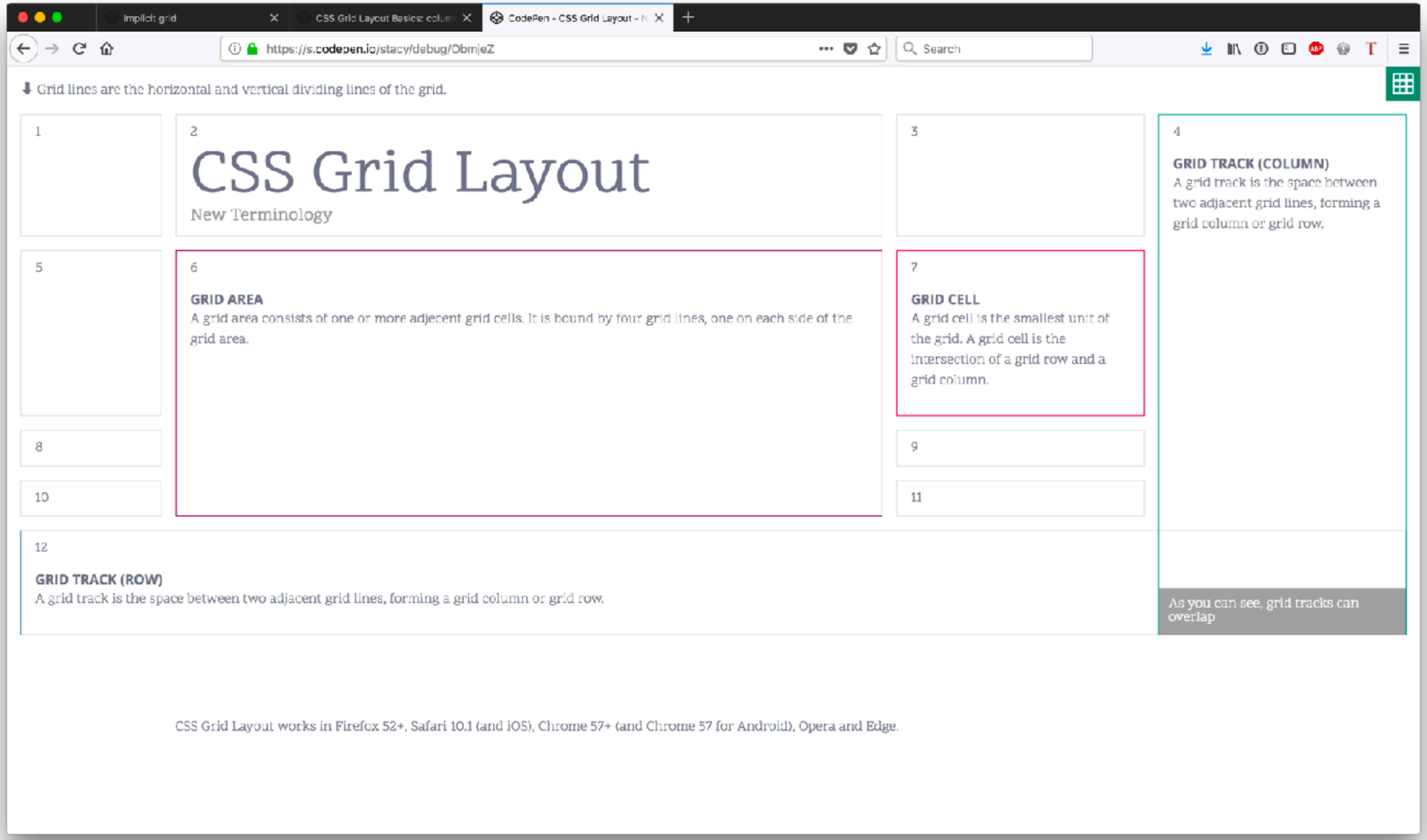
Either horizontal or vertical

CELL



A single cell within a track

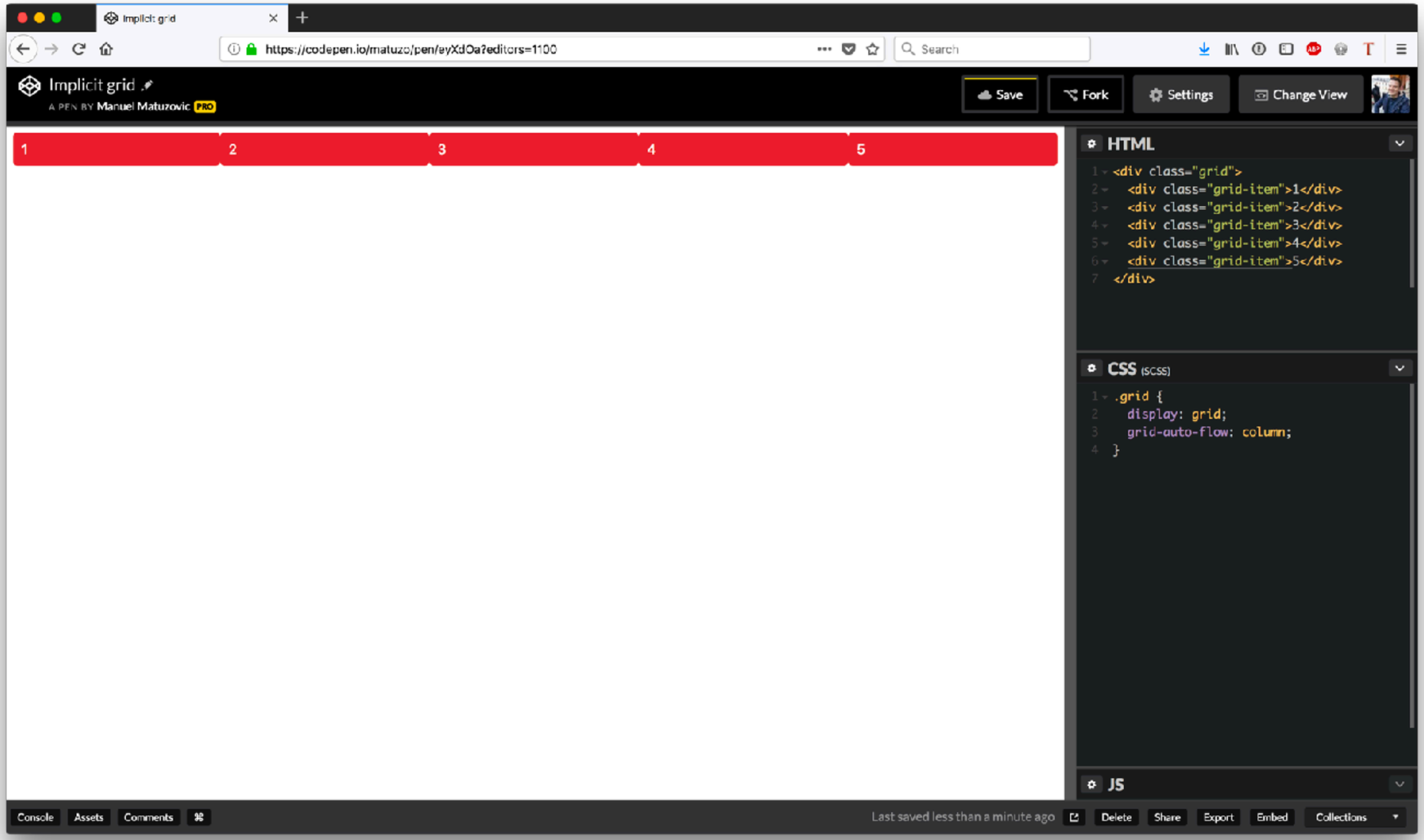
Illustrations: https://gridbyexample.com/what



GRID TERMINOLOGY

s.codepen.io/stacy/debug/ObmjeZ

EXPLICIT GRID VS. IMPLICIT GRID



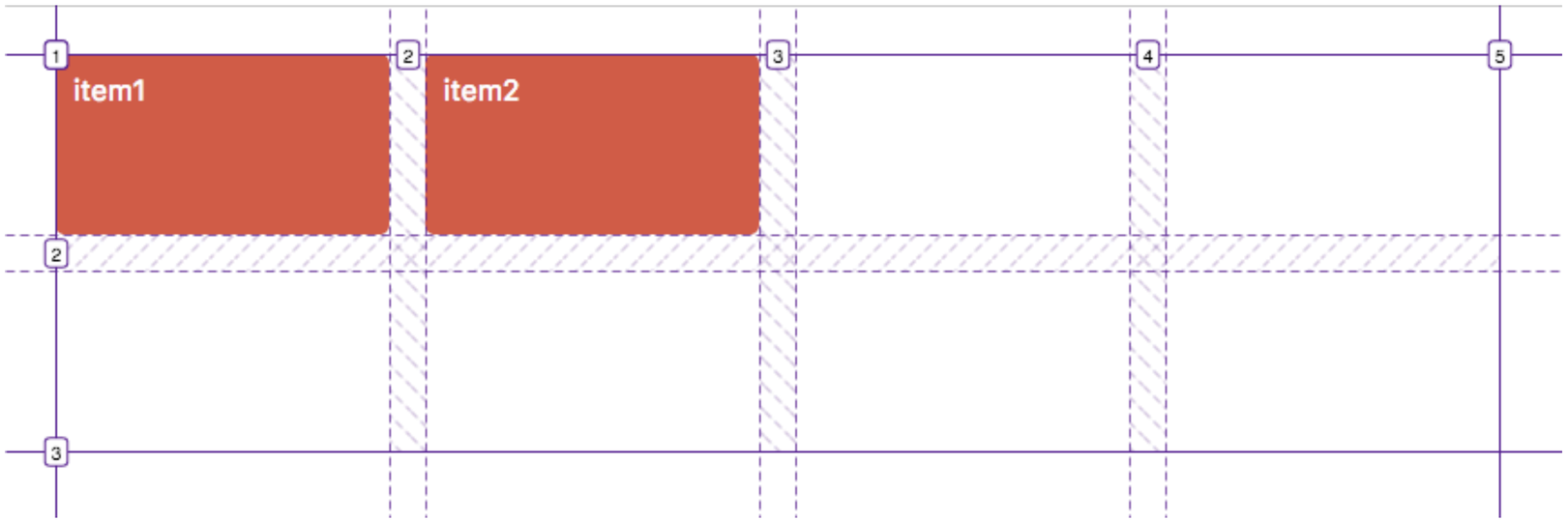
IMPLICIT GRIDS

bit.ly/grid_implicit

Explicit grids

- We can define a fixed number of lines and tracks that form a grid by using the properties grid-template-rows, grid-template-columns, and grid-template-areas.

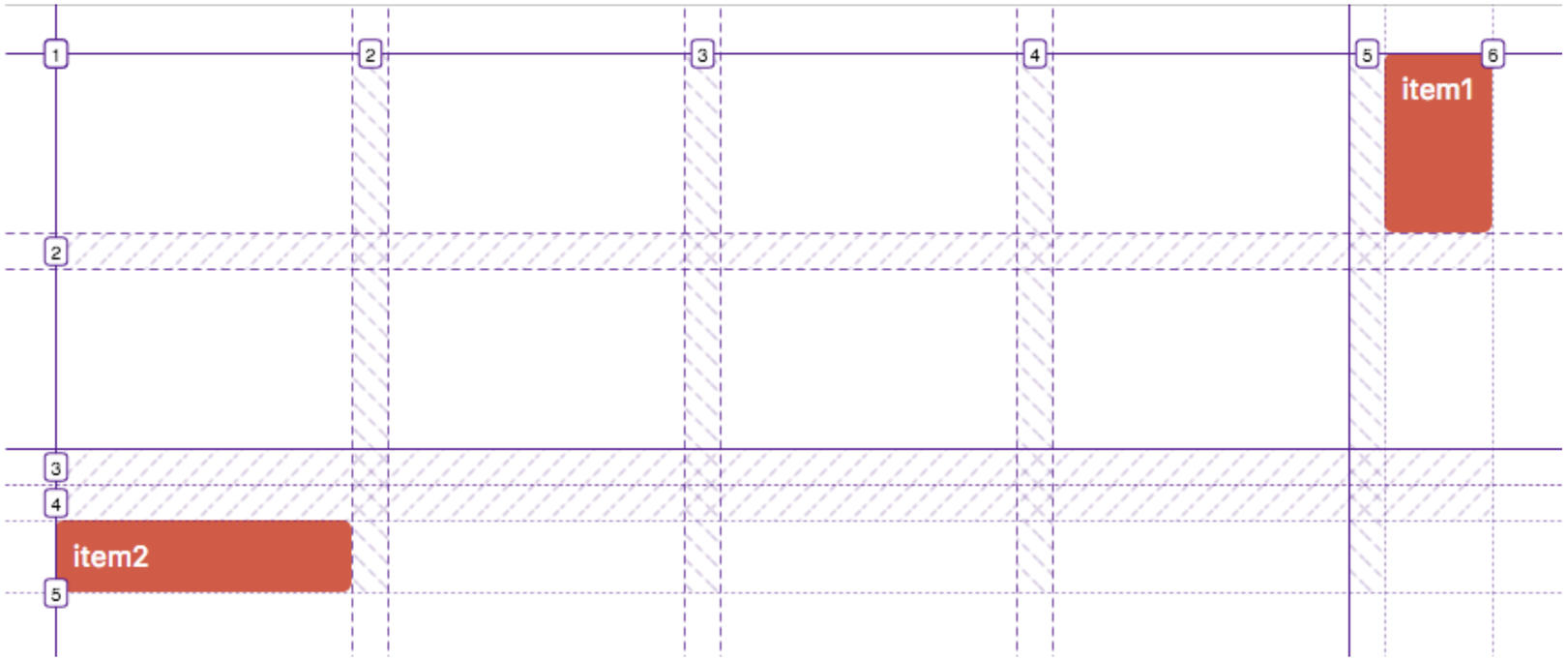
- This manually defined grid is called the explicit grid.



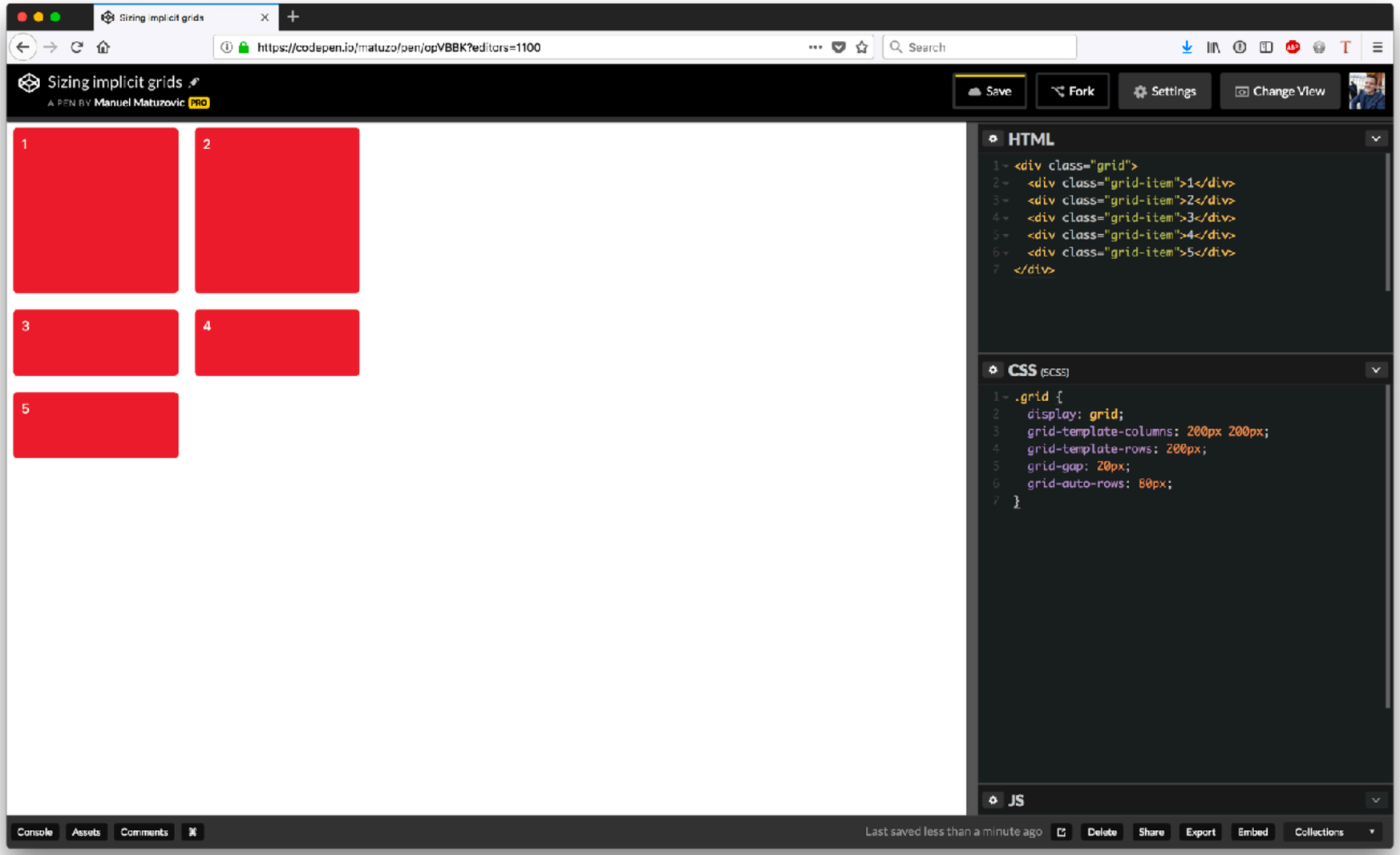
EXPLICIT GRID

Implicit grids

- The grid container automatically generates grid tracks by adding grid lines to the grid
  - … if there are more grid items than cells in the grid
  - …when a grid item is placed outside of the explicit grid.
- The widths and heights of the implicit tracks are set automatically. By default, they are only big enough to fit the placed grid items.



IMPLICIT GRIDS



SIZING IMPLICIT GRIDS

bit.ly/grid_implicit2

Grid Basics

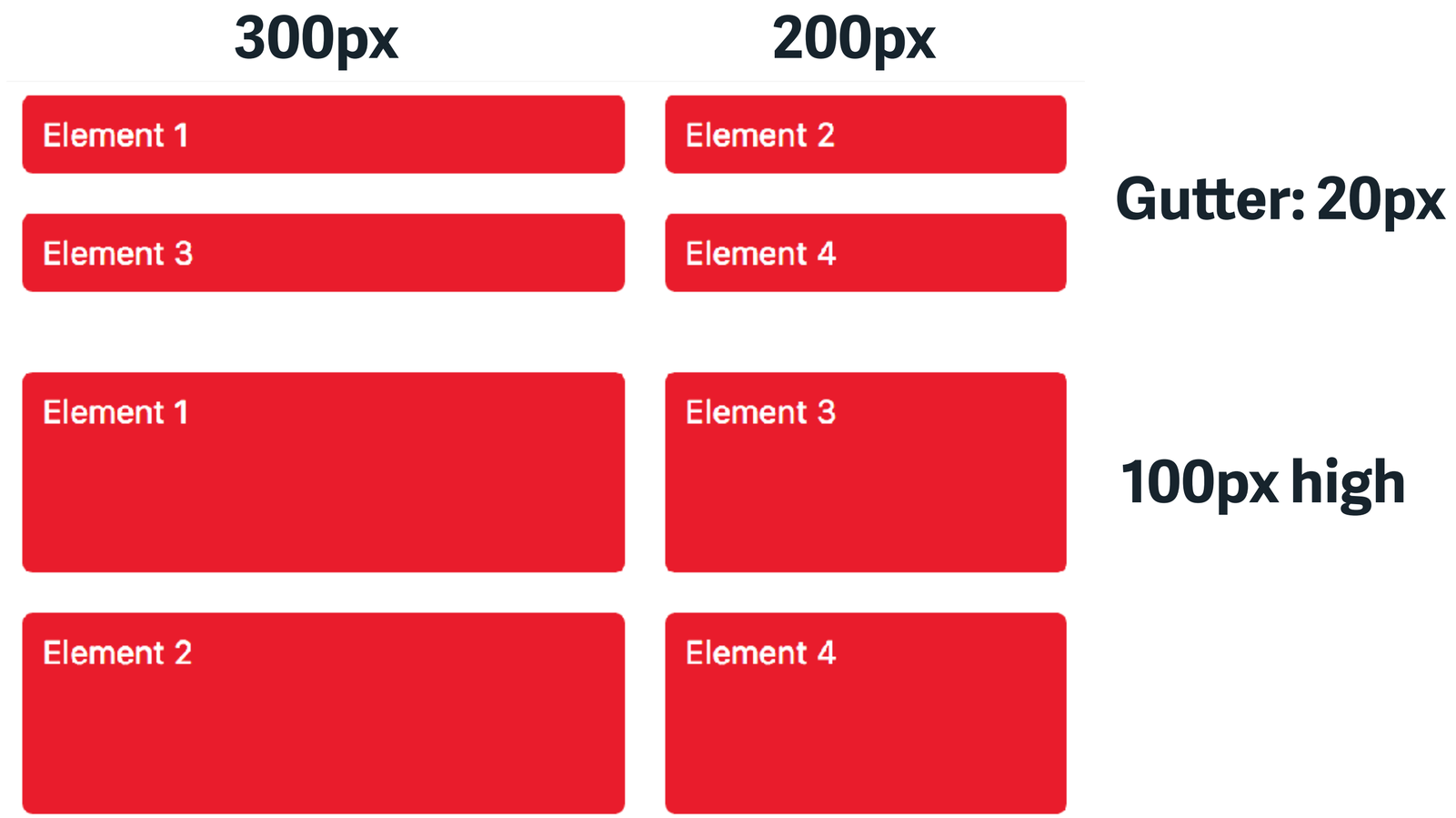
Sizing implicit columns

```
.container {
  grid-auto-columns: 80px;
}
```

Sizing implicit rows

```
.container {
  grid-auto-rows: 80px;
}
```

EXERCISE 1



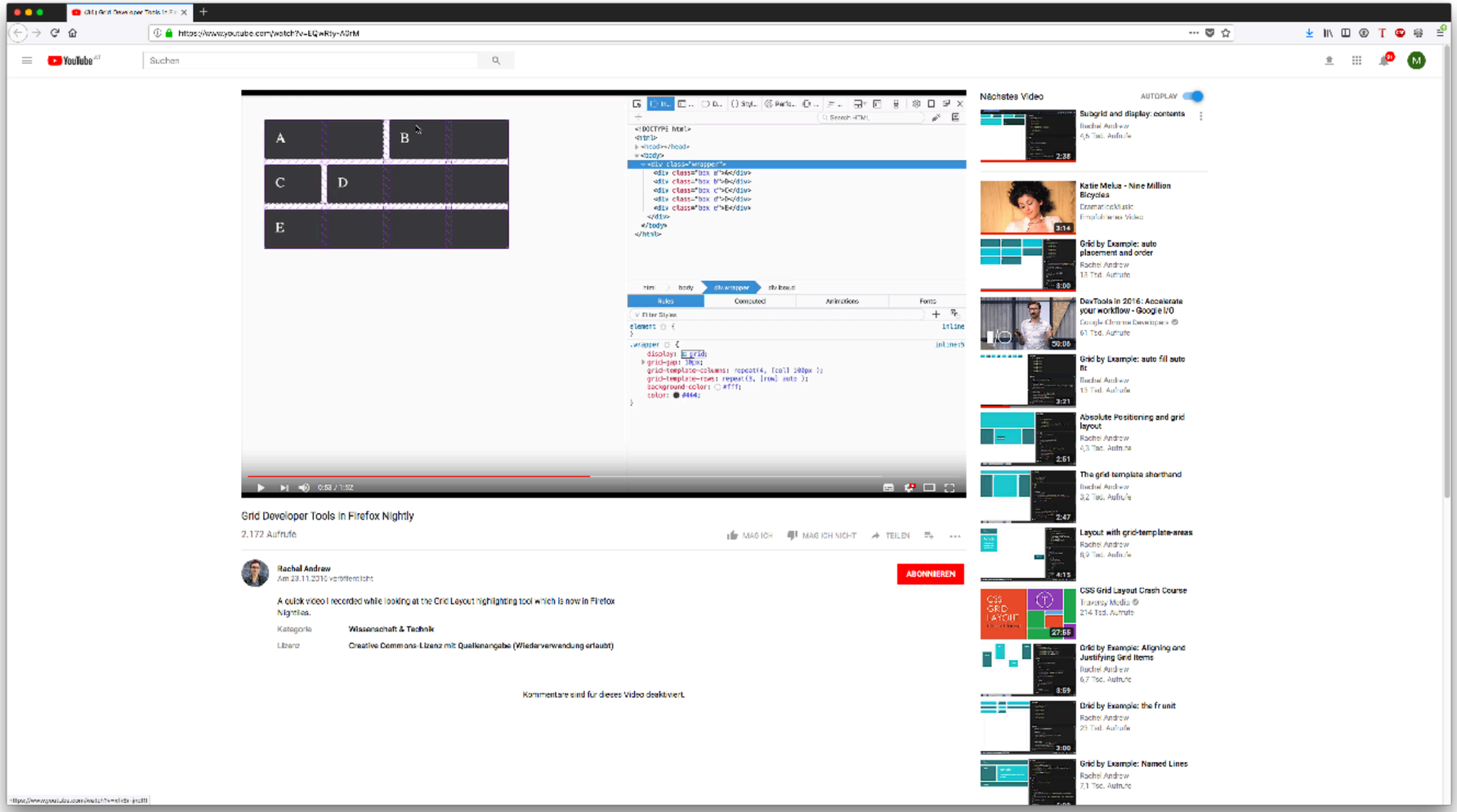
BIT.LY/GRID_EXERCISE1

DEBUGGING

Debugging

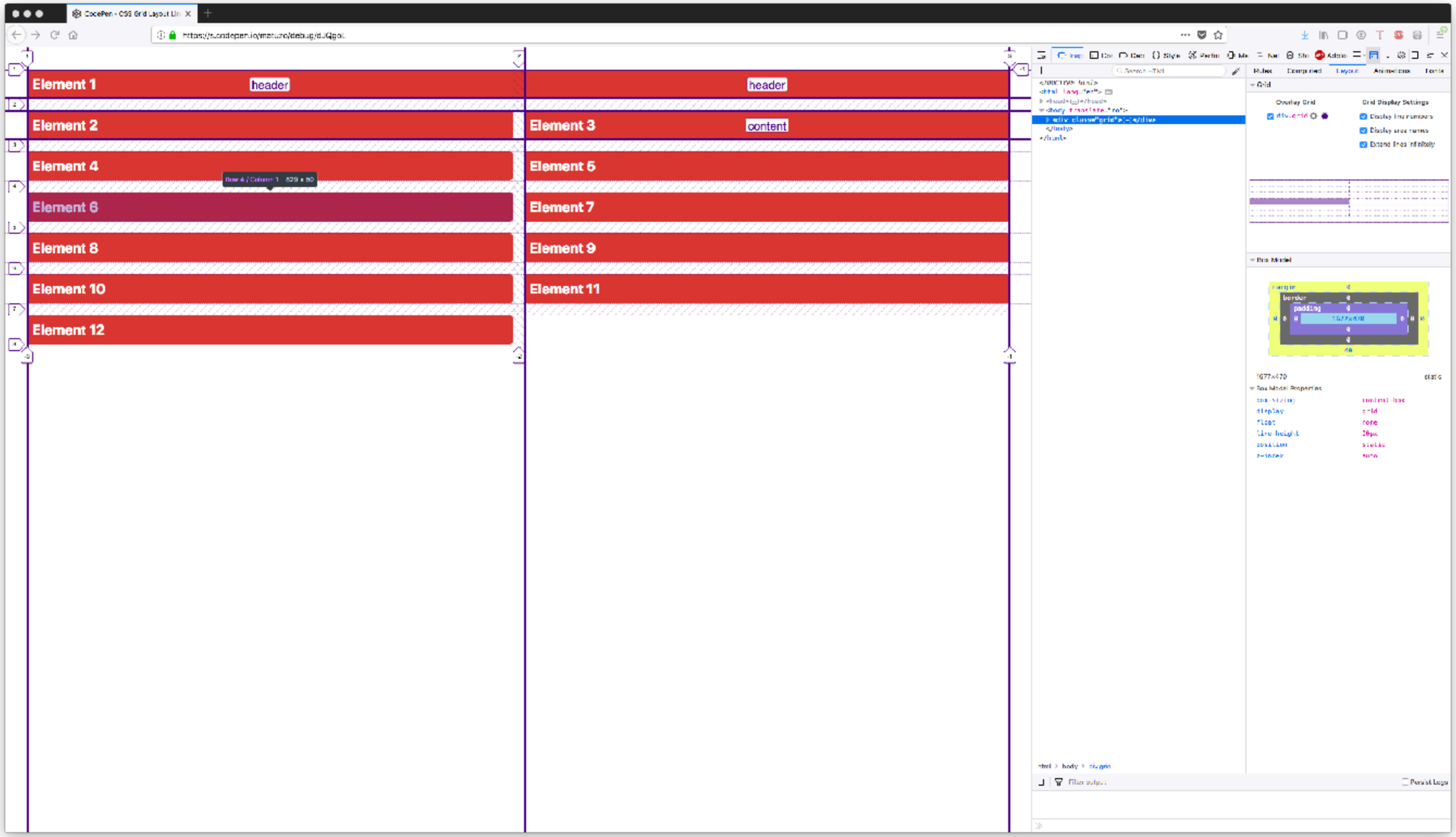
Firefox is currently the best browser for debugging.

https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector/How_to/Examine_grid_layouts



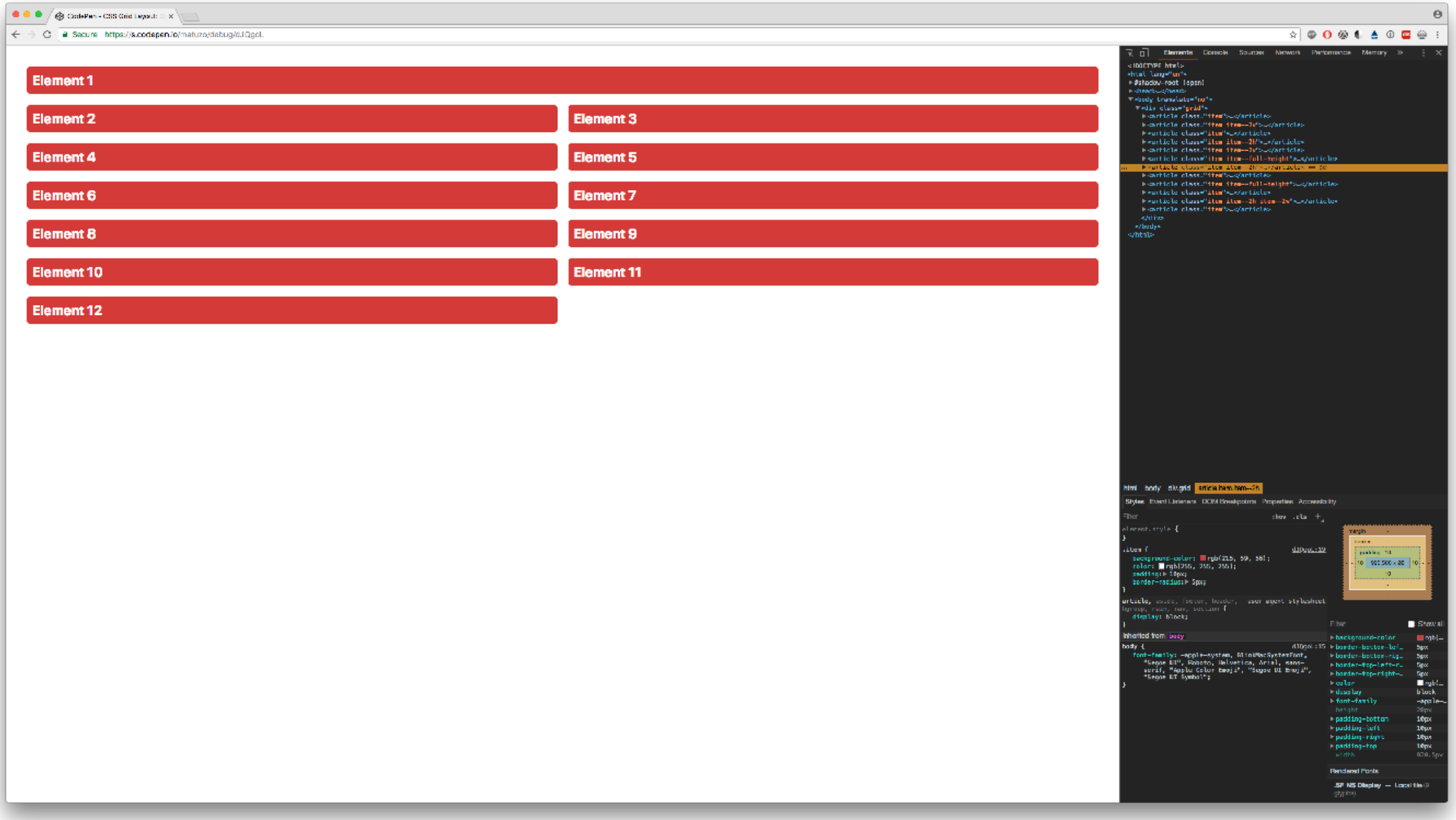
GRID DEBUGGING IN FIREFOX

www.youtube.com/watch?v=EQwRty-A0rM



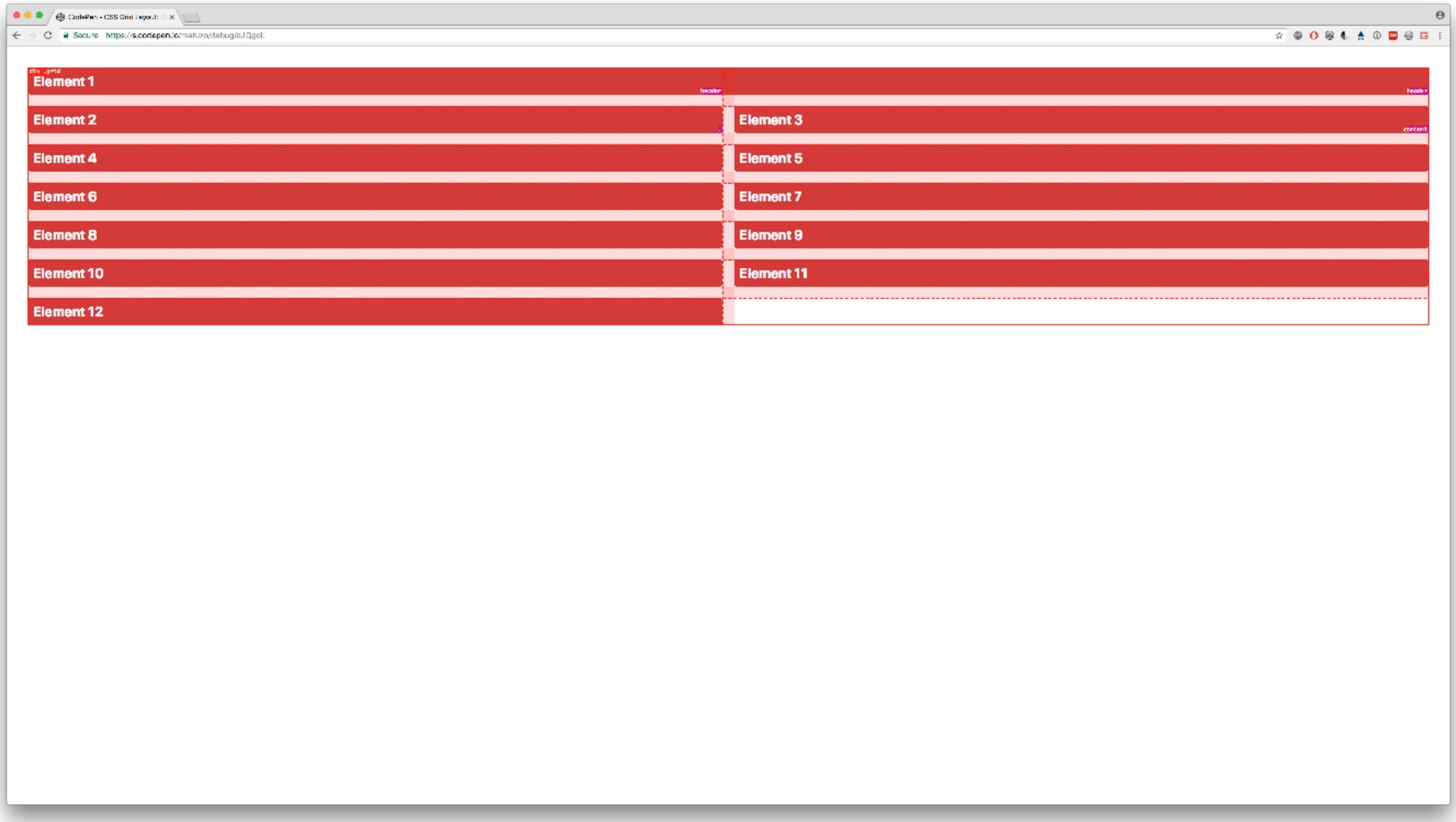
DEBUGGING IN FIREFOX

bit.ly/grid_debug



DEBUGGING IN CHROME

bit.ly/grid_debug

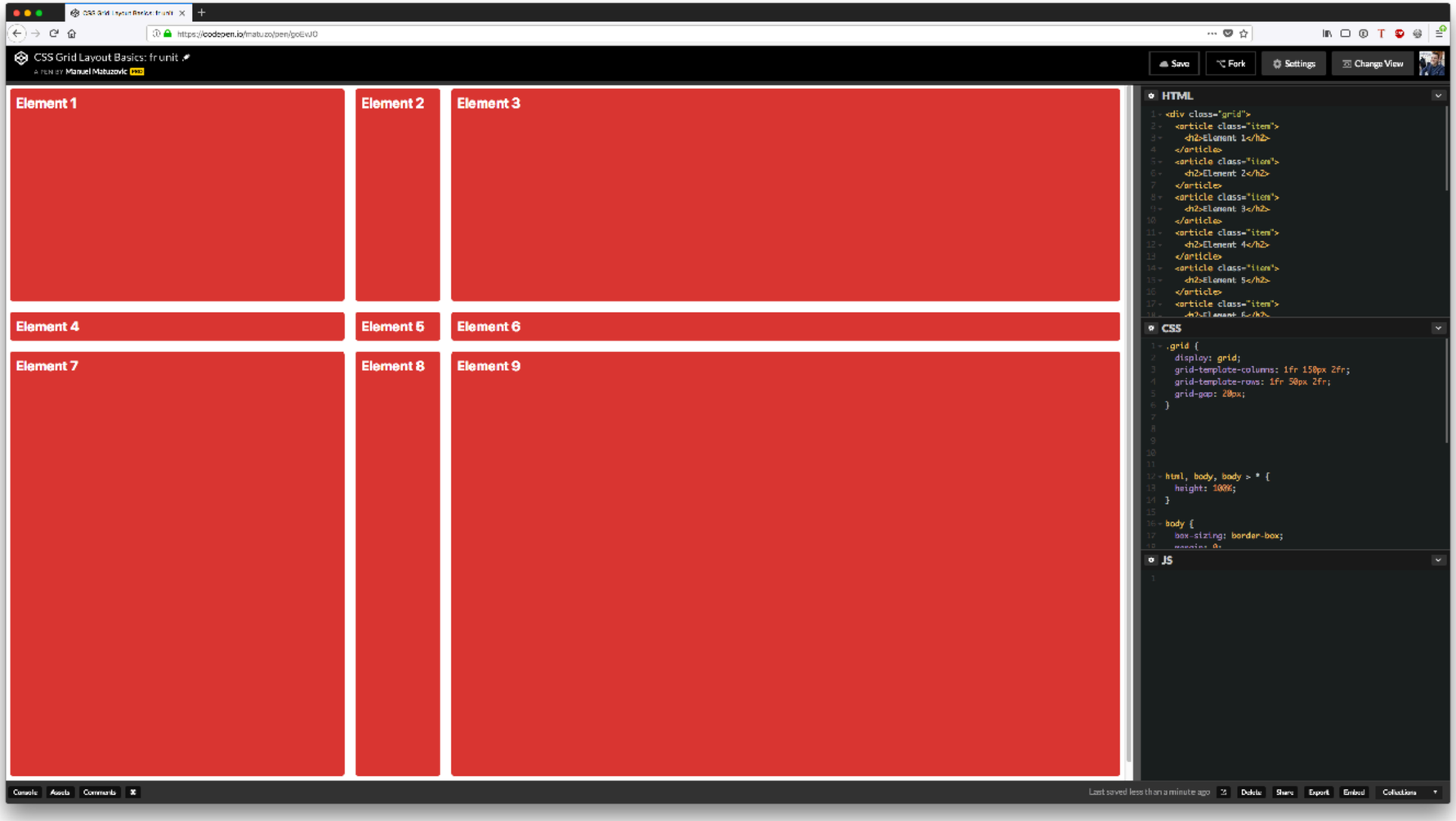


GRIDMAN - CSS GRID INSPECTOR. ULTRA FAST!

Chrome

GRID BASICS

fr unit, repeat, minmax



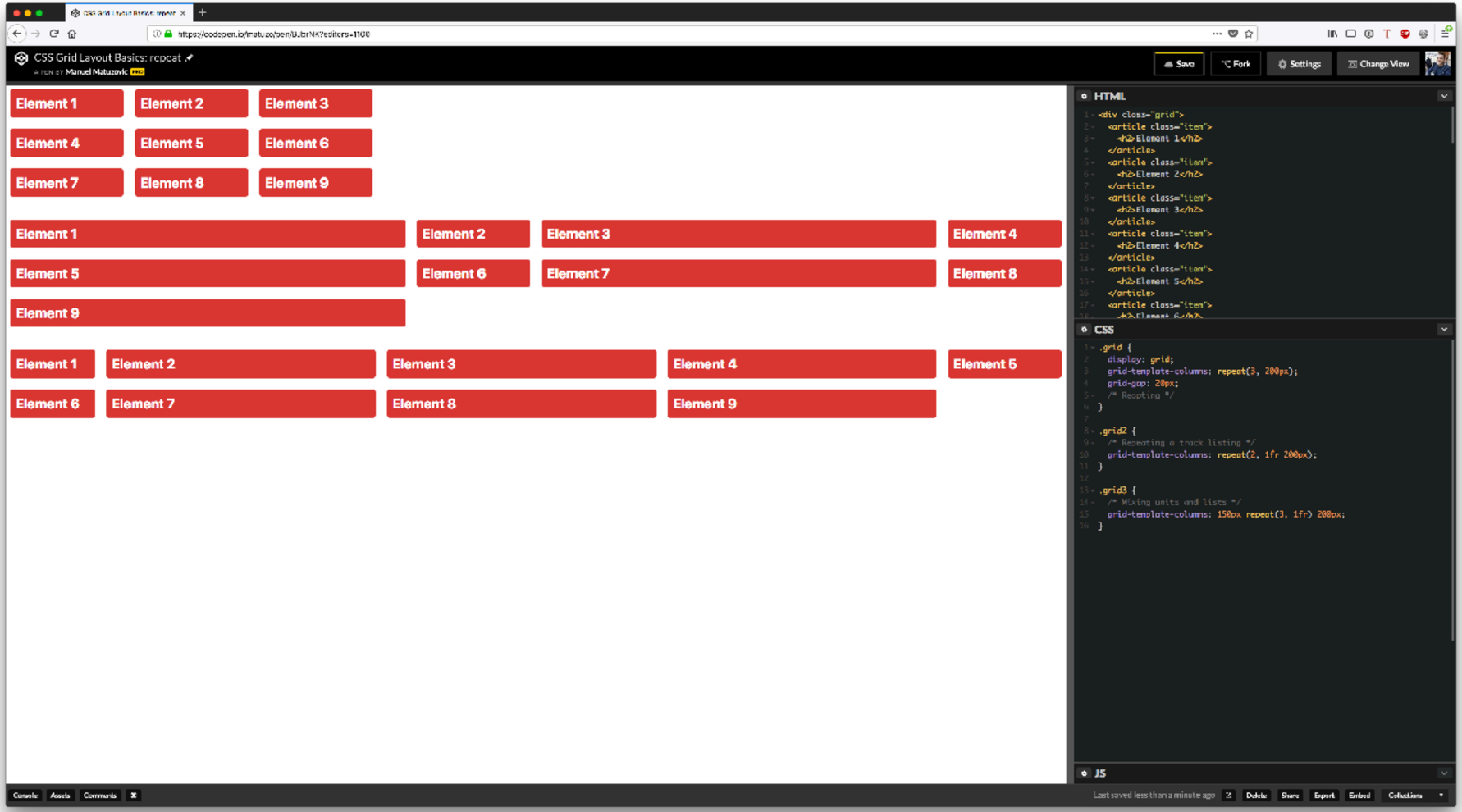
FR UNIT

bit.ly/grid_fr

Grid Basics

Flexible units (fr)

```
.container {
  grid-template-columns: 1fr 150px 2fr;
  grid-template-rows: 1fr 50px 2fr;
}
```



REPEATING TRACKS

bit.ly/grid_repeating

Grid Basics

Repeating tracks or track lists

```
.container {
  grid-template-columns: repeat(3, 200px);
  grid-template-columns: repeat(2, 1fr 200px);
  grid-template-columns: 150px repeat(3, 1fr) 200px;
}
```

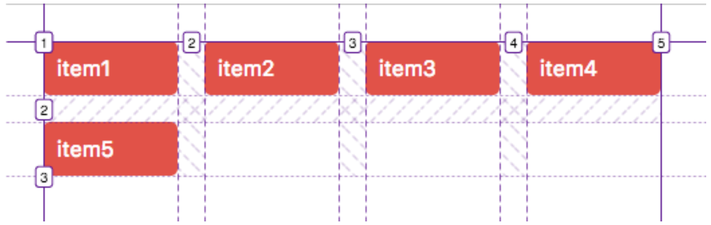
```css
.grid {
  grid-template-columns: repeat(4, 100px);
}
```


```css
.grid {
  grid-template-columns: repeat(auto-fill, 100px);
}
```

AUTO-FILLING TRACKS

bit.ly/grid_autofill

```
.grid {
  grid-template-columns: repeat(auto-fit, 100px);
}
```



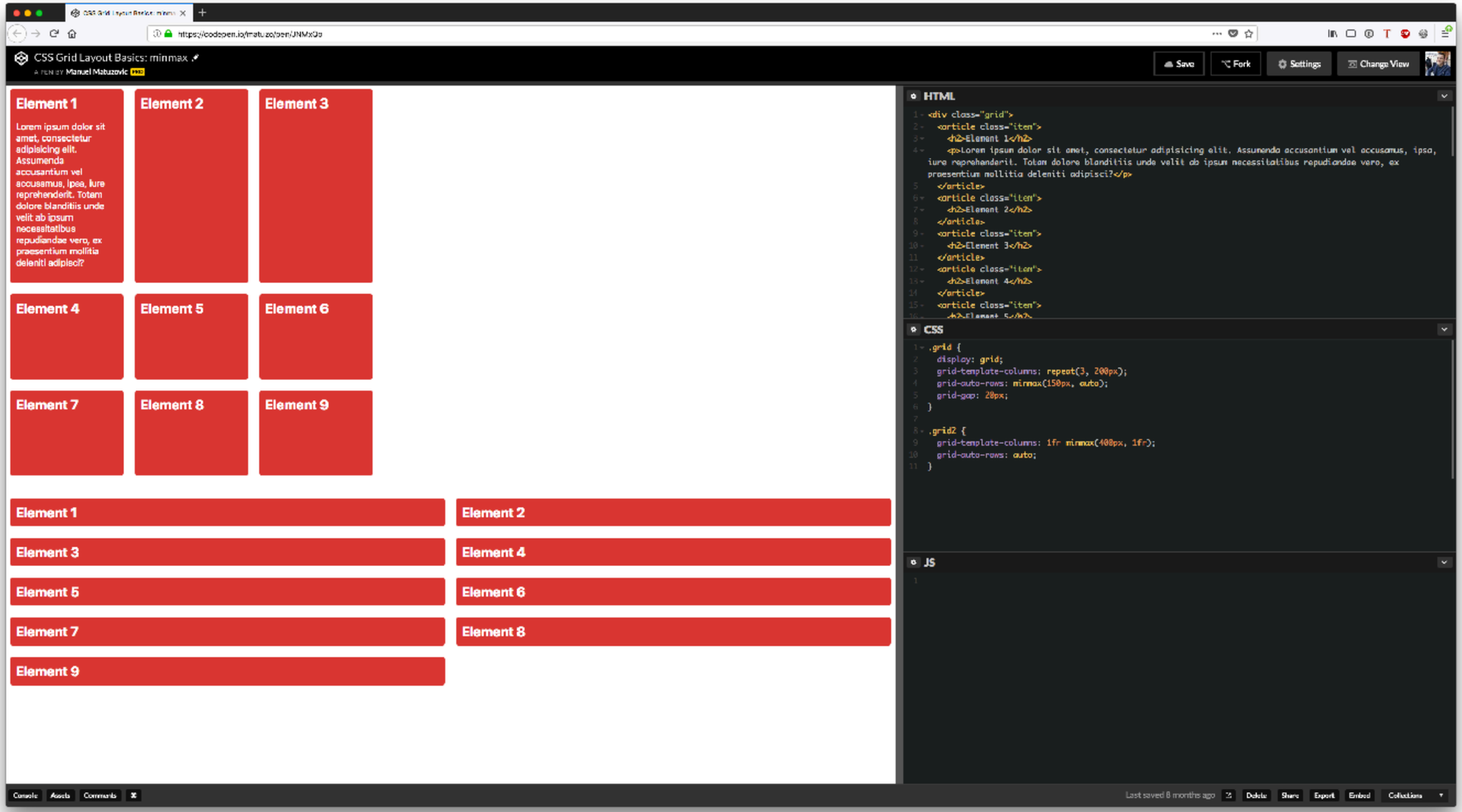
AUTO-FITTING TRACKS

bit.ly/grid_autofit

Grid Basics

Auto-filling and auto-fitting tracks

```
.container {
  grid-template-columns: repeat(auto-fill, 200px);
  grid-template-columns: repeat(auto-fit, 100px);
}
```



MINMAX

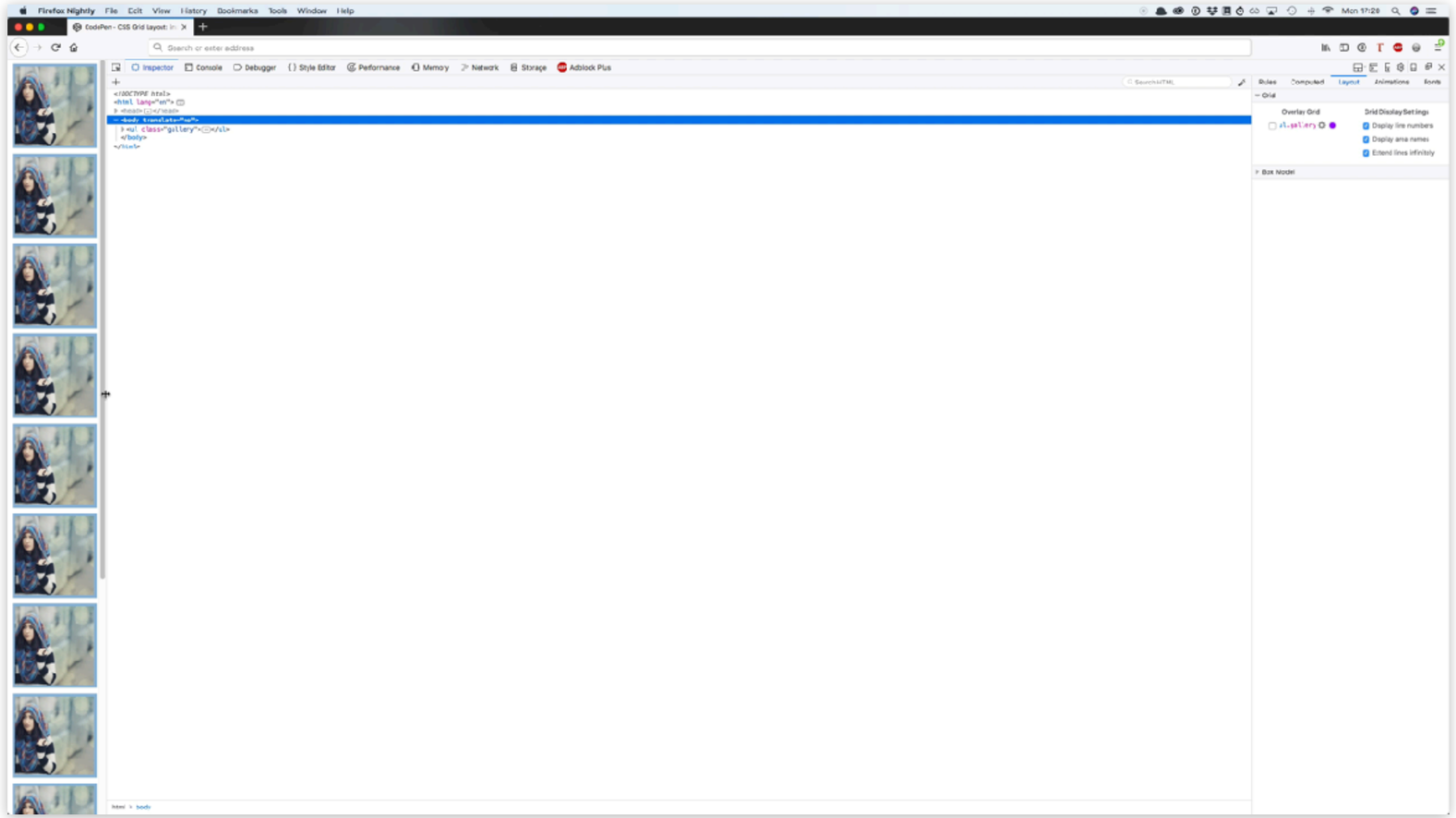
bit.ly/grid_minmax

Grid Basics

Defining min and max values

```
.container {
  grid-auto-rows: minmax(150px, 250px);
  grid-template-columns: 1fr minmax(400px, 1fr);
}
```

EXERCISE 2

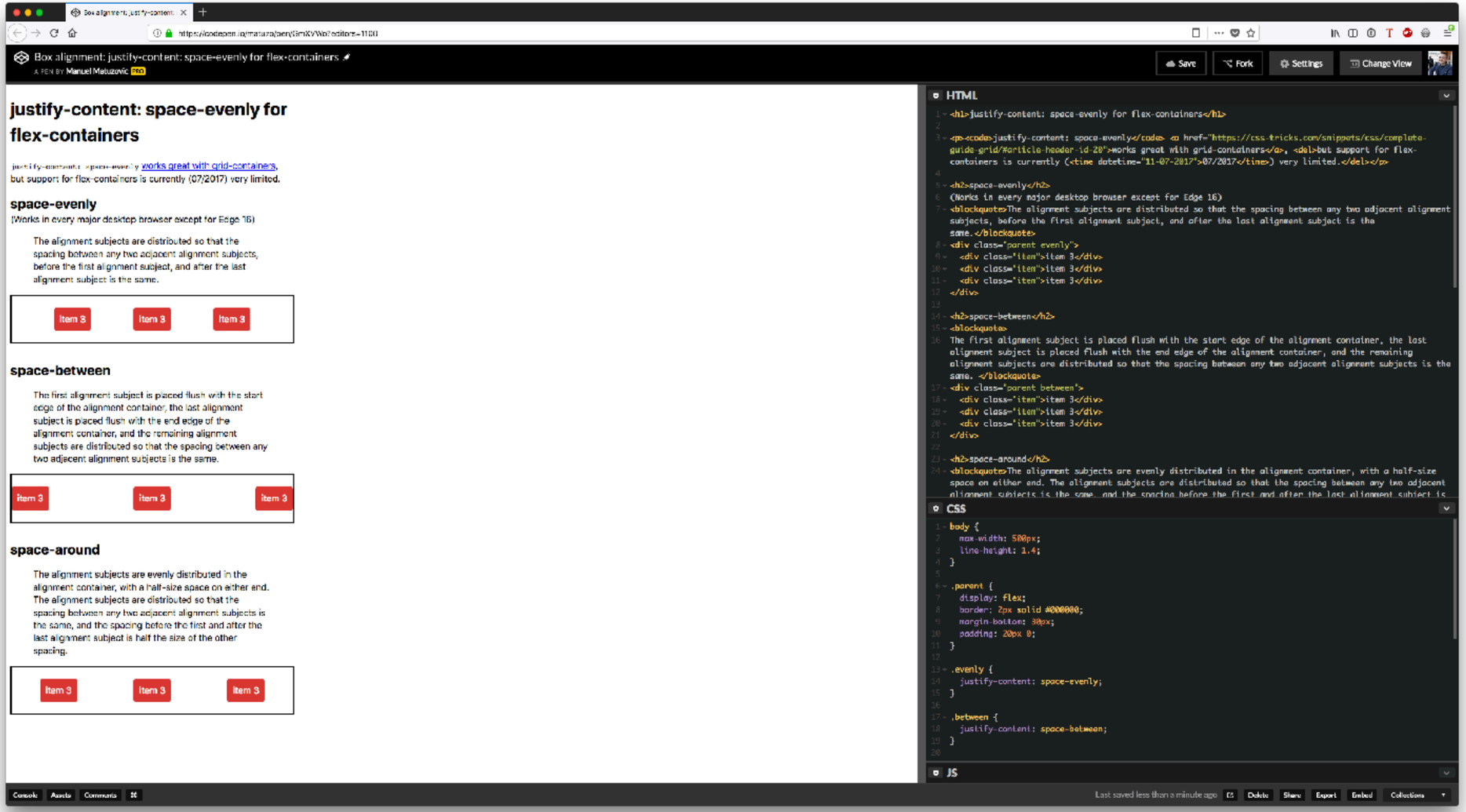


BIT.LY/GRID_EXERCISE2

CSS BOX ALIGNMENT MODULE

Box Alignment Module

- Separate specification for alignment properties.
- justify-items, align-items, justify-self, align-self, justify-content, align-content, grid-gap
- place-items, place-self, gap
- Properties do not just apply to Grid and Flexbox (In the future also to block layout).

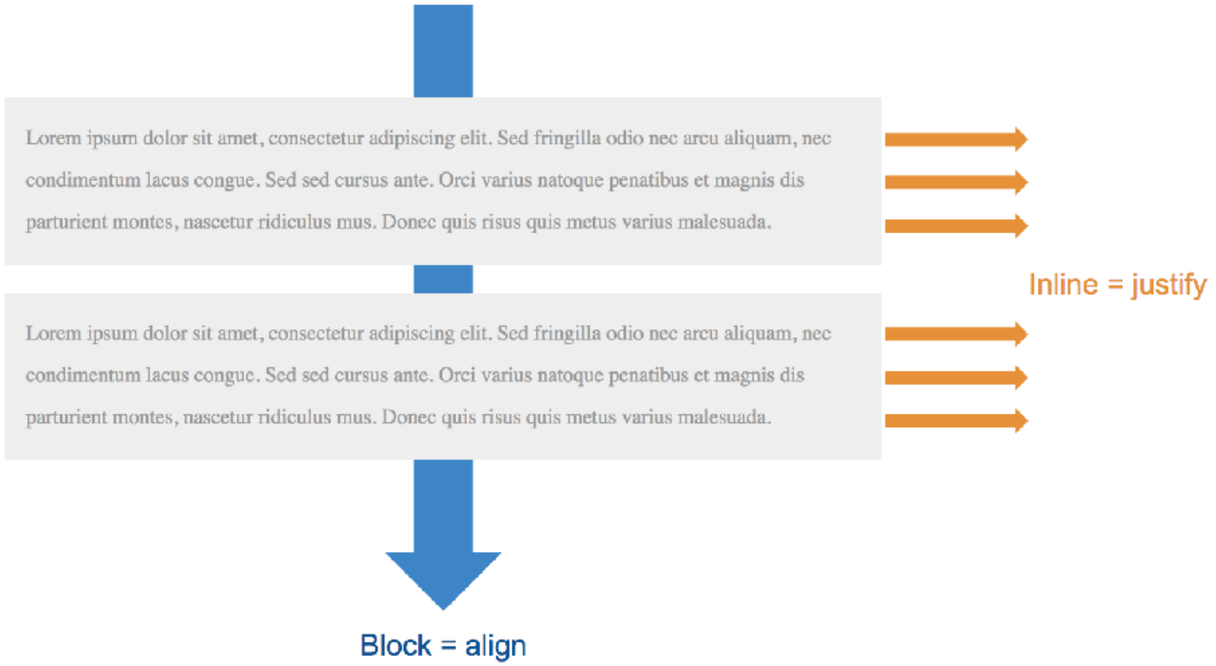


JUSTIFY-CONTENT: SPACE-EVENLY

bit.ly/grid_align3

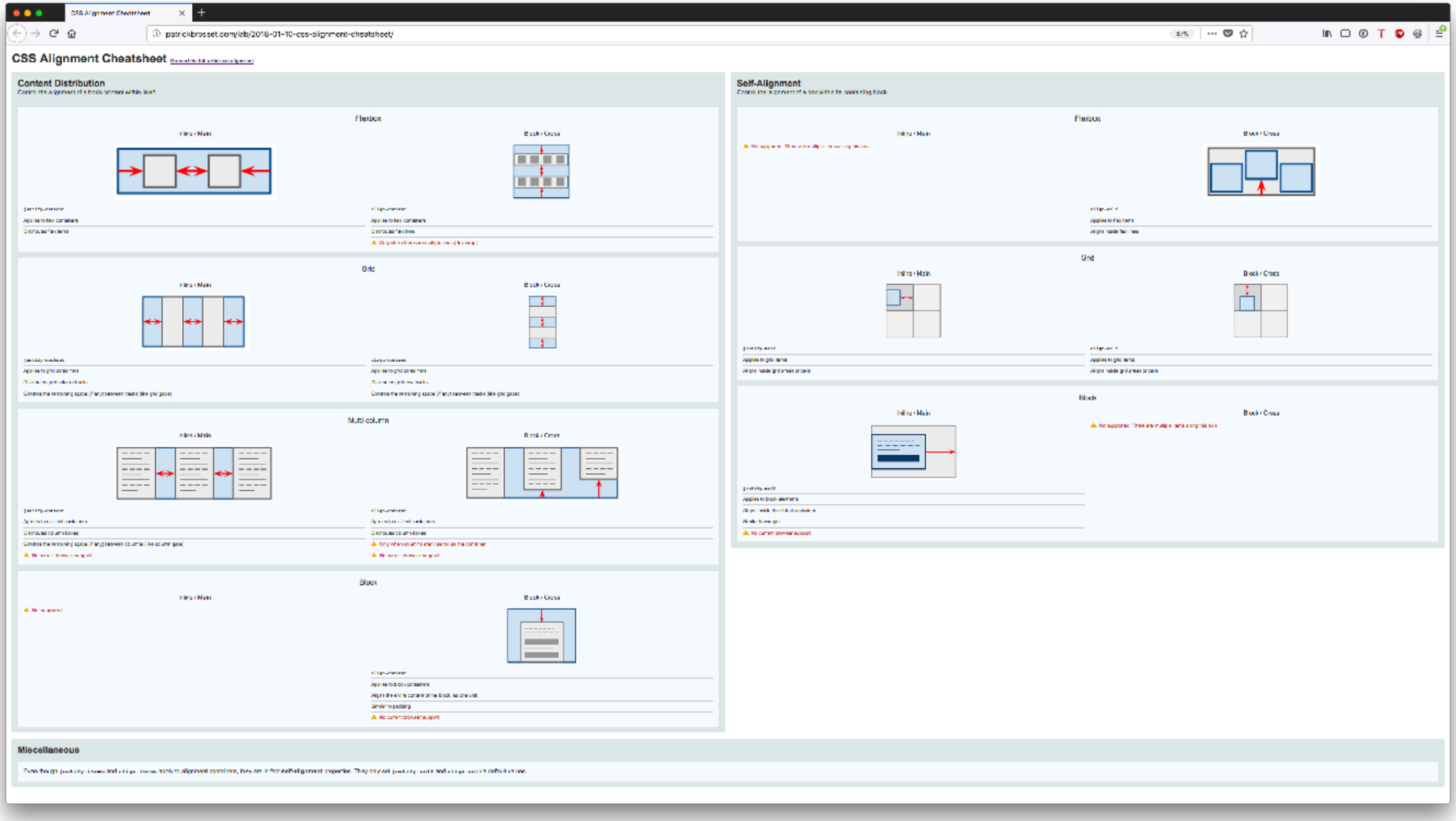
Box Alignment Module

- z-axis aside, we have two axes on which we can work on.
- The inline axis is the one each line of text runs on.
- The other one is the block axis and it’s the one along which blocks are stacked.



DEFAULT AXES

https://medium.com/@patrickbrosset/demystifying-css-alignment-2d3ea7a02a36



ALIGNMENT CHEATSHEET

patrickbrosset.com/lab/2018-01-10-css-alignment-cheatsheet/

BOX ALIGNMENT

justify-items, align-items, justify-self, align-self

justify-content, align-content

Box Alignment

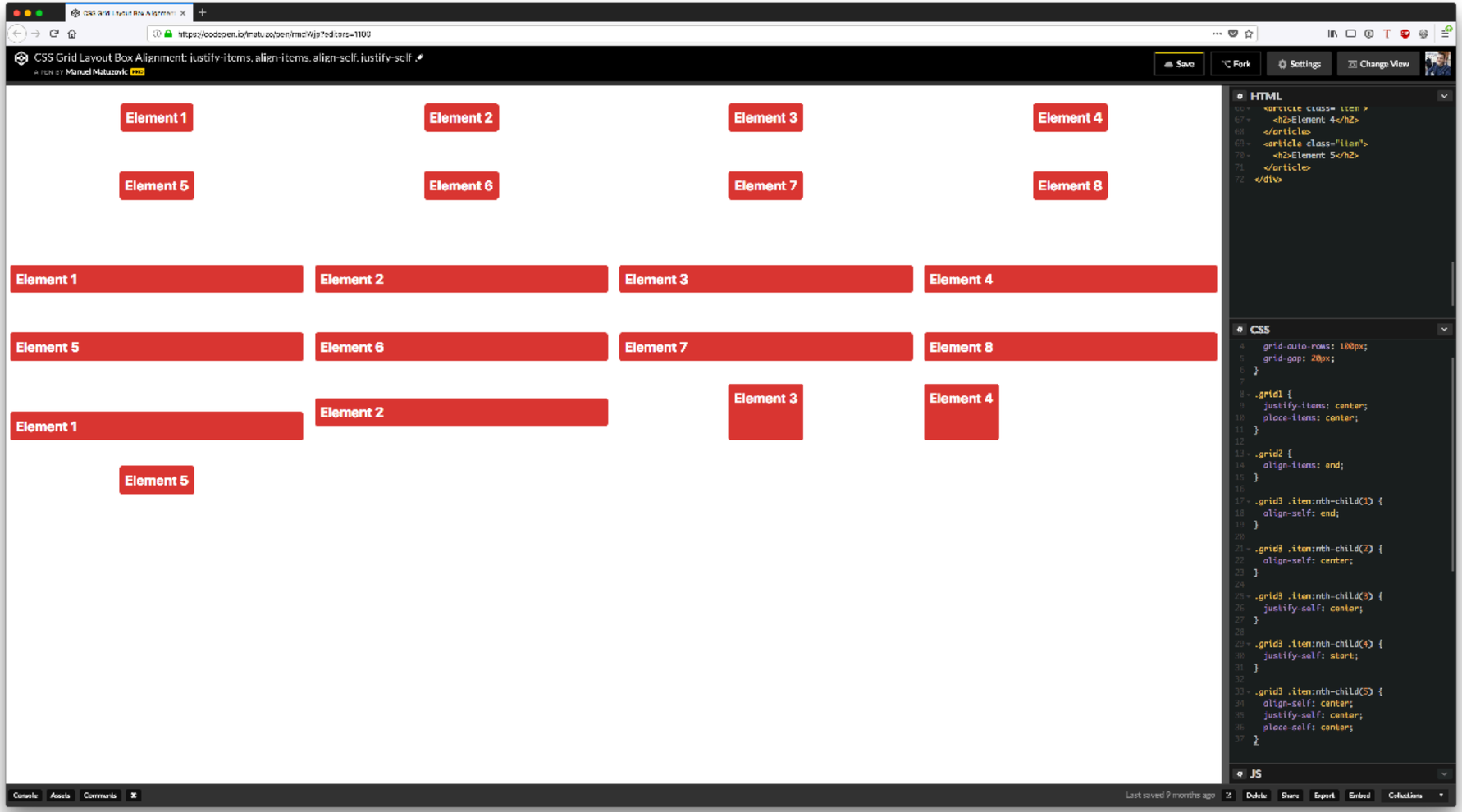
Inline tracks

```
.container {
  justify-items: center;
}
```

Block tracks

```
.container {
  align-items: start;
}
```

Values: start, end, center, stretch (default)



ALIGNING CELLS

bit.ly/grid_align

Box Alignment

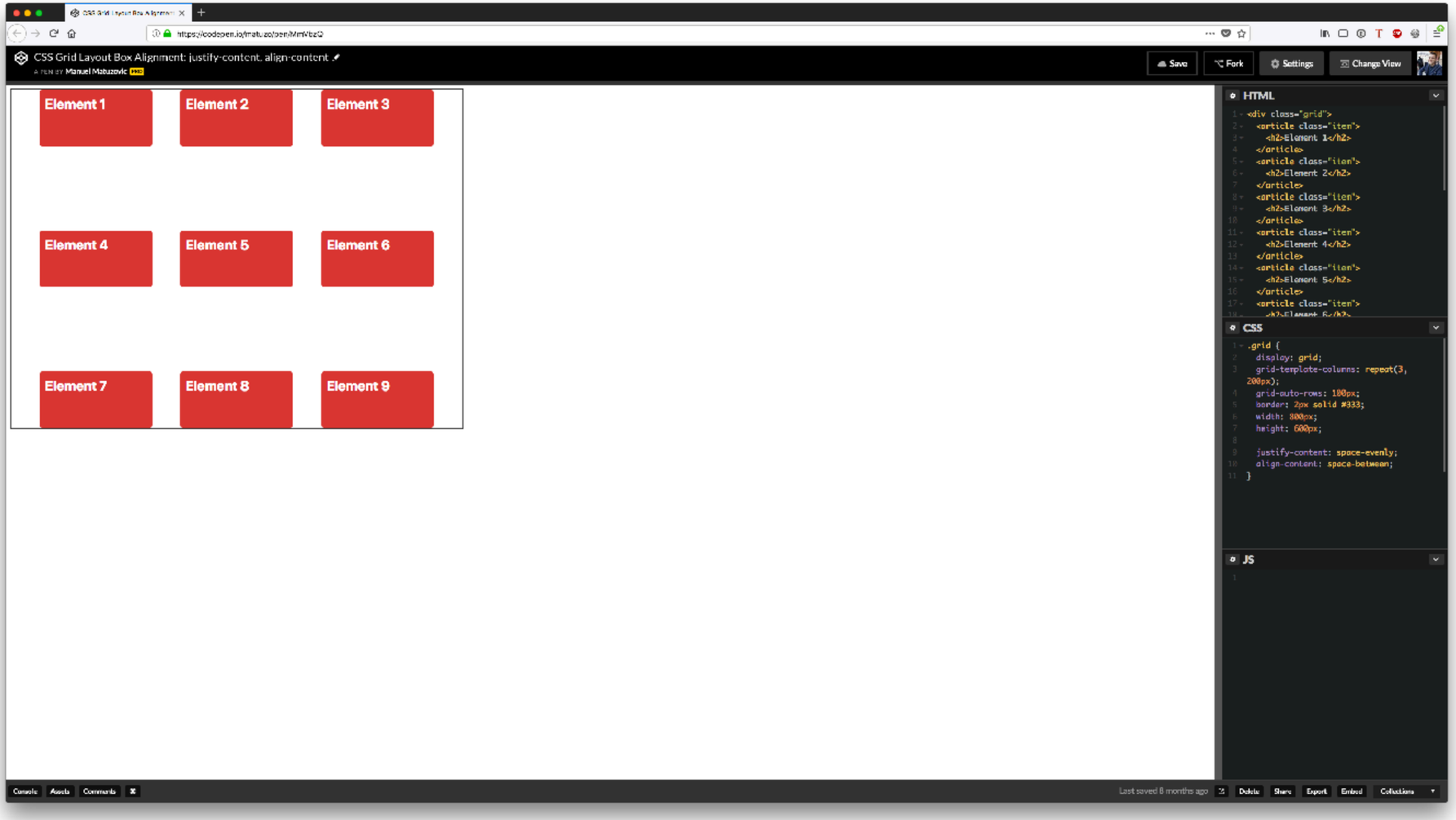
Inline cell alignment

```
.grid-item {
  justify-self: center;
}
```

Block cell alignment

```
.grid-item {
  align-self: start;
}
```

Values: start, end, center, stretch (default)



ALIGNING THE WHOLE GRID

bit.ly/grid_align2

Box Alignment

Inline grid positioning

```
.grid-item {
  justify-content: center;
}
```

Block positioning

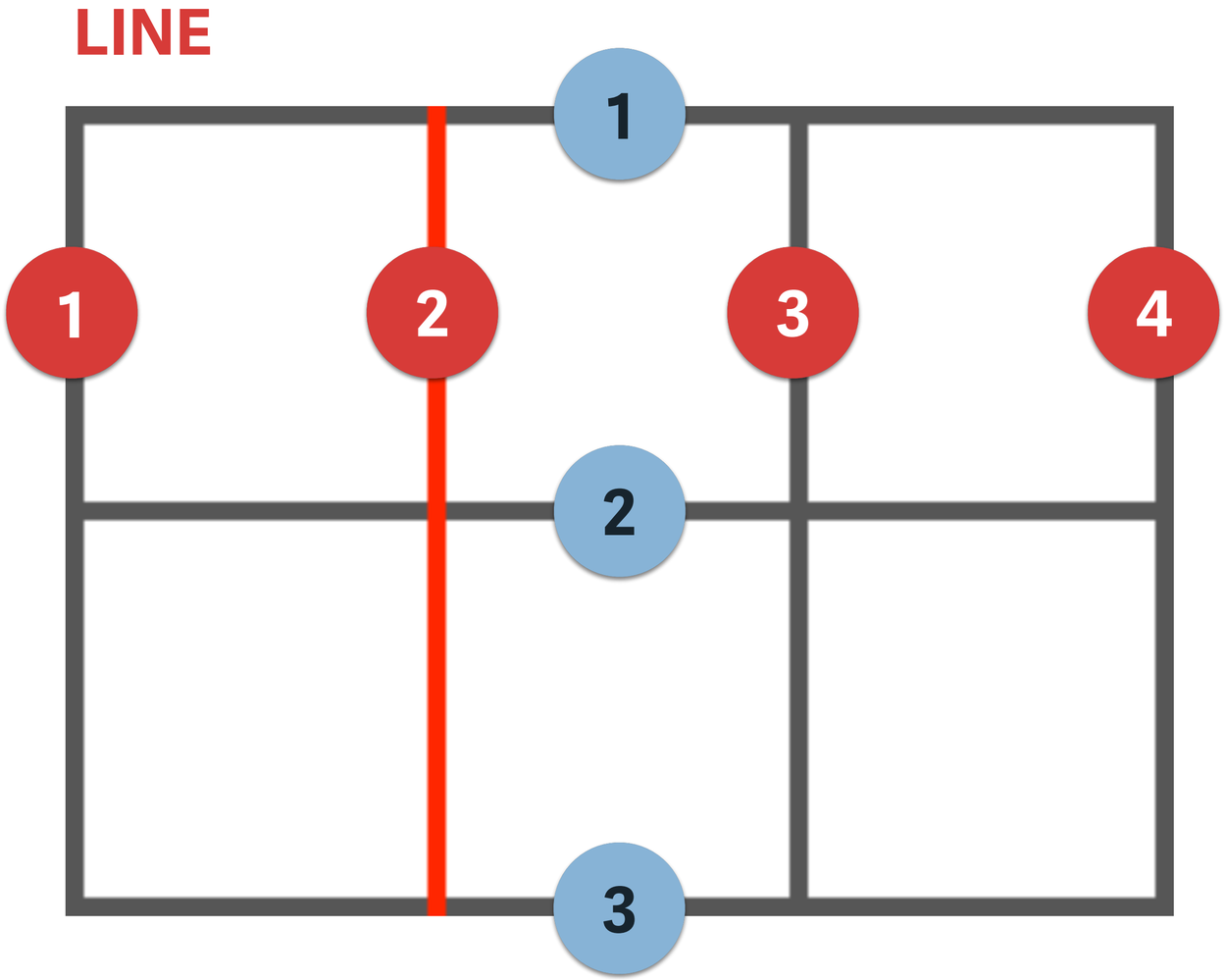
```
.grid-item {
  align-content: end;
}
```

Values: start (default), end, center, space-between, space-around, space-evenly

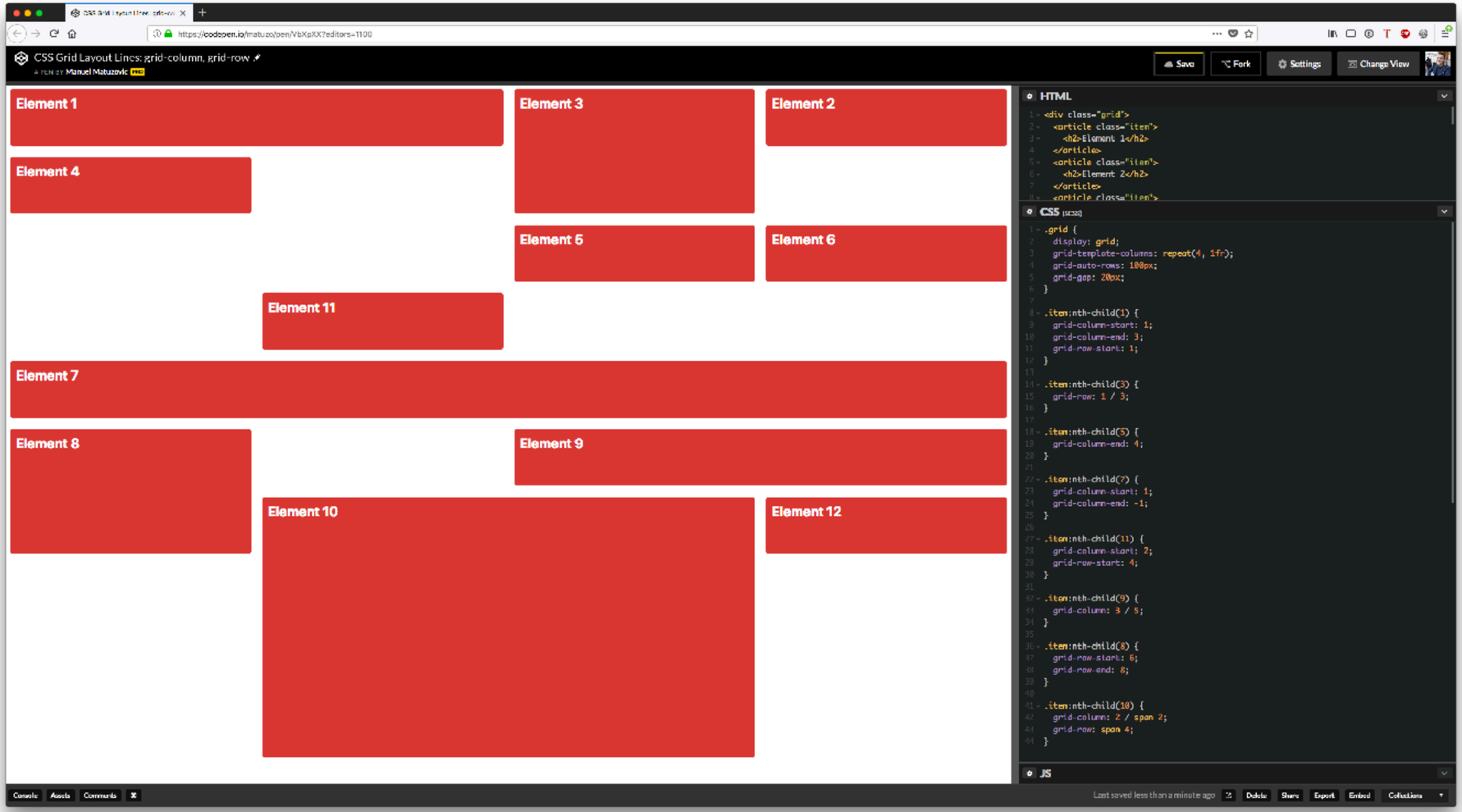
LINES

grid-column, grid-row, grid-column-start,
grid-column-end, grid-row-start, grid-row-end

Terminology



Illustrations: https://gridbyexample.com/what



PLACING CELLS ON LINES

bit.ly/grid_lines

Lines

Cell start and end point.
(vertical track)

Value: Line number

Shorthand:

```
.item {
  grid-column-start: 2;
  grid-column-end: 4;
}
```

```
.item {
  grid-column: 2 / 4;
}
```

Lines

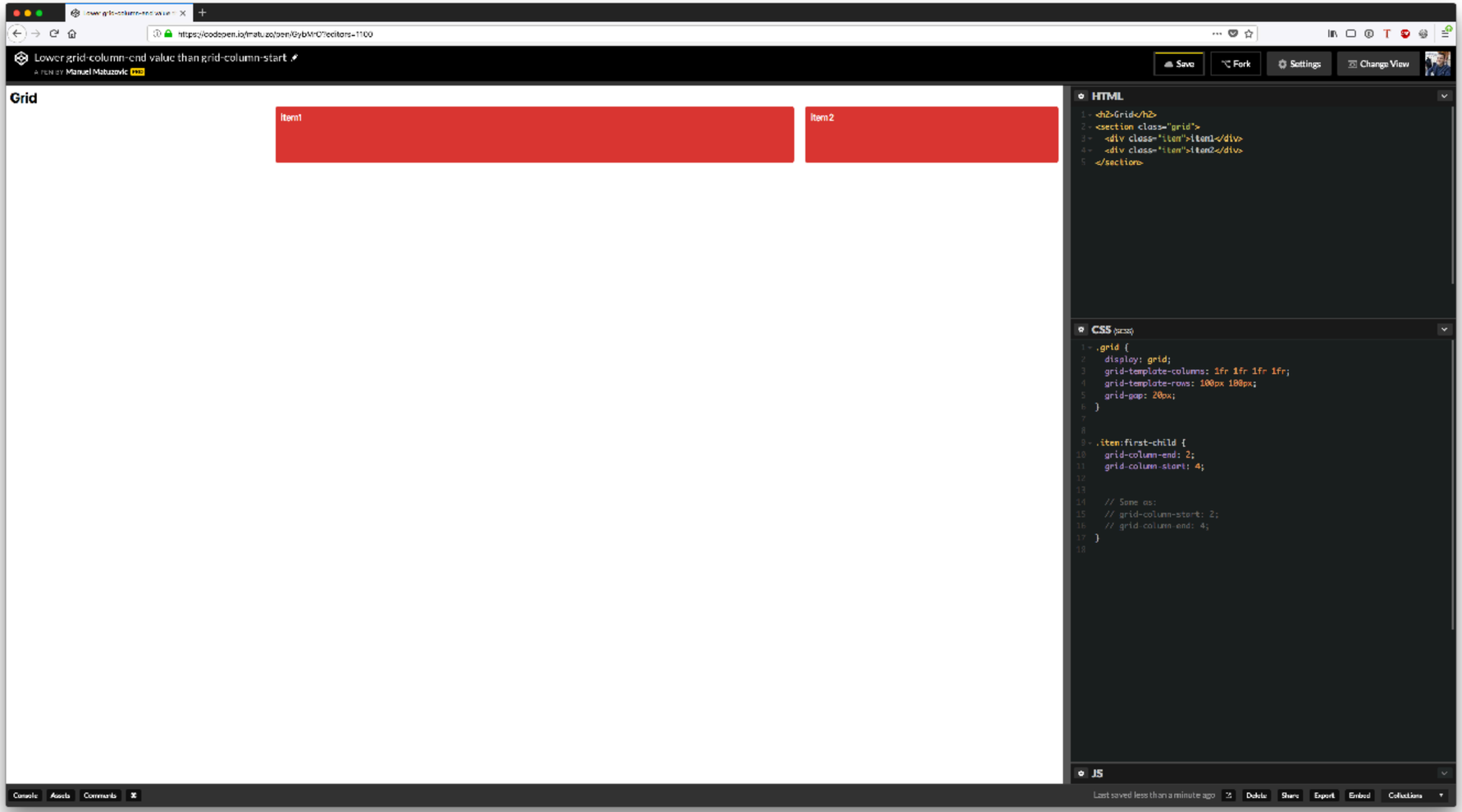
Cell start and end point.
(horizontal track)

Value: Line number

Shorthand:

```
.item {
  grid-row-start: 1;
  grid-row-end: 3;
}
```

```
.item {
  grid-row: 1 / 3;
}
```



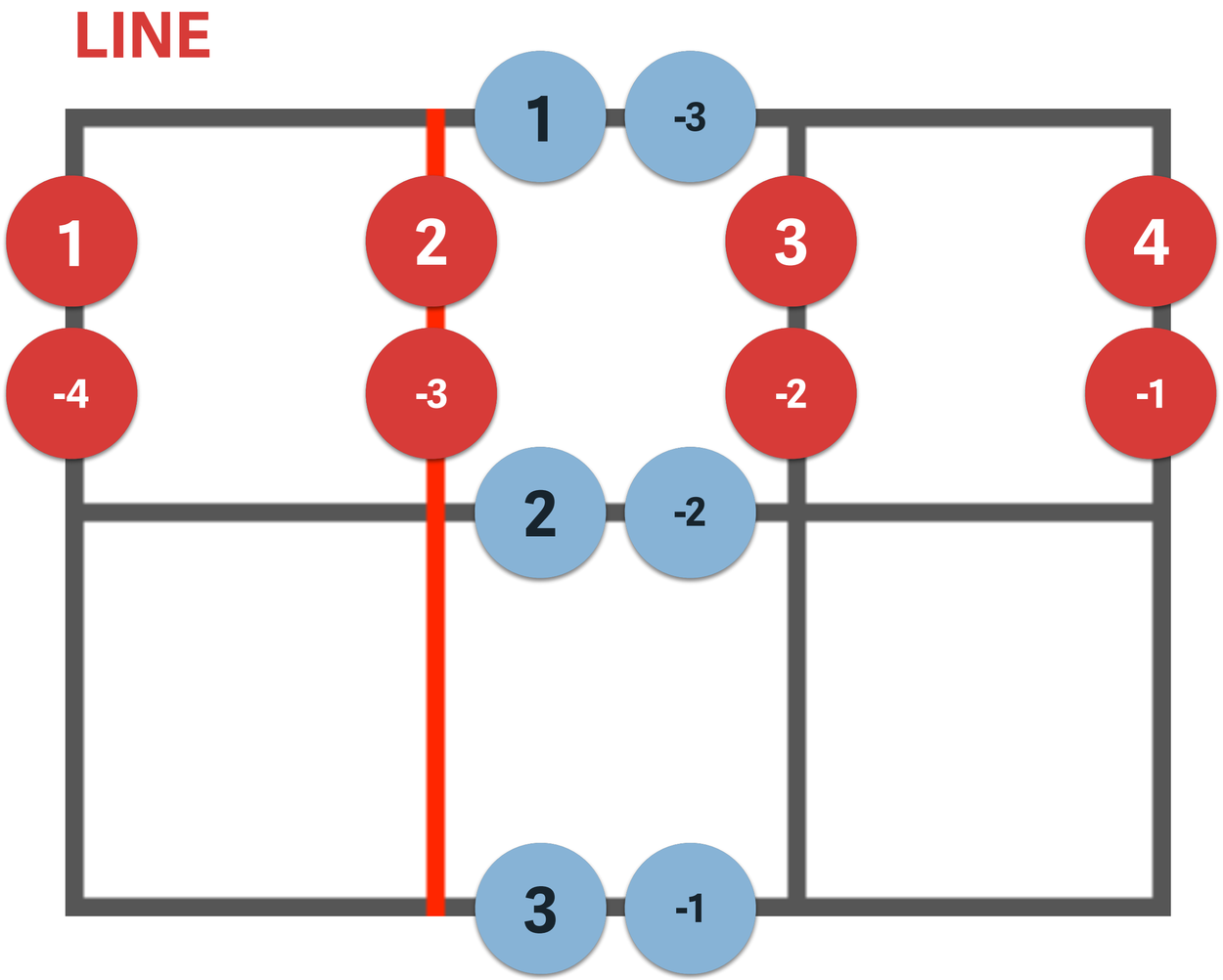
LOWER END VALUE THAN START VALUE

bit.ly/grid_lines7

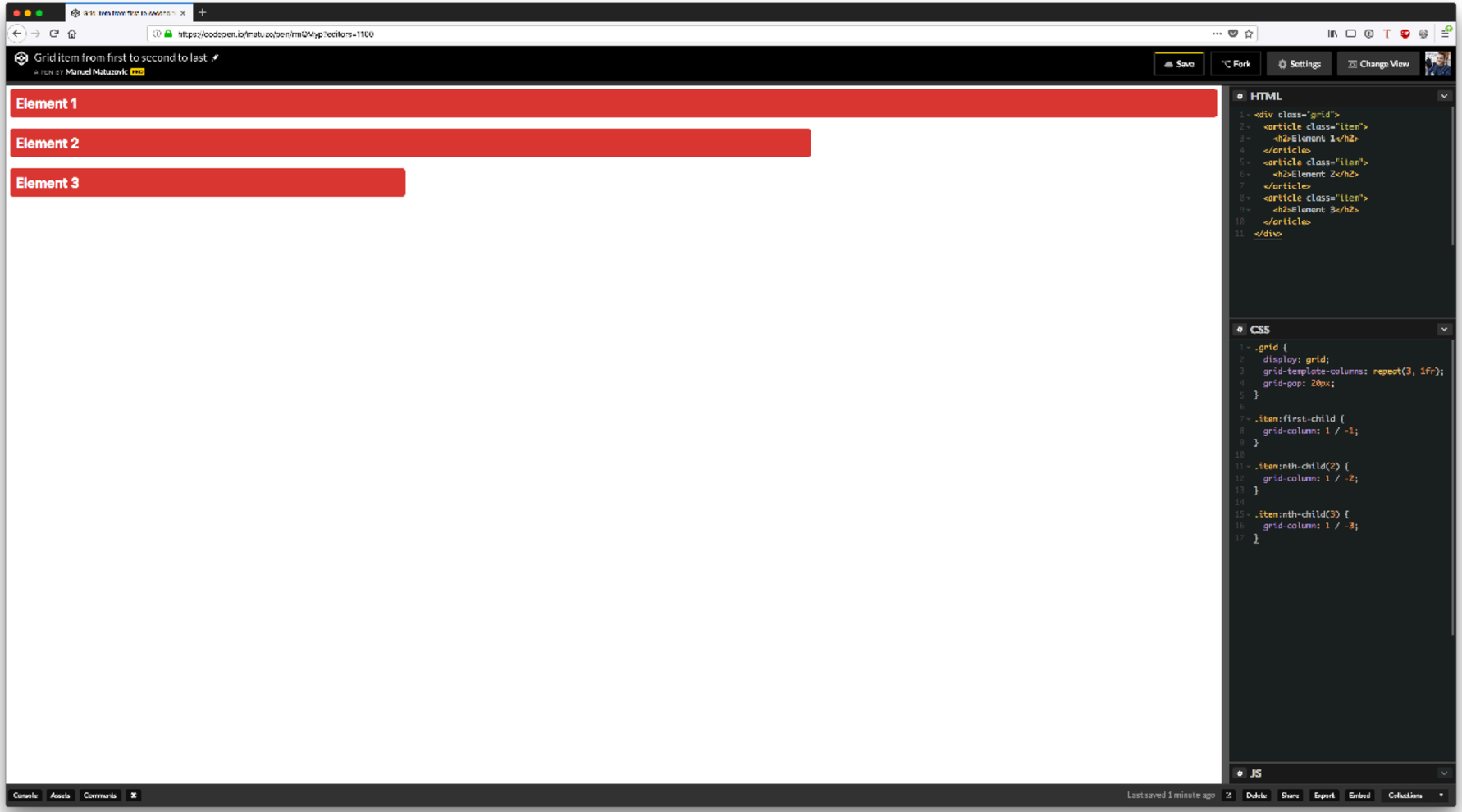
Negative lines values

- Each line is represented by a positive and a negative number.
- For example, `grid-column: 2 / -1;` spans from the second line to the last line.
- It’s possible to use negative values in both `grid-row/column-start` and `grid-row/column-end`.

Terminology

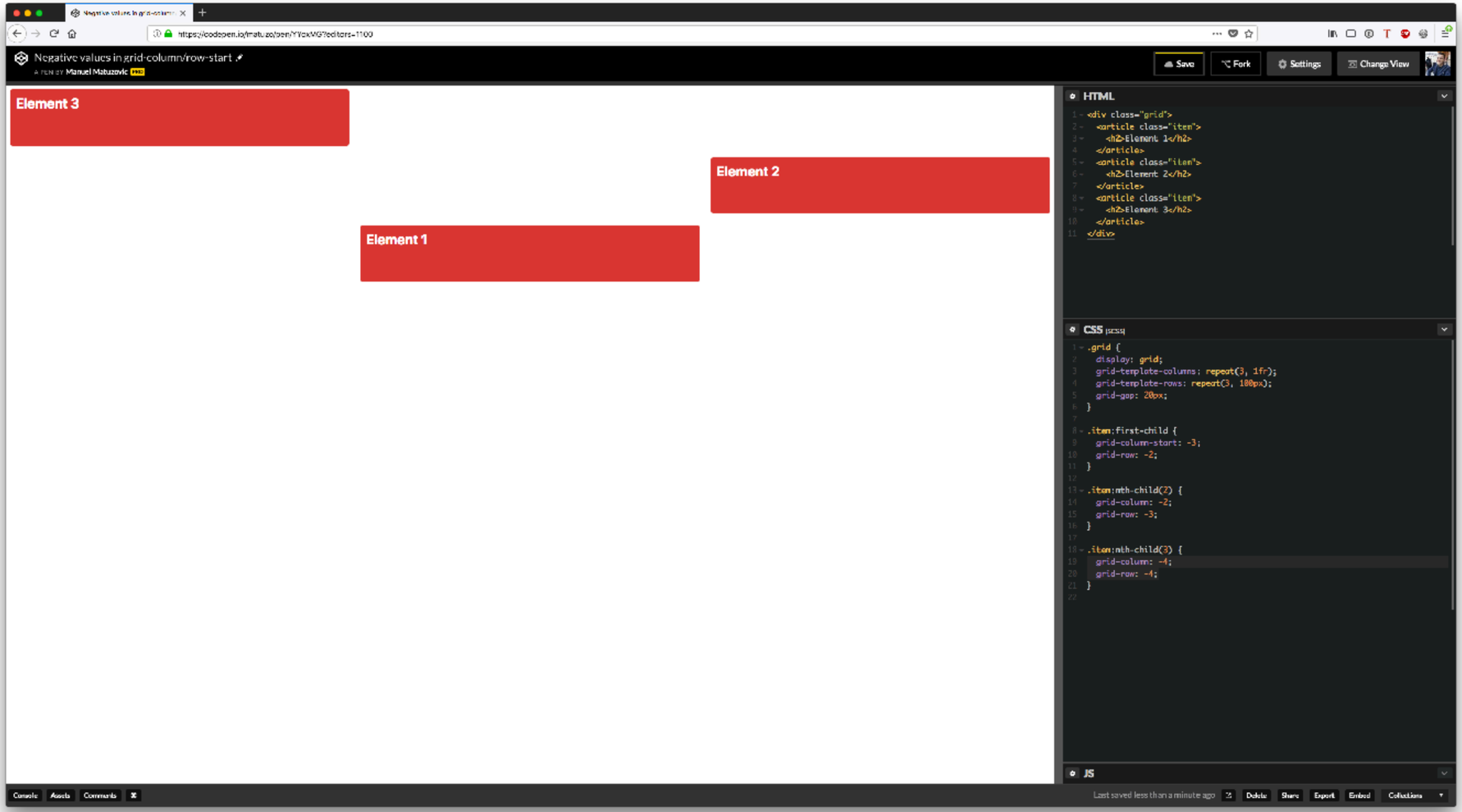


Illustrations: https://gridbyexample.com/what



NEGATIVE LINE VALUES

bit.ly/grid_lines5

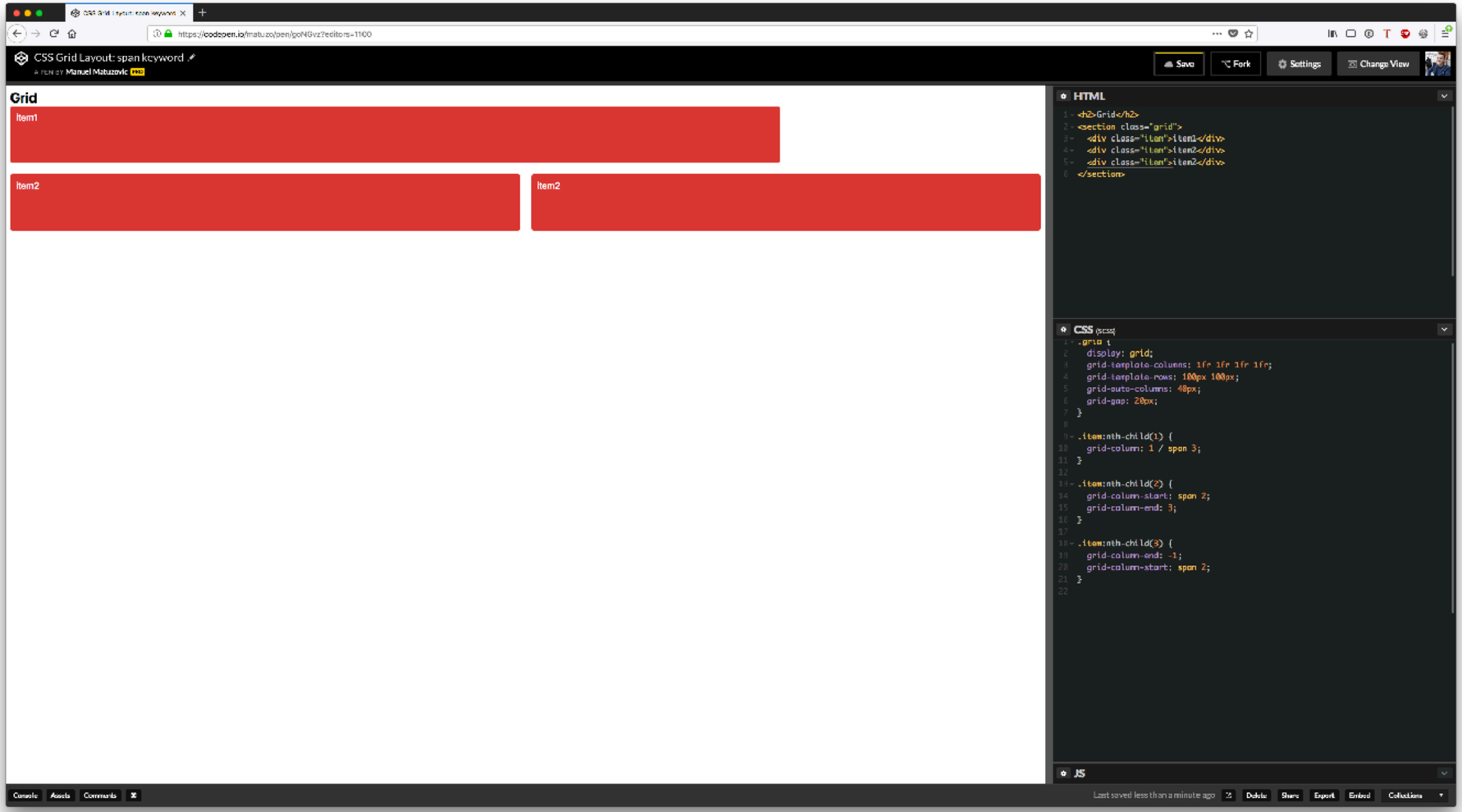


NEGATIVE LINE VALUES

bit.ly/grid_lines6

span Keyword

- A grid item spans a single cell by default.
- That can be changed by applying the span keyword.
- For example, grid-column-start: 1 and grid-column-end: span 2 will make the item span two cells, from the first to the third line.
- The span keyword may also be used on start properties.
- For example, with grid-column-end: -1 and grid-column-start: span 2 the item will be placed at the last line and span 2 cells, from the last to third to last line.



THE SPAN KEYWORD

bit.ly/grid_lines8

Lines

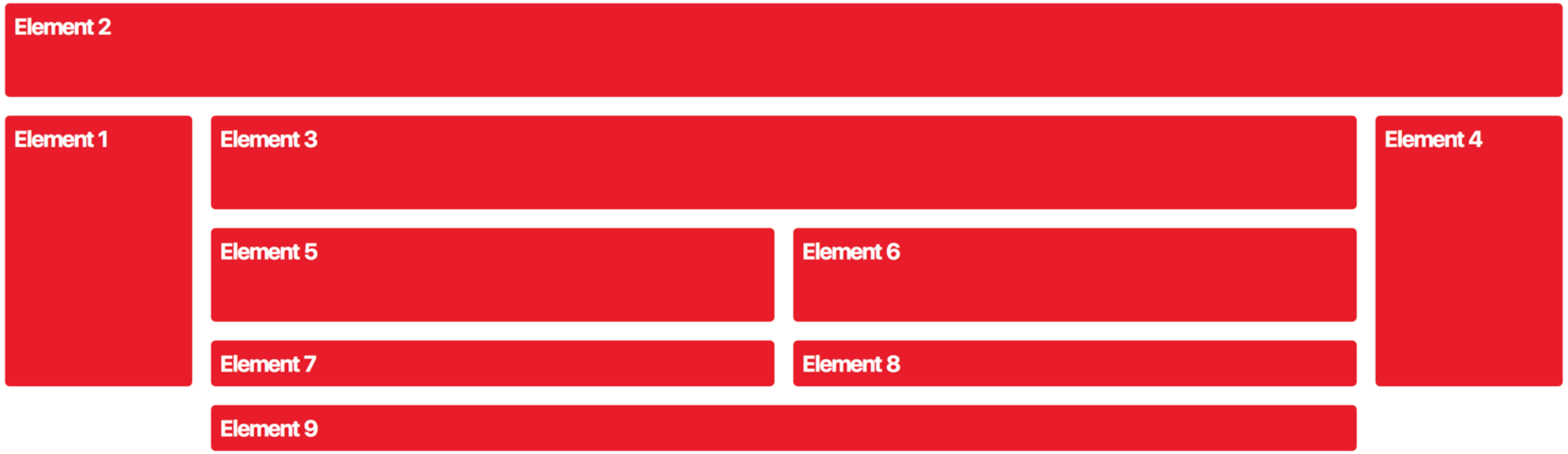
Span keyword

```
.item {
  grid-row: 3 / span 2;
  grid-column: span 2;
}
```

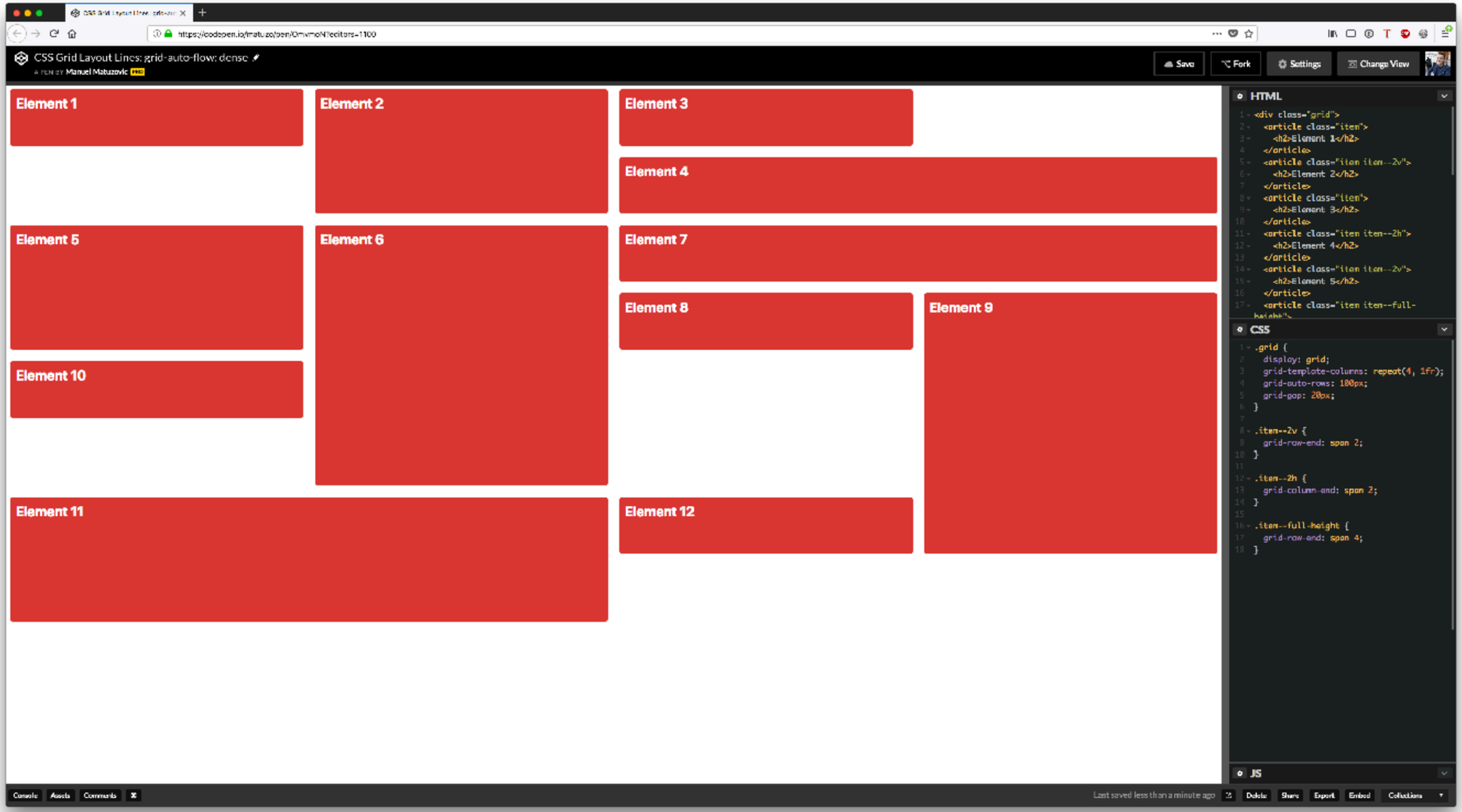
Spanning from line `n` to the last line.

```
.item {
  grid-column: 2 / -1;
}
```

EXERCISE 3

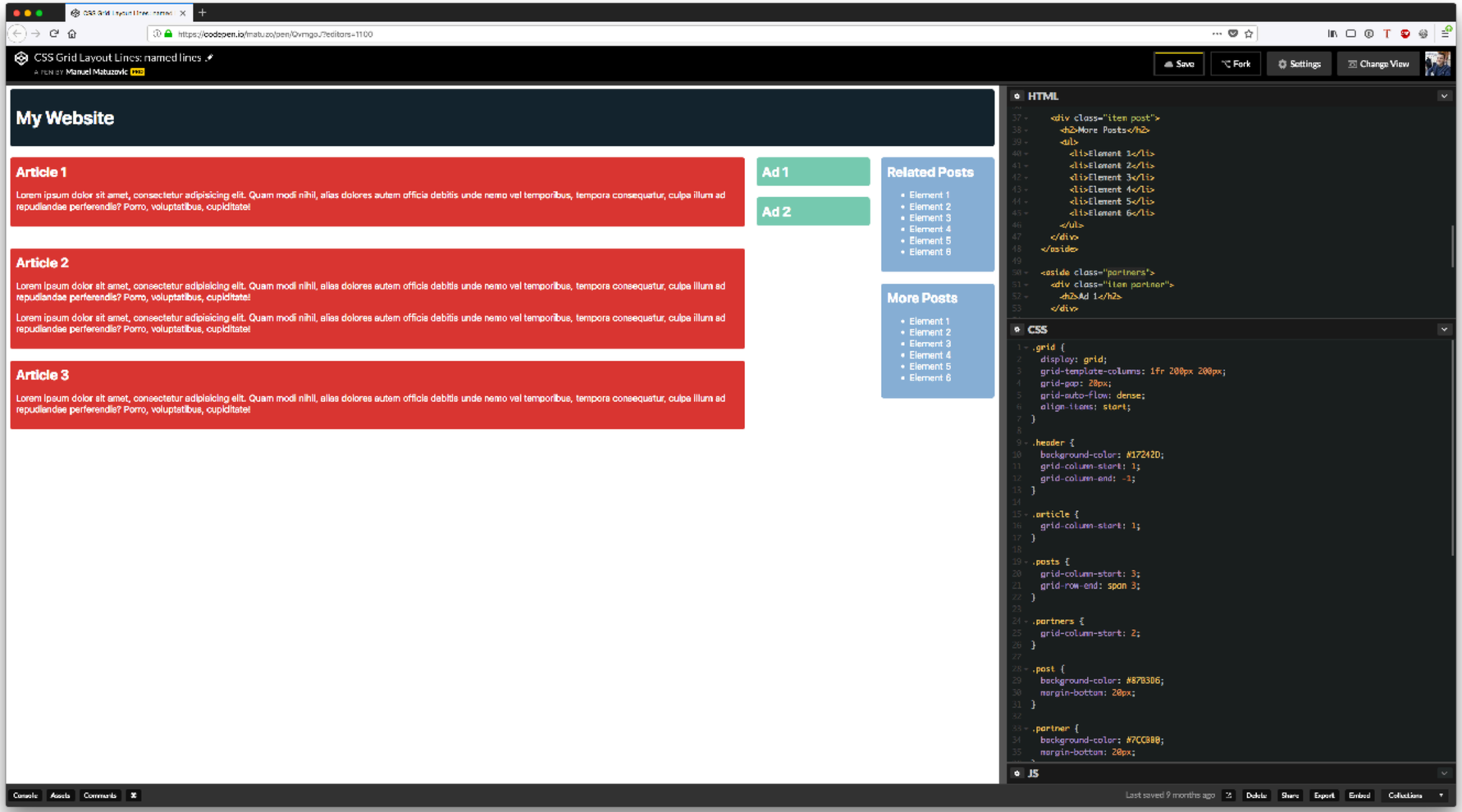


BIT.LY/GRID_EXERCISE3



DENSE PACKING MODE

bit.ly/grid_dense



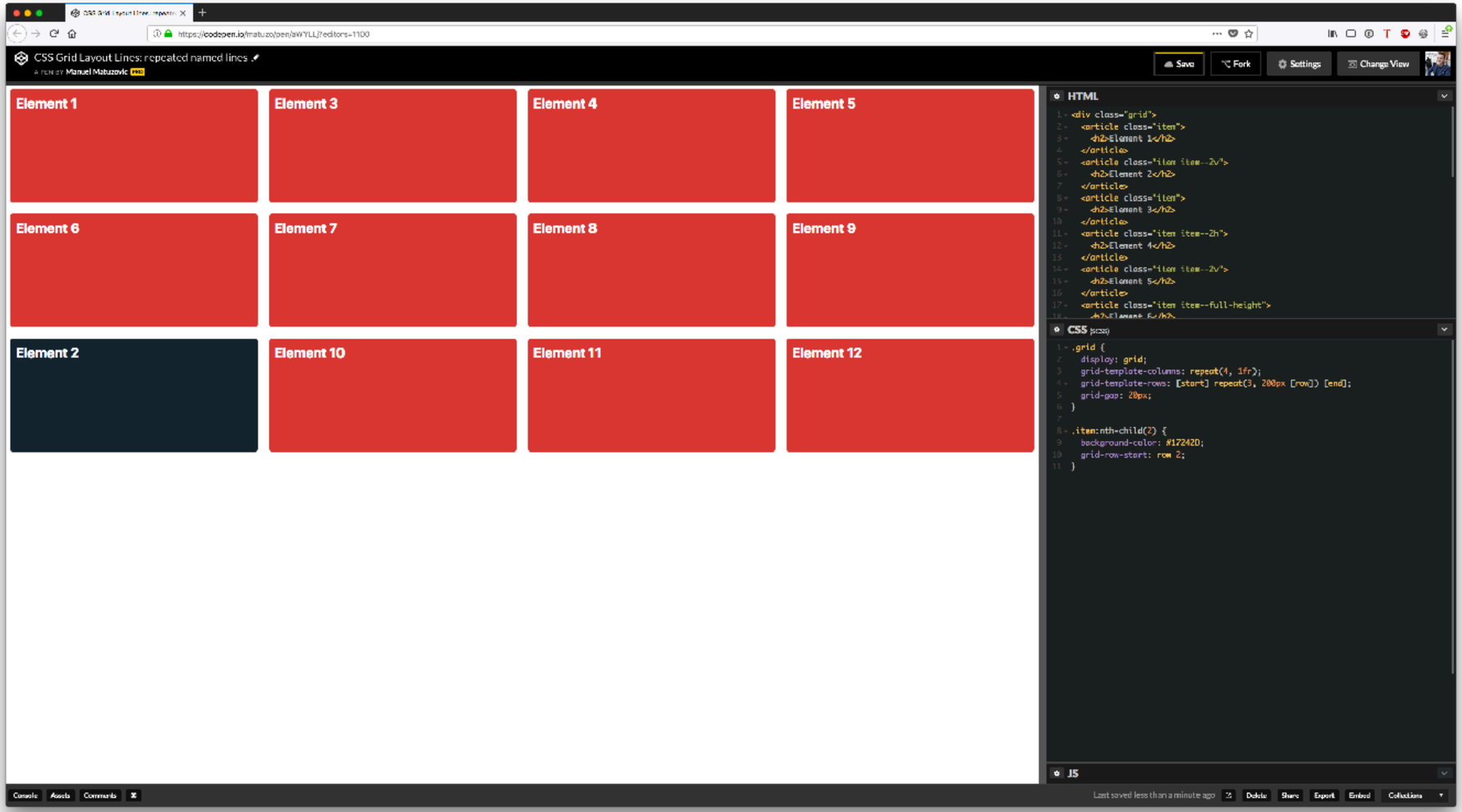
NAMING LINES

bit.ly/grid_lines2

Lines

Naming lines

```
.container {
  grid-template-columns:
    [line1] 20px [line2] 20px [line3] 20px [line4];
}


.grid-item {
  grid-column: line2 / line4;
}
```



REPEATED NAMED LINES

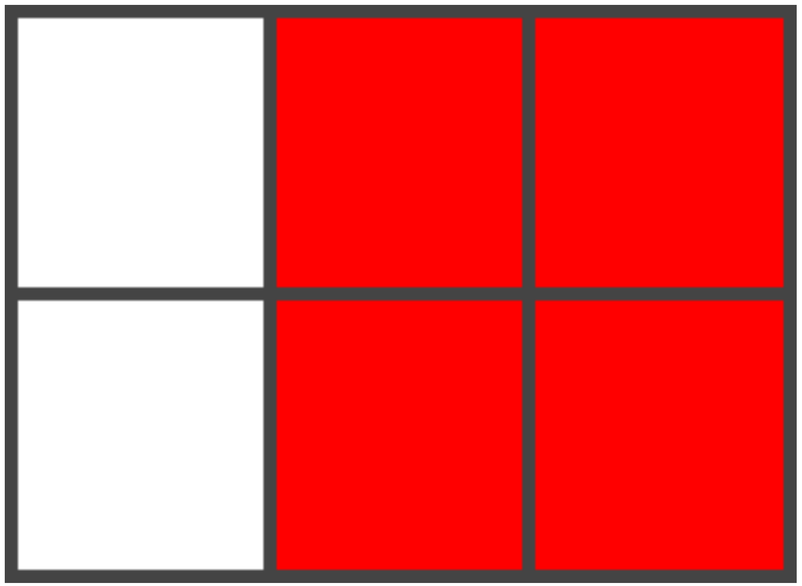
bit.ly/grid_lines3

GRID AREAS

grid-area

Terminology

AREA



One or more cells

Illustrations: https://gridbyexample.com/what

Areas

Defining areas

```
section {
  grid-area: main;
}


aside {
  grid-area: sidebar;
}
```

Areas

Defining layouts with areas

```
.grid {
  grid-template-areas: "main sidebar“;
  grid-template-columns: 1fr 2fr;
}

section {
  grid-area: main;
}


aside {
  grid-area: sidebar;
}
```

Areas

Defining layouts with areas

```
.grid {
  grid-template-areas: "content sidebar"
                       "content sidebar"
                       "content sidebar";
}
```

Areas

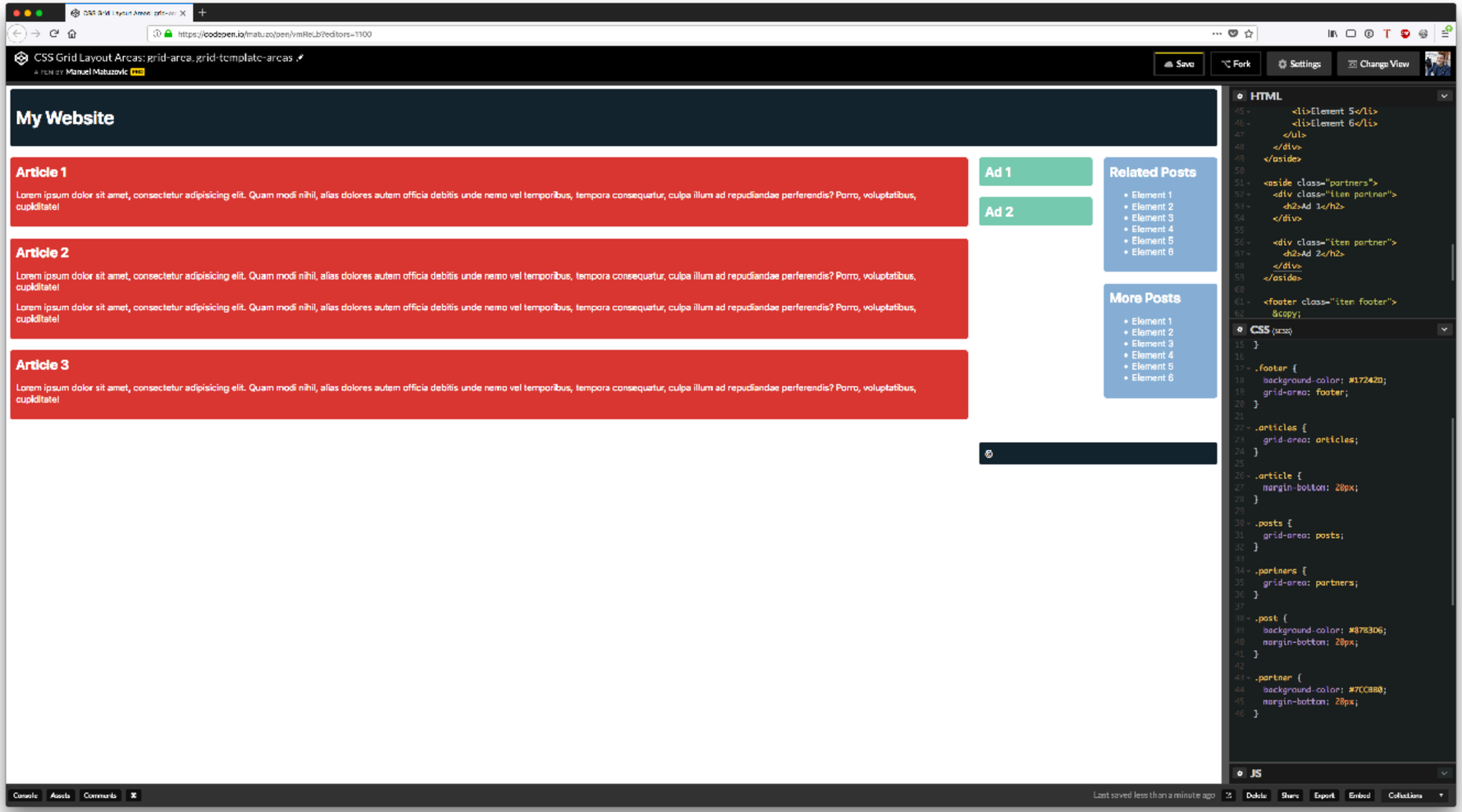
Defining layouts with areas

```
.grid {
  grid-template-areas: "header header“
                       "content sidebar"
                       "footer footer";
}
```

Areas

Skipping areas

```
.grid {
  grid-template-areas: "header header"
                       "content sidebar"
                       ". footer";
}
```



GRID-AREA LAYOUTS

bit.ly/grid_area2

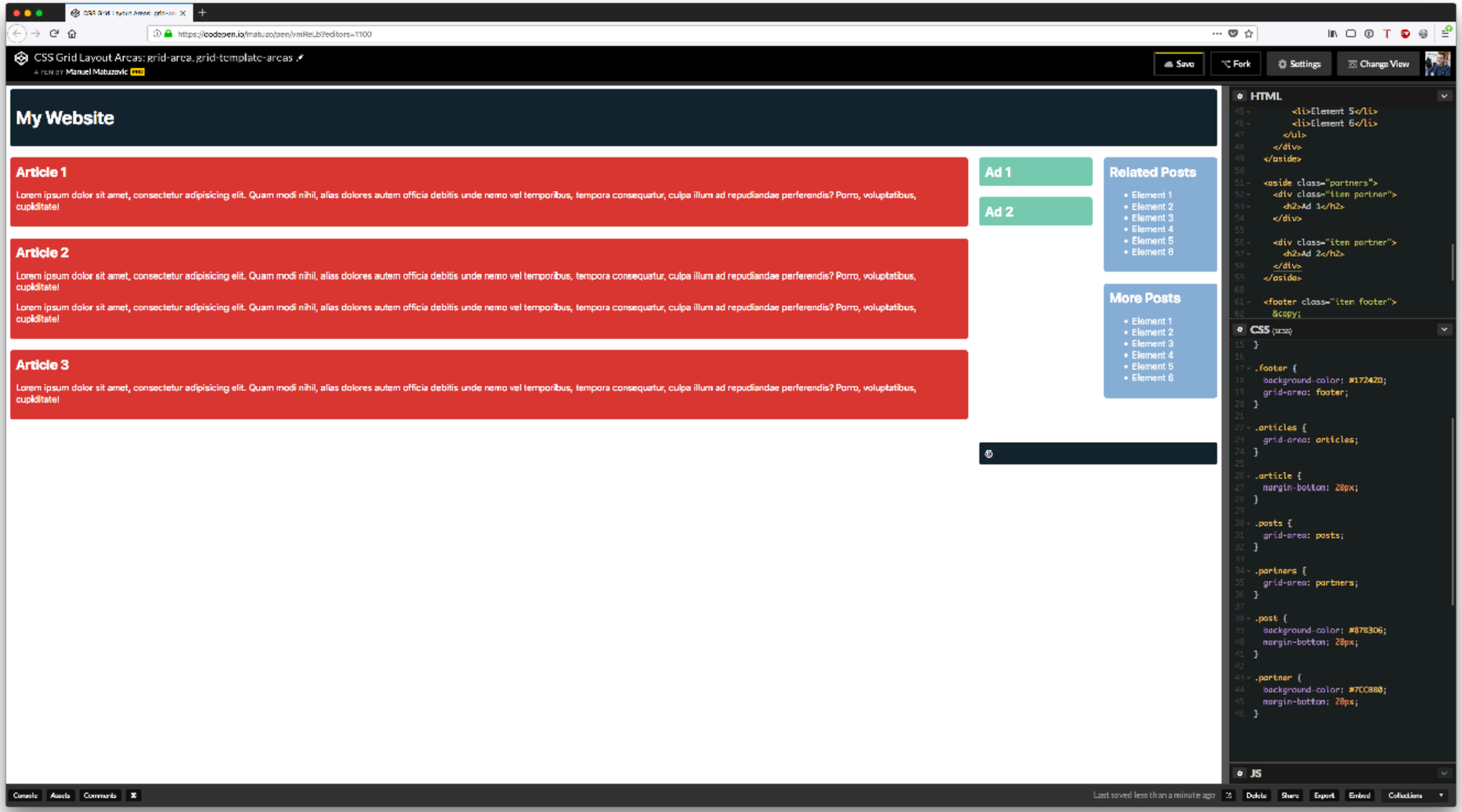
Areas

“Magic lines”.

By defining areas we get automatically named lines for free.

[AREA NAME] + -start or -end suffix.

```
.grid {
  grid-template-areas: "header header"
                       "content sidebar"
                       "footer footer";
}

.grid-item {
  grid-row-start: content-end;
  grid-column-start: sidebar-start;
}
```



GRID-AREA MAGIC LINES

bit.ly/grid_area3

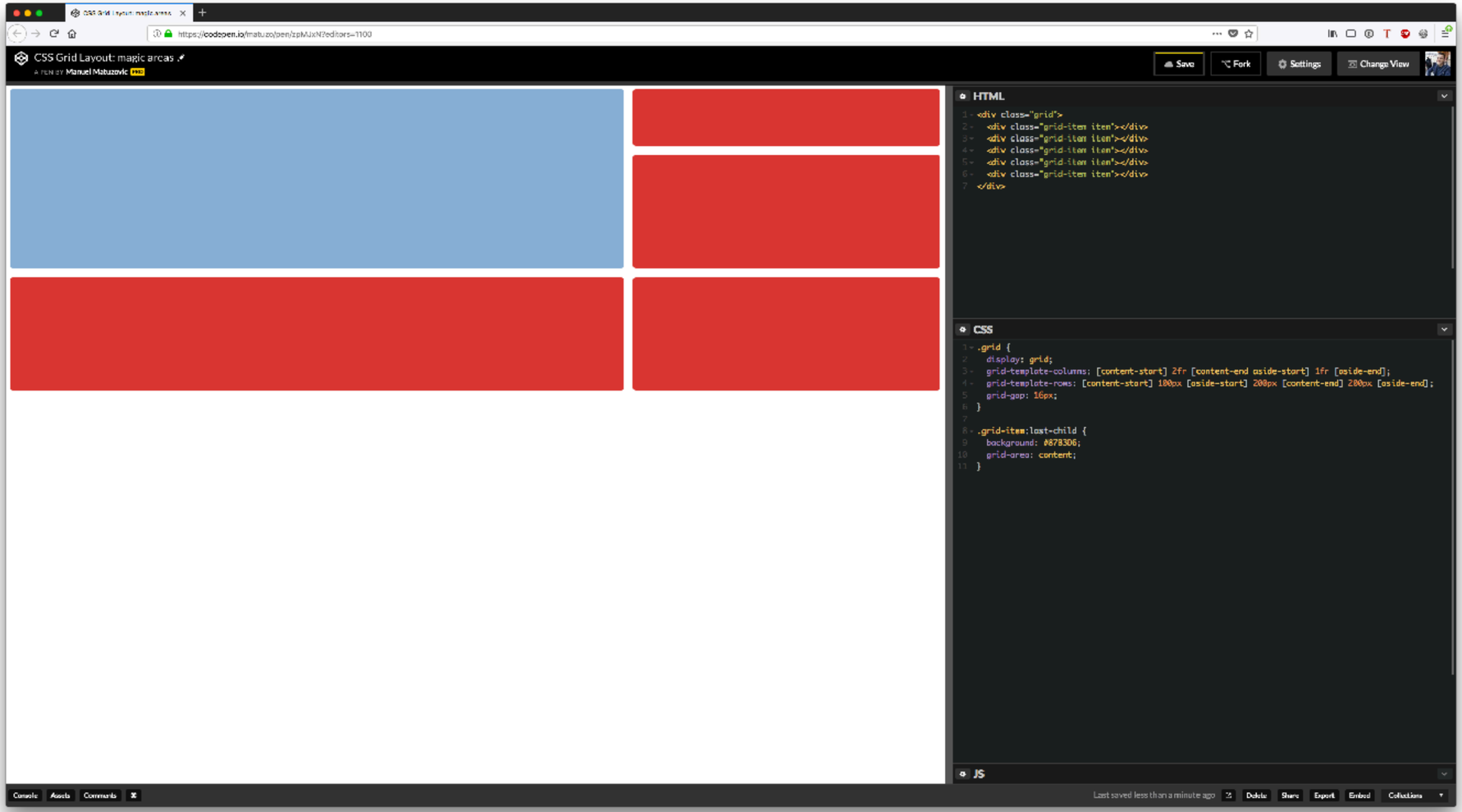
Areas

“Magic areas”.

By defining named lines we get automatically areas for free.

http://bit.ly/grid_area

```
.grid {
  display: grid;
  grid-template-columns: 2fr [aside-start] 1fr [aside-end];
  grid-gap: 16px;
}


.grid-item:last-child {
  background: blue;
  grid-area: aside;
}
```



GRID-AREA MAGIC AREAS

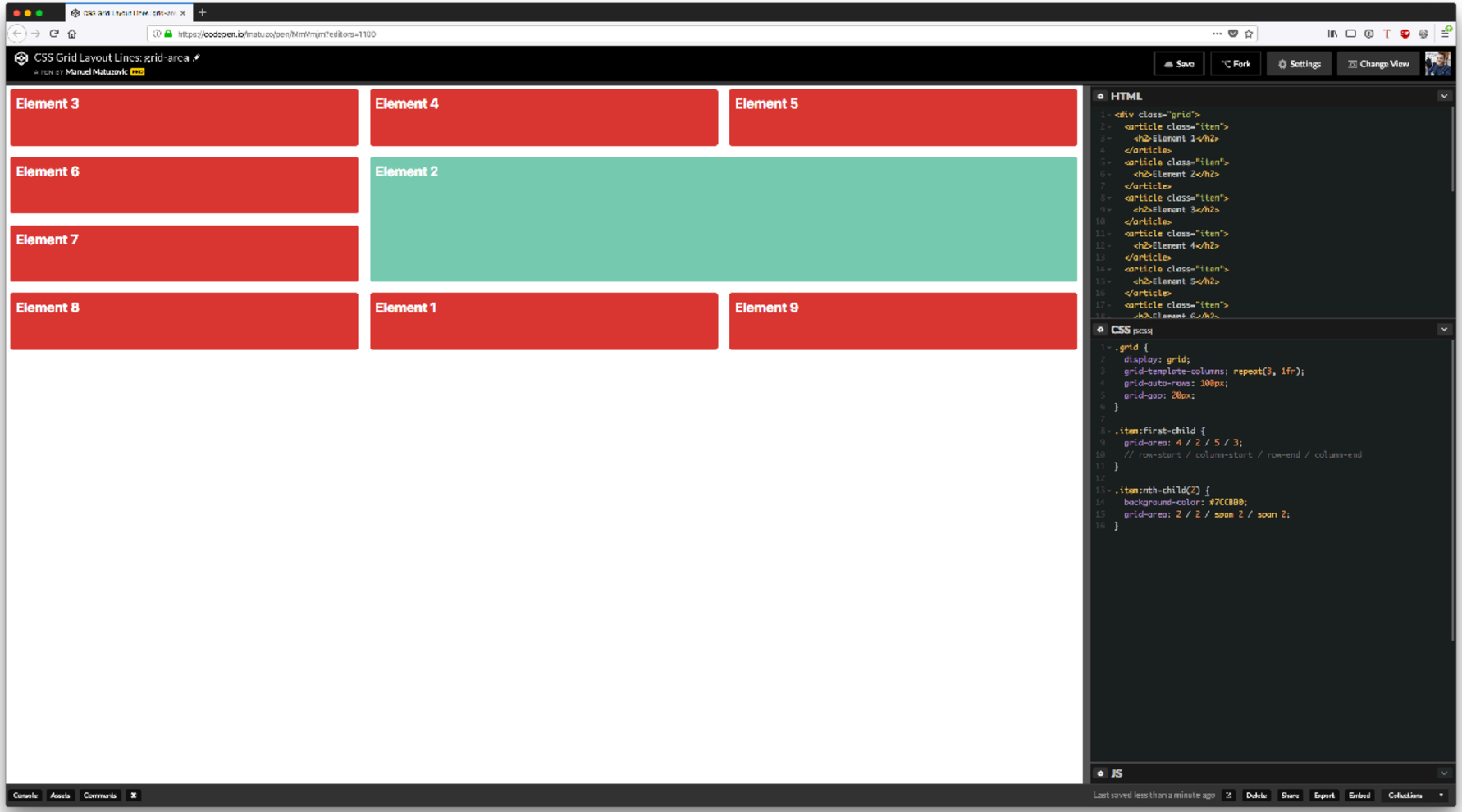
bit.ly/grid_area

Areas

Using grid-area as a shorthand.

grid-row-start
grid-column-start
grid-row-end
grid-column-end

```
.grid-item {
  grid-area: 4 / 2 / 5 / 3;
}


.grid-item {
  grid-area: 2 / 2 / span 2 / span 2;
}
```



GRID-AREA SHORTHAND

bit.ly/grid_area1

SHORTHANDS

grid, grid-template

Shorthands

grid-template shorthand

row track-listing /
column track-listing

```
.grid {
  grid-template-columns: 1fr 600px 1fr;
  grid-template-rows: 200px auto 80px;
}
```

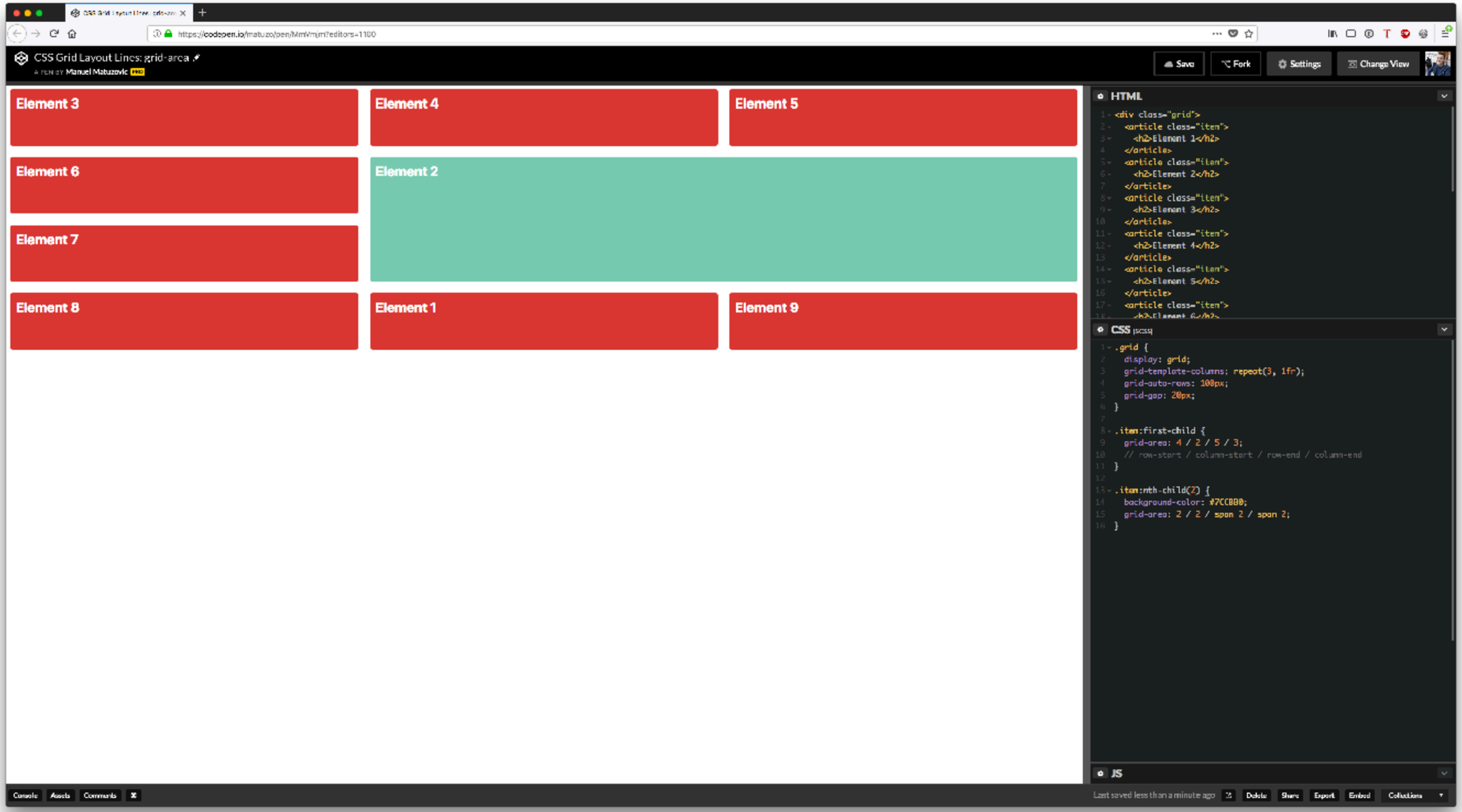
```
.grid {
  grid-template: 200px auto 80px / 1fr 600px 1fr;
}
```

Shorthands

grid-template shorthand with areas

Areas + row tracks / column track-listing

```
.grid {
  grid-template-columns: 1fr 600px 1fr;
  grid-template-rows: 200px auto 80px;
  grid-template-areas: "header header header"
                       ". content sidebar"
                       "footer footer footer";
}

.grid {
  grid-template: "header header header" 200px
                 ". content sidebar"
                 "footer footer footer" 80px /
                 1fr 600px 1fr;
}
```



GRID-TEMPLATE SHORTHAND

bit.ly/grid_short4

Shorthands

grid shorthand

grid-template-areas
grid-template-rows /
grid-template-columns,
grid-auto-flow,
grid-auto-rows /
grid-auto-columns

```
.grid {
  grid-template-rows: 200px auto;
  grid-template-columns: 1fr auto 1fr;
}
```

```
.grid {
  grid: 200px auto / 1fr auto 1fr;
}
```

Shorthands

grid shorthand

grid-template-areas
grid-template-rows /
grid-template-columns,
grid-auto-flow,
grid-auto-rows /
grid-auto-columns

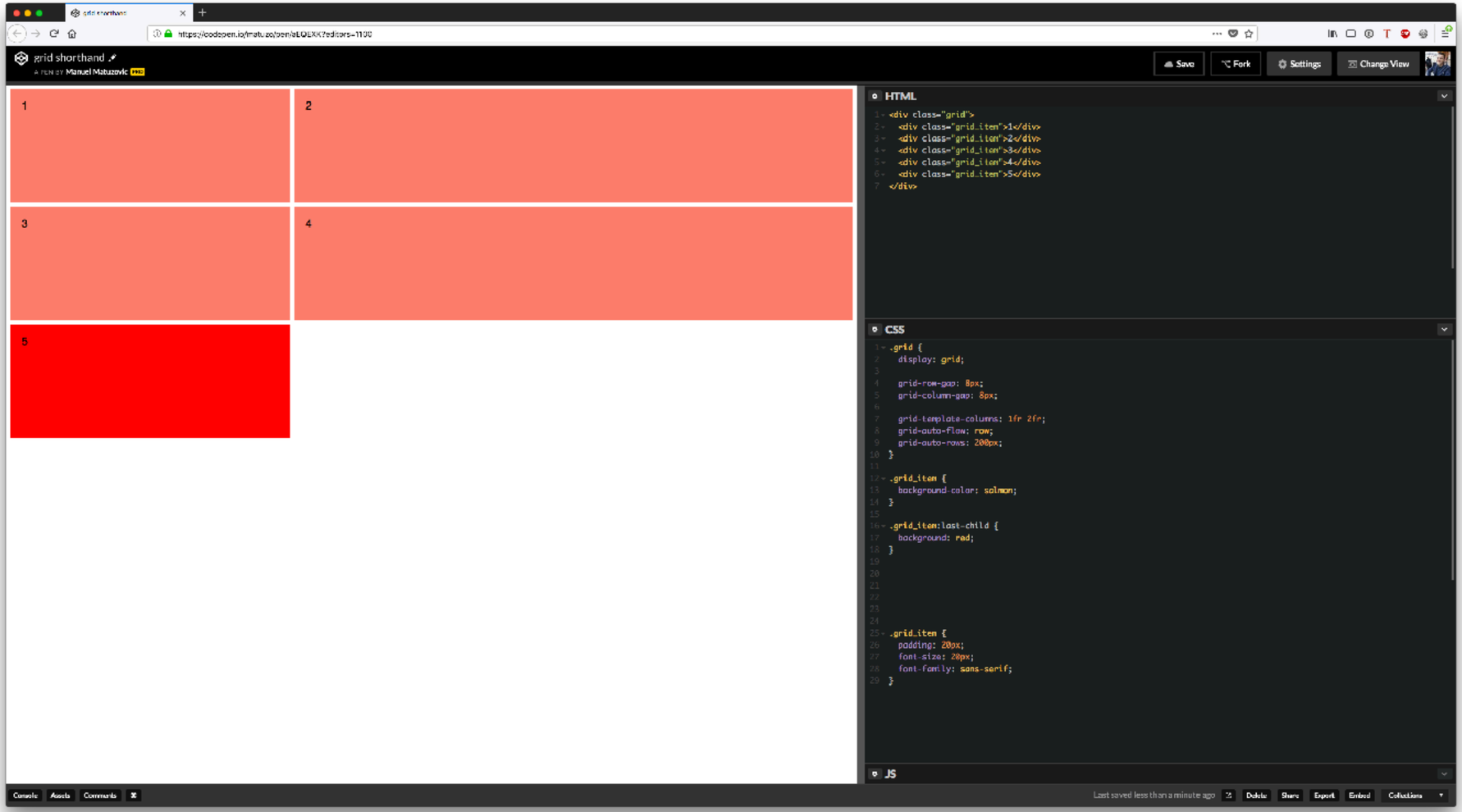
```
.grid {
  display: grid;
  grid-template-rows: 200px 100px;
  grid-auto-flow: column;
  grid-auto-columns: 150px;
}

.grid {
  grid: 200px 100px / auto-flow 150px;
}
```

Shorthands

grid shorthand

grid-template-areas
grid-template-rows /
grid-template-columns,
grid-auto-flow,
grid-auto-rows /
grid-auto-columns

```
.grid {
  display: grid;
  grid-template-columns: 1fr 2fr;
  grid-auto-flow: row;
  grid-auto-rows: 200px;
}

.grid {
  grid: auto-flow 200px / 1fr 2fr;
}
```



GRID SHORTHAND

bit.ly/grid_short

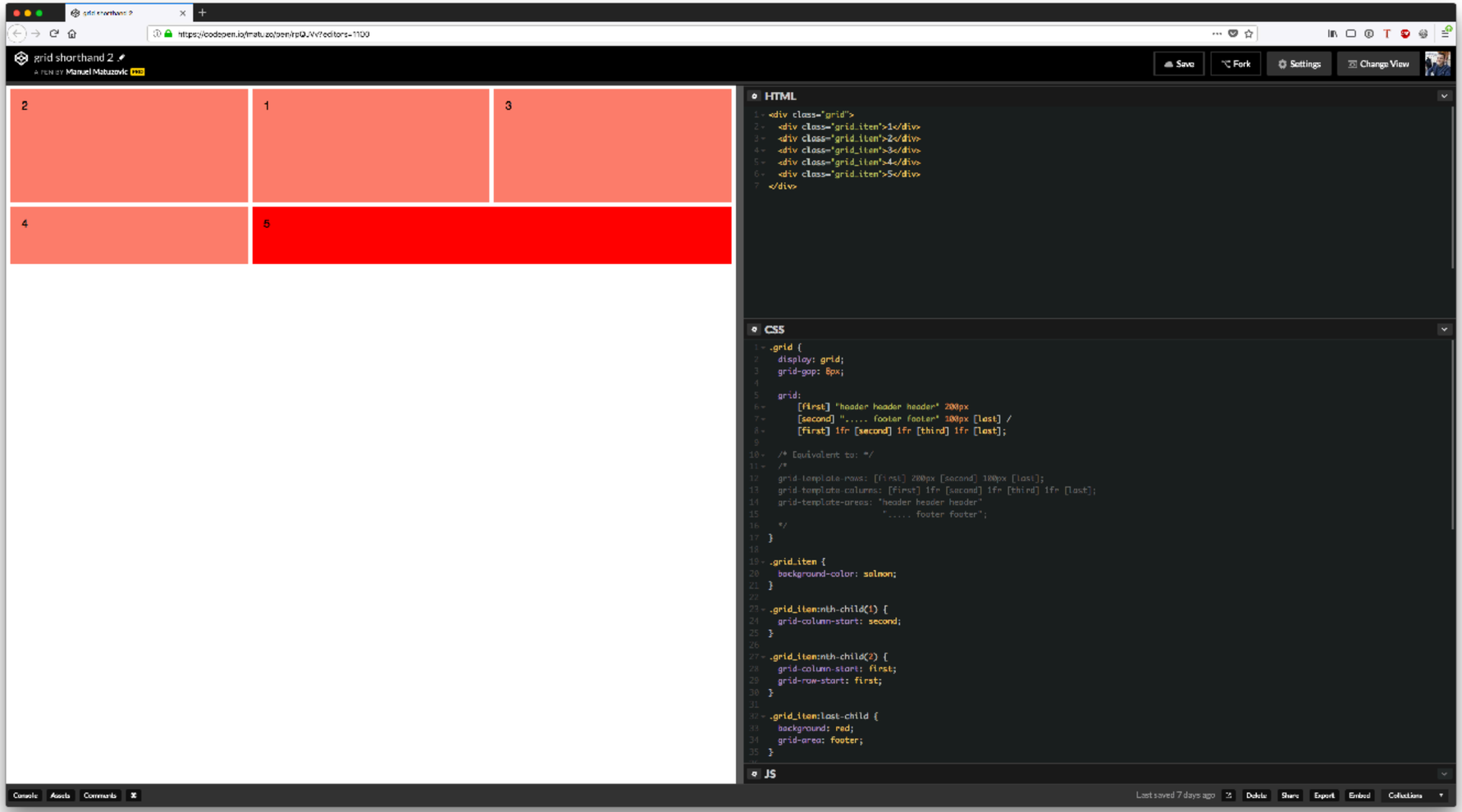
Shorthands

grid shorthand with named lines

grid-template-areas
grid-template-rows /
grid-template-columns,
grid-auto-flow,
grid-auto-rows /
grid-auto-columns

https://bit.ly/grid_short2

```
.grid {
  grid-template-rows: [first] 200px [second] 100px [last];
  grid-template-columns: [first] 1fr [second] 1fr [third] 1fr [last];
  grid-template-areas: "header header header"
                       ".... footer footer";
}

.grid {
  grid:
    [first] "header header header" 200px
    [second] ".... footer footer" 100px [last] /
    [first] 1fr [second] 1fr [third] 1fr [last];
}
```



GRID SHORTHAND WITH NAMED LINES

bit.ly/grid_short2

Shorthands

grid shorthand

Imitating size in Post-CSS.

http://bit.ly/grid_short3

```
.grid {
  display: grid;
  grid: 400px / 500px;
}
```

GRID SHORTHAND

bit.ly/grid_short3

Shorthands

grid shorthand

Resetting grid properties.

```
.grid {
  display: grid;
  grid-template-rows: 200px 100px;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-areas: "header header header"
                       "..... footer footer";
  grid-auto-flow: column;
  grid: none;
}
```

CSS INTRINSIC AND EXTRINSIC SIZING

min-content, max-content, fit-content

CSS Intrinsic And Extrinsic Sizing

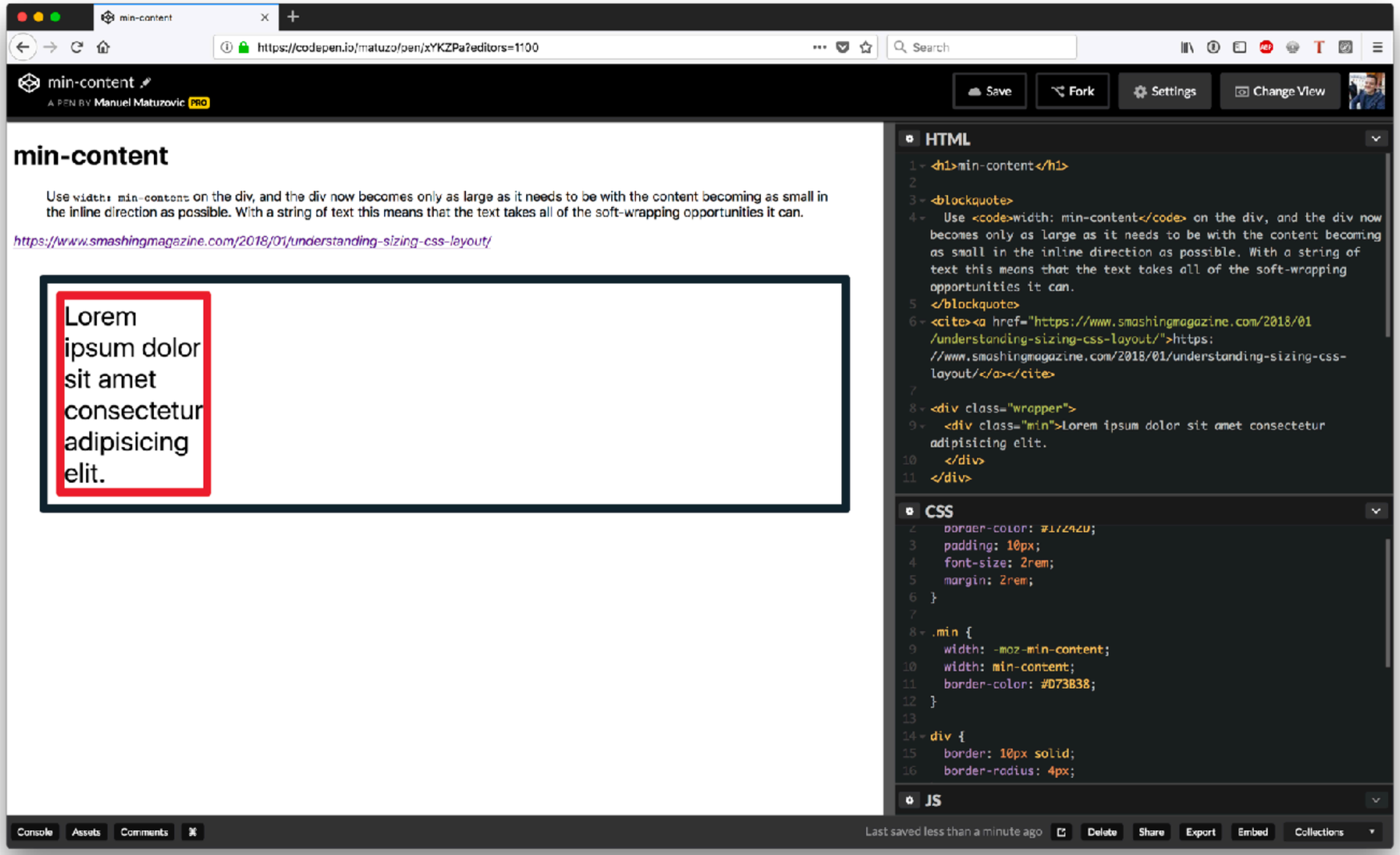
- The module defines additional ways to control size.
- Keywords that represent content-based "intrinsic" sizes and context-based "extrinsic" sizes.
- min-content, max-content, fit-content.
- min-content and max-content can be used for any of the properties that normally take a length (width, height, min-width, flex-basis,...).

Shorthands

min-content

The item becomes as large as it needs to be with the content becoming as small in the inline direction as possible.

```
.item {
  width: -moz-min-content;
  width: min-content;
}
```



MIN-CONTENT

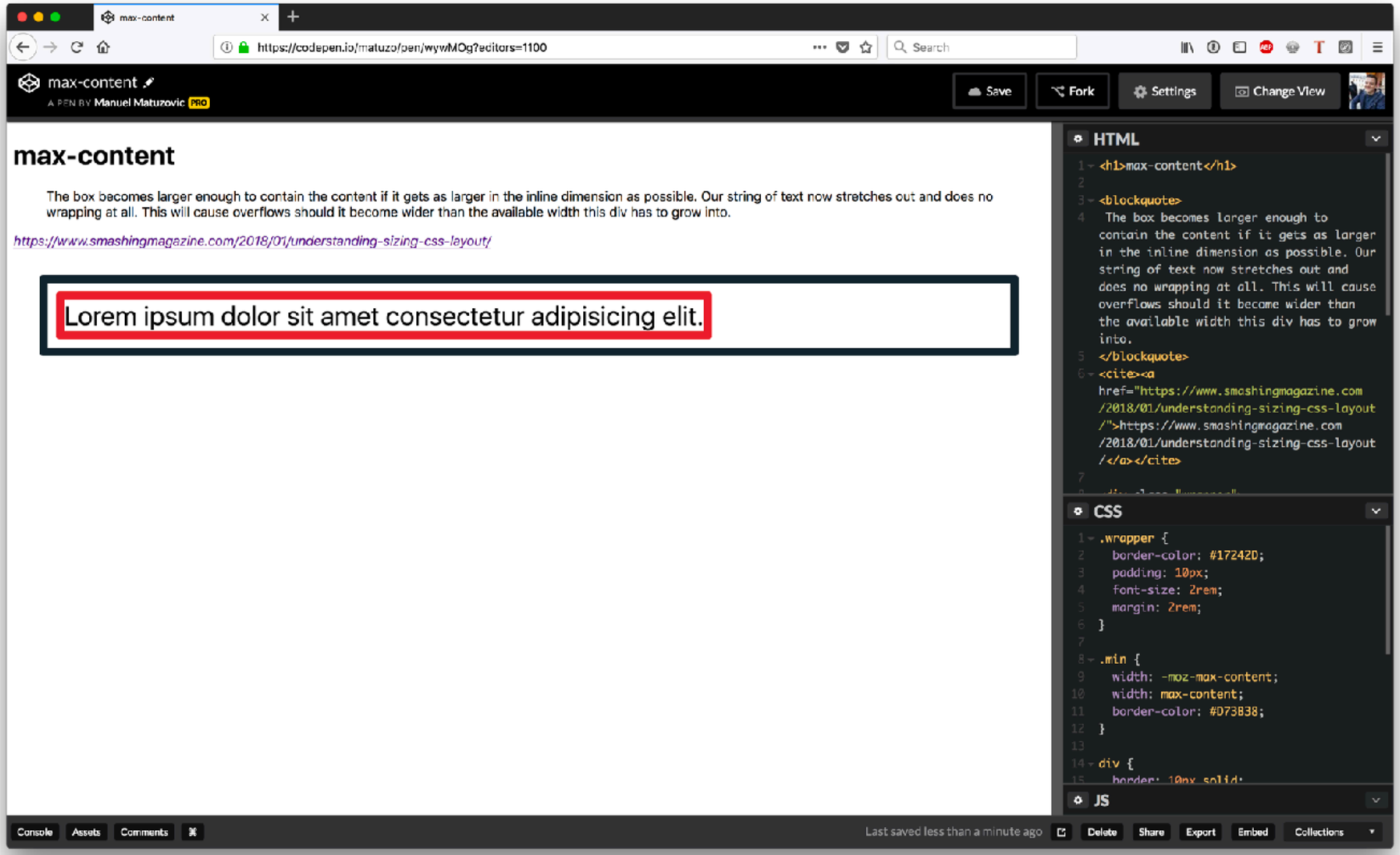
bit.ly/grid_min

Shorthands

max-content

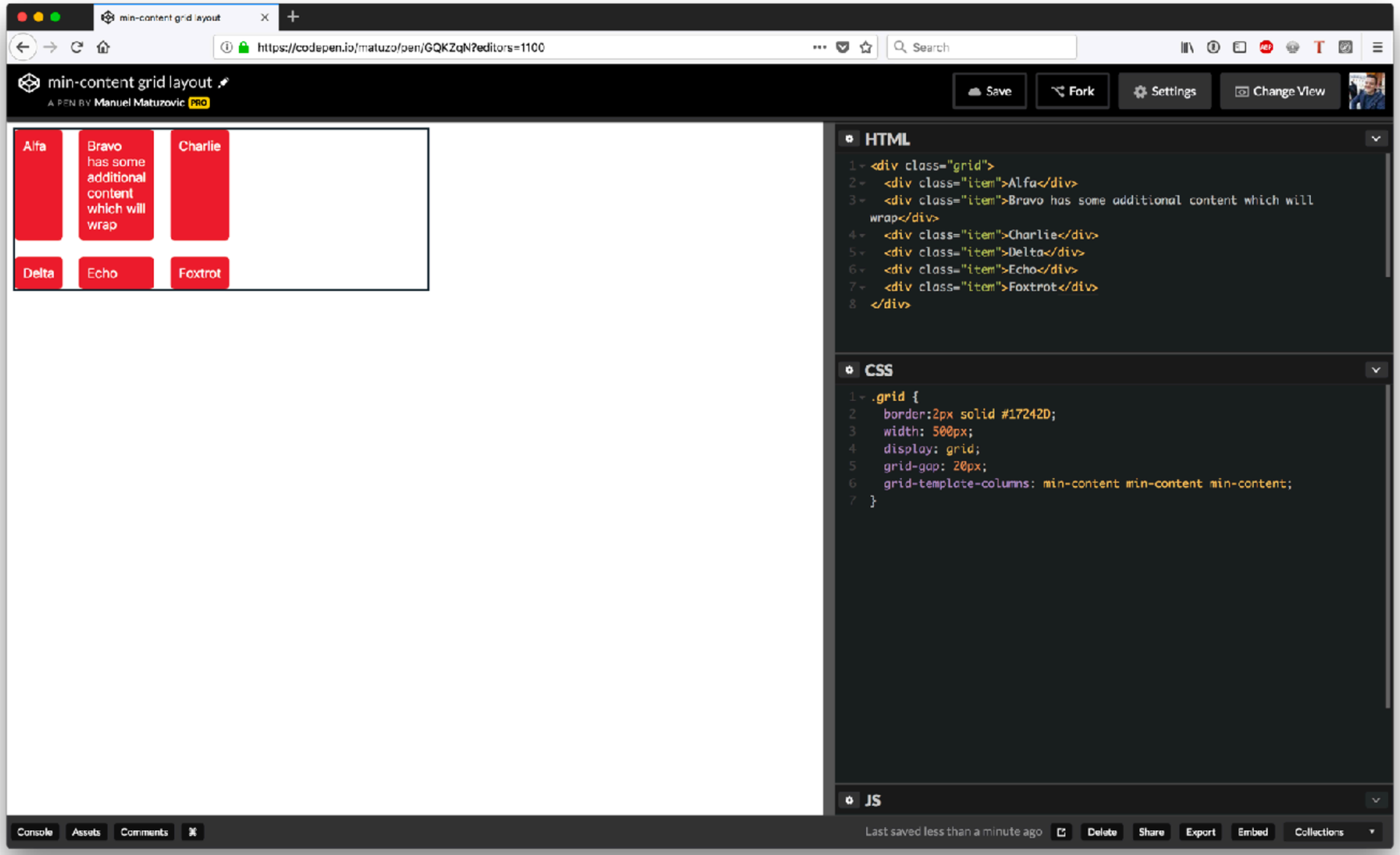
The item becomes large enough to contain the content.

```
.item {
  width: -moz-max-content;
  width: max-content;
}
```



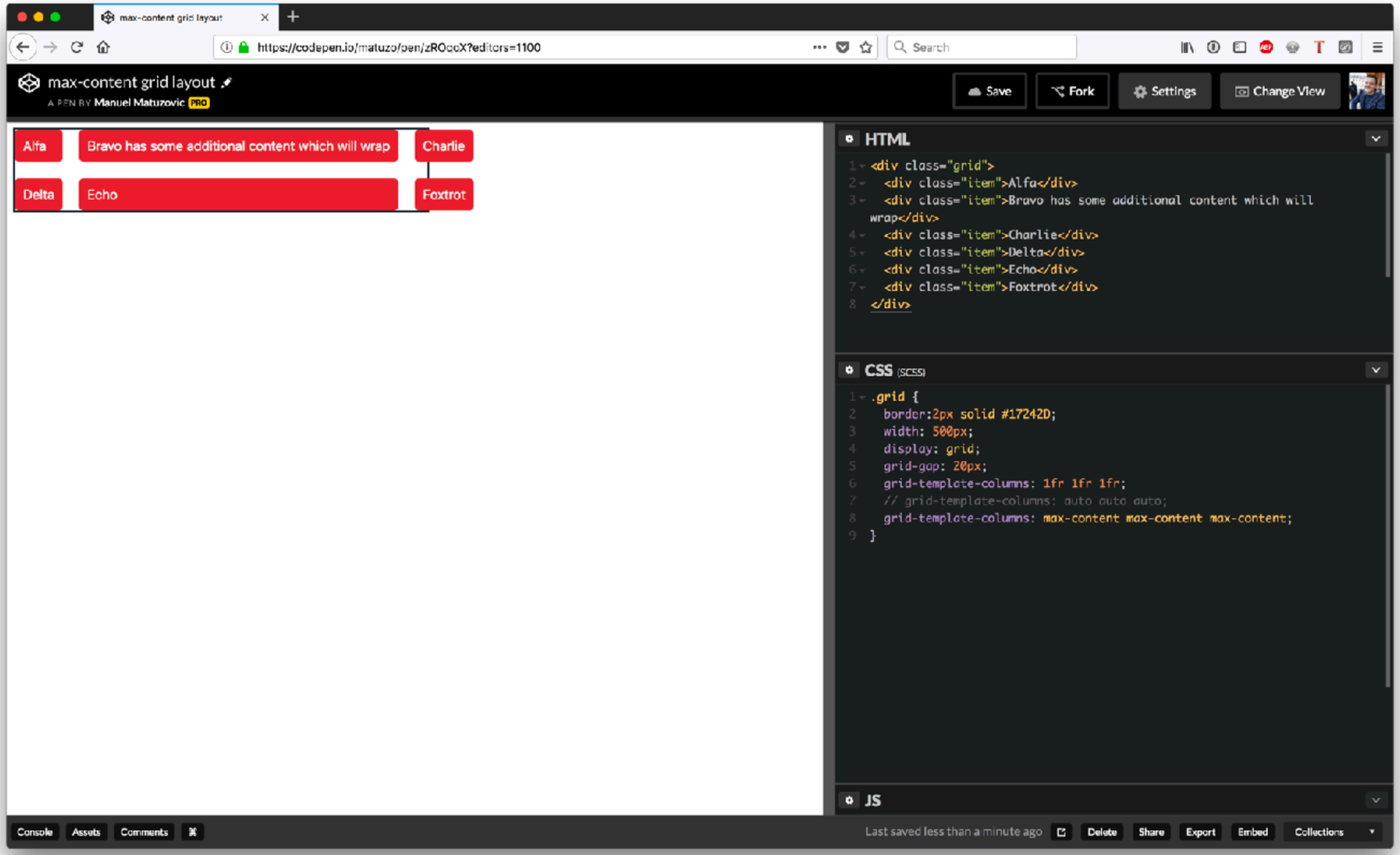
MAX-CONTENT

bit.ly/grid_max



MIN-CONTENT FOR TRACK SIZING

bit.ly/grid_min2



MAX-CONTENT FOR TRACK SIZING

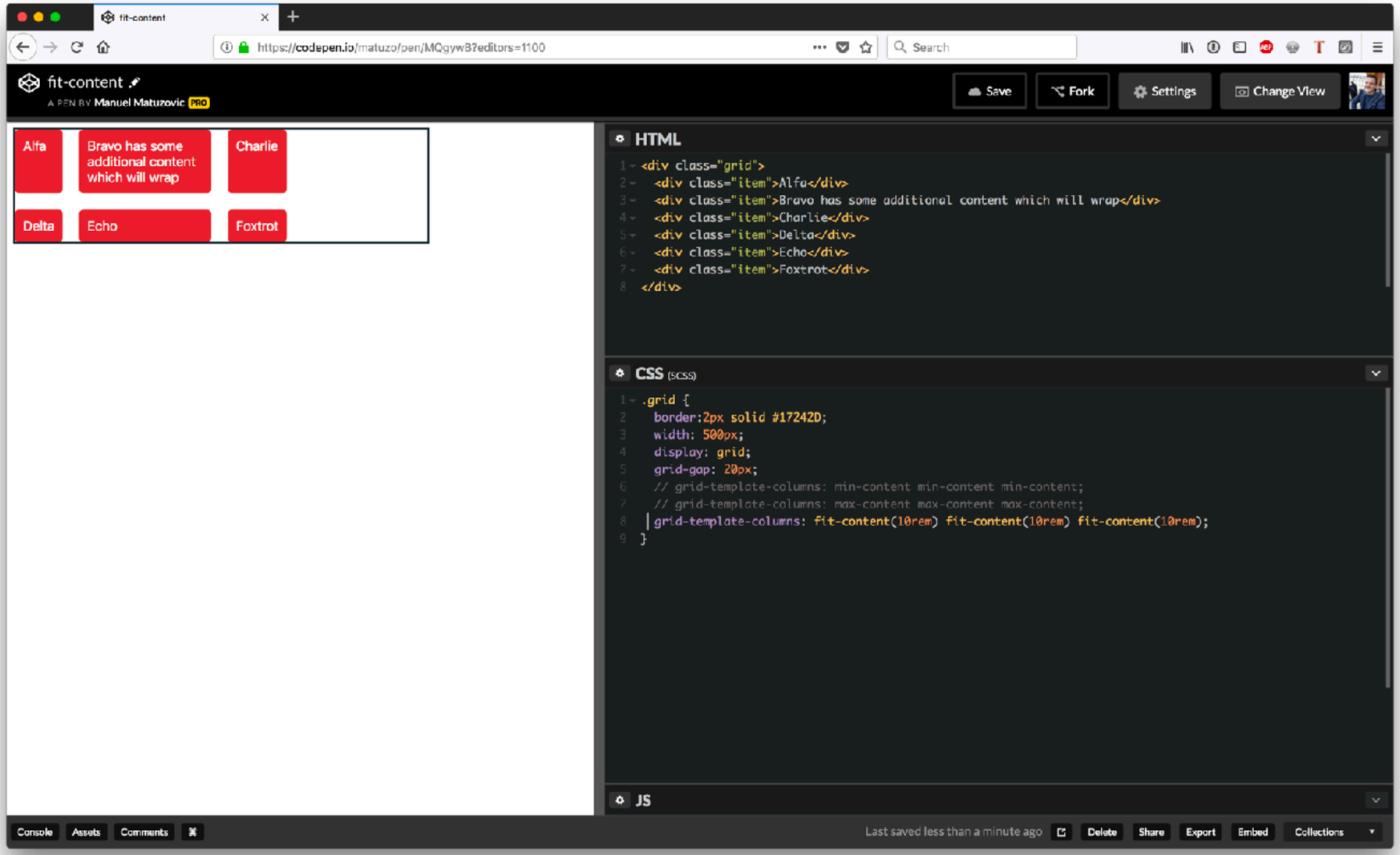
bit.ly/grid_max2

Shorthands

fit-content()

fit-content() takes a length or percentage as a value. When you use fit-content for track sizing, the track will act like max-content until it gets to the size of the passed in value.

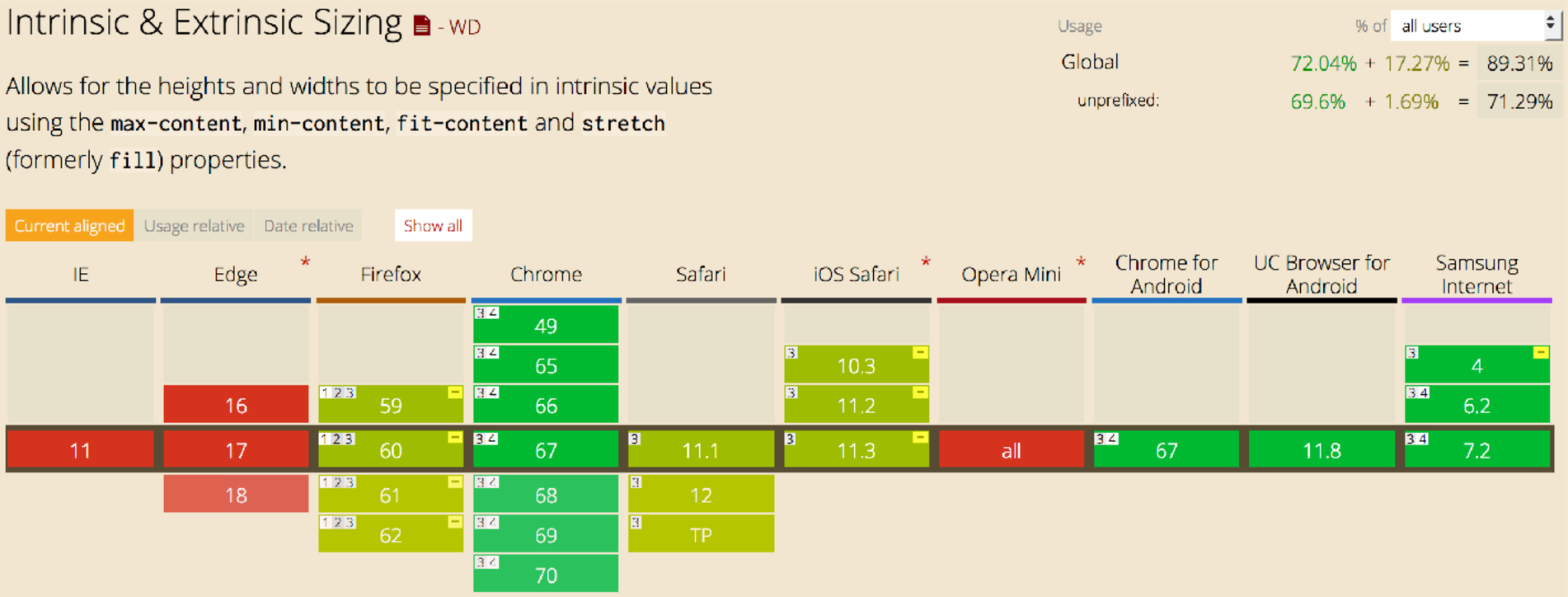
```
.grid {
  display: grid;
  grid-template-columns: fit-content(10rem) fit-content(10rem) fit-content(10rem);
}
```



FIT-CONTENT

bit.ly/grid_fit

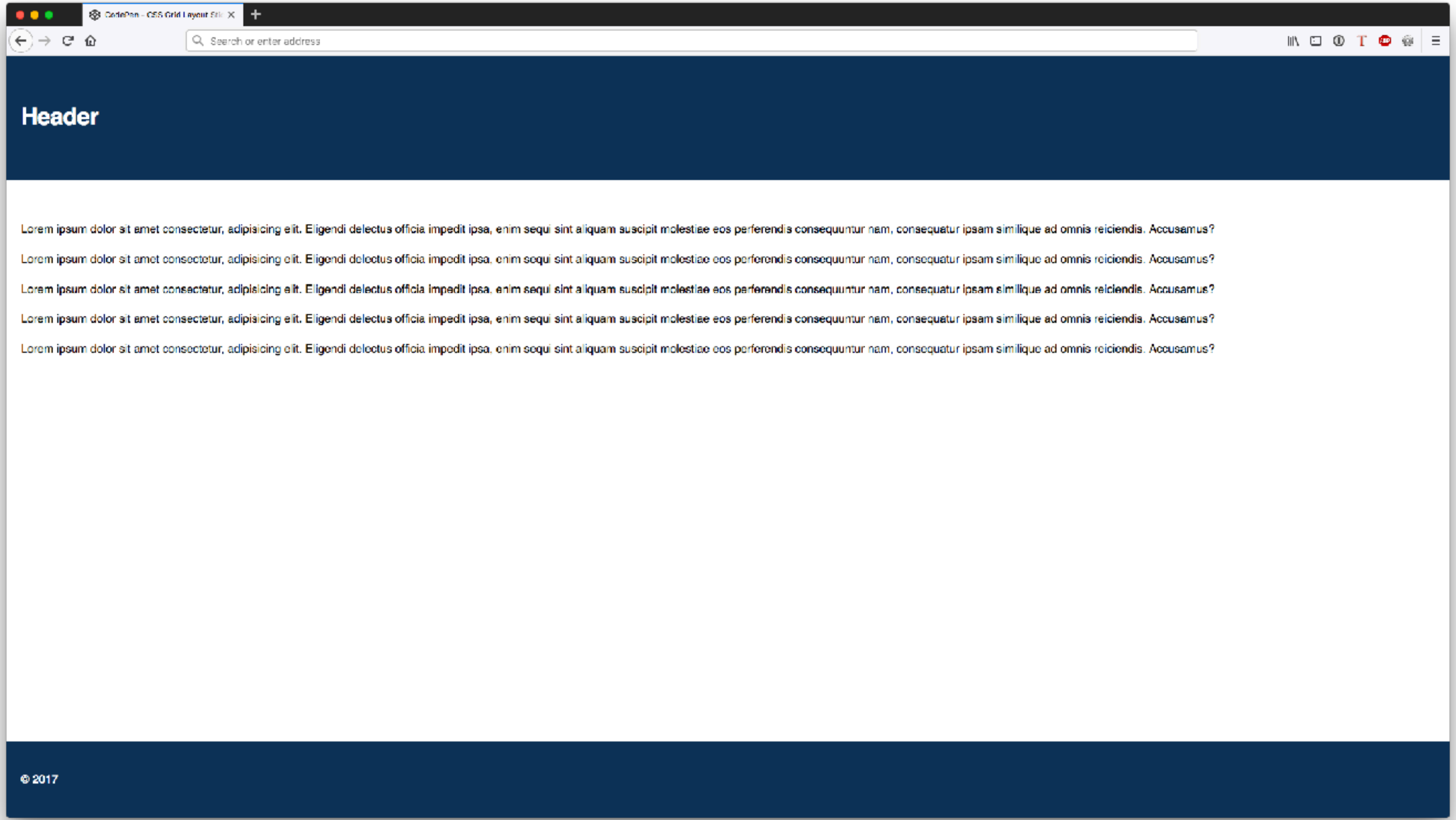
Browsersupport



PATTERNS

Some common patterns

EXERCISE 4



EXERCISE: STICKY FOOTER

https://bit.ly/grid_sticky

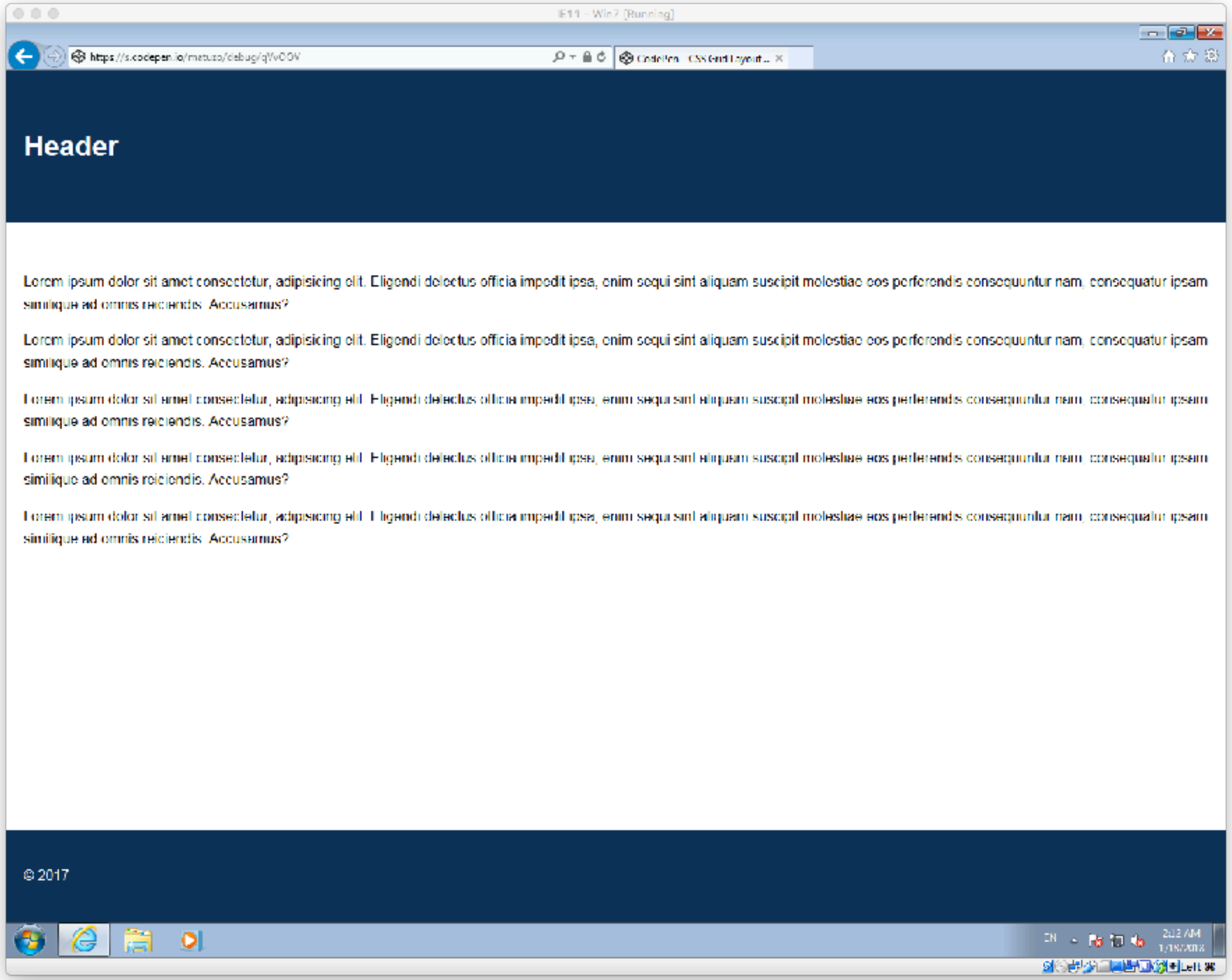
Sticky footer

EXERCISE 4

- **Try to implement a sticky footer pattern with grid.**
- Footer Sticky at the bottom of the page.
- Footer moves with the content if the content is longer than the page.
- Flexible height for header and footer.

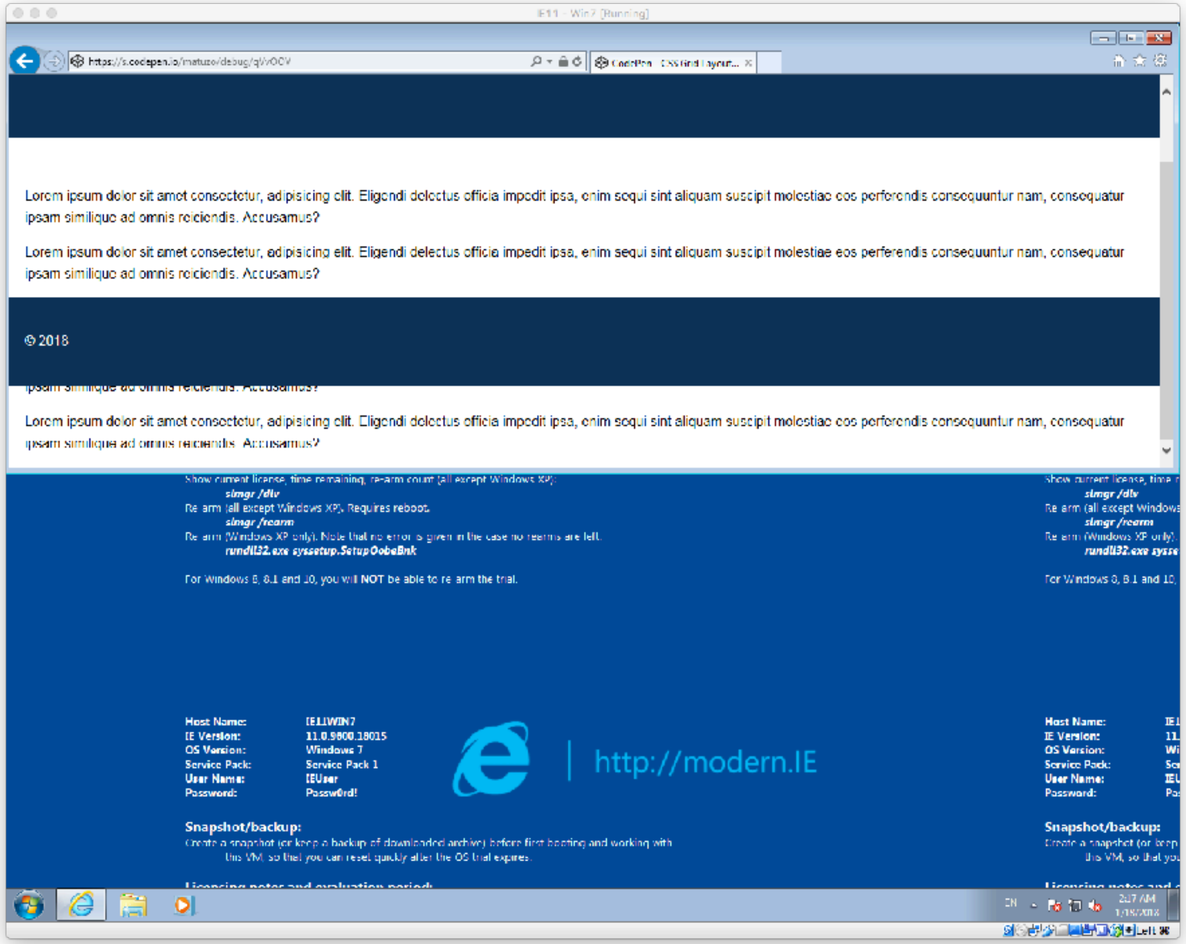
https://bit.ly/grid_sticky

EXERCISE 4.5



STICKY FOOTER IN IE: 🤔

https://bit.ly/grid_sticky_ie

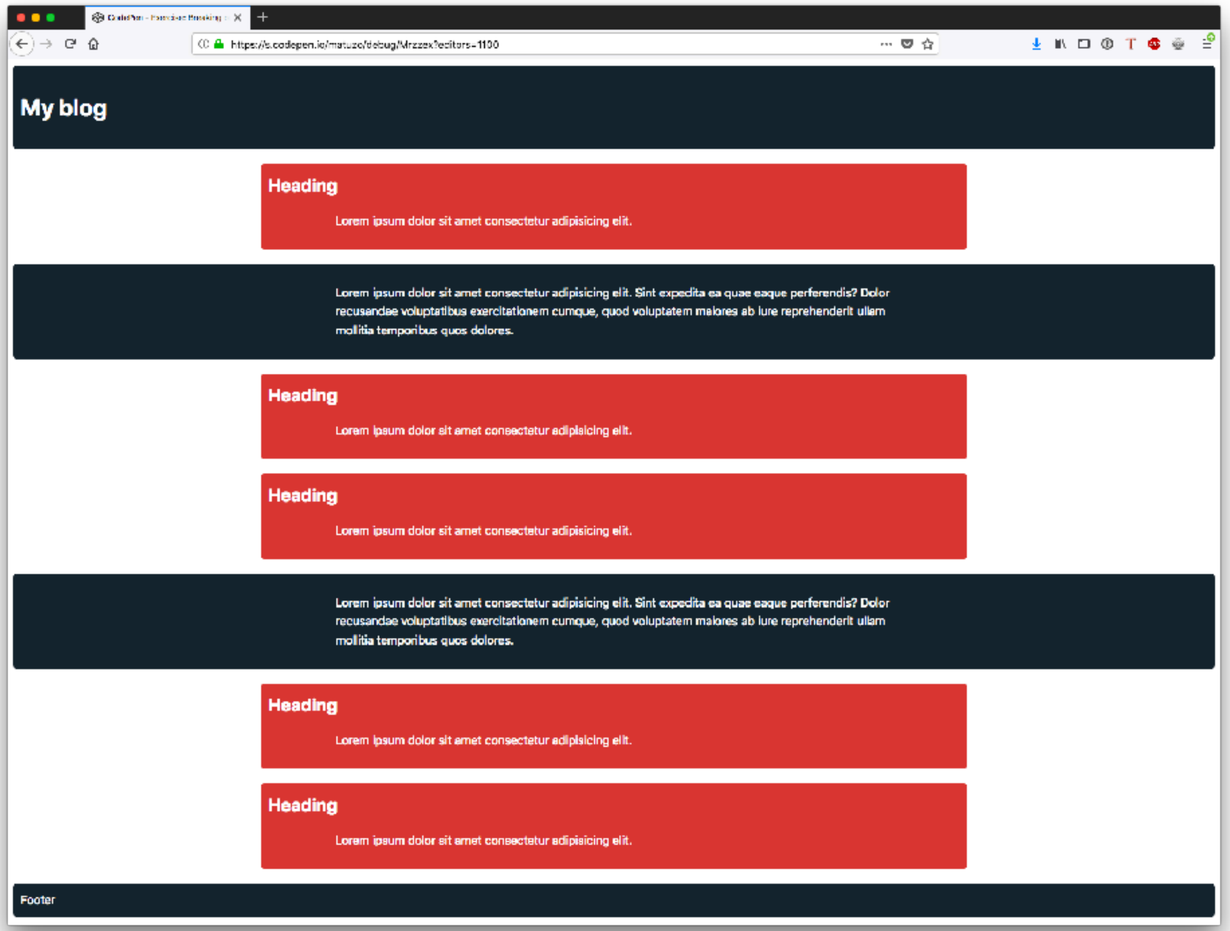


EXERCISE 4.5

STICKY FOOTER IN IE: 😞

https://bit.ly/grid_sticky_ie

EXERCISE 5



EXERCISE: BREAKING OUT

http://bit.ly/grid_break

ACCESSIBILITY

„Authors must use order and the grid-placement properties only for visual, not logical, reordering of content. Style sheets that use these features to perform logical reordering are non-conforming. “

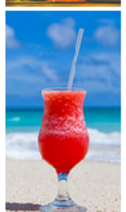
– https://www.w3.org/TR/css-grid-1/#order-accessibility



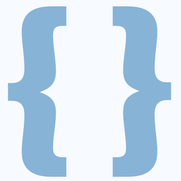








```
<ul>
  <li>
    <a href="#">
      <img src="/drink3.jpg" alt="A delicious drink" />
    </a>
  </li>
  …
</ul>
```



```
ul {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(180px, 1fr));
  grid-gap: 20px;
  grid-auto-rows: 180px;
}
```

A grid with as many 180px wide columns as possible

```
.long {
  grid-row: span 2;
}

.large {
  grid-row: span 2;
}

.large1 {
  grid-column: span 2 / -1;
}

.large2 {
  grid-column: 1 / span 2;
}
```

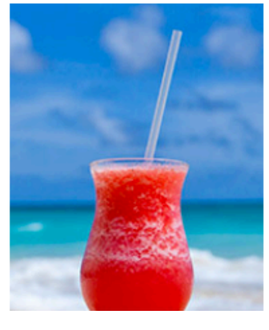


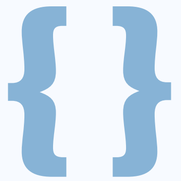












```css
ul {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(180px, 1fr));
  grid-gap: 20px;
  grid-auto-rows: 180px;
  grid-auto-flow: dense;
}
```

grid-auto-flow tries to fill the gaps in a grid

Accessibility

- Planing and design start with the content and source order.
- If something needs to be moved significantly, move it in the DOM and not visually.
- Don’t be tempted to compromise on semantics and flatten the structure due to design or developer experience reasons .

Development process

- Start with a semantic and well structured document.
- Create the grid and check if the structure still makes sense.
- Test your pages and components with the keyboard.
- If required, change the source order.

Links

- http://adrianroselli.com/2015/09/source-order-matters.html
- http://adrianroselli.com/2015/10/html-source-order-vs-css-display-order.html
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/CSS_Grid_Layout_and_Accessibility
- https://www.w3.org/TR/css-grid-1/#order-accessibility

STACKING ORDER

Using z-index with grid items

Stacking order

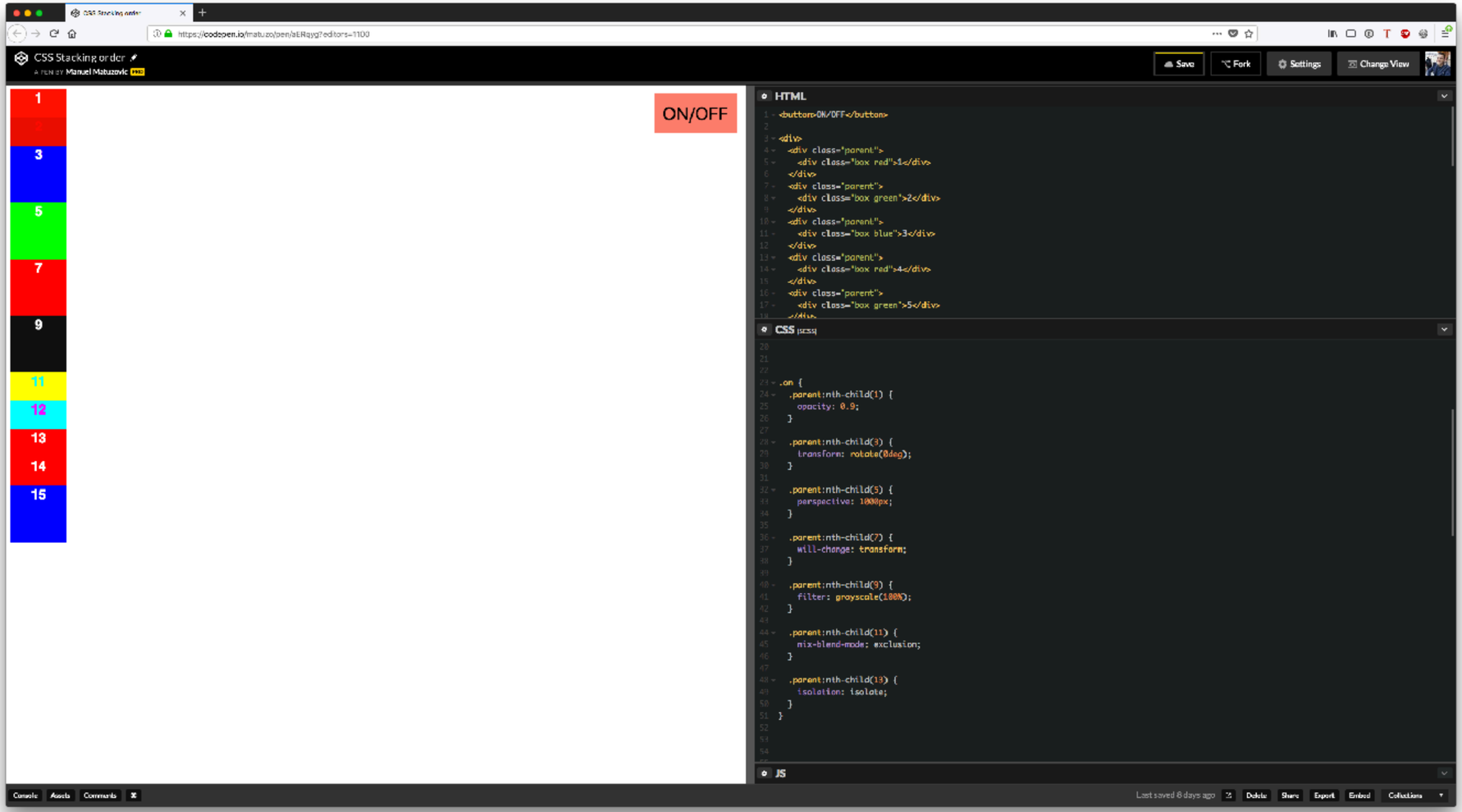
- The rendering order of elements on the z-axis can be influenced by changing the `z-index` property.
- This applies only to items which form a *stacking context*.
- A stacking context is formed, anywhere in the document, by any element in the following scenarios:

Stacking context

- position with a value of relative, absolute, fixed or sticky.
- opacity less than 1.
- mix-blend-mode value other than normal.
- Elements with any of the following properties with values other than none: transform, filter, perspective, clip-path, mask / mask-image / mask-border
- Element with an isolation value isolate.

Stacking context

- Elements with a -webkit-overflow-scrolling value touch.
- Elements with a will-change value specifying any property that would create a stacking context on non-initial value.
- Flex items.
- Grid items.



STACKING ORDER

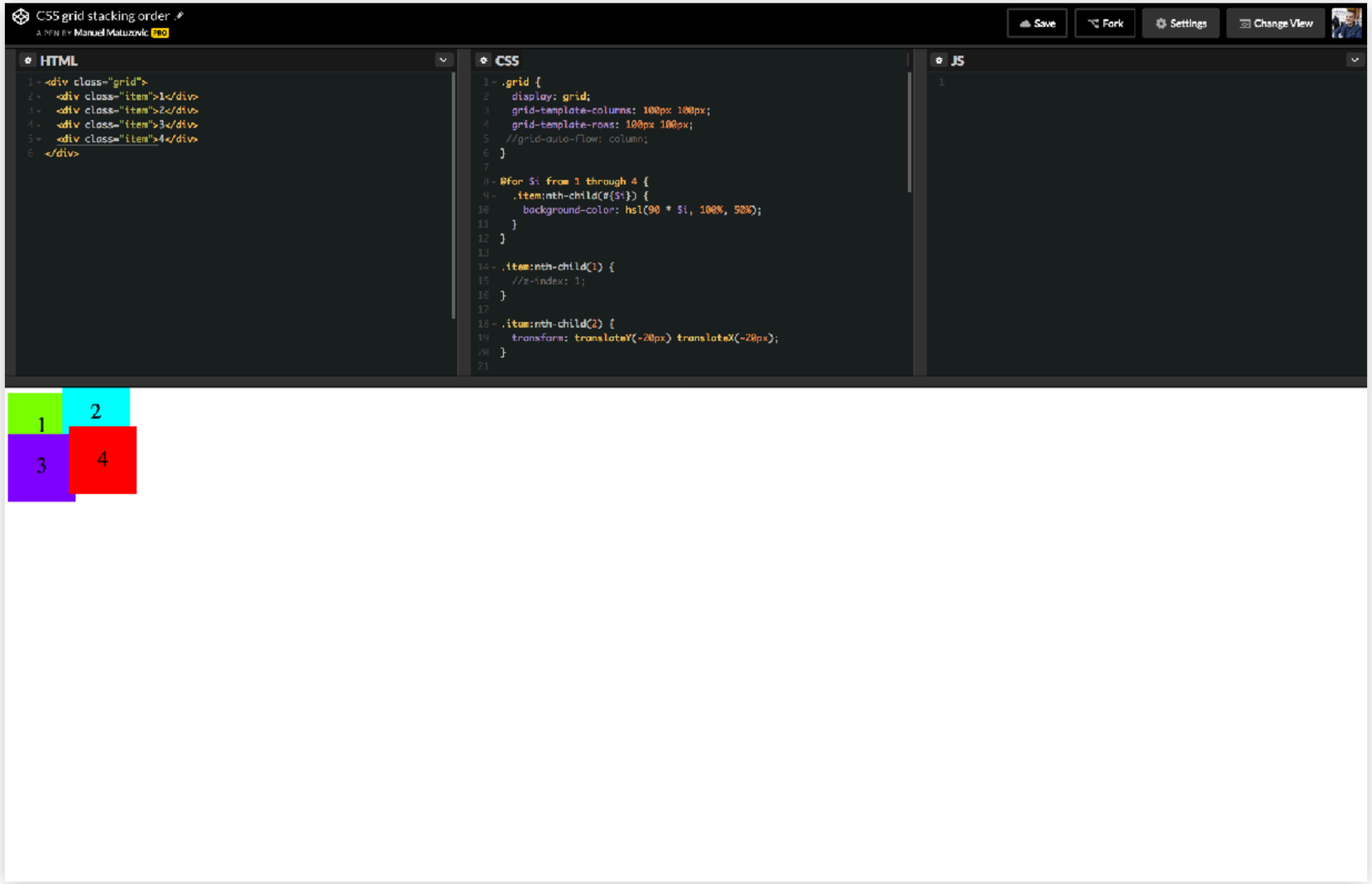
bit.ly/grid_stacking

Stacking order

It’s possible to stack grid items.

Their positions on the z-axis is determined by the `z-index` property.

```
.grid-item {
  grid-column-start: 1;
}


.grid-item2 {
  grid-column-start: 1;
  z-index: 1;
}
```

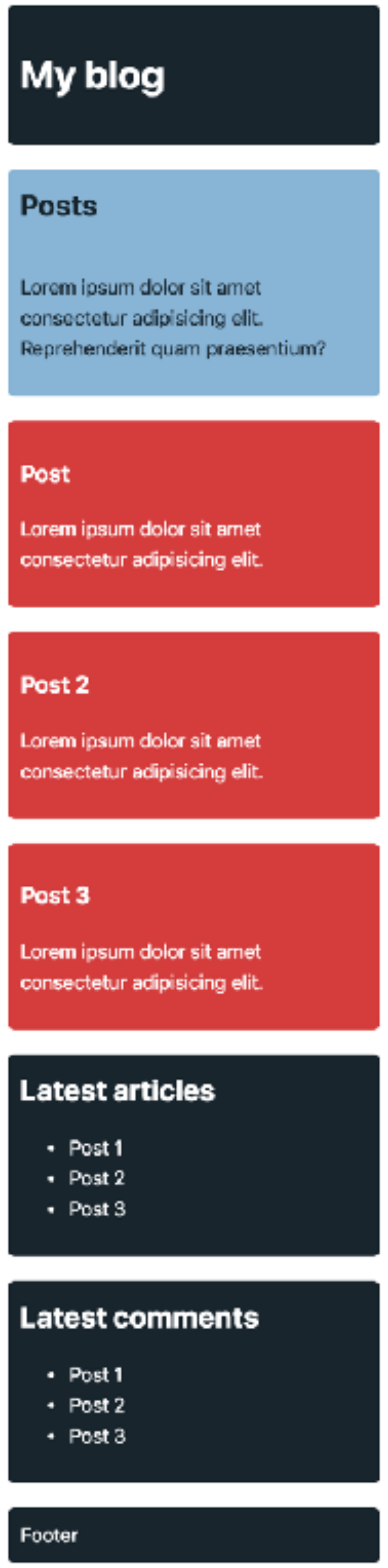


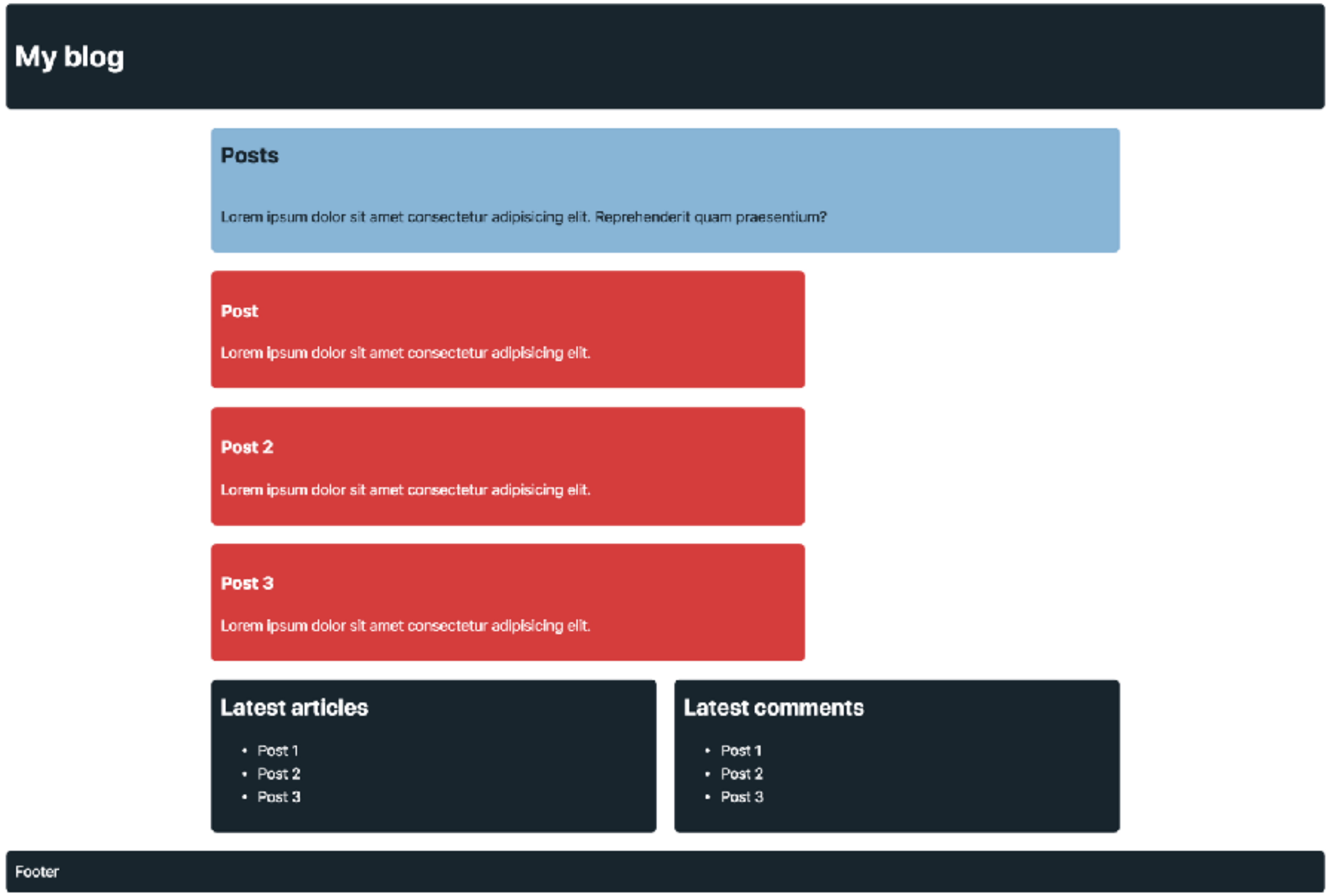
STACKING ORDER GRID

bit.ly/grid_stacking1

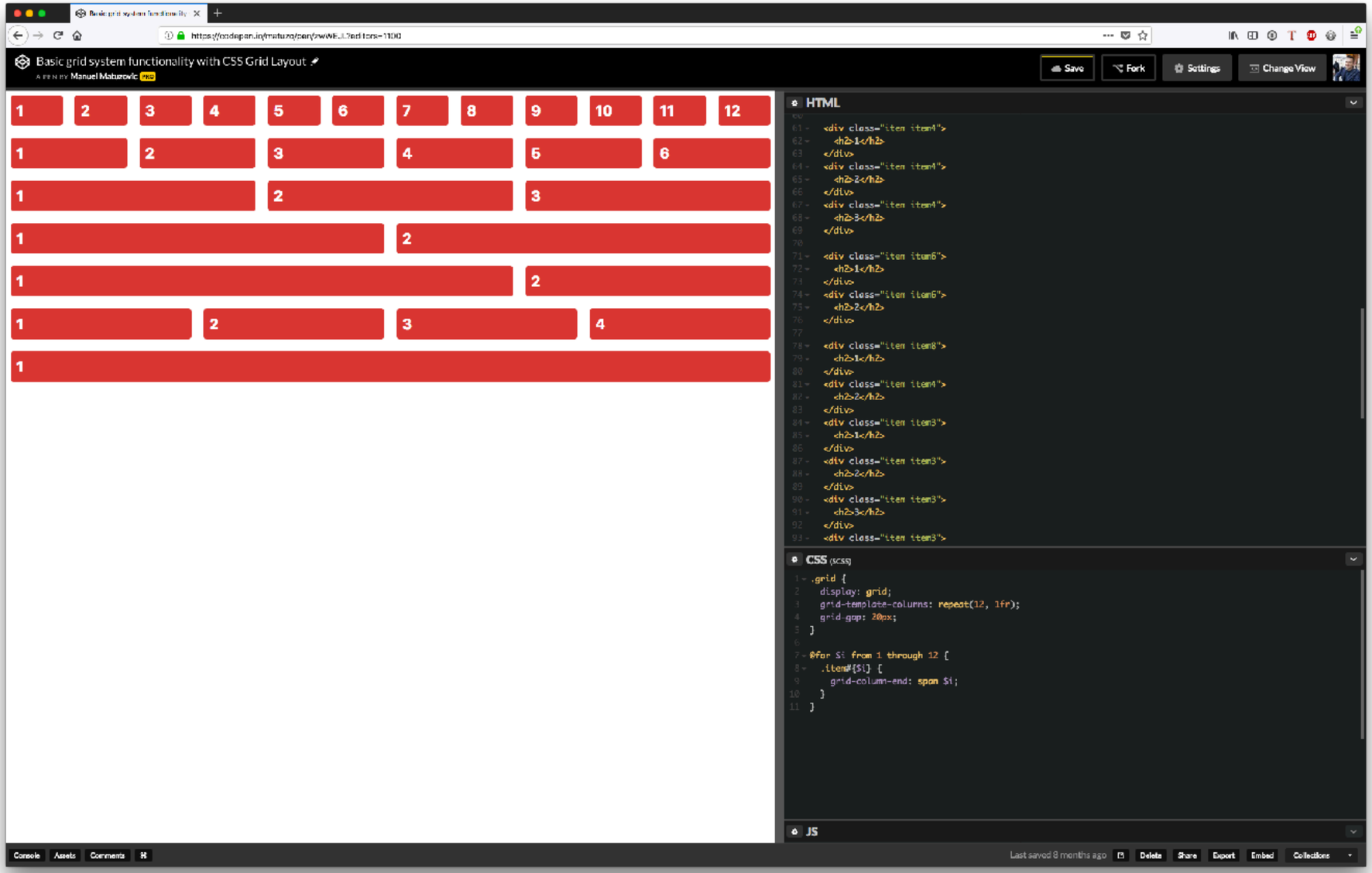
RESPONSIVE WEBDESIGN

EXERCISE 6



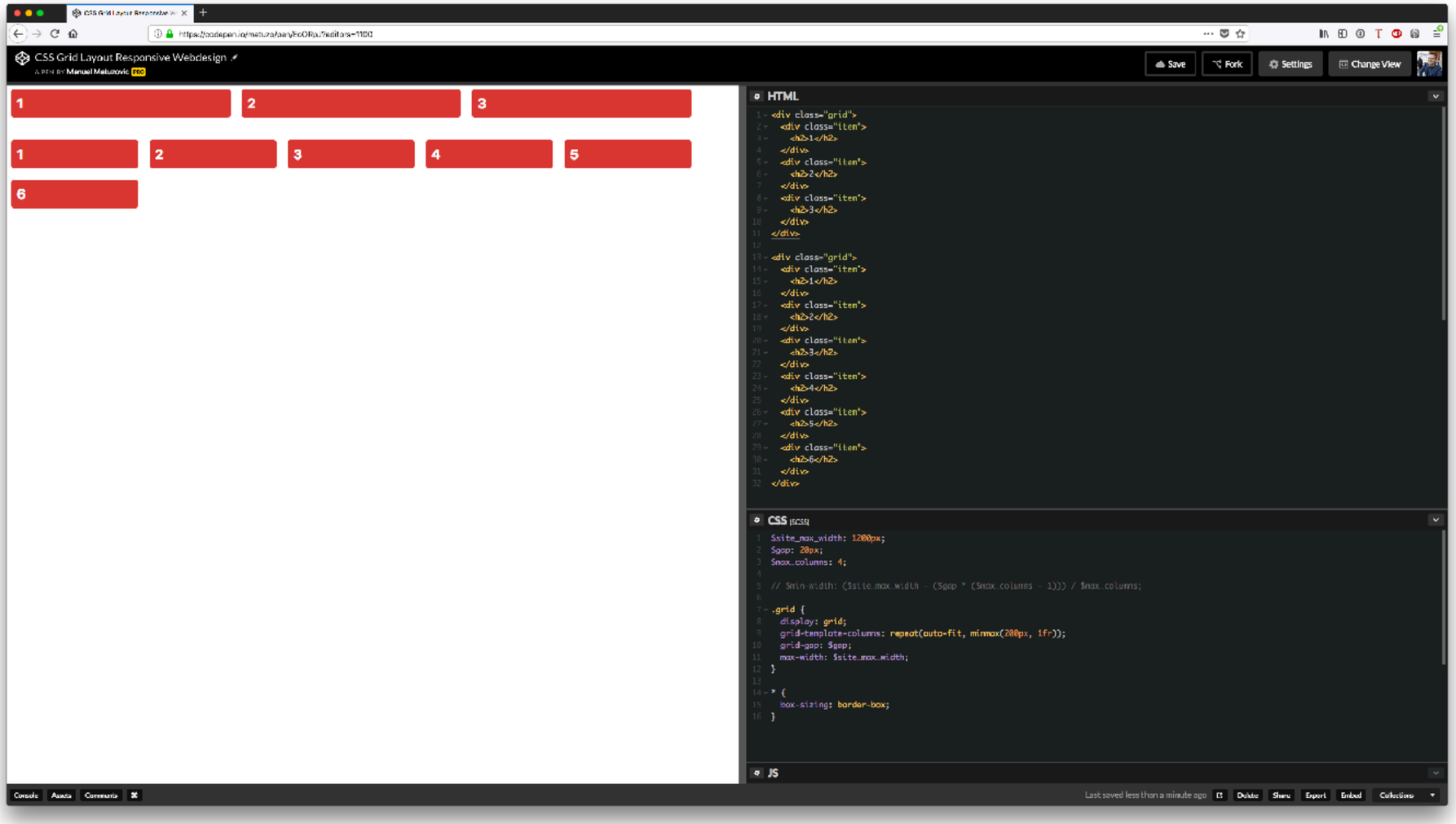


BIT.LY/GRID_MQ



BASIC GRID SYSTEM FUNCTIONALITY

bit.ly/grid_mq3



RWD WITHOUT MEDIA QUERIES

bit.ly/grid_mq4

PROGRESSIVE ENHANCEMENT

Progressive Enhancement

- Progressive enhancement is a philosophy.
- One principle: fault tolerance.
- Browsers must ignore anything they don’t understand in HTML and CSS.
- Fault tolerance is the reason progressive enhancement works.
- The core focus is the content and everything builds upon that.

https://adaptivewebdesign.info/1st-edition/read/chapter-1.html

“Progressive enhancement ensures that all content (that is to say the information contained in a website) is both available to and usable by anyone, regardless of her location, the device she is using to access that information, or the capabilities of the program she is using to access that content”

– Aaron Gustafson

Progressive Enhancement

- PE doesn’t require that you provide the same experience in different browsers.
- PE doesn’t preclude you from using the latest and greatest technologies.
- PE just asks you to apply technologies in an intelligent way, layer-upon-layer.
- Progressive enhancement = excellent customer service.

Progressive Enhancement in HTML

Due to HTMLs fault tolerance this markup will work even in very old (e.g. IE 8) or basic (e.g. Lynx) browsers.

```
<header>
  <h1>My site</h1>
  <nav>
    <ol>
      <li><a href="#links">Links</a></li>
    </ol>
  </nav>
</header>
```

Progressive Enhancement in HTML

Using the picture element to provide modern browsers with webp images.

Browsers the either don’t understand webp or the picture element fall back to the img.

```
<picture>
  <source srcset="opera.webp" type="image/webp">
  <img src="opera.jpg" alt="The Oslo Opera House">
</picture>
```

Progressive Enhancement in HTML

Instead of seeing nothing when an image fails to load, users will at least know what's in the picture ("*The Oslo Opera House*").

```
<img src="opera.jpg" alt="The Oslo Opera House">
```

Progressive Enhancement in HTML

Using the title attribute on an abbr tag enhances the experience for both blind and sighted users.

```
<abbr title="World Wide Web Consortium">W3C</abbr>
```

Progressive Enhancement in CSS

A browser reads each rule set and examines it. If it encounters something it doesn’t understand, it experiences a parsing error.

```
p {
  color: red;
  font-weight: bold;
}
```

Progressive Enhancement in CSS

Employ parsing errors to advantage by providing fallback declarations.

```
div {
  width: 50vw;
  width: 50vmax;
}
```

Progressive Enhancement in CSS

By preferring (min-width) over max-width in media queries, not only smartphone users but users of browsers than can’t interpret media queries will get at least some kind of experiences.

```
div {
  width: 90%;
}

@media (min-width: 48em) {
  div {
    display: flex;
  }
}
```

Graceful Degradation

Progressive Enhancement

PROGRESSIVE ENHANCEMENT IN MEDIA QUERIES

Maven

Progressive Enhancement in JS

Only capable browsers will cache contents offline and improve performance and user experience.

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('sw.js').then(function(reg)
{
    // ...
  }).catch(function(error) {
    // ...
  });
};
```

Progressive Enhancement

- float and clear have no effect
- vertical-align has no effect
- flex items become grid items
- block, inline-block or table-cell items become grid items.
- column- properties have no effect

Progressive Enhancement

It’s not necessary to create multiple layouts.

```
.grid {
  display: flex;
  flex-wrap: wrap;
  display: grid;
  grid-template-columns: repeat(5, 1fr);
}
```

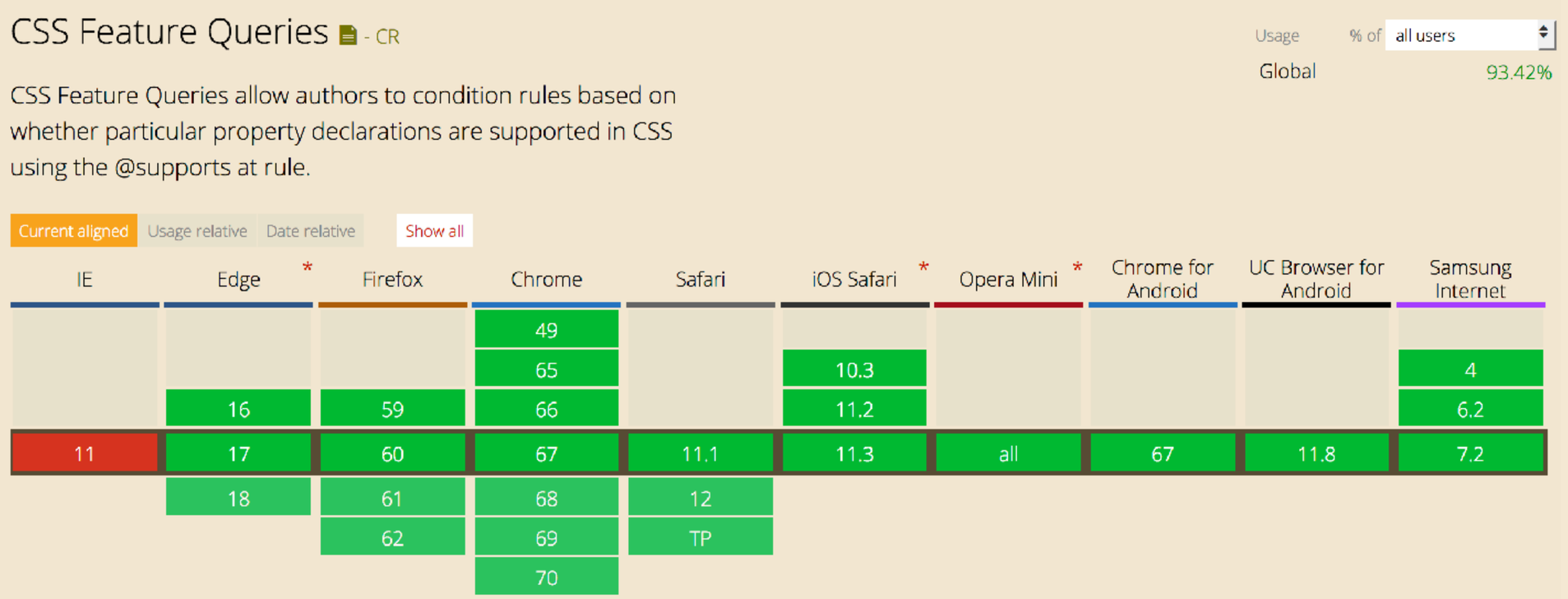
Progressive Enhancement

Feature Queries.

Check for property + value.

```
@supports (display: grid) {
  .grid {
    margin-right: 0;
  }
}
```

Feature Queries



http://caniuse.com/#search=feature%20queries

Detecting Support

- We don’t need tools like Modernizr.
- All browsers that understand Grid unterstand Feature Queries as well.
- It might be necessary to differentiate between the old and the new spec.
- In newer versions of Edge queries for both the old and the new syntax evaluate to true.

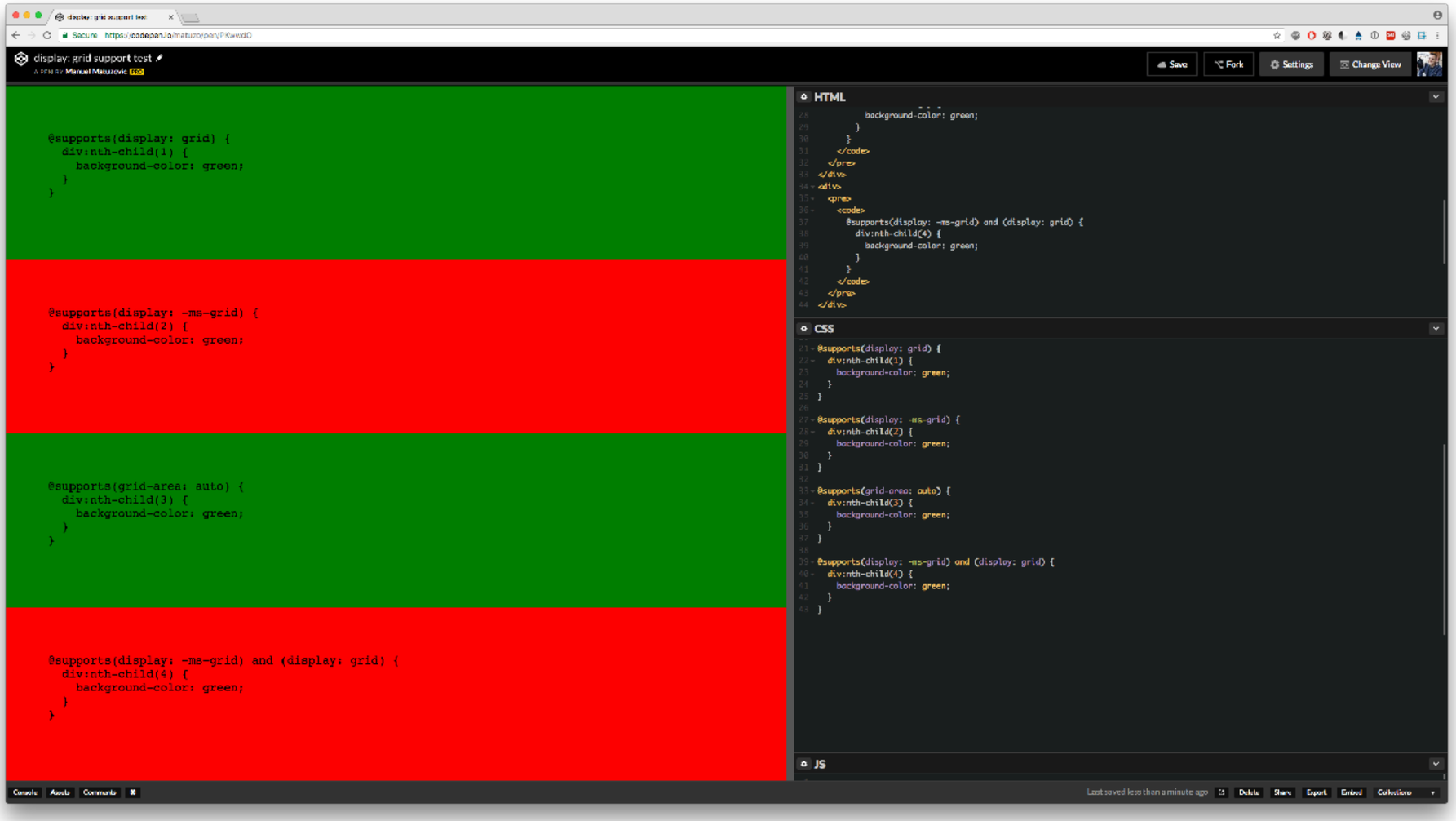
Feature detection

Detecting the new spec

```
@supports(display: grid) {
  div {
    margin: 0;
  }
}
```

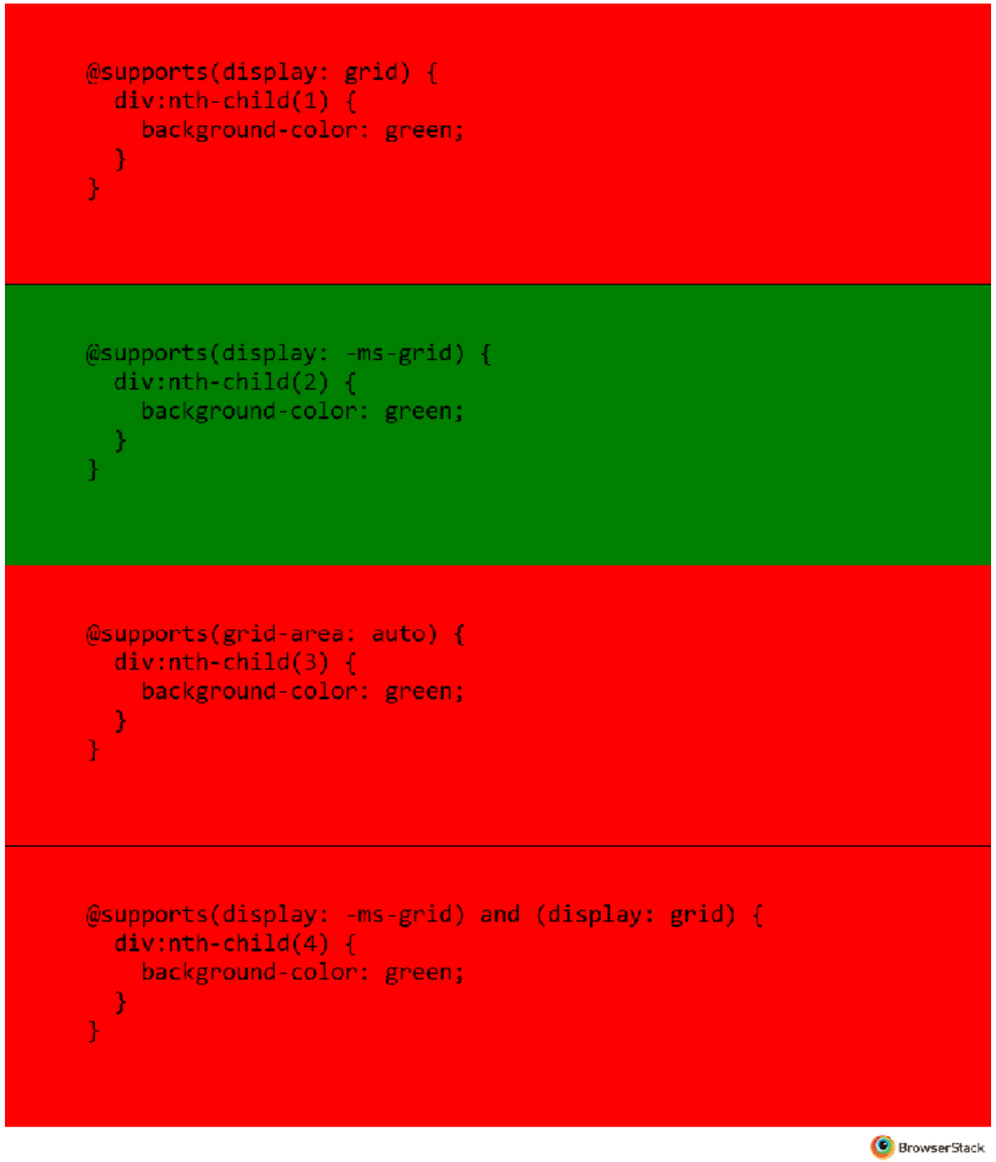
Detecting the old spec

```
@supports(display: -ms-grid) {
  div {
    margin: 0;
  }
}
```



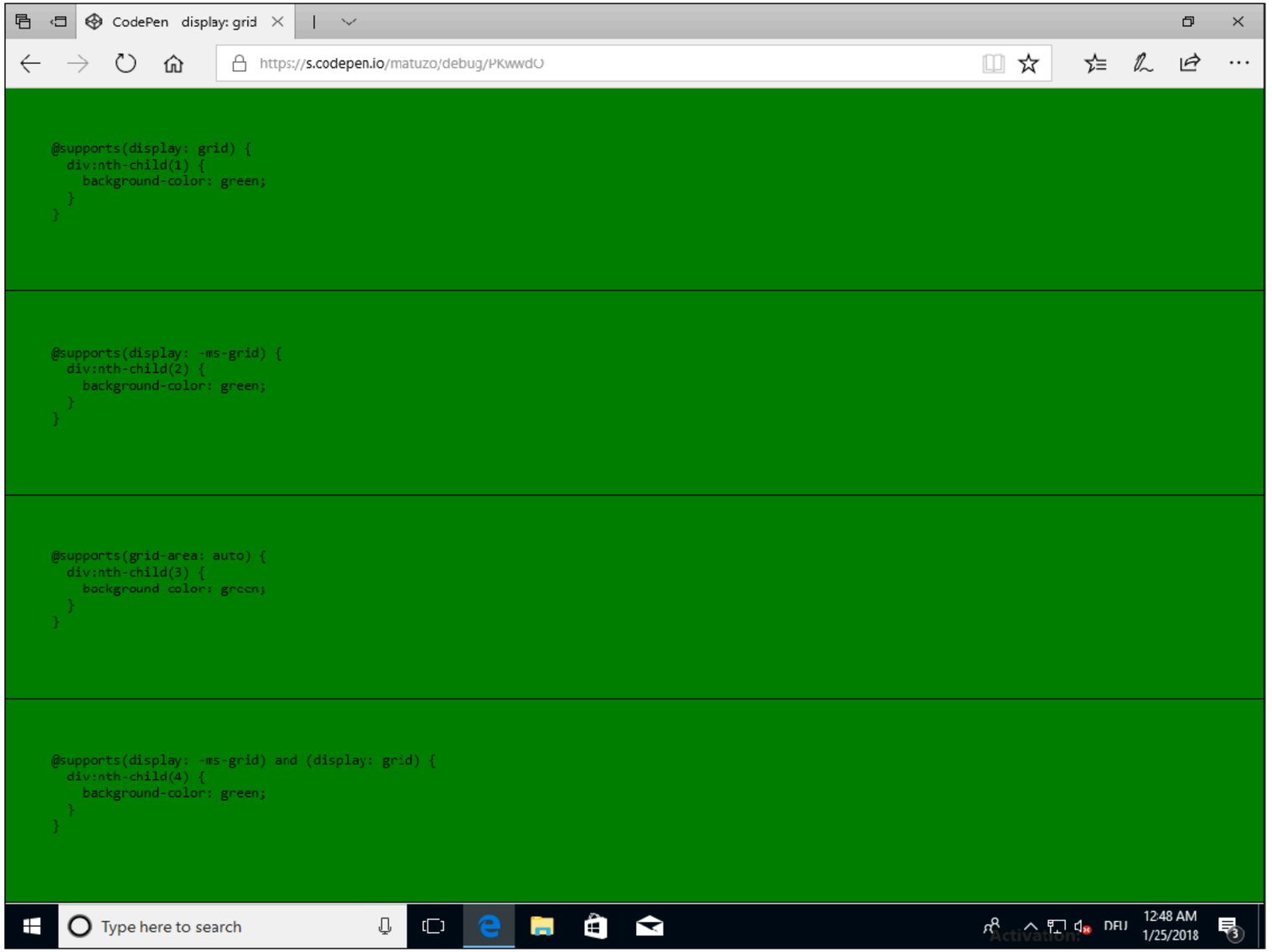
GRID FEATURE TEST IN CHROME 66

bit.ly/grid_feature1



GRID FEATURE TEST IN EDGE 15

bit.ly/grid_feature1



GRID FEATURE TEST IN EDGE 16

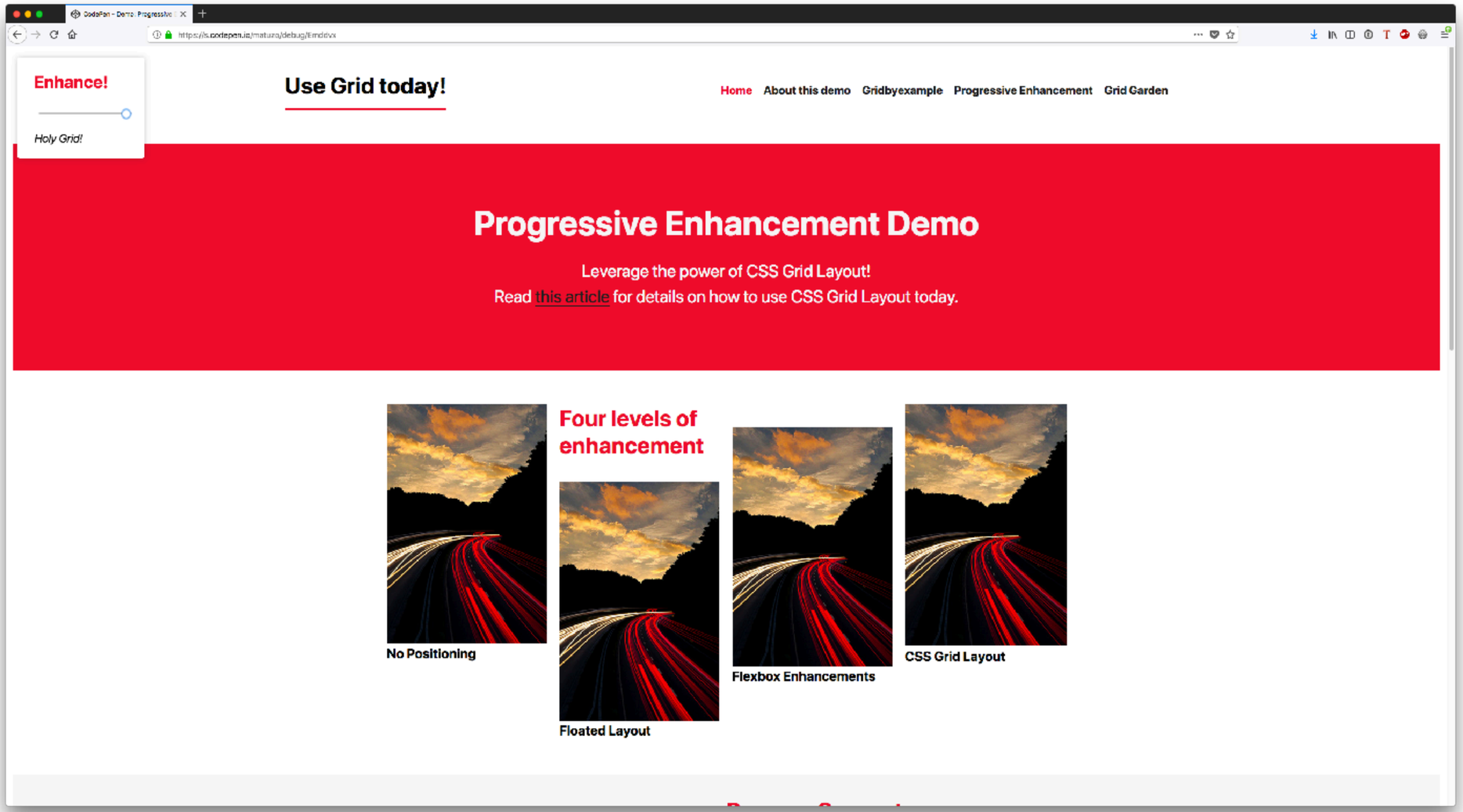
bit.ly/grid_feature1

Progressive Enhancement

Feature Queries.

Only change what's necessary.

```
.grid-item {
  float: left;
  margin-right: 20px;
}

@supports(display: flex) {
  .grid-item {
    margin-right: 0;
  }
}
```



PROGRESSIVELY ENHANCED GRID

bit.ly/grid_pe

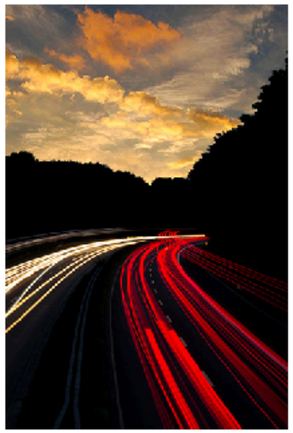
Progressive Enhancement Example

Markup: section, heading and 4 articles

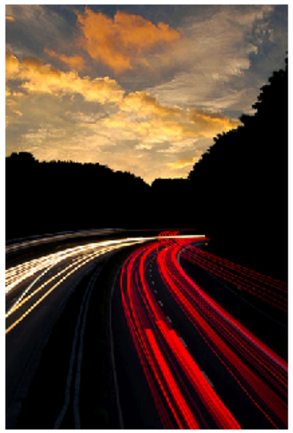
```
<section>
  <h2>Four levels of enhancement</h2>

  <article>
    <h3>No Positioning</h3>
  </article>
  <article>
   …
   </article>
</section>
```

Four levels of enhancement

No Positioning



Floated Layout



Flexbox Enhancements

EXAMPLE: HTML

bit.ly/grid_pe2

Progressive Enhancement Example

Floating articles

```
article {
  float: left;
  width: 24.25%;
}

article:not(:last-child) {
  margin-right: 1%;
}

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

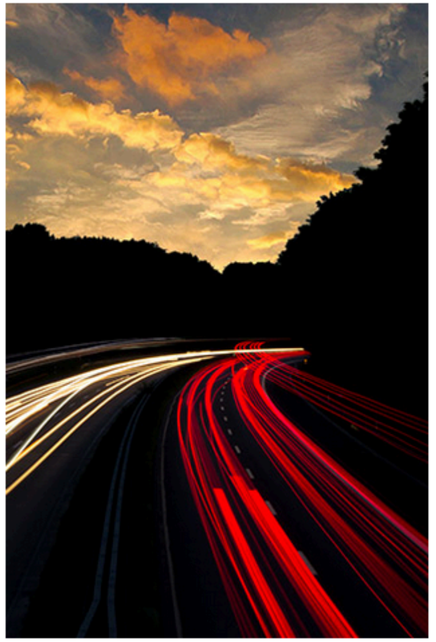
Four levels of enhancement

No Positioning

Floated Layout

Flexbox Enhancements

CSS Grid Layout



EXAMPLE: FLOAT

bit.ly/grid_pe3

Progressive Enhancement Example

Enhancing with Flexbox.

```
article {
  float: left;
  width: 24.25%;

  display: flex;
  flex-direction: column;
}

h3 {
  order: 1;
}
```

Four levels of enhancement

No Positioning

Floated Layout

Flexbox Enhancements

CSS Grid Layout

EXAMPLE: FLEXBOX

bit.ly/grid_pe4

Progressive Enhancement Example

Enhancing with Grid.

```
section {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: 20px;
  max-width: 1300px;
}
```

Four levels of enhancement

No Positioning

Floated Layout

Flexbox Enhancements

CSS Grid Layout

EXAMPLE: GRID

bit.ly/grid_pe1

Progressive Enhancement Example

Floating articles

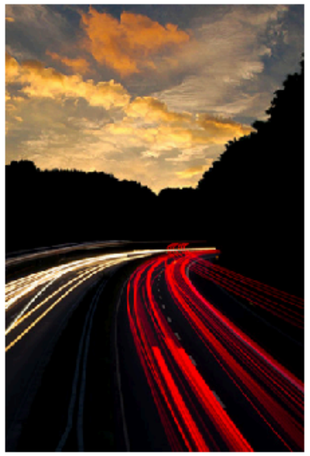
```
article {
  float: left;
  width: 24.25%;
}

article:not(:last-child) {
  margin-right: 1%;
}
```

Progressive Enhancement Example

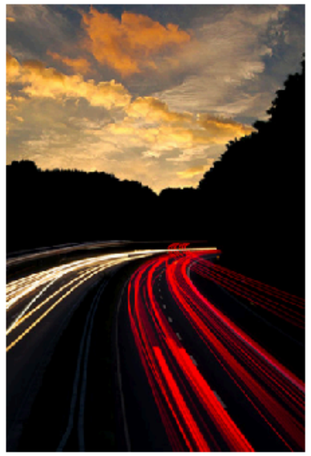
Overwrite only the properties that actually need overwriting.

```
@supports(display: grid) {
  article {
    width: auto;
  }

  article:not(:last-child) {
    margin-right: 0;
  }
}
```

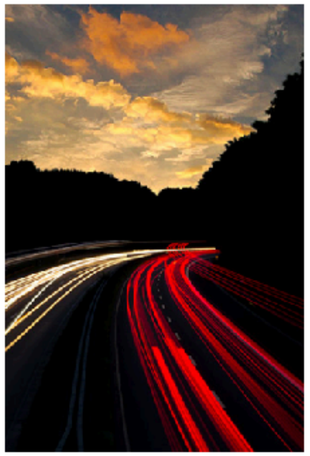
Four levels of enhancement



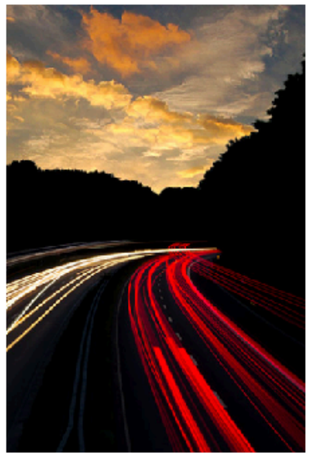
No Positioning



Floated Layout



Flexbox Enhancements



CSS Grid Layout

EXAMPLE: GRID

bit.ly/grid_pe1

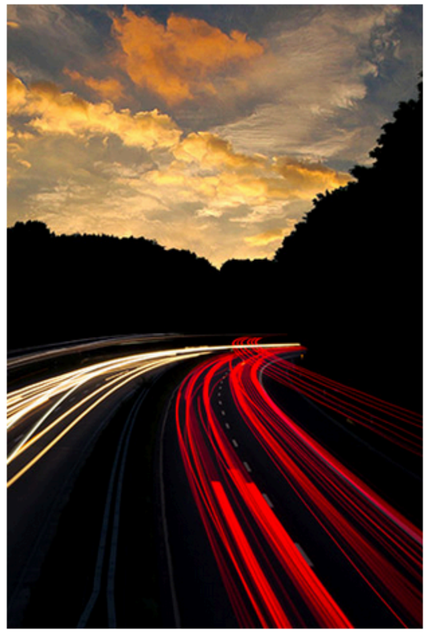
Progressive Enhancement Example

Placing the heading in the second row and aligning the articles.

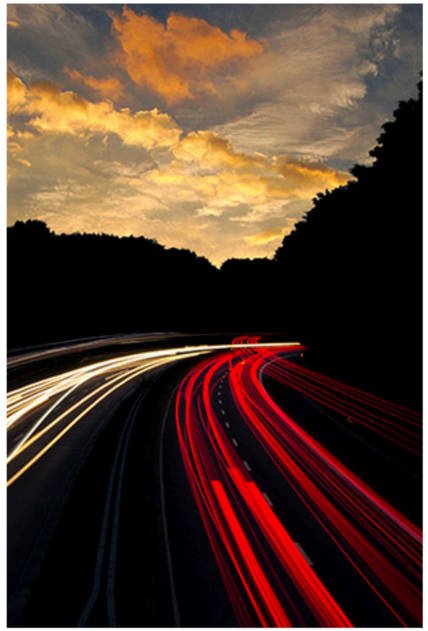
```
article {
  float: left;
  width: 24.25%;
  display: flex;
  flex-direction: column;
  grid-row: span 2;
}

article:nth-of-type(2) {
  grid-column: 2;
  grid-row: 2 / span 2;
}

h2 {
  grid-row: 1;
  grid-column: 2;
  font-size: 50px;
  margin: 0;
}
```

No Positioning

Four levels of enhancement



Floated Layout

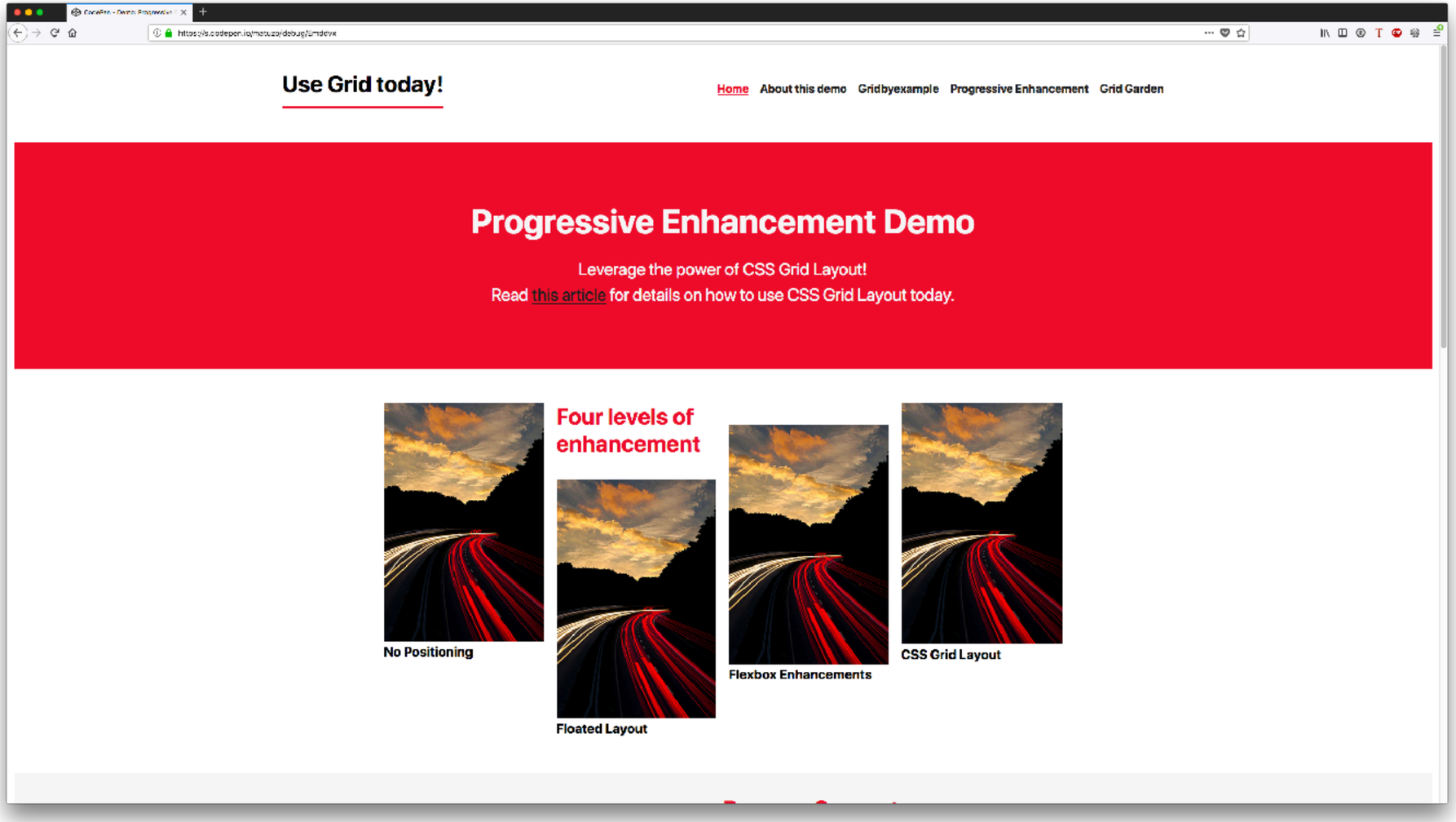


Flexbox Enhancements

CSS Grid Layout

EXAMPLE: GRID

bit.ly/grid_pe1



CHROME - MACOS

Enhance!

Holy Grid!

Use Grid today!

Home About this demo Gridbyexample Progressive Enhancement Grid Garden

Progressive Enhancement Demo

Leverage the power of CSS Grid Layout!
Read this article for details on how to use CSS Grid Layout today.

Four levels of enhancement

No Positioning **Floated Layout** **Flexbox Enhancements** **CSS Grid Layout**

Browser Support

Works even in very old browsers

IE 11

Enhance!

Holy Grid!

Use Grid today!

Home About this demo Gridbyexample Progressive Enhancement Grid Garden

Progressive Enhancement Demo

Leverage the power of CSS Grid Layout!
Read this article for details on how to use CSS Grid Layout today.

Four levels of enhancement

No Positioning

Floated Layout

Flexbox Enhancements

CSS Grid Layout

Browser Support

IE 10

Use Grid today!

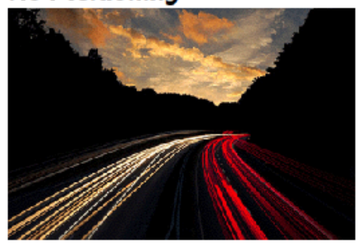
Home About this demo Gridbyexample Progressive Enhancement Grid Garden

Progressive Enhancement Demo

Leverage the power of CSS Grid Layout today!
Read this article for details on how to use CSS Grid Layout today.

Four levels of enhancement

No Positioning

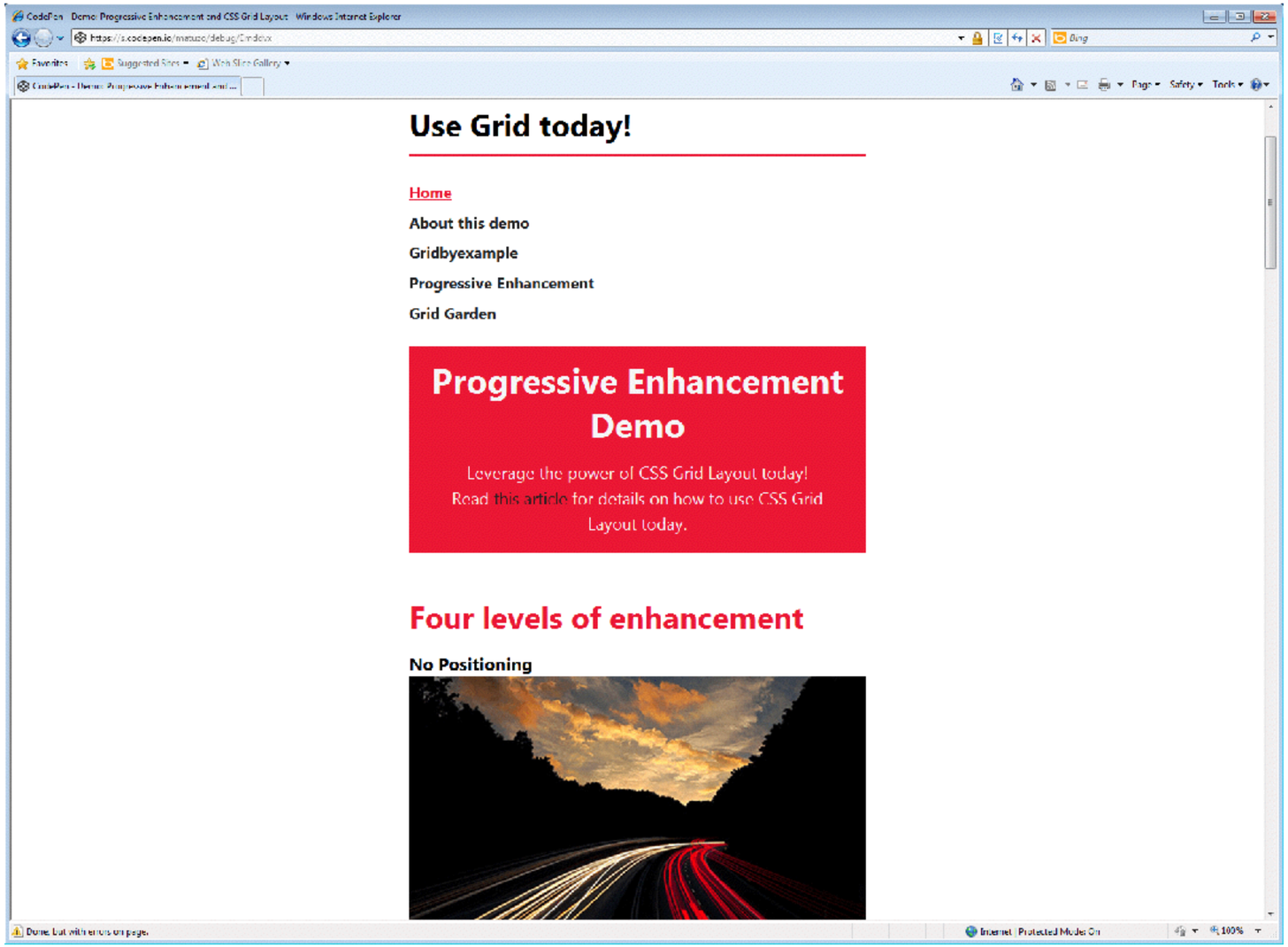


Floated Layout

Flexbox Enhancements

CSS Grid Layout

IE 9


CodePen Demo: Progressive Enhancement and CSS Grid Layout Windows Internet Explorer
Use Grid today!
Home
About this demo
Gridbyexample
Progressive Enhancement
Grid Garden
Progressive Enhancement Demo
Leverage the power of CSS Grid Layout today! Read this article for details on how to use CSS Grid Layout today.
Four levels of enhancement
No Positioning

IE 8

Progressive Enhancement

- Use vendor prefixes only sparingly, if ever.
- Grid is build in a way to coexist with other properties.
- `@support() {}` works in all browsers that support grid.
- It’s not necessary to create multiple layouts.
- Please! Don’t use poly fills.

Grid properties in IE10 - Edge 15

Level 1 Property	IE 10 Implementation
grid-template-columns	-ms-grid-columns
grid-template-rows	-ms-grid-rows
grid-template-areas	X
grid-template	X
grid-auto-columns	X
grid-auto-rows	X
grid-auto-flow	X

Grid properties in IE10 - Edge 15

Level 1 Property	IE 10 Implementation
grid	X
grid-row-start	-ms-grid-row
grid-column-start	-ms-grid-column
grid-row-end	X (-ms-grid-row-span)
grid-column-end	X (-ms-grid-column-span)
grid-row	Yes (only start value)
grid-column	Yes (only start value)

Grid properties in IE10 - Edge 15

Level 1 Property	IE 10 Implementation
grid-area	X
grid-row-gap	X
grid-column-gap	X
grid-gap	X
align-self	-ms-grid-column-align
justify-self	-ms-grid-row-align

Is prefixing grid properties worth it?

- No auto-placement
- No grid-gap
- No areas
- Box alignment not implemented

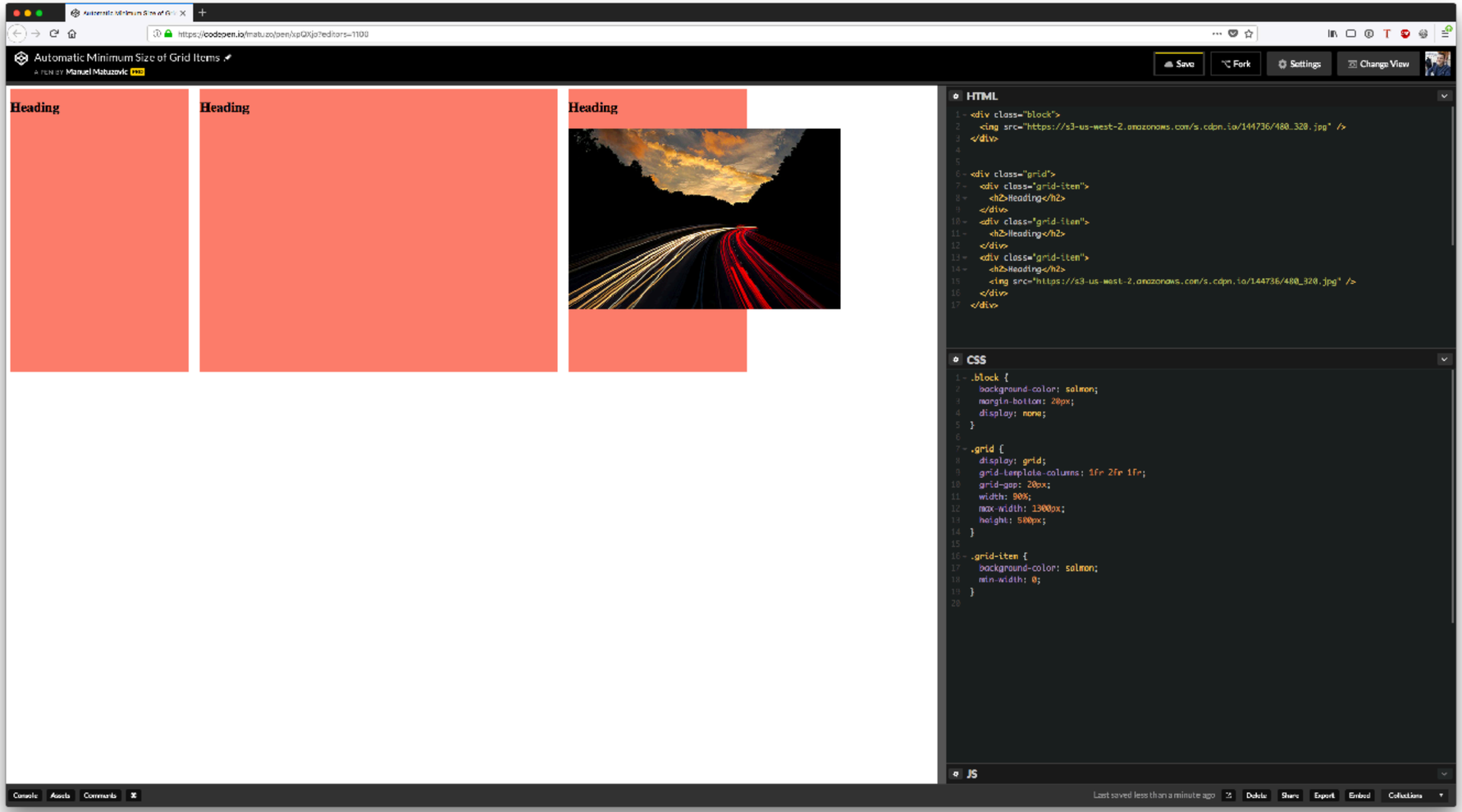
„…you can’t simply run Autoprefixer and consider the job done.“

Links

- https://rachelandrew.co.uk/css/cheatsheets/grid-fallbacks
- https://rachelandrew.co.uk/archives/2016/11/26/should-i-try-to-use-the-ie-implementation-of-css-grid-layout
- https://hacks.mozilla.org/2016/08/using-feature-queries-in-css/
- https://www.smashingmagazine.com/2017/07/enhancing-css-layout-floats-flexbox-grid

MISC

Nesting, Animation, Bugs & more



AUTOMATIC MINIMUM SIZE OF GRID ITEMS

bit.ly/grid_sizing

Automatic Minimum Size of Grid Items

- By default, a grid item cannot be smaller than the size of its content.
- Grid items have an initial size of min-width: auto and min-height: auto.
- That applies to flex items as well.
- https://codepen.io/matuzo/pen/NXEvor

Nesting

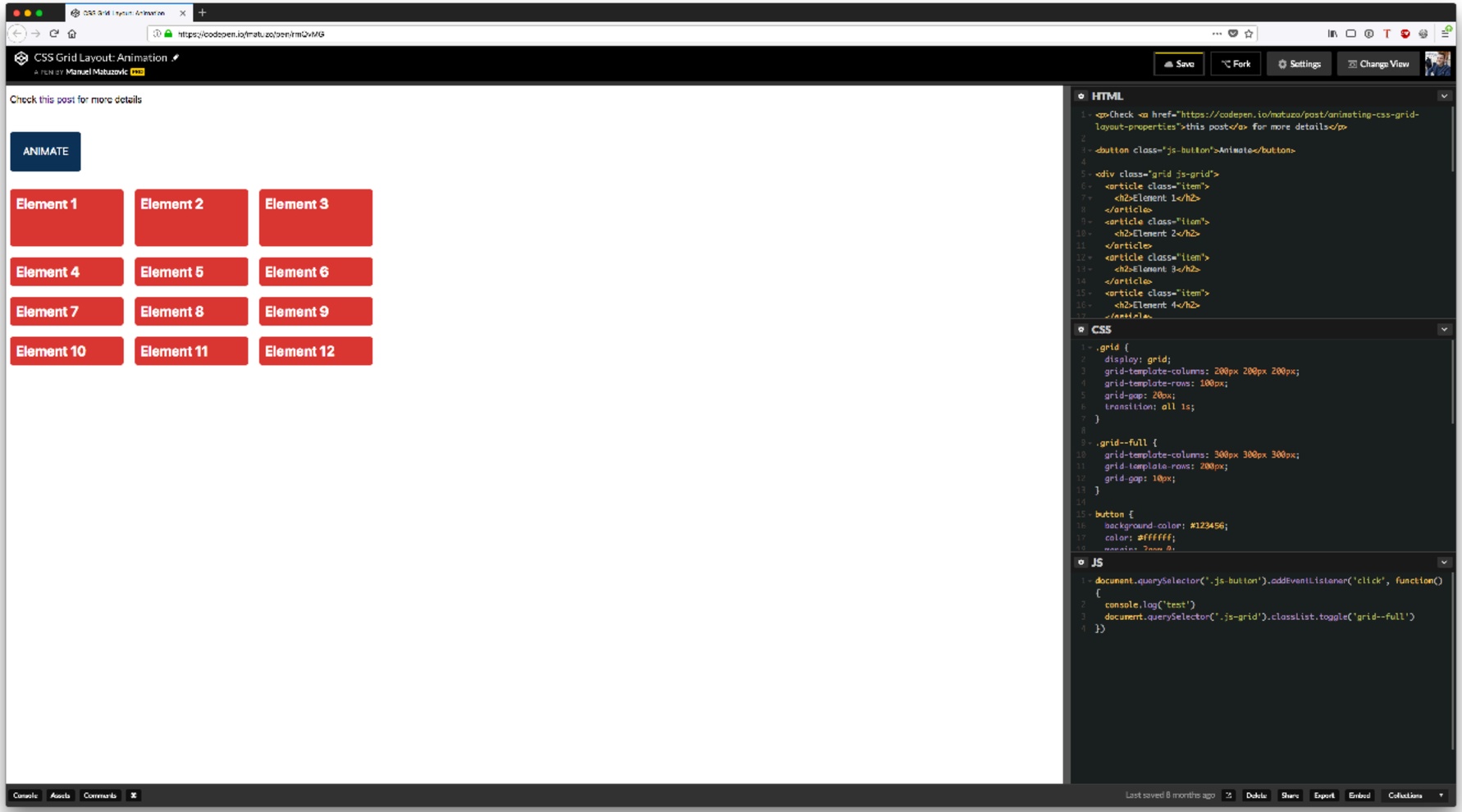
- A grid item can also be a grid container.
- Parent grid properties don’t apply for nested grid items.
- subgrid will probably come with level 2 of the specification.
- Workaround for subgrid: display: contents (http://gridbyexample.com/video/subgrid-display-contents/)

Animation

- There are 5 animatable grid properties:
  - grid-gap, grid-row-gap, grid-column-gap as length, percentage, or calc.
  - grid-template-columns, grid-template-rows as a simple list of length, percentage, or calc, provided the only differences are the values of the length, percentage, or calc components in the list.
  - Blog post about CSS Grid animation.

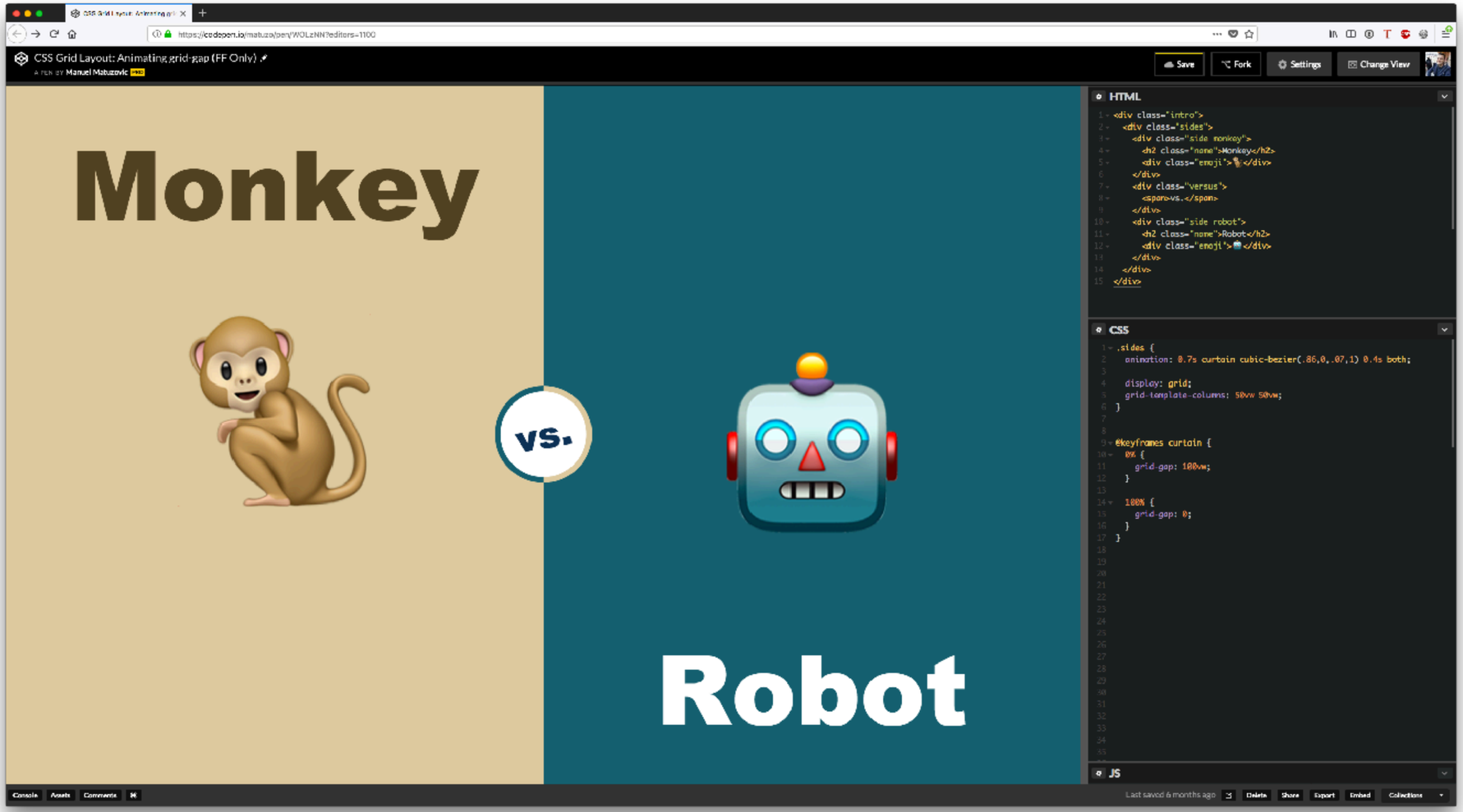
Browser	grid-gap, grid-row-gap, grid-column-gap	grid-template-columns	grid-template-rows
Firefox 55, Firefox 53 Mobile	✅	❌	❌
Safari 11.0.2	❌	❌	❌
Chrome 65	❌	❌	❌
Chrome for Android 63, Opera Mini 32	❌	❌	❌
Edge 16	✅	❌	❌

GRID PROPERTY ANIMATION BROWSER SUPPORT



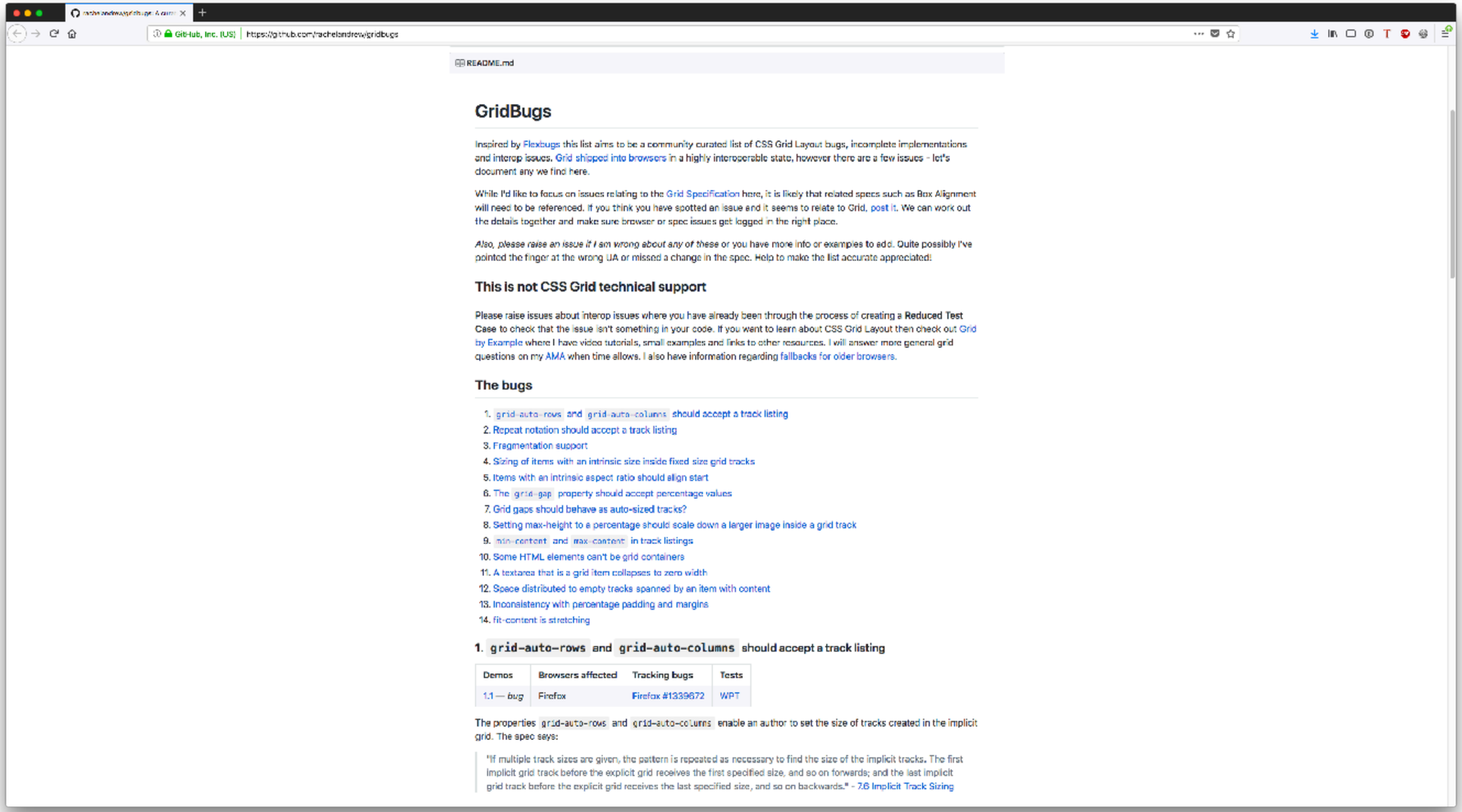
ANIMATION

bit.ly/grid_animation



ANIMATION

bit.ly/grid_animation2



README.md

GridBugs

Inspired by Flexbugs this list aims to be a community curated list of CSS Grid Layout bugs, incomplete implementations and interop issues. Grid shipped into browsers in a highly interoperable state, however there are a few issues - let's document any we find here.

While I'd like to focus on issues relating to the Grid Specification here, it is likely that related specs such as Box Alignment will need to be referenced. If you think you have spotted an issue and it seems to relate to Grid, post it. We can work out the details together and make sure browser or spec issues get logged in the right place.

Also, please raise an issue if I am wrong about any of these or you have more info or examples to add. Quite possibly I've pointed the finger at the wrong UA or missed a change in the spec. Help to make the list accurate appreciated!

This is not CSS Grid technical support

Please raise issues about interop issues where you have already been through the process of creating a **Reduced Test Case** to check that the issue isn't something in your code. If you want to learn about CSS Grid Layout then check out Grid by Example where I have video tutorials, small examples and links to other resources. I will answer more general grid questions on my AMA when time allows. I also have information regarding fallbacks for older browsers.

The bugs

1. `grid-auto-rows` and `grid-auto-columns` should accept a track listing
2. Repeat notation should accept a track listing
3. Fragmentation support
4. Sizing of items with an intrinsic size inside fixed size grid tracks
5. Items with an intrinsic aspect ratio should align start
6. The `grid-gap` property should accept percentage values
7. Grid gaps should behave as auto-sized tracks?
8. Setting max-height to a percentage should scale down a larger image inside a grid track
9. `min-content` and `max-content` in track listings
10. Some HTML elements can't be grid containers
11. A textarea that is a grid item collapses to zero width
12. Space distributed to empty tracks spanned by an item with content
13. Inconsistency with percentage padding and margins
14. fit-content is stretching

1. `grid-auto-rows` and `grid-auto-columns` should accept a track listing

Demos	Browsers affected	Tracking bugs	Tests
1.1 — *bug*	Firefox	Firefox #1339672	WPT

The properties `grid-auto-rows` and `grid-auto-columns` enable an author to set the size of tracks created in the implicit grid. The spec says:

> "If multiple track sizes are given, the pattern is repeated as necessary to find the size of the implicit tracks. The first implicit grid track before the explicit grid receives the first specified size, and so on forwards; and the last implicit grid track before the explicit grid receives the last specified size, and so on backwards." - 7.6 Implicit Track Sizing

BUGS

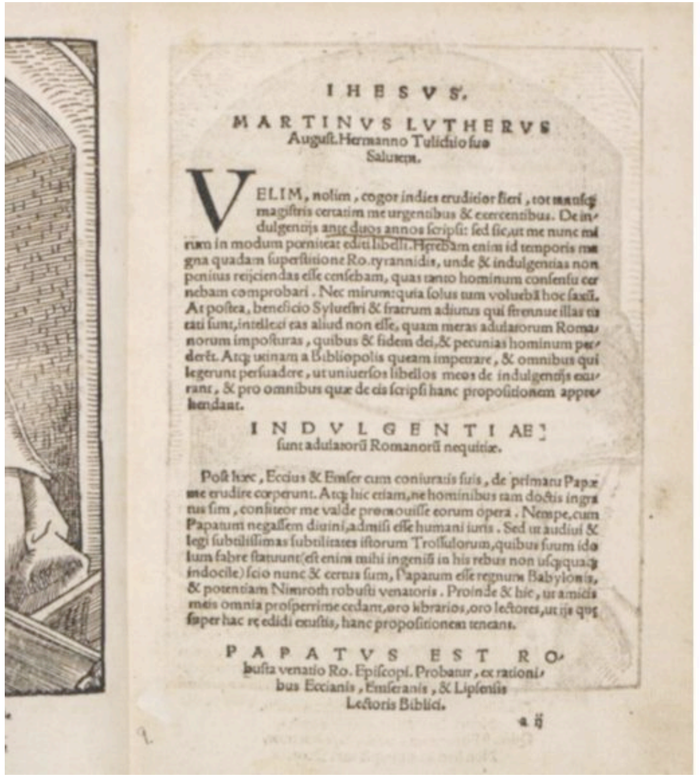
github.com/rachelandrew/gridbugs

Is there a grid system for CSS Grid Layout?

- Grid is a grid system!
- You don’t even need extra markup for defining columns or rows.
- It might make sense to use mixins in order to enforce a specific style or set of techniques.

PSEUDO ELEMENTS

bit.ly/grid_pseudo



IHESVS.

MARTINVS LVTHERVS
Augustl. Hermanno Tulichio suo
Salutem.

VELIM, nolim, cogor indies eruditior fieri, tot manus magistris certatim me urgentibus & exercentibus. De indulgentijs ante duos annos scripsi: sed sic, ut me nunc miserum in modum pœniteat editi libelli. Heredam enim id temporis magna quadam superstitione Ro. tyrannidis, unde & indulgentias non penitus reijciendas esse censebam, quas tanto hominum consensu cernebam comprobari. Nec mirum: quia solus tum volueba hoc fassi. At postea, beneficio Syluestri & fratrum adiutus qui strenue illas traiecti sunt, intellexi eas aliud non esse, quam meras adulatorum Romanorum impostoras, quibus & fidem dei, & pecunias hominum perderet. Atq; utinam a Bibliopolis queam imperare, & omnibus qui legerunt persuadere, ut uniuersos libellos meos de indulgentijs exurant, & pro omnibus quæ de eis scripsi hanc propositionem apprehendant.

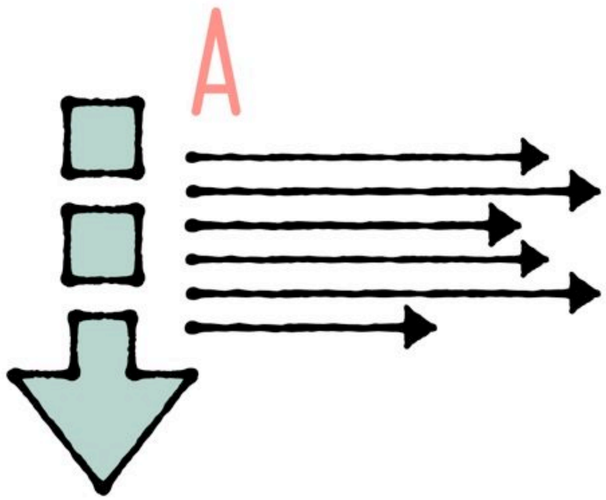
INDVLGENTIAE
sunt adulatorū Romanorū nequitiæ.

Post hæc, Eccius & Emser cum coniuratis suis, de primatu Papæ me erudire cœperunt. Atq; hic etiam, ne hominibus tam doctis ingratus sim, confiteor me valde promouisse eorum opera. Nempe, cum Papatum negassem diuini, admisi esse humani iuris. Sed ut audiui & legi subtilissimas subtilitates istorum Trossulorum, quibus suum idolum fabre statuunt (est enim mihi ingenium in his rebus non usq; quaq; indocile) scio nunc & cernis sum, Papatum esse regnum Babyloniæ, & potentiam Nimroth robusti venatoris. Proinde & hic, ut amicis meis omnia prosperrime cedant, oro librarios, oro lectores, ut ijs qui super hac re edidi exustis, hanc propositionem teneant.

PAPATVS EST RO-
busta venatio Ro. Episcopi. Probatur, ex rationibus Eccianis, Emseranis, & Lipsensis
Lectoris Biblici.

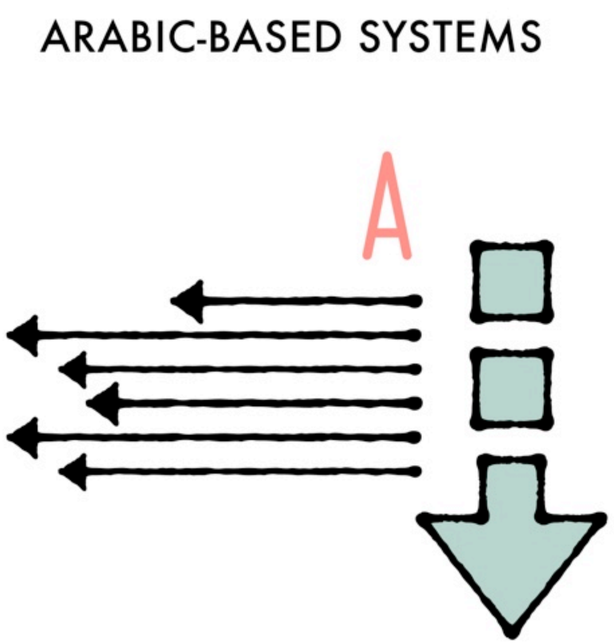
a ij

LATIN-BASED SYSTEMS



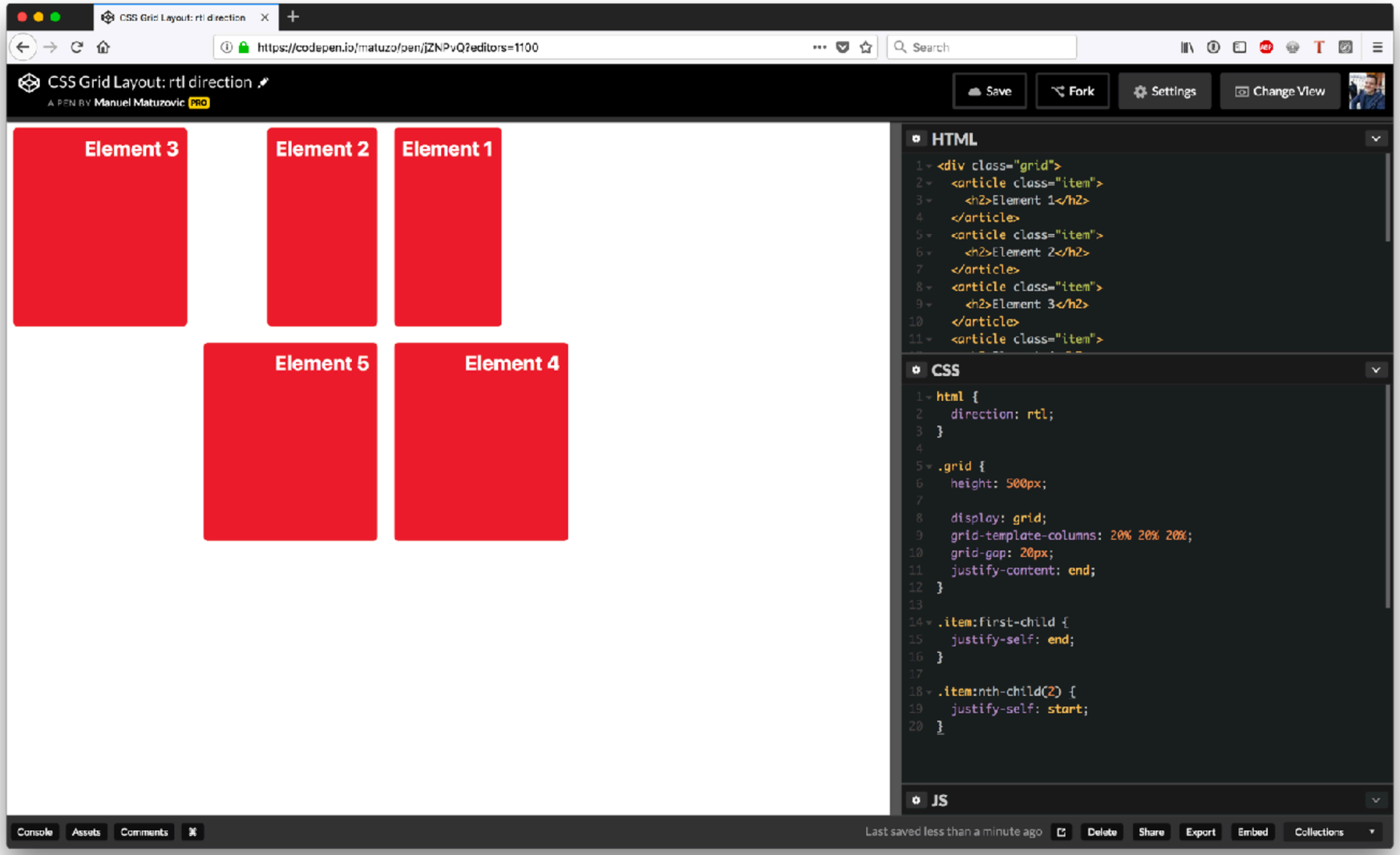
WRITING MODE: LATIN-BASED SYSTEMS

24ways.org/2016/css-writing-modes



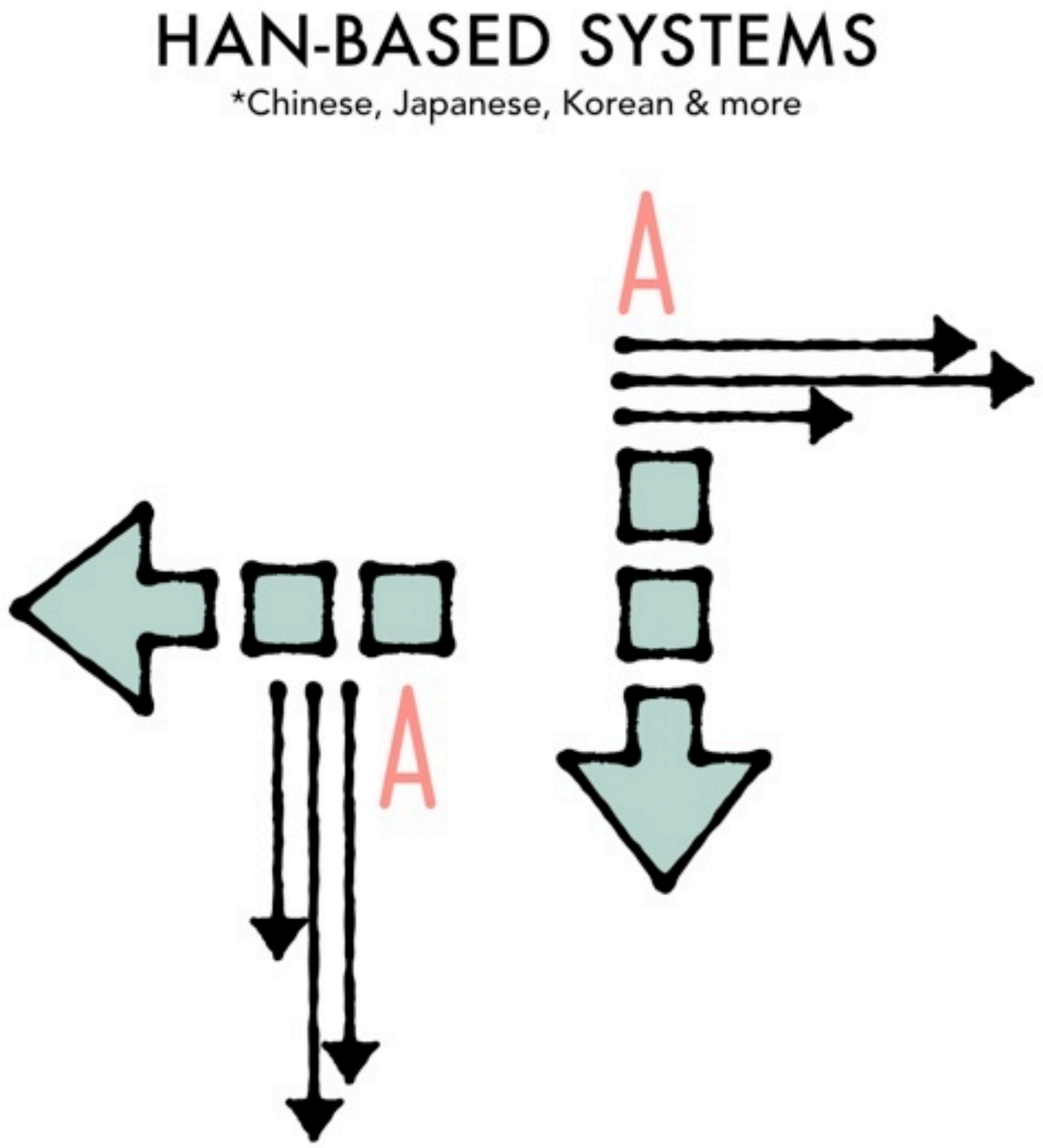
WRITING MODE: ARABIC-BASED SYSTEMS

24ways.org/2016/css-writing-modes



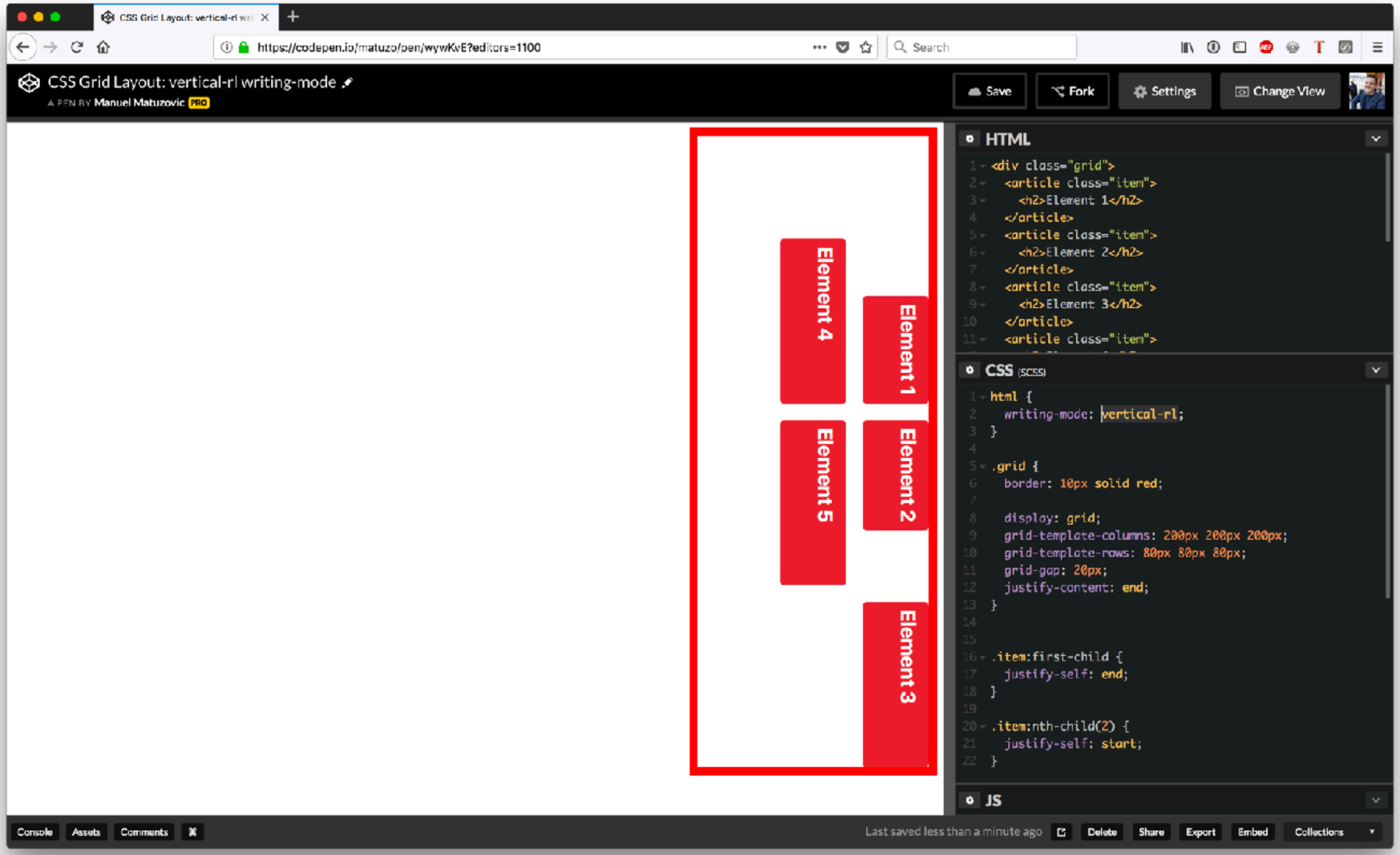
WRITING MODES & CSS GRID

bit.ly/grid_rtl



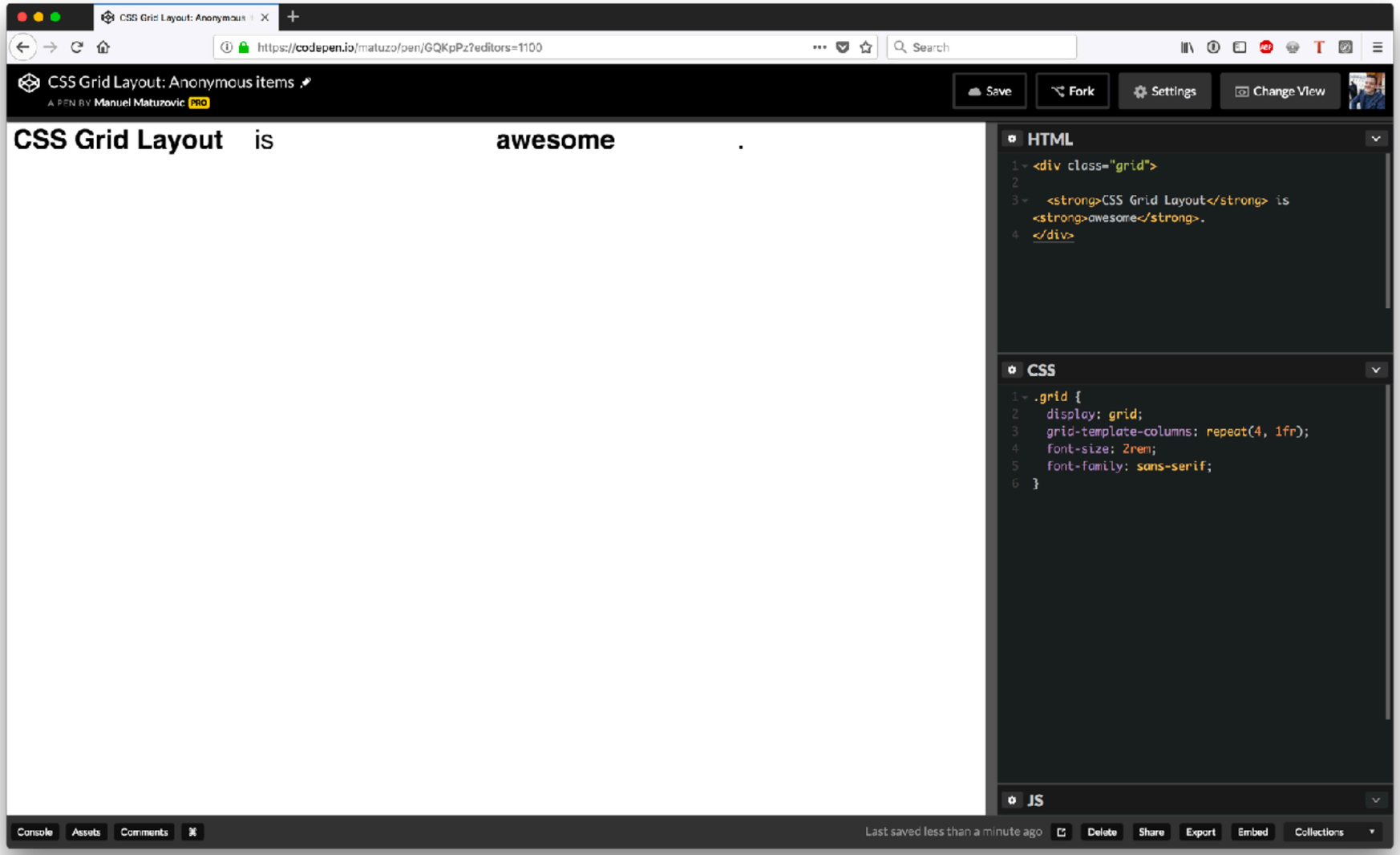
WRITING MODE: HAN-BASED SYSTEMS

24ways.org/2016/css-writing-modes



WRITING MODES & CSS GRID

bit.ly/grid_rl



ANONYMOUS ITEMS

bit.ly/grid_anon

WHAT'S NEXT?

Nesting, Debugging, RWD

Gaps

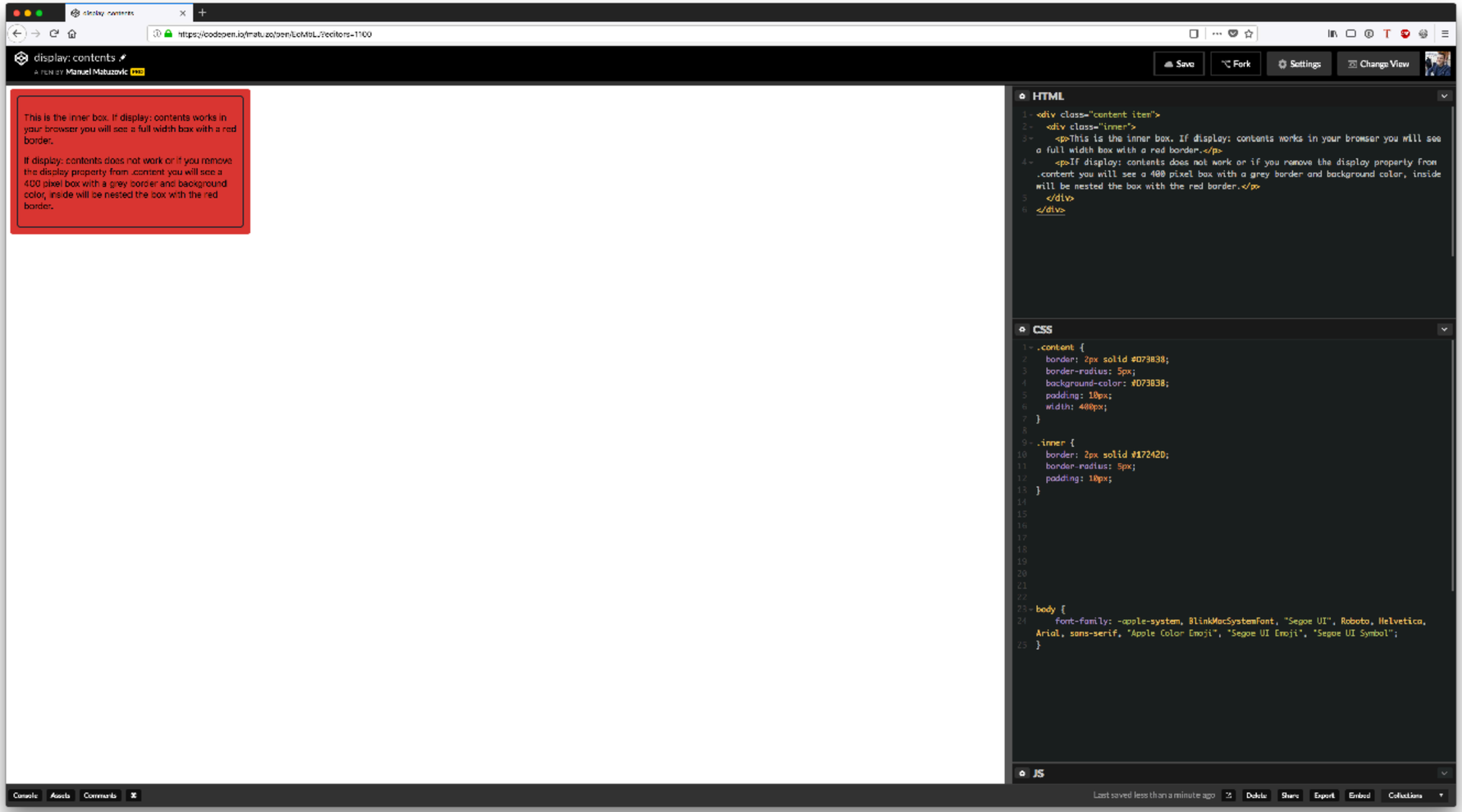
- grid-column-gap, grid-row-gap and grid-gap will be renamed to column-gap, row-gap and a gap.
- It is likely that browsers will alias the old names to the new for the foreseeable future.
- The new gap properties are going to be part of the Box Alignment specification.
- They’re not limited to Grid. Available to other layout modes where they make sense (Flexbox!).

Subgrid

- Only direct children of a grid container become grid items. Children of grid items are no affected.
- It’s possible to nest grids, but the child grids have no relationship to the parent.
- `subgrid` has been deferred to Level 2 due to lack of implementation and desire for further discussion.

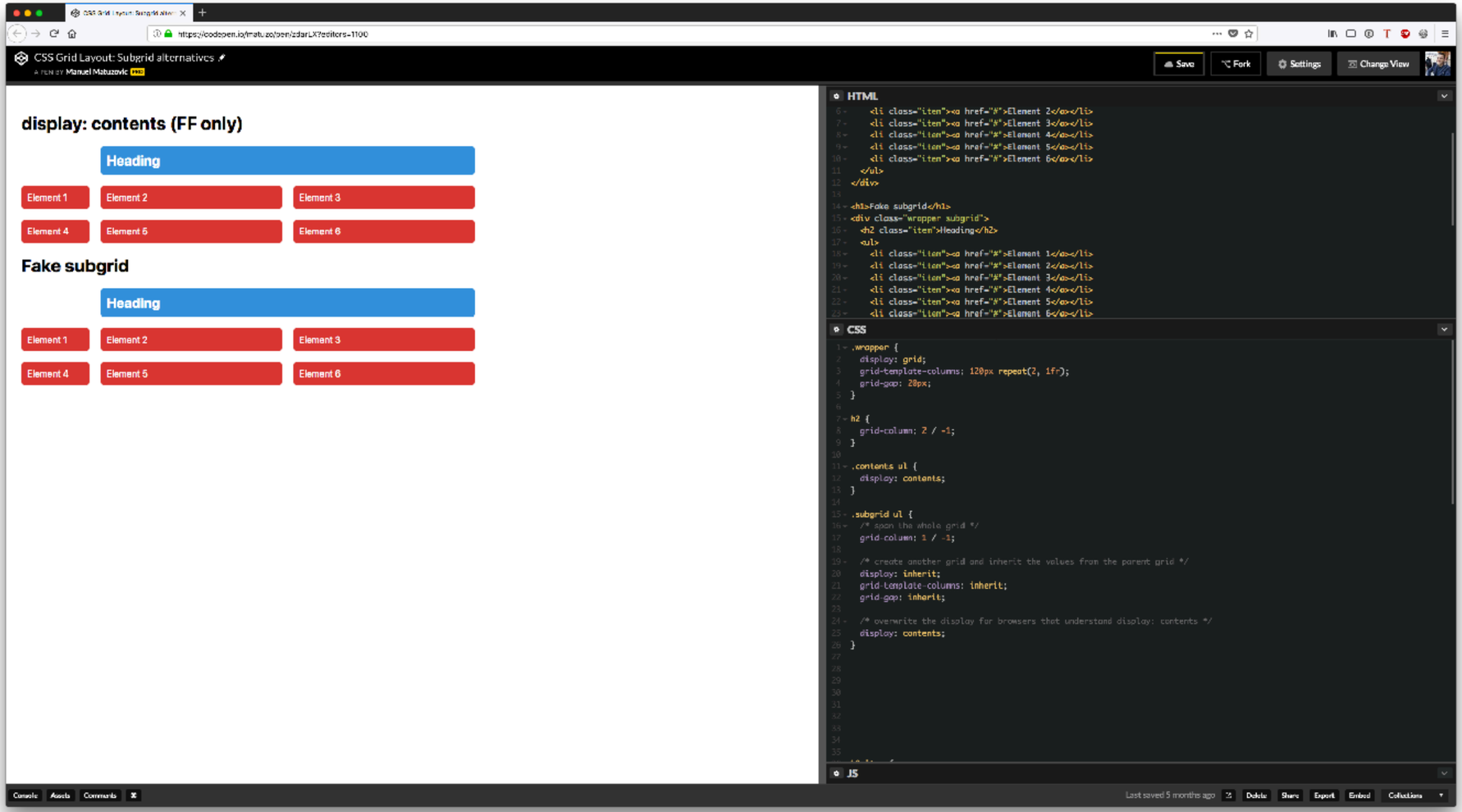
display:contents

- The element itself does not generate any boxes, but its children and pseudo-elements still generate boxes as normal.
- Behaves as if it had been replaced with its children. Child elements take up it’s place in the document tree.
- Could be used in order to simulate display: subgrid.



DISPLAY: CONTENTS

bit.ly/grid_contents2



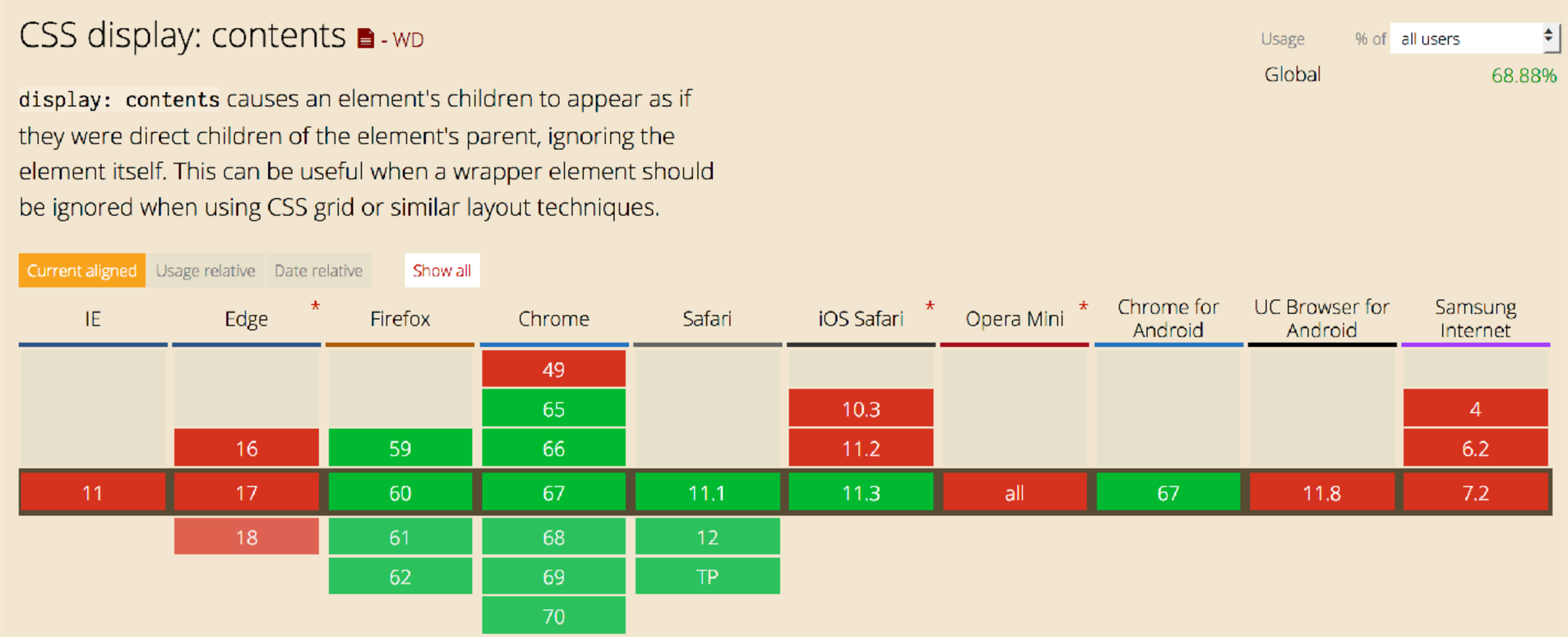
DISPLAY: CONTENTS

bit.ly/grid_contents1

Subgrid with display:contents?

- Issue 1: You cannot apply backgrounds and borders to an item with display: contents.
- Issue 2: Items behave as if they were direct child items, they don't become direct child items. That's why direct child item selectors wouldn't work.
- Issue 3: You lose auto-placement of the parent grid item, which could be undesirable.

Browsersupport



display: contents is enabled by default in Blink and WebKit and it will be probably shipped in Chrome 65 and Safari 11.1.

Masonry Layout ???

- Grid was designed for the creation of two dimensional grids, which a Masonry Layout isn't.
- Masonry Layouts with Grid are currently only possible with nasty hacks and fixed heights.
- There's an issue in the unofficial level 2 draft which suggests to at least consider implementing a Masonry feature.

Styling cells, tracks and areas

- Styling lines, cells, tracks and areas is currently not possible.
- It would be nice, though.
- Issue on Github.

More

- Creating non-rectangular grid areas.
- Flowing content through grid cells or areas.
- Solving the keyboard/layout disconnect.
- Read What next for CSS Grid Layout? by Rachel for more details.

WRAP-UP

Wrap-UP

- Use Grid now (when it makes sense). It has been developed for 5 years and it’s implemented by all browser vendors. It’s not the same disaster as Flexbox.
- Use Grid for the over all page layout as well as on a component level.
- Use it for two-dimensional layouting.
- Leverage the power of progressive enhancement.
- Make sure your grids are accessible.

Wrap-UP

- Explicit grids. (grid-template-rows and grid-template-columns)
- Implicit grids (grid-auto-rows and grid-auto-columns)
- Flow (column, row, dense)
- Lines (grid-column, grid-row, span keyword)
- Cells
- Tracks (minmax, repeat, auto-fill, auto-fit)
- Areas (layouts, magic lines, magic areas)

Wrap-UP

- Box Alignment Module (justify-items, align-items, justify-self, align-self, justify-content, align-content, grid-gap (gap)).
- MQ breakpoints vs. auto-fill and auto-fit.
- Animation.
- Level 2 Spec.

Links

- http://gridbyexample.com/
- https://css-tricks.com/snippets/css/complete-guide-grid/
- https://developer.mozilla.org/en/docs/Web/CSS/grid
- https://www.w3.org/TR/css-grid-1/
- https://css-tricks.com/getting-started-css-grid/
- https://css-tricks.com/collection-interesting-facts-css-grid-layout/
- https://css-tricks.com/difference-explicit-implicit-grids/