Information Technology 3

Grid Layout Revisited

Kristof Michiels

Revisiting?

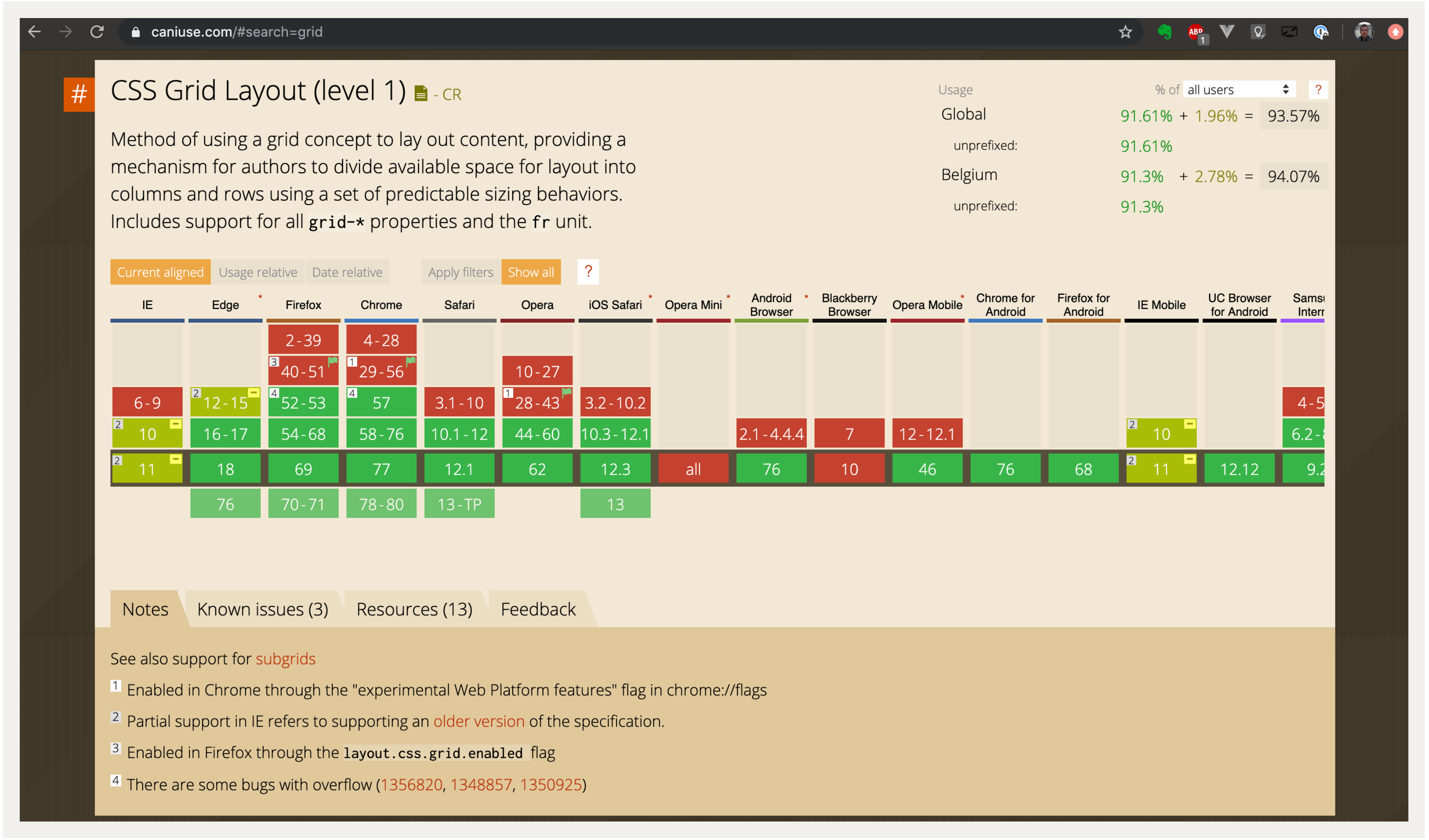
Revisiting? Yes!

- Yes, because mastering grid layout is one of the most important aspects of CSS
- Yes, because specification is being updated to address some of the limitations that existed in the first version of Grid
- Yes, because expert knowledge of Grid Layout will help you ace your IT3 exam

A short history of CSS Grid

- CSS Grid arrived in 2017 with massive browser support
- Much anticipated after it first appeared in 2012 in Internet Explorer alone
- Designers could now dump floating as a layout technique. Lucky them, lucky you :-)

Support has risen to more than 94%


caniuse.com/#search=grid

CSS Grid Layout (level 1) - CR

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

Usage — % of all users

Global	91.61% + 1.96% =	93.57%
unprefixed:	91.61%	
Belgium	91.3% + 2.78% =	94.07%
unprefixed:	91.3%	

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Sams Interr
		🟥 2-39	🟥 4-28												
		🟥 [3] 40-51	🟥 [1] 29-56		🟥 10-27										
🟥 6-9	🟨 [2] 12-15	🟩 [4] 52-53	🟩 [4] 57	🟥 3.1-10	🟥 [1] 28-43	🟥 3.2-10.2									🟥 4-5
🟨 [2] 10	🟩 16-17	🟩 54-68	🟩 58-76	🟩 10.1-12	🟩 44-60	🟩 10.3-12.1		🟥 2.1-4.4.4	🟥 7	🟥 12-12.1			🟨 [2] 10		🟩 6.2-
🟨 [2] 11	🟩 18	🟩 69	🟩 77	🟩 12.1	🟩 62	🟩 12.3	🟥 all	🟩 76	🟥 10	🟩 46	🟩 76	🟩 68	🟨 [2] 11	🟩 12.12	🟩 9.2
	🟩 76	🟩 70-71	🟩 78-80	🟩 13-TP		🟩 13									

Notes | Known issues (3) | Resources (13) | Feedback

See also support for subgrids

[1] Enabled in Chrome through the "experimental Web Platform features" flag in chrome://flags

[2] Partial support in IE refers to supporting an older version of the specification.

[3] Enabled in Firefox through the `layout.css.grid.enabled` flag

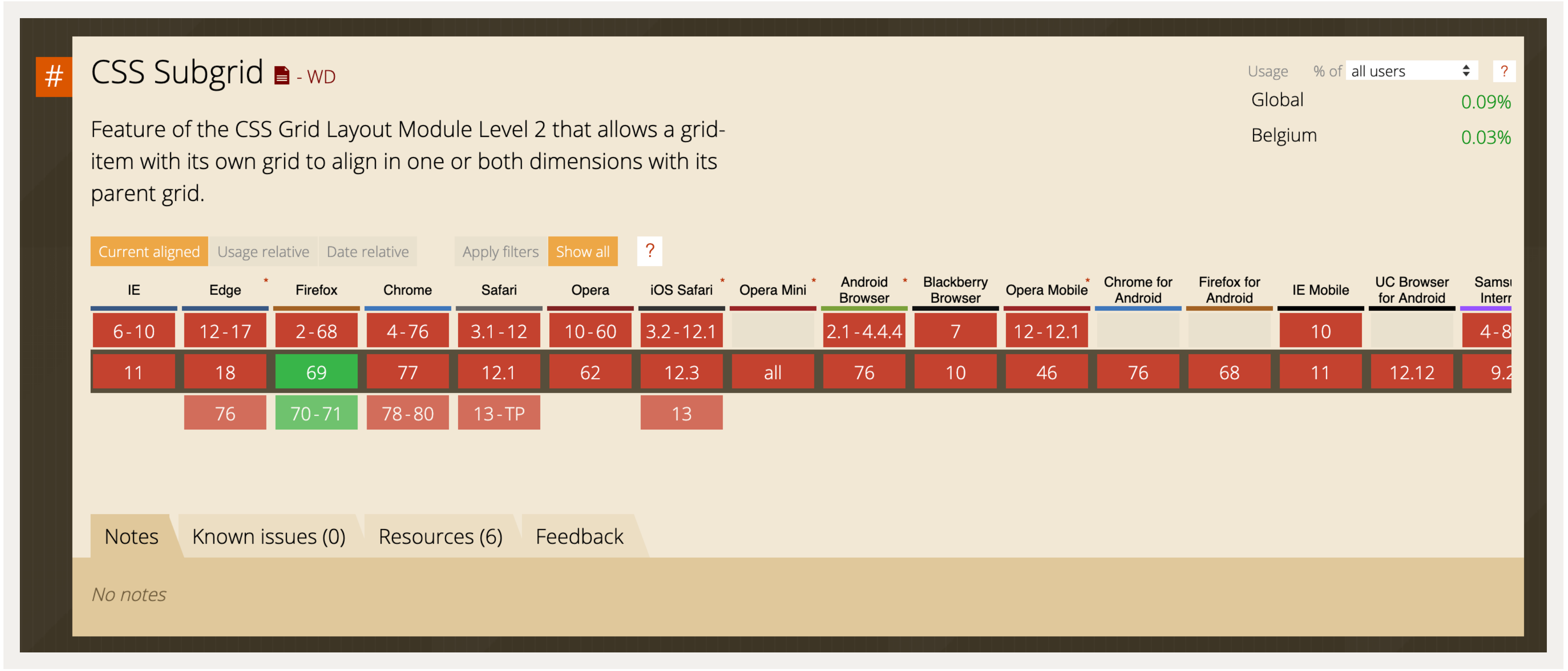
[4] There are some bugs with overflow (1356820, 1348857, 1350925)

https://caniuse.com/#search=grid

Support has risen to more than 94%

- Was 90% at the end of 2018 when you learned to work with grid
- Coming from 83% at the end of 2017
- This means we can now finally drop @supports (grid-area: auto) {}
- This means you can start using grid for the mobile version of your site

Level 2 of specification being implemented in Firefox



# CSS Subgrid - WD

Feature of the CSS Grid Layout Module Level 2 that allows a grid-item with its own grid to align in one or both dimensions with its parent grid.

Usage % of all users ?

Global	0.09%
Belgium	0.03%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Blackberry Browser	Opera Mobile *	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Samsı Interr
6-10	12-17	2-68	4-76	3.1-12	10-60	3.2-12.1		2.1-4.4.4	7	12-12.1			10		4-8
11	18	69	77	12.1	62	12.3	all	76	10	46	76	68	11	12.12	9.2
	76	70-71	78-80	13-TP		13									

Notes | Known issues (0) | Resources (6) | Feedback

No notes

https://caniuse.com/#search=grid

Level 2 of specification being implemented in Firefox

- E.g. subgrids currently only marginally supported
- We'll leave it untouched for now
- Adoption might rise quickly in the near future
- I'll keep you posted :-)

What again is Grid Layout?

Grid Layout?

- The CSS grid layout Module defines a two-dimensional grid layout system
- Once a grid has been established on a containing element, the direct children of that element can be placed into a flexible or fixed layout grid
- The grid can be redefined using media queries
- Grid is a very flexible module, so there are a number of ways to use it

Grid Basics

Defining a grid

A grid is defined using a new value of the display property, display: grid

```
<div class="wrapper">
    <div class="box a">A</div>
    <div class="box b">B</div>
    <div class="box c">C</div>
    <div class="box d">A</div>
    <div class="box e">B</div>
    <div class="box f">C</div>
</div>
```

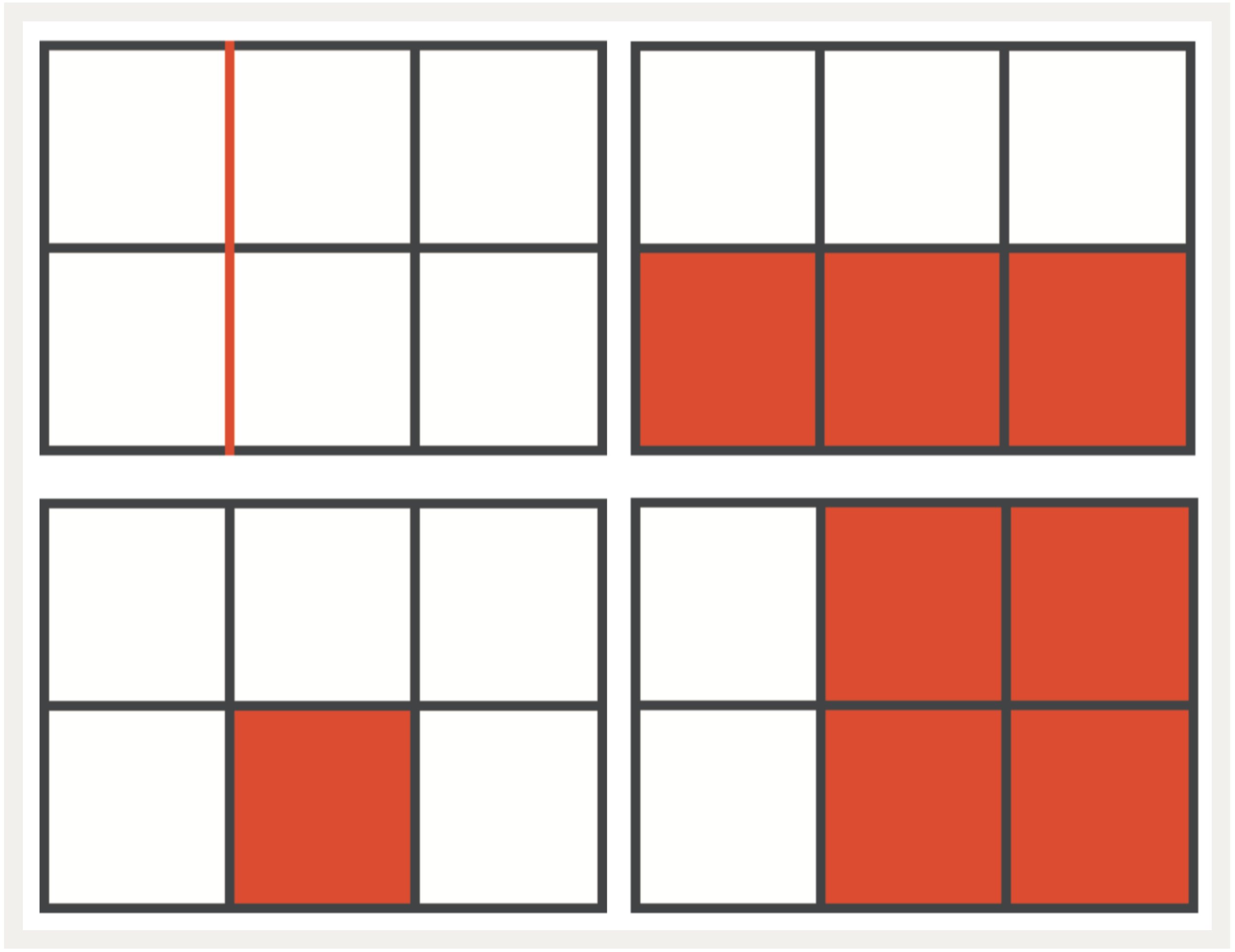
```
.wrapper {
    display: grid;
}
```

Describing rows and columns

- Grids have rows and columns, which the CSS Grid Layout Module gives us properties to describe
- Child elements place themselves automatically on the grid according to Grid's auto-placement rules
- These simply fill each cell in turn with a direct child of the grid container

```css
.wrapper {
    display: grid;
    grid-template-columns: 10rem 10rem 10rem;
    grid-template-rows: 10rem 10rem;
}
```

Terminology


Clockwise: column line 2, track between row lines 2 and 3, a grid cell, a grid area

The explicit and the implicit grid

- In the example we created an explicit grid: the child elements automatically slotted into the cells created by those grid tracks
- If we don’t create enough cells, or place something outside of the explicit grid, Grid creates implicit grid tracks for us
- This means that we can remove the grid-template-rows property, and the items will still place themselves on the grid
- The rows are then auto-sized. An auto-sized track will be large enough to fit the content.

Gaps between grid tracks

We can do this by using the gap property, or individual properties of column-gap and row-gap

```
.wrapper {
    display: grid;
    grid-template-columns: 10rem 10rem 10rem;
    gap: 10rem;
}
```

Grid extra's

Sizing implicit rows

Auto-sized is the initial value of implicit tracks. We can however specify a size for them using the grid-auto-rows and grid-auto-columns properties.

```
.wrapper {
    display: grid;
    grid-template-columns: 10rem 10rem 10rem;
    grid-auto-rows: 10rem;
    gap: 10rem;
}
```

The minmax() function

In the previous example we could run into a situation where the content overflows the fixed-height track. We can achieve more flexibility by using the minmax() function for our track sizing. It allows us to pass in a minimum and a maximum value.

```css
.wrapper {
    display: grid;
    grid-template-columns: 10rem 10rem 10rem;
    grid-auto-rows:minmax(10rem, auto);
    gap: 10rem;
}
```

The fr unit

- Tracks can be created with any valid CSS length unit, or with percentages
- You can also size tracks using the Grid-specific fr unit
- This value represents a fraction of the available space in the grid container

```
.wrapper {
    width: 60rem;
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    grid-auto-rows:minmax(10rem, auto);
    gap: 10rem;
}
```

The repeat() notation

- With repeat(), we place comma-separated values between parentheses
- The value before the comma stands for the number of times a pattern should repeat
- The value after the comma refers to the pattern
- We can repeat a single track value or a track listing

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

The auto-fill keyword

- Say we want to have as many tracks (with a minimum size) as will fit into our grid container
- This enables a responsive number of column tracks without relying on media queries to add breakpoints
- This is achieved by using the auto-fill keyword instead of a number before the comma in our repeat notation

```css
.wrapper {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(20rem, 1fr));
}
```

Grid items by line number

Placing grid items by line number

- In the previous example we auto-placed the grid items
- We can of course place them manually using css
- The simplest method to use is line-based placement

```css
.a {
    grid-row: 1 / 3;
    grid-column: 1 / 3;
}
```

Overlapping items on the grid

- When placing items using lines, we can place an item into the same cell as another item
- Items that are lower in the source display on top of items that come before them
- we can use the z-index property to change the stacking order of items

Grid template areas

Positioning using grid template areas

Here we create named grid areas and use grid-template-areas to describe where on the grid they sit

```css
.a {grid-area: area-a;}
.b {grid-area: area-b;}
.c {grid-area: area-c;}
.d {grid-area: area-d;}
.e {grid-area: area-e;}
.f {grid-area: area-f;}

.wrapper {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-areas:
        "area-a area-b area-c"
        "area-d area-e area-f";
}
```

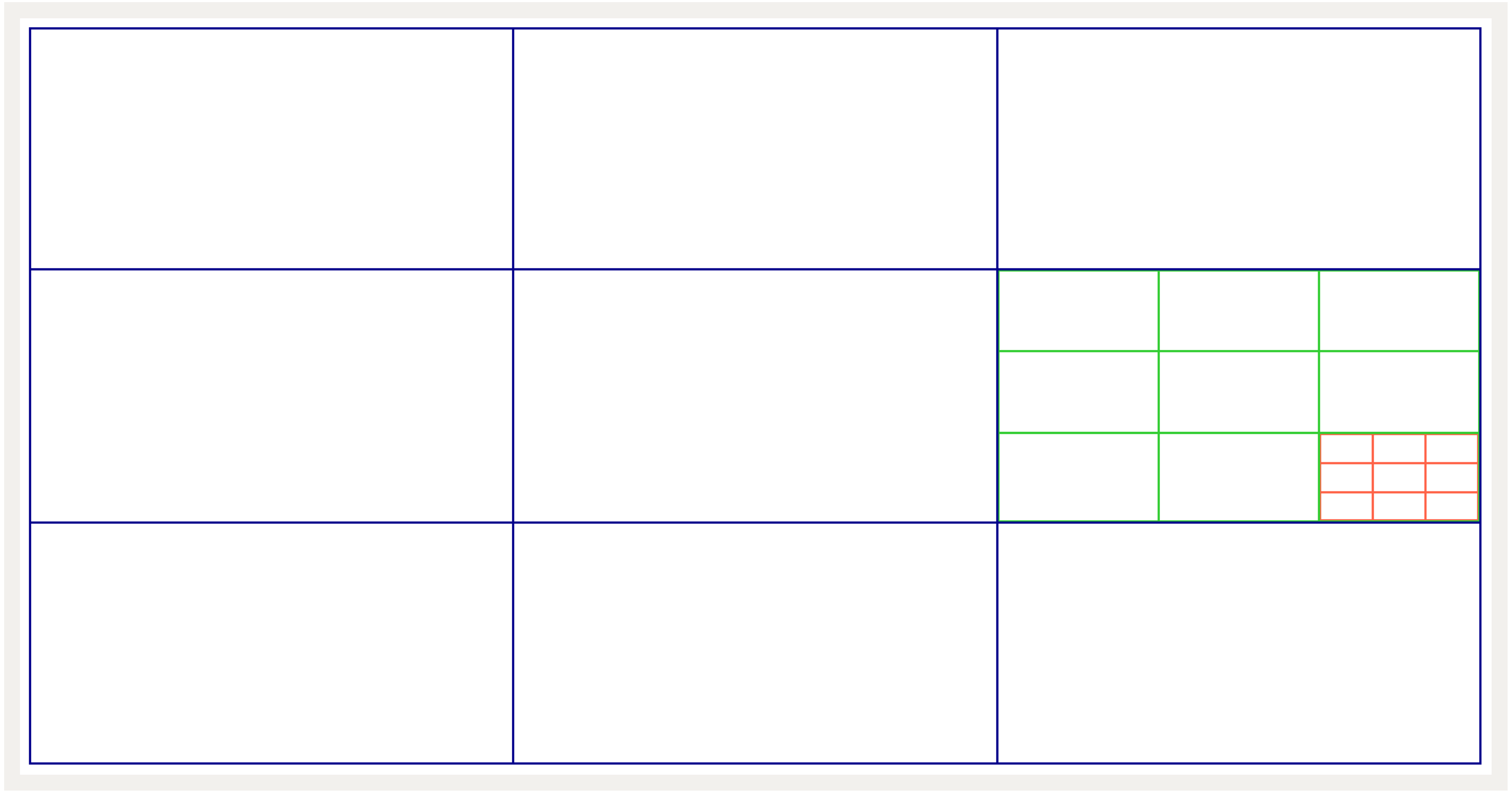
Positioning using grid template areas

Another possibility:

```
.wrapper {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-areas:
        "area-a area-a area-a"
        ".      area-b area-b"
        ".      area-c area-d"
        "area-f area-f area-f"
        "area-e area-e area-e";
}
```

Grids in grids?

Grids within grids



https://codepen.io/admkrm/full/EbREvK

Grids within grids

- This is an important one
- Understanding this 100% will enable you to look at a design and immediately see where you need one or more grids
- Each web page may contain any number of grids
- It is also possible that a grid element can become a grid in its own right: these grids we call nested grids
- When designing a page: develop your ideas on paper first and determine where you will apply your grids
- Don’t take the one mother-of-all-grids approach: you don’t need this kind of complexity

Exercise

Exercise

- Search the web for at least 12 beautiful web sites with interesting layouts
- Tip: search on dribbble.com
- Take screenshots of these 12 examples
- Build a grid based homepage called "Beautiful grids"
- Make it a credible site, with a real-world look and feel
- Use Grid Layout to lay out the screenshots with their full references
- Site is responsive, designed mobile first
- Bonus points for using the new features
- Deadline = 2 weeks

IT3 - Grid Layout Revisited