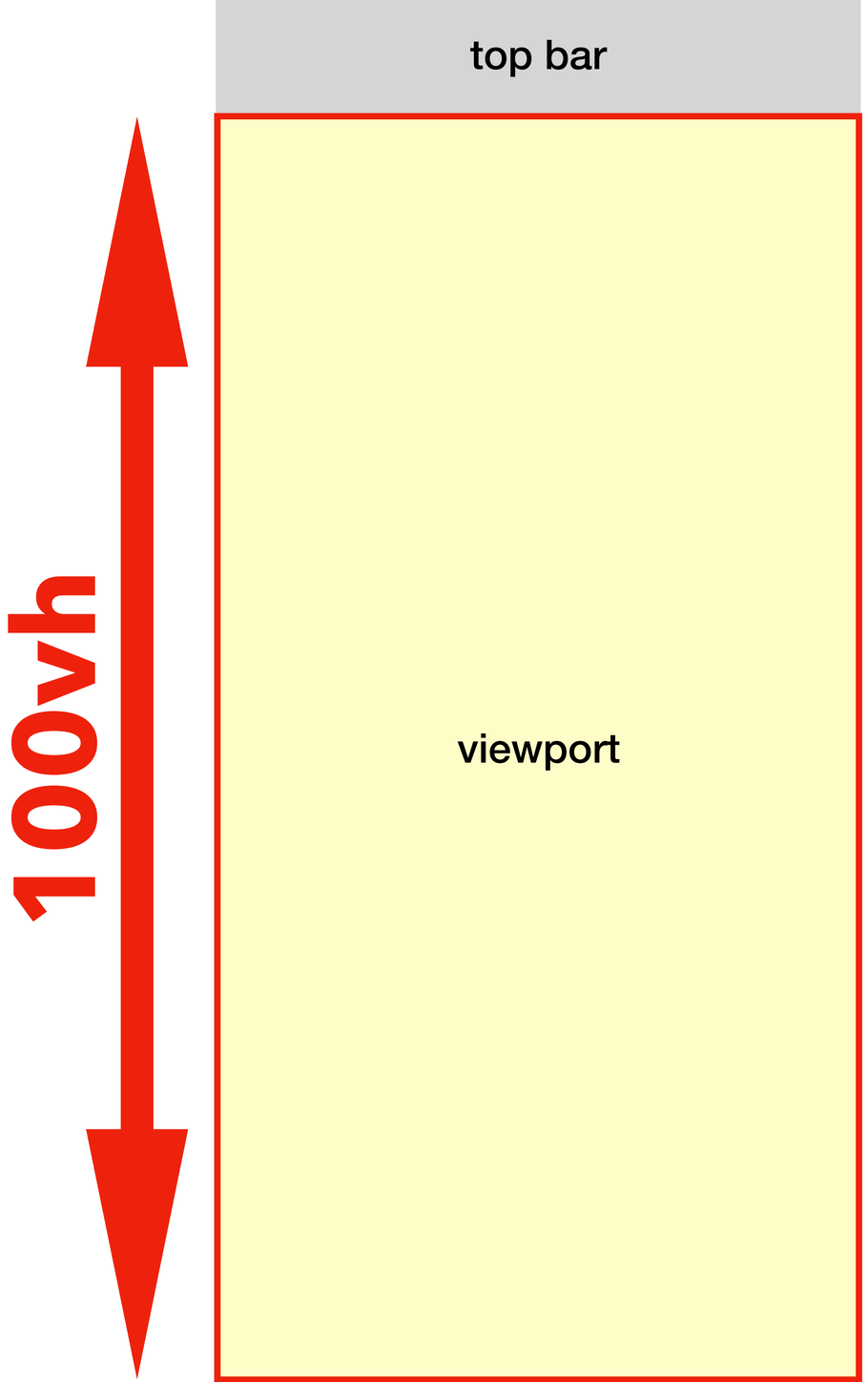
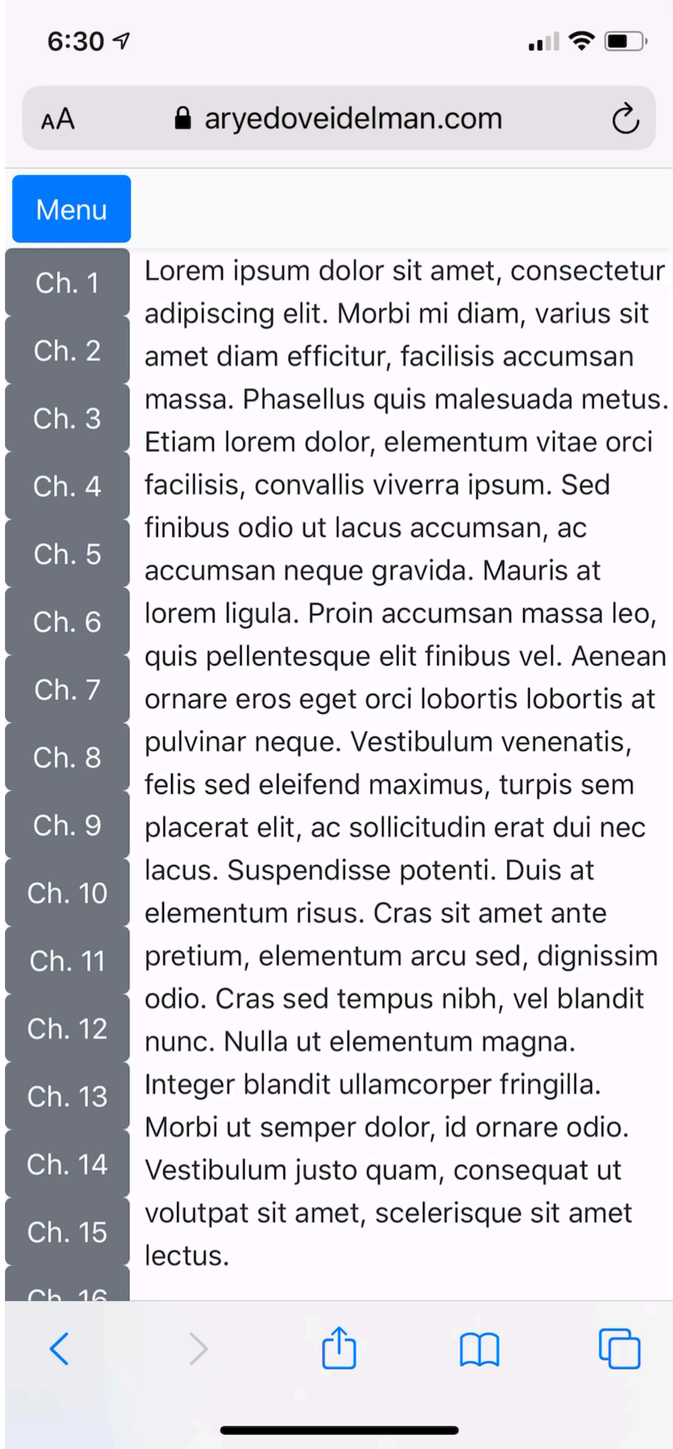
CSSWG Meeting
Jan 2020, Spain

VH Units

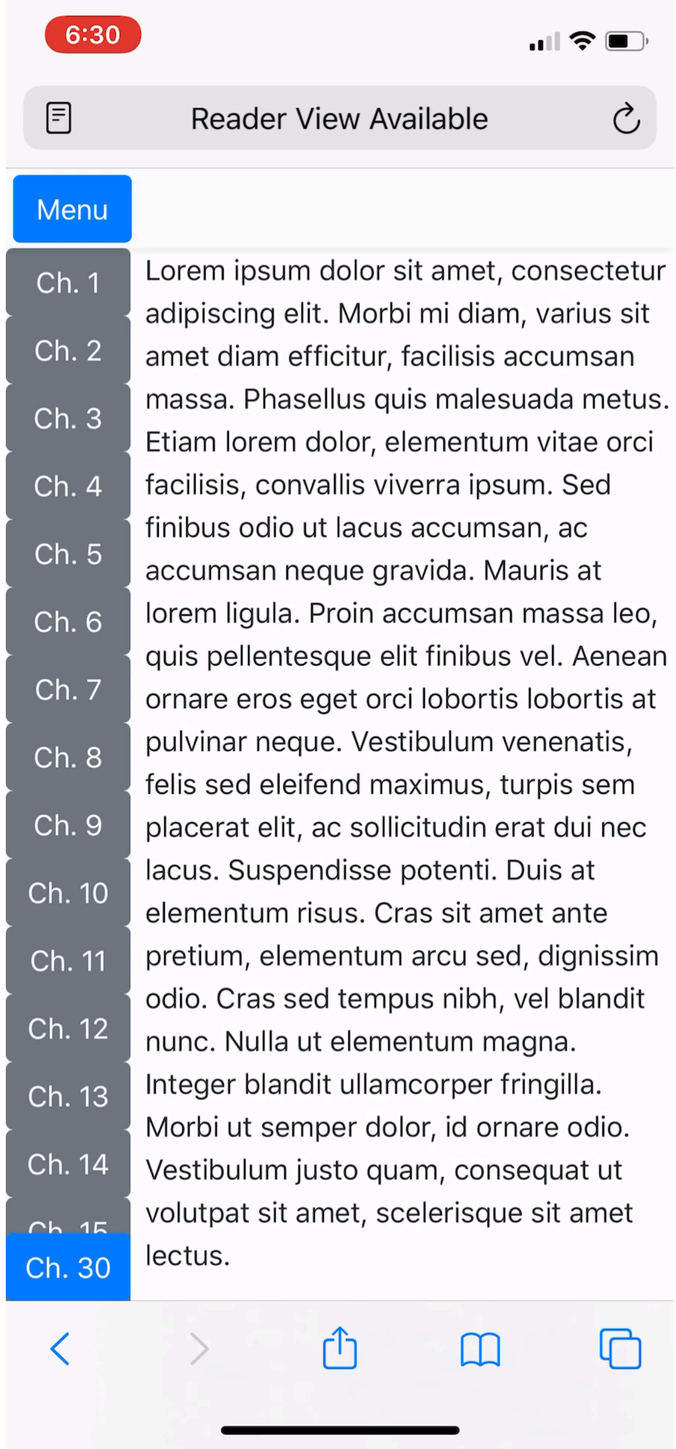

6:34
AA
cdpn.io
0
25
50
75
100
125
150
175
200
225
250
275
300
325
350
375
400
425
450
475
500
525
550
575
600
625


top bar
100vh
viewport

Lossy



Lossless



top bar

vhc

content

bottom bar

top bar

vh

content

top bar

content

controls

dynamic variable of current viewport size
(let any keyboard/controls go overtop me)

`env(foo)`

dynami
of what
(subtrac
any key

Things…

- We already tried not having interop.
- Implementers tried making everything magically shrink & grow – it had performance problems.
  And Authors don’t always want that.
- Developers currently try to solve this with JS.
- 100vh, 100% body, window.innerHeight don’t match.
  Fixed positioning to visual viewport, not layout viewport.
- May want to apply same solution to VW & scrollbars.

Assumptions

- We can’t redefine the current VH unit.
- Authors have a legit use for all the possibilities.
- Best solution puts the burden, and choice, on Authors.
- Fixed units for all 3 measurements, animatable.

Proposals

- VH unit = equal to 1% of the height of the initial containing block with user agent chrome minimized.
- VHC unit = equal to 1% of the height of the initial containing block with user agent chrome maximized.

- env(inset-collapsable-height).
- `height: calc(100vh - env(inset-collapsable-height, 0px));`

Custom Origins

[css-cascade] Custom cascade origins #4470

New issue

Open mirisuzanne opened this issue on Oct 29, 2019 · 1 comment

mirisuzanne commented on Oct 29, 2019 • edited

This relates to the Cascade Specification, along with a number of "specificity" concerns and proposals (such as #2272 & #3890 & the `:where()` selector).

Much of my work with design systems has revolved around helping companies define layers of abstraction: building tokens, then defaults, then patterns, components etc. That's a common approach, whether we call it OOCSS or Atomic Design or ITCSS or something else. In order to do that, we often have to be very careful with matching specificity to layer – so components override patterns, and so on – and third-party tools can easily break a delicate balance.

It strikes me that cascading origins & `!important` are designed to solve that same problem on a larger scale (UA, user, author), and then reverse-order for `!important` styles. It's a pretty clever solution, but `!important` is a blunt instrument for handling layers inside the author origin.

I doubt most developers think about cascading origins, or the role importance plays in it - and at this point they don't really need to for practical reasons. I don't have a full solution here, but a rough sense that providing control of custom cascade origins (within/around the author origin) might help:

- provide a useful tool for solving many issues seen as "a specificity problem"
- help teach the powerful concepts already built into the core of the language
- make it more clear how CSS and `!important` are designed, and how they work under the hood

A few notes on finding a syntax/approach that would work:

- The property-by-property `!important` approach (or `!default` proposal which I like) is useful in other situations, but too narrowly applied for this particular use-case
- The selector-specificity `:where()` approach is both narrowly-applied and removes all

Assignees

No one assigned

Labels

Agenda+ F2F

css-cascade-5

Projects

None yet

Milestone

No milestone

2 participants

Project to Modernize the Cascade

CSS Cascade: Importance, Specificity, Overrides & Control

- Was designed for simpler times
- Today, can easily have hundreds of developers writing CSS over 10+ years on one project
- Completing a project ticket can’t require refactoring the overall architecture of all styles

Increasingly-Desperate Solutions

- Just have A Good Plan™. Be organized.
- OOCSS, BEM, SMACSS.
- Only ever use classes (only one class at a time),
  to flatten out specificity,
  basically completely removing the cascade from Author styles.
- Overuse ***!important*** to win specificity wars.
- CSS-in-JS (where now CSS loads in random order).
- Inline-style everything.

What About [as designed]?

- IDs increase specificity, but can only be used one per page.
- Element selectors work well for simple default styles, but aren't reusable enough for design/code systems. Too dependent on DOM structure.
- Which leaves classes and attribute selectors.
- And ***!important***.
- Authors are completely flattening the cascade to avoid specificity, instead of using it.

tailwindcss

Search the docs (Press "/" to focus)

A utility-first CSS framework for rapidly building custom designs.

Tailwind CSS is a highly customizable, low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override.

Get Started | Why Tailwind?

```
<div class="md:flex bg-white rounded-lg p-6">
  <img class="h-16 w-16 md:h-2 rounded-full mx-auto md:mx-
  <div class="text-center md:text-left">
    <h2 class="text-lg">Erin Lindford</h2>
    <div class="text-purple-500">Customer Support</div>
    <div class="text-gray-600">erinlindford@example.com</d
    <div class="text-gray-600">(555) 765-4321</div>
  </div>
</div>
```

Erin Lindford
Customer Support

Trying to keep everything minimum-specificity and inline. Separate every style from each other.

Most CSS frameworks do too much.

They come with all sorts of predesigned components like buttons, cards, and alerts that might help you move quickly at first, but cause more pain than they cure when it comes time to make your site stand out with a custom design.

Tailwind is different.

Instead of opinionated predesigned components, Tailwind provides low-level utility classes that let you build completely custom designs without ever leaving your HTML.

Cries for Help / Future Hopes

- Scoped Styles
- Web Components
- Encapsulation & Isolation

Instead of letting community of Authors destroy the cascade, can the CSSWG modernize it?

Extend Selectors

[css-selectors] Making the Tag-layer and ID-layer #4690

New issue

Open **jensimmons** opened this issue 7 minutes ago · 0 comments

jensimmons commented 7 minutes ago • edited

In an effort to modernize the CSS Cascade, and make it more suitable for projects with multiple Authors writing code over many years, **@mirisuzanne** is thinking about Origins (and how we might extend their power), and about the Specificity of Selectors.

For context:
The Selectors Level 4 defines three levels of sectors / three levels of specificity:

> A selector's specificity is calculated for a given element as follows:
> count the number of ID selectors in the selector (= A)
> count the number of class selectors, attributes selectors, and pseudo-classes in the selector (= B)
> count the number of type selectors and pseudo-elements in the selector (= C)
> ignore the universal selector

Miriam wrote on Twitter:

> Specificity only has three layers, & only the middle (classes/attributes) layer is flexible enough for most use cases. IDs can only be used once per page, & tags need to be reused semantically. They're useful but very limited.
>
> When everything is in a single layer, minute changes can have major impact. Everything relies on careful counting of selectors, and careful management of source order. Specificity should be more robust.
>
> We need ways to create custom tag-layer – & reusable id-layer – selectors.

Let's talk about this. What would it look like to extend selectors so that:

Assignees

No one assigned

Labels

- **css-cascade-5**
- **selectors-5**

Projects

None yet

Milestone

No milestone

1 participant

Custom Origins

§ 6.1. Cascading Origins

Each style rule has a ***cascade origin***, which determines where it enters the cascade. CSS defines three core origins:

Author Origin
: The author specifies style sheets for a source document according to the conventions of the document language. For instance, in HTML, style sheets may be included in the document or linked externally.

User Origin
: The user may be able to specify style information for a particular document. For example, the user may specify a file that contains a style sheet or the user agent may provide an interface that generates a user style sheet (or behaves as if it did).

User Agent Origin
: Conforming user agents must apply a default style sheet (or behave as if they did). A user agent's default style sheet should present the elements of the document language in ways that satisfy general presentation expectations for the document language (e.g., for visual browsers, the EM element in HTML is presented using an italic font). See e.g. the HTML user agent style sheet. [HTML]

Extensions to CSS define the following additional origins:

Animation Origin
: CSS Animations [css-animations-1] generate "virtual" rules representing their effects when running.

Transition Origin
: Like CSS Animations, CSS Transitions [css-transitions-1] generate "virtual" rules representing their effects when running.

§ 6. Cascading

The ***cascade*** takes an unordered list of declared values for a given property on a given element, sorts them by their declaration's precedence as determined below, and outputs a single cascaded value.

The cascade sorts declarations according to the following criteria, in descending order of priority:

¶ **Origin and Importance**

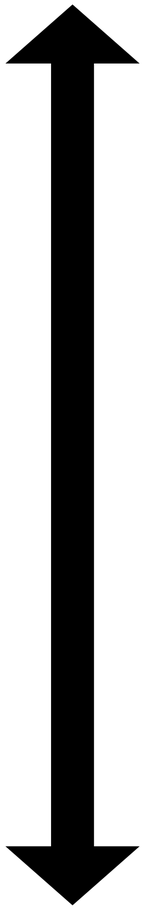
The origin of a declaration is based on where it comes from and its importance is whether or not it is declared '!important' (see below). The precedence of the various origins is, in descending order:

1. Transition declarations [css-transitions-1]
2. Important user agent declarations
3. Important user declarations
4. Important author declarations
5. Animation declarations [css-animations-1]
6. Normal author declarations
7. Normal user declarations
8. Normal user agent declarations

Declarations from origins earlier in this list win over declarations from later origins.


Most important
!important UA Styles
!important User Styles
!important Author Styles
Author Styles
User Styles
UA Styles
Least important

Most important



Least important

!important UA Styles

!important User Styles

!important Author – Custom Origin 1

!important Author – Custom Origin 2

!important Author – Custom Origin 3

Author Styles – Custom Origin 3

Author Styles – Custom Origin 2

Author Styles – Custom Origin 1

User Styles

UA Styles

portant

portant

portant

portant

		Components			
igin 4		Layouts	Site styles		Site Styles
igin 3	Overrides	Patterns	Base Theme		Ad network
igin 2	Design System	Defaults	CMS Extend	New Styles	3rd-party libs
igin 1	Reset	Reset	CMS Core	Old Styles	Bootstrap

User

UA

Advantages / Usecases

- Could be used to help the specificity wars between frameworks and Author styles.
- Could reinvigorate the intended use of importance, not as a new specificity, but as a balance of power.
- People could use a Origin with more importance instead of using `!important` whenever they need a style to override something that has more specificity. Which leaves ! important for Authors to use for a different use. (But also, this might perhaps be just as annoying as !important.)

usecases

Mandy Michael
@Mandy_Kerr

Replying to @jensimmons

This would be super handy for overriding 3rd parties like adtech and video tech etc...we often have specifically issues with that kind of stuff at work because of poorly written CSS so this would give us a cleaner way to resolve issues without needing more complex css!

jae anne (new! decade! new! decade!)
@dulcedejae

Replying to @jensimmons and @MiriSuzanne

I've had to write workarounds for this kind of problem a lot so personally I'd find this proposal super helpful. these days I'm using CSS-in-js in great part because it lets you control this kind of "inheritance" stuff more predictably than selector precedence

HJ Chen
@hj_chen

Replying to @jensimmons

This is an intriguing prospect because in my mind that allows for authors to retain a low specificity graph in their own stylesheet, which makes for a cleaner codebase in spite of the inclusion of 3rd party libraries.
Would support it being developed

[Mia | Miriam] Suzanne?
@MiriSuzanne

Replying to @jensimmons @jacobmparis and @ryanflorence

Which does raise the "defaults" use-case here. Think how much more CSSRemedy could do if we didn't have to keep specificity low.

Matthew Fung
@MattFung_

Replying to @jensimmons

This would be absolutely tremendous for anyone that's tried to refactor CSS on a large website before. Having the ability to overrule some 10,000 lines of wild west CSS would be an absolute blessing, especially if you're working with these files simultaneously. Sign me up.

Erica ☠️✨ @_erica · 3h
Replying to @zeldman and @jensimmons
HELL YES

zeldman @zeldman · 4h
Replying to @jensimmons
FULL STEAM AHEAD
1 11

Next Steps

- Is this something to pursue?
  Anyone going to object?

- Miriam Suzanne

- Bigger meta project to “modernize the Cascade” – to explore a range of ideas.

Masonry Layout

[css-grid-2] Masonry layout #4650

New issue

Open MatsPalmgren opened this issue 16 days ago · 9 comments

MatsPalmgren commented 16 days ago • edited

Overview

This is a proposal to extend CSS Grid to support masonry layout in one of the axes while doing normal grid layout in the other. I'll use `grid-template-rows/columns: masonry` to specify the masonry-axis in the examples (and I'll call the other axis the grid-axis). Here's a simple example:

```
<style>
.grid {
  display: inline-grid;
  grid: masonry / 50px 100px auto;
  grid-auto-columns: 200px;
  grid-gap: 10px;
  border: 1px solid;
}
item { background: silver; }
</style>

<div class="grid">
  <item style="border:10px solid">1</item>
  <item>2</item>
  <item>3</item>
  <item style="height:50px">4</item>
  <item>5</item>
  <item>6</item>
</div>
```

Result:

Assignees

No one assigned

Labels

Agenda+ F2F
css-grid-2
css-grid-3

Projects

None yet

Milestone

No milestone

5 participants

masonry.desandro.com

Masonry | Layout | Options | Methods | Events | Extras | FAQ

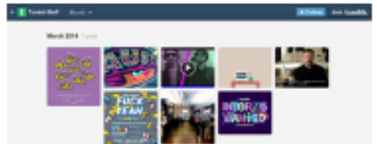
Masonry

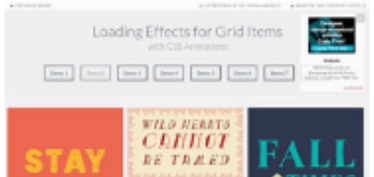
Cascading grid layout library

What is Masonry?

Masonry is a JavaScript grid layout library. It works by placing elements in optimal position based on available vertical space, sort of like a mason fitting stones in a wall. You've probably seen it in use all over the Internet.

Erik Johansson

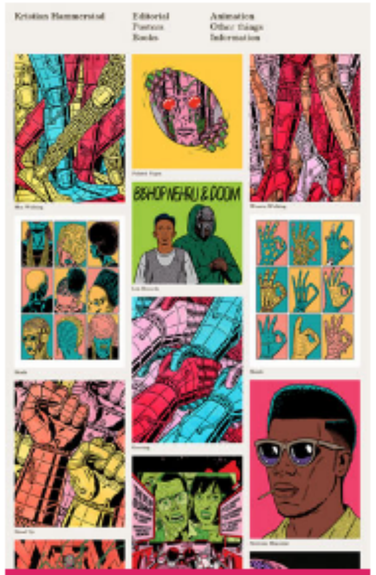




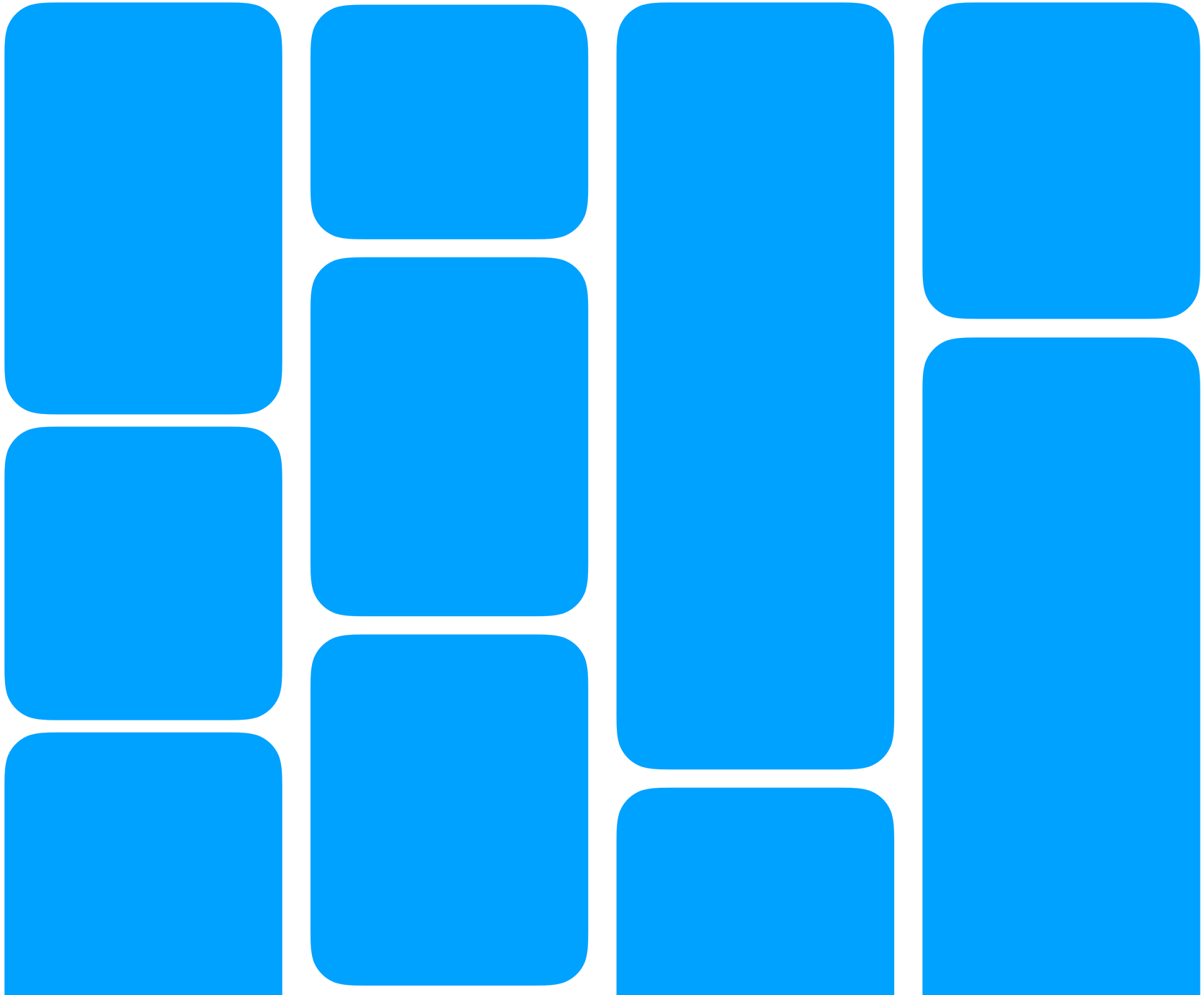
Download masonry.pkgd.min.js

Download these docs

Masonry on GitHub



Kristian Hammerstad



1

2

3

4

5

6

7

8

9

10

mountain cabin design

All Pins

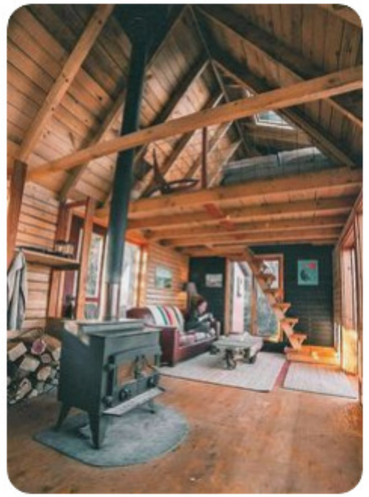
Home

Following

Jen

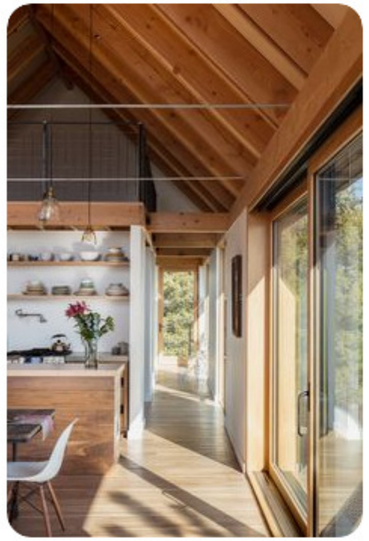
Rustic Living Room Decor Ideas Inspired By Cozy Mountain...

Rustic Living Room Decor Ideas Inspired By Cozy Mountain...



60 small mountain cabin plans with loft inspirational pin by...

Inspiring Small Log Cabin Designs



Boulder Mountain Cabin / HMH Architecture + Interiors

Boulder Mountain Cabin / HMH Architecture + Interiors

Cozy mountain retreat with Scandinavian vibe on beautiful...

Top 60 Best Log Cabin Interior Design Ideas - Mountain...

Top 60 Best Log Cabin Interior Design Ideas - Mountain...

25+ Rustic Living Room Ideas To Fashion Your Revamp Around

60 small mountain cabin plans with loft beautiful pin by...

1

2

3

4

5

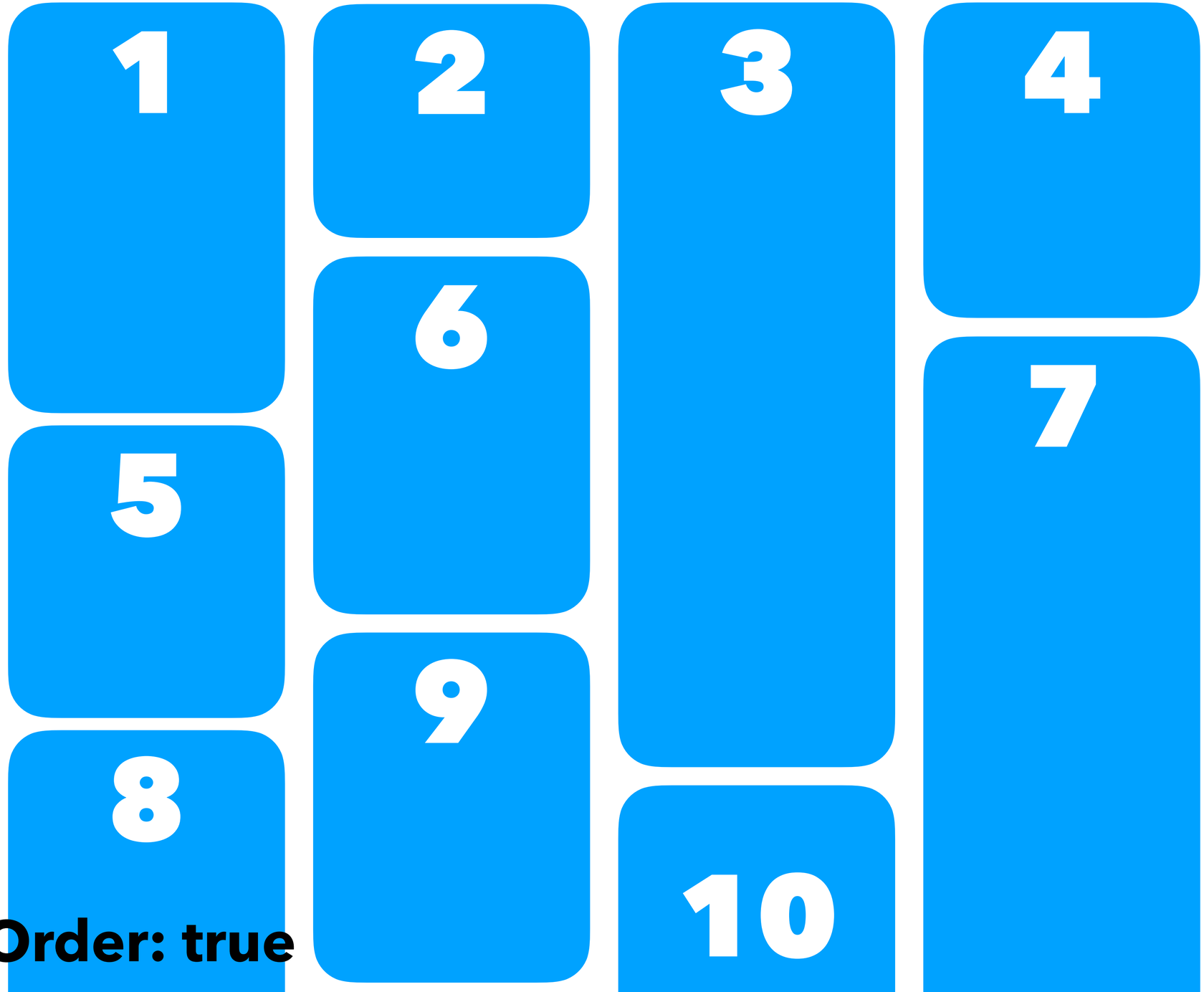
6

7

8

9

10



horizontalOrder: true