Please MAKE A COPY of this document before proceeding.

Before your session:

- Check your network connectivity and browser compatibility
- Sign up for a Hopin demo or contact us at blinkon@chromium.org with questions

Additional resources:

- Hopin Knowledge Center
- Quick Troubleshooting Reference Guide
- Sessions Tutorial - During the Event
- Tips for Speaking in and Moderating Sessions
    - Your session is an Open Session (anyone can participate on screen)

** Please note that this slide is for speakers only. Please feel free to delete this slide before sharing.*

To join your session:

- Head to the calendar invite or the Sessions tab in Hopin and find the Session you'll be hosting
- Click Share Audio and Video in the center of the Session screen
    - If you haven't allowed access to your camera or mic in the event, you'll be prompted to do so. Once you see yourself on screen, you're live to your audience.

To share your screen:

- Click the Screen sharing icon at the bottom of the page
- Switch to Chrome Tab on the popup window and choose the required browser tab
- Check the Share audio box
- Click Share to start sharing

Your session will be automatically recorded.

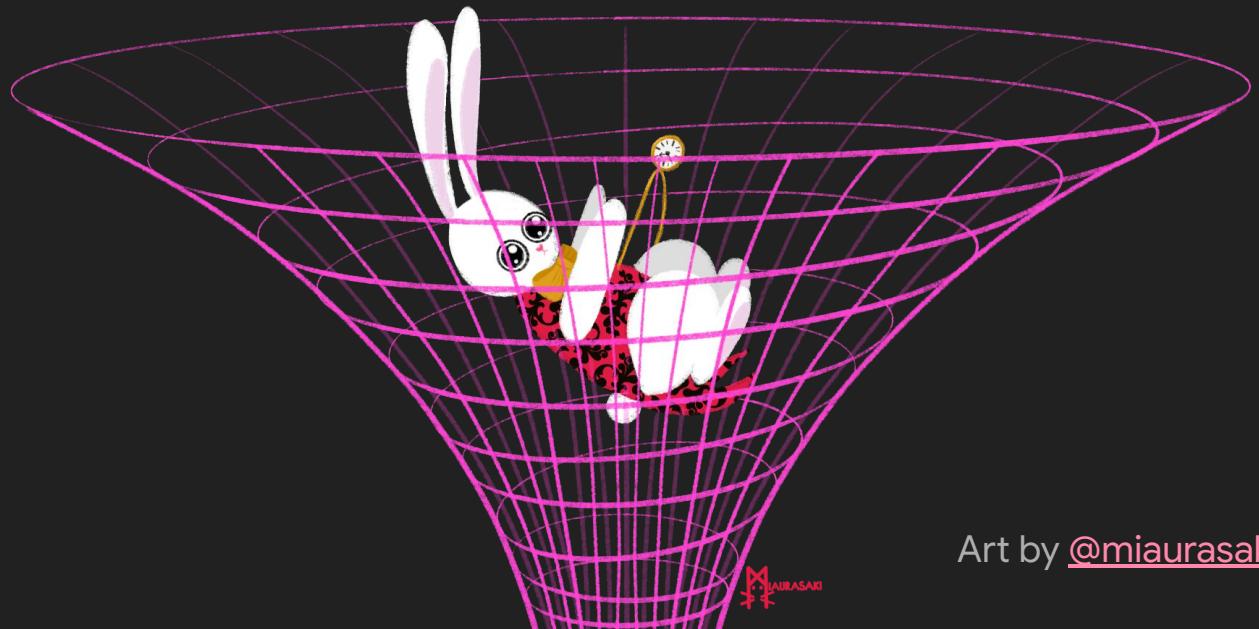
** Please note that this slide is for speakers only. Please feel free to delete this slide before sharing.*

Down the rabbit hole of GridNG

Ethan Jimenez (he/him)

Microsoft

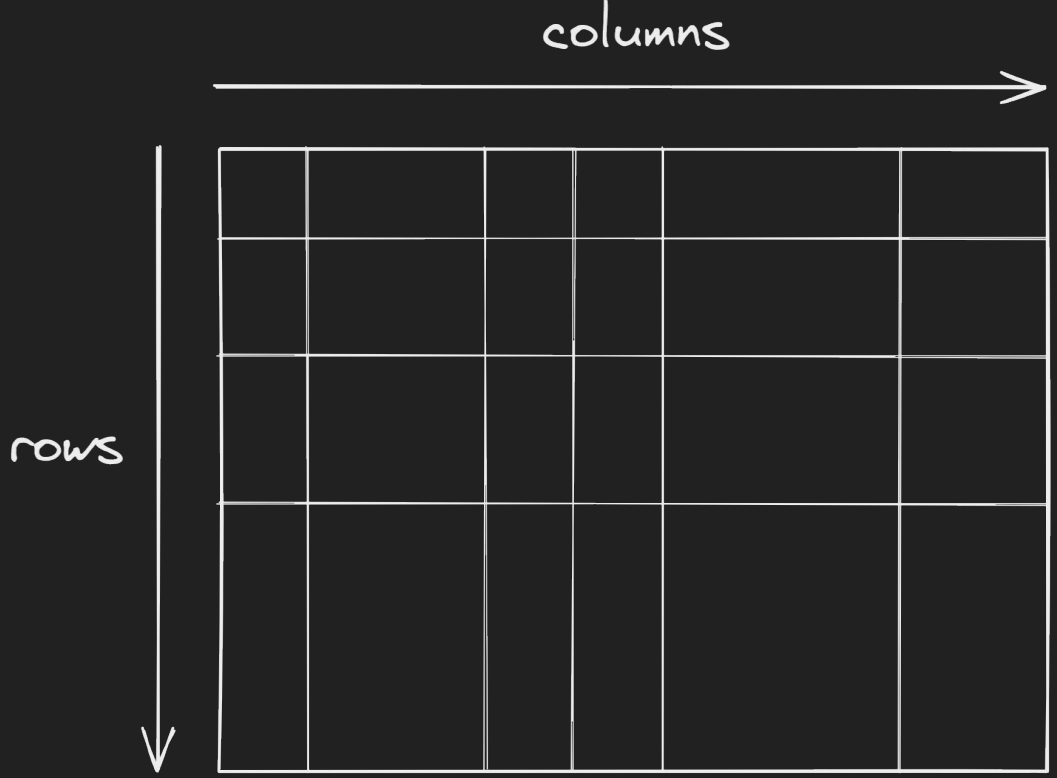
2022.05.18



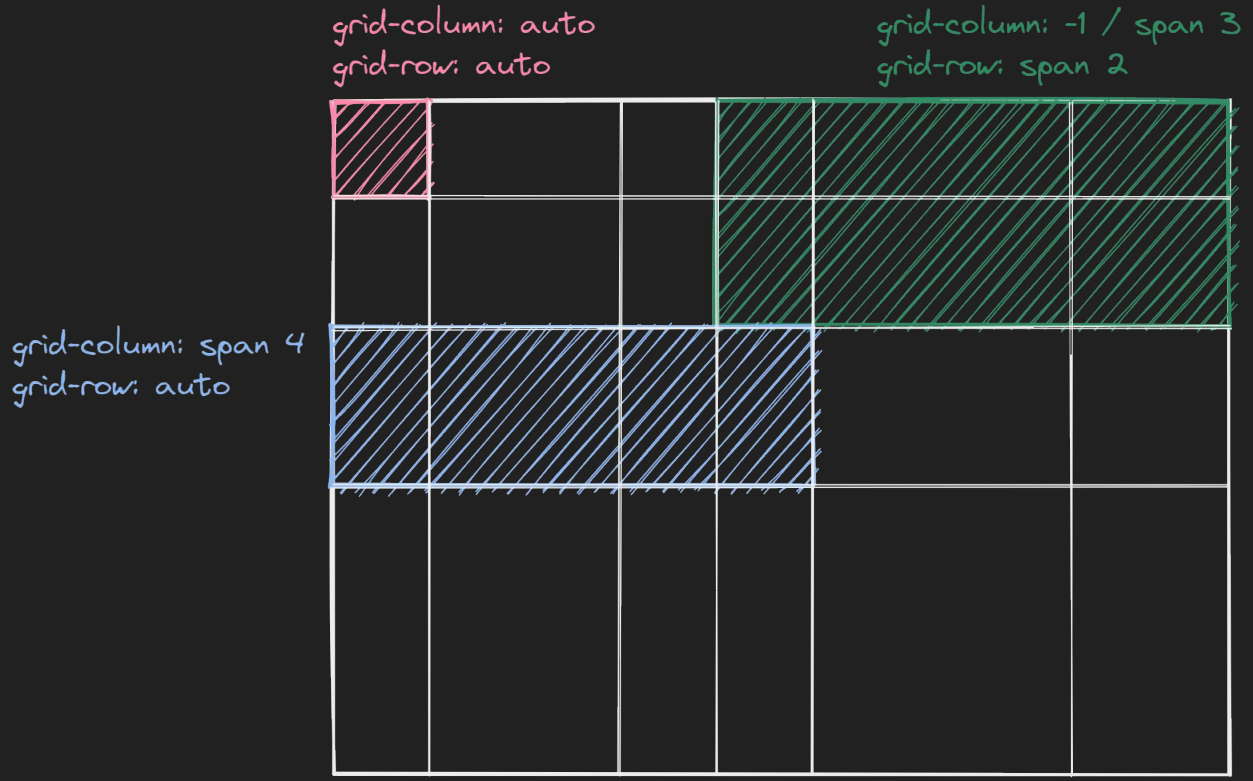
Art by @miaurasaki

Quick recap of grid concepts

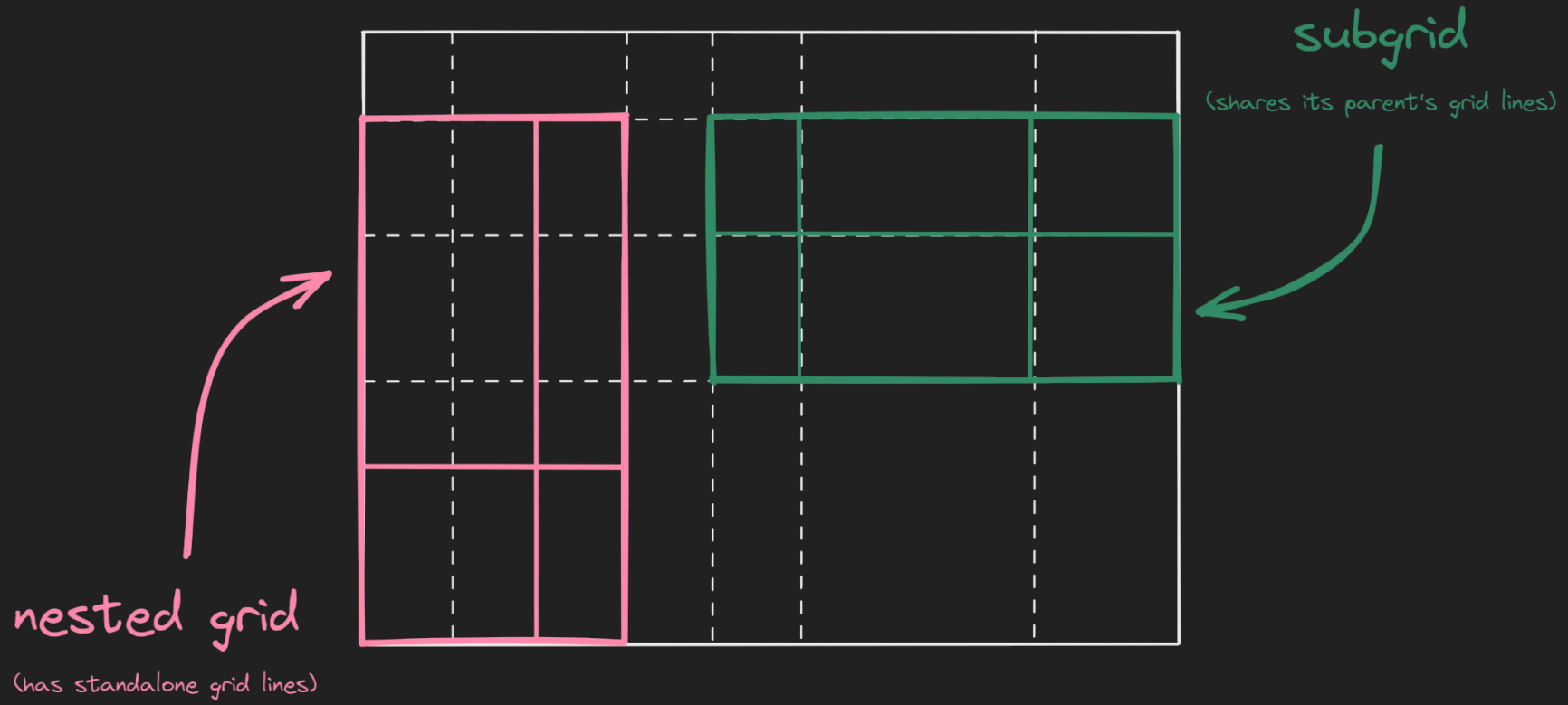
It all starts with...



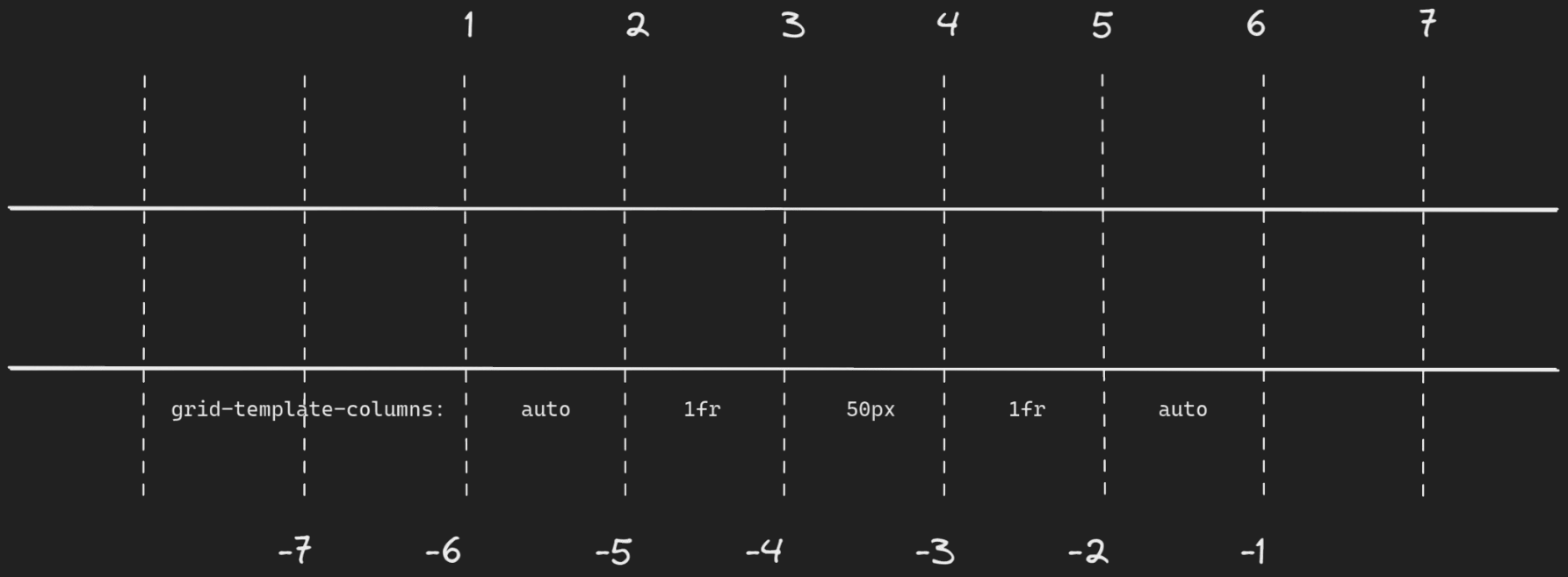
Grid items and placement



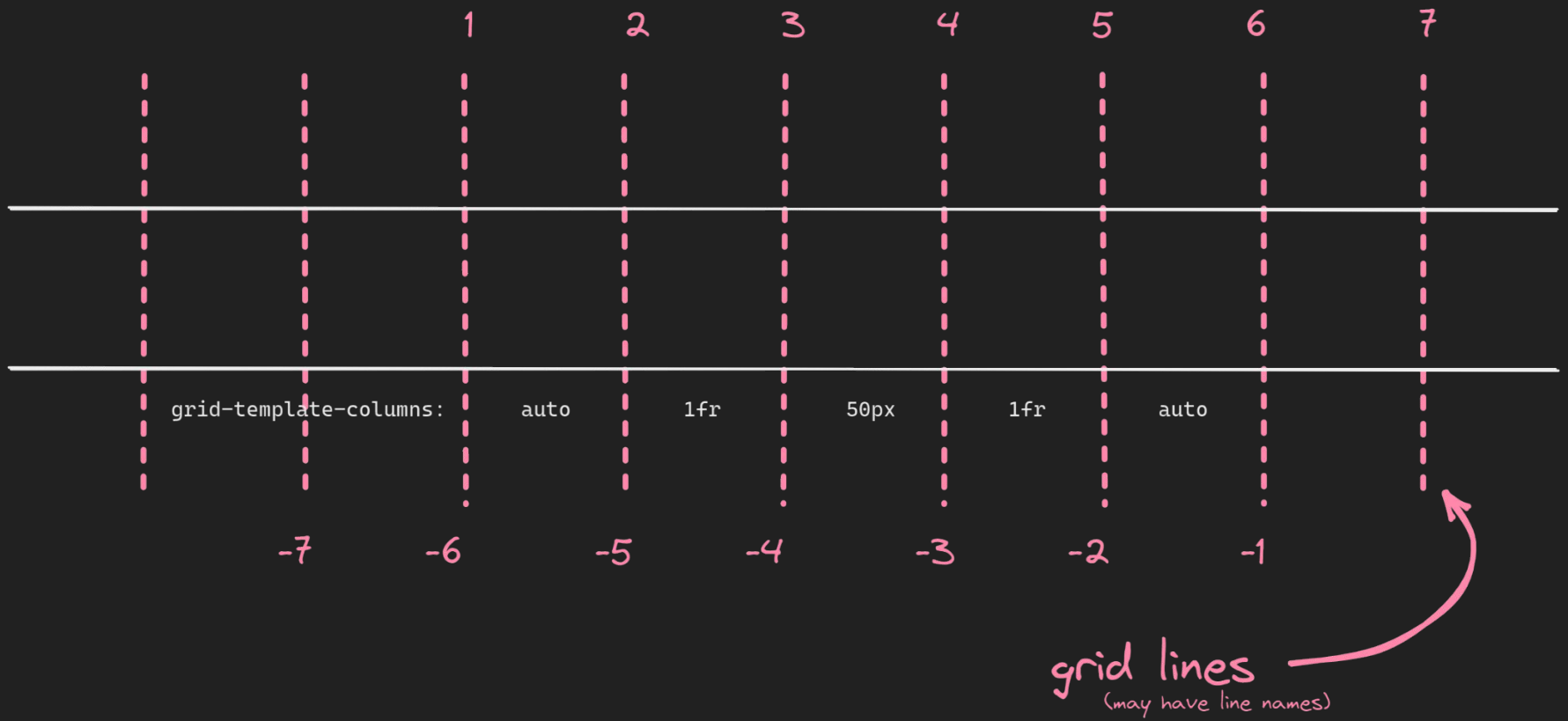
Grids within a grid



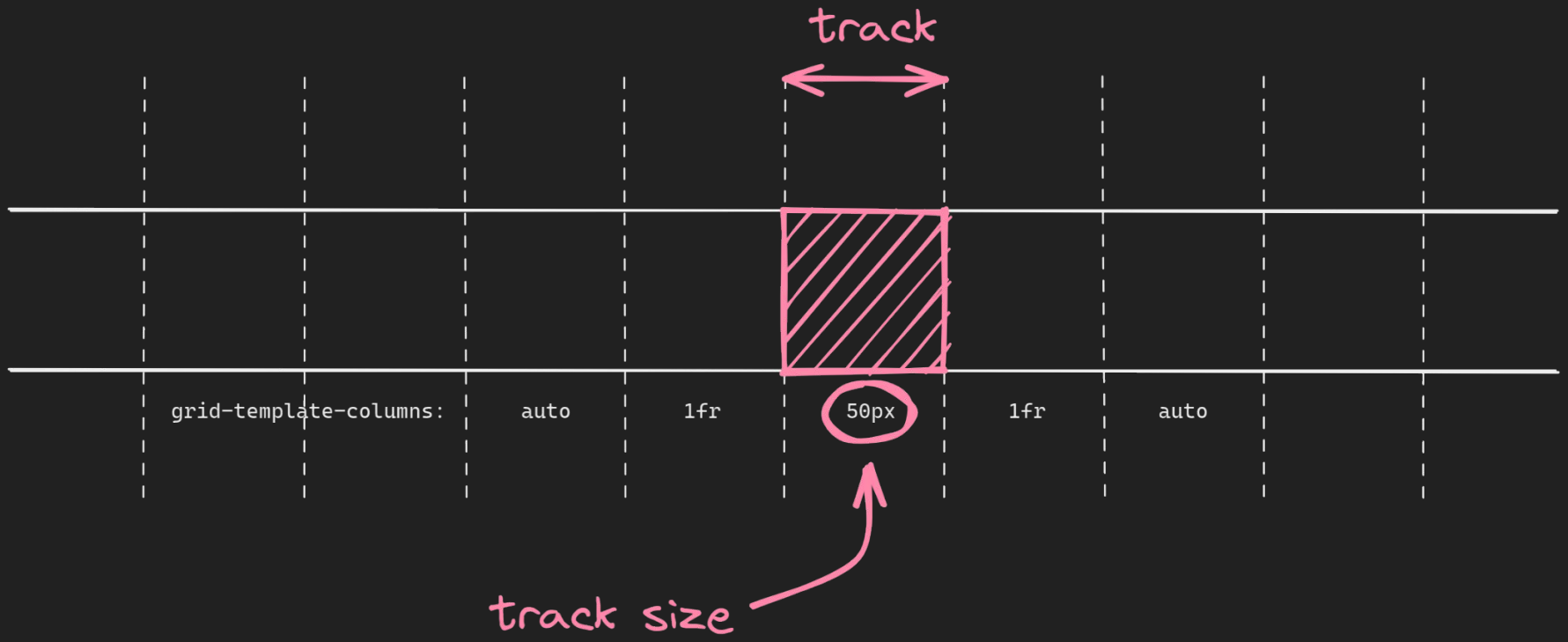
Track collection



Grid lines

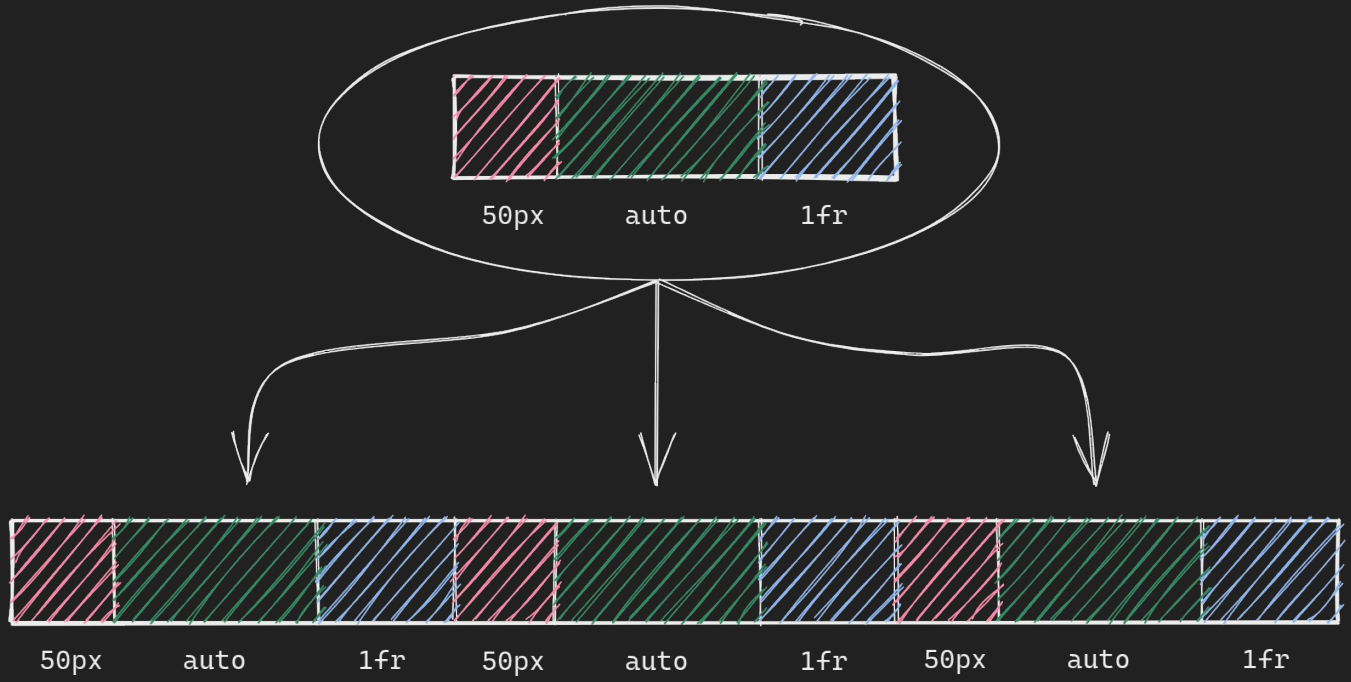

1
2
3
4
5
6
7
grid-template-columns:
auto
1fr
50px
1fr
auto
-7
-6
-5
-4
-3
-2
-1
grid lines
(may have line names)

Track definition



Repeater

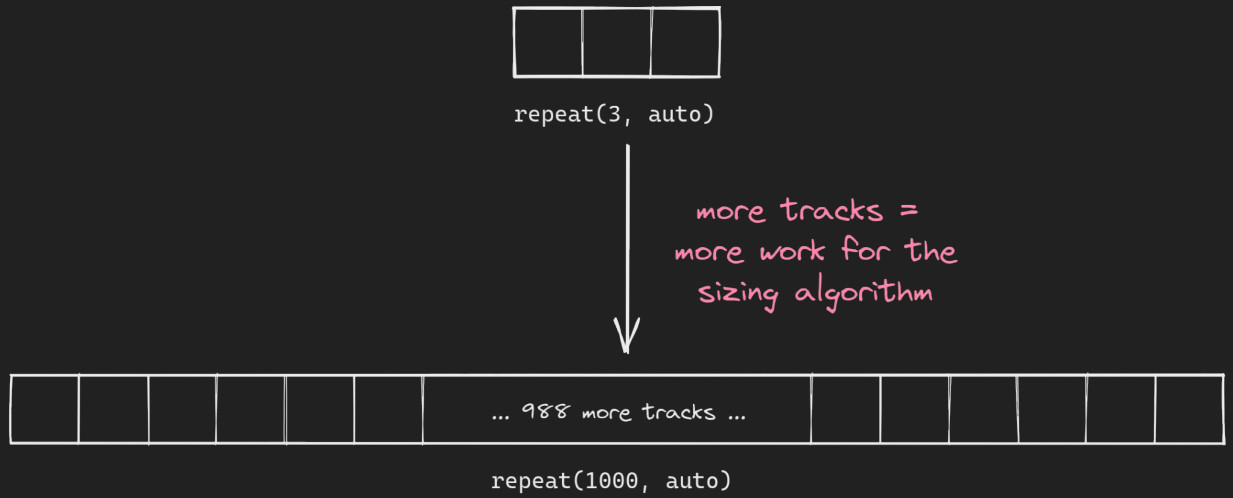
```
repeat(3, 50px auto 1fr)
```



What GridNG does differently

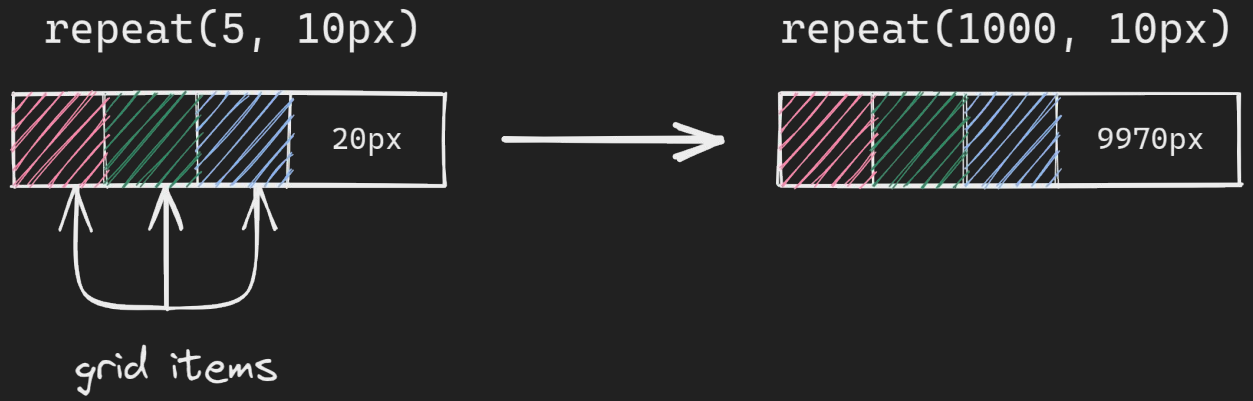
Deliberate features

- Avoid expanding repeaters as much as possible
  - How to deal with `grid-template-columns: repeat(1000, auto)`?



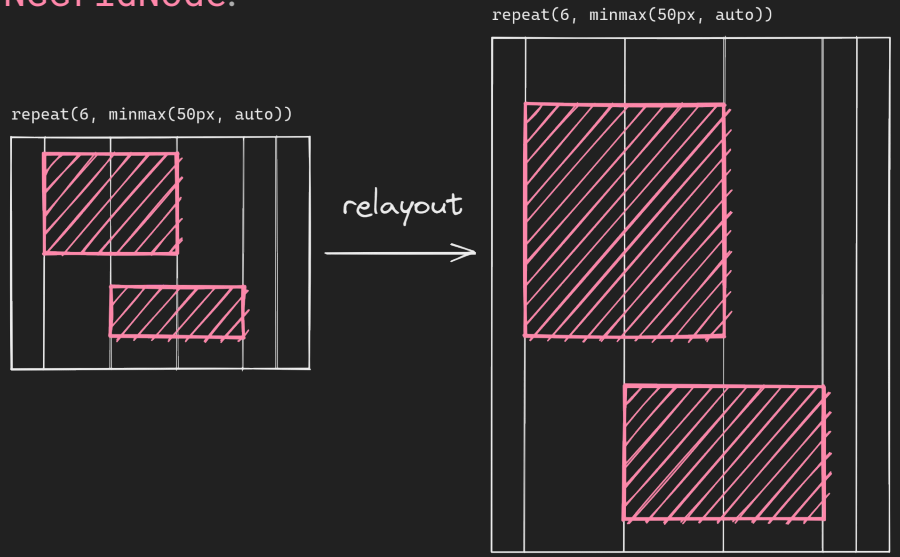
Deliberate features

- Avoid expanding repeaters as much as possible
  - Complexity is tied to the number of tracks and grid items that the author defines.
  - Changing the hard-coded max number of tracks that a grid can have does not affect performance or memory usage of our implementation.



Deliberate features

- Keep the state of grid algorithms in stack
  - In other words... pass all context for grid methods via parameters.
  - Couple of methods rely on this to invalidate cached results.
    - E.g., resolved grid item positions in `NGGridNode`.



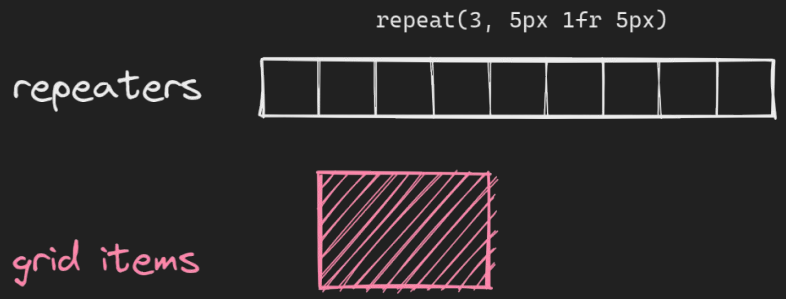
Don’t expand repeaters!

repeat(3, 5px 1fr 5px)

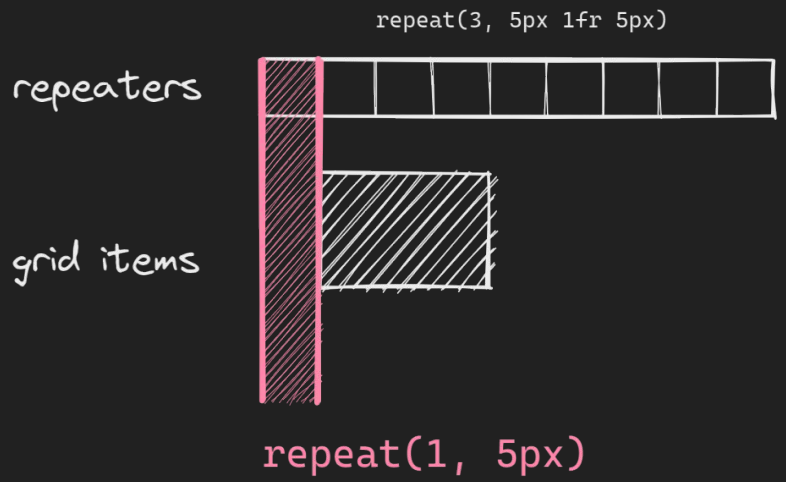
repeaters

= 15px 3fr 15px

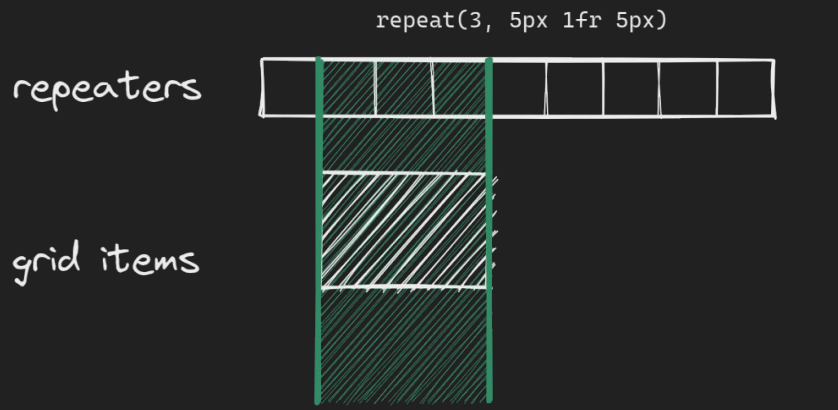
Don’t expand repeaters!



Don’t expand repeaters!

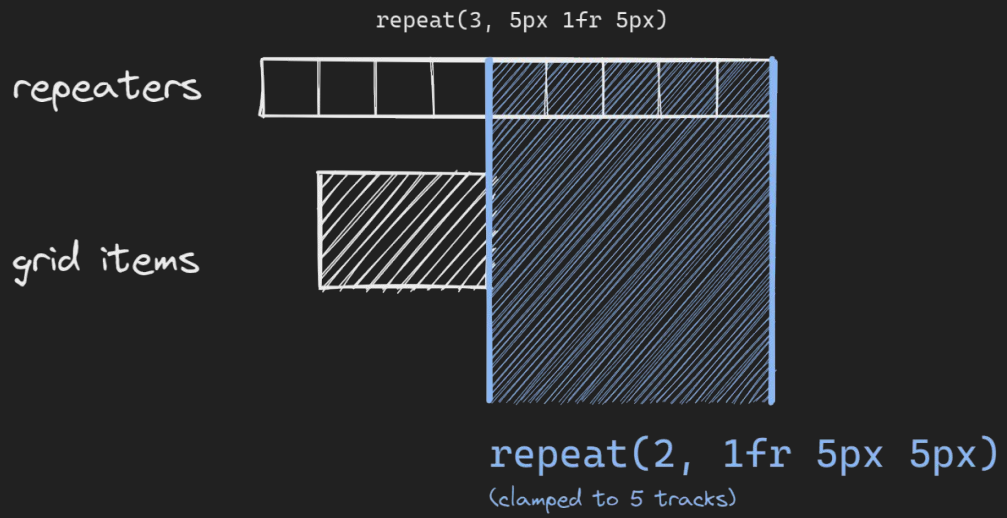


Don’t expand repeaters!

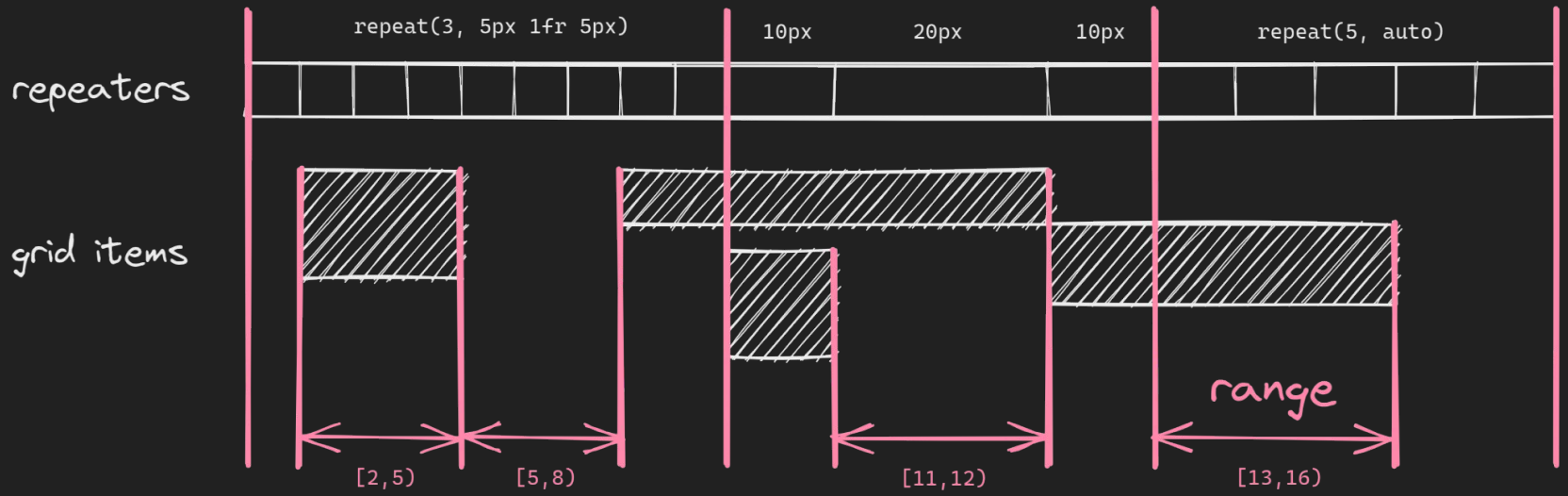


`repeat(1, 1fr 5px 5px)`

Don’t expand repeaters!

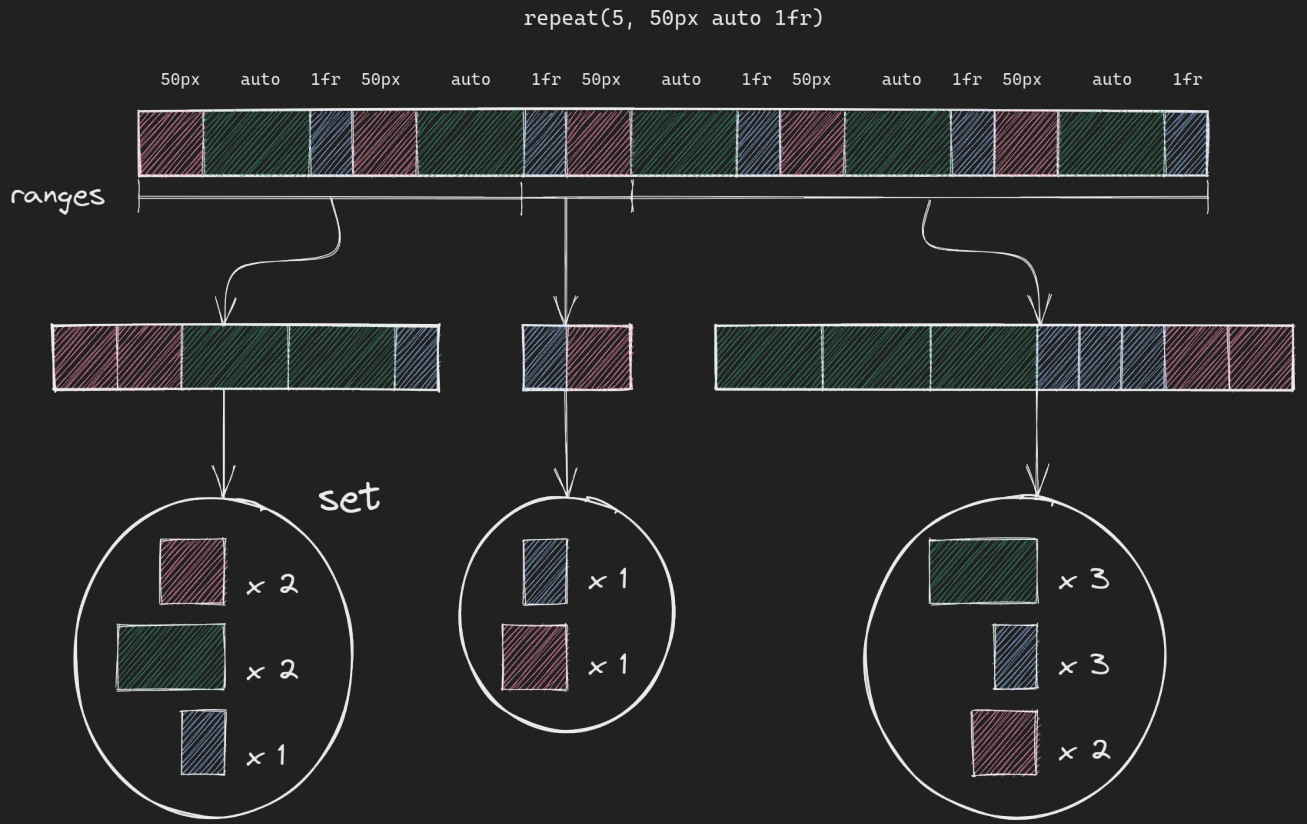


Track ranges




```
ranges = [[ 1,  2), [ 2,  5), [ 5,  8),
          [ 8, 10), [10, 11), [11, 12),
          [12, 13), [13, 16), [16, 18)]
```

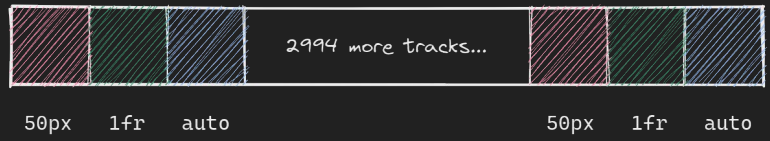
Sets of track definitions



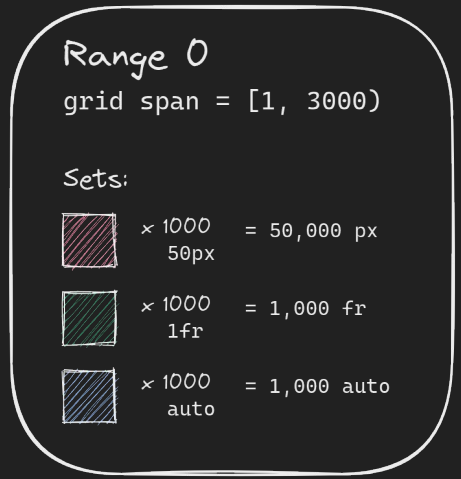
Not expanding repeaters = lower complexity

`repeat(1000, 50px 1fr auto)`

Legacy track collection

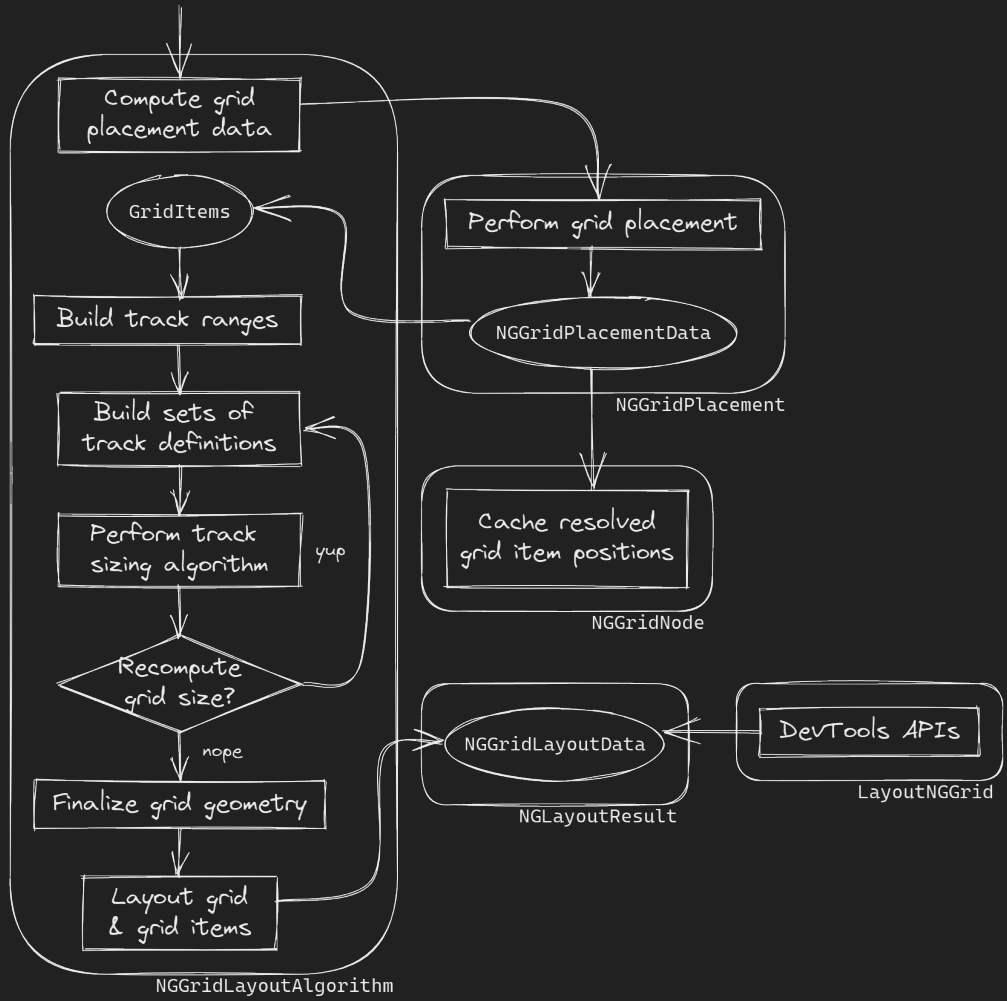


GridNG track collection



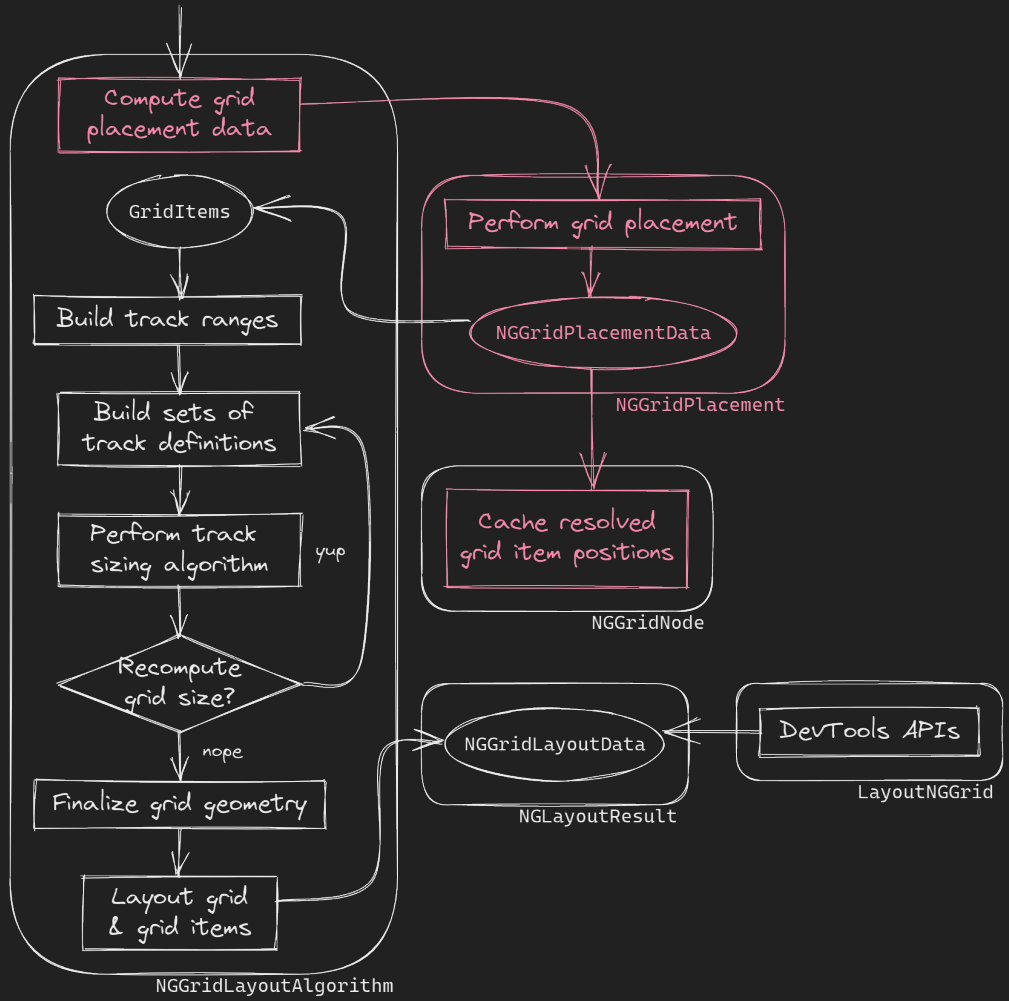
GridNG pipeline

... before subgrid



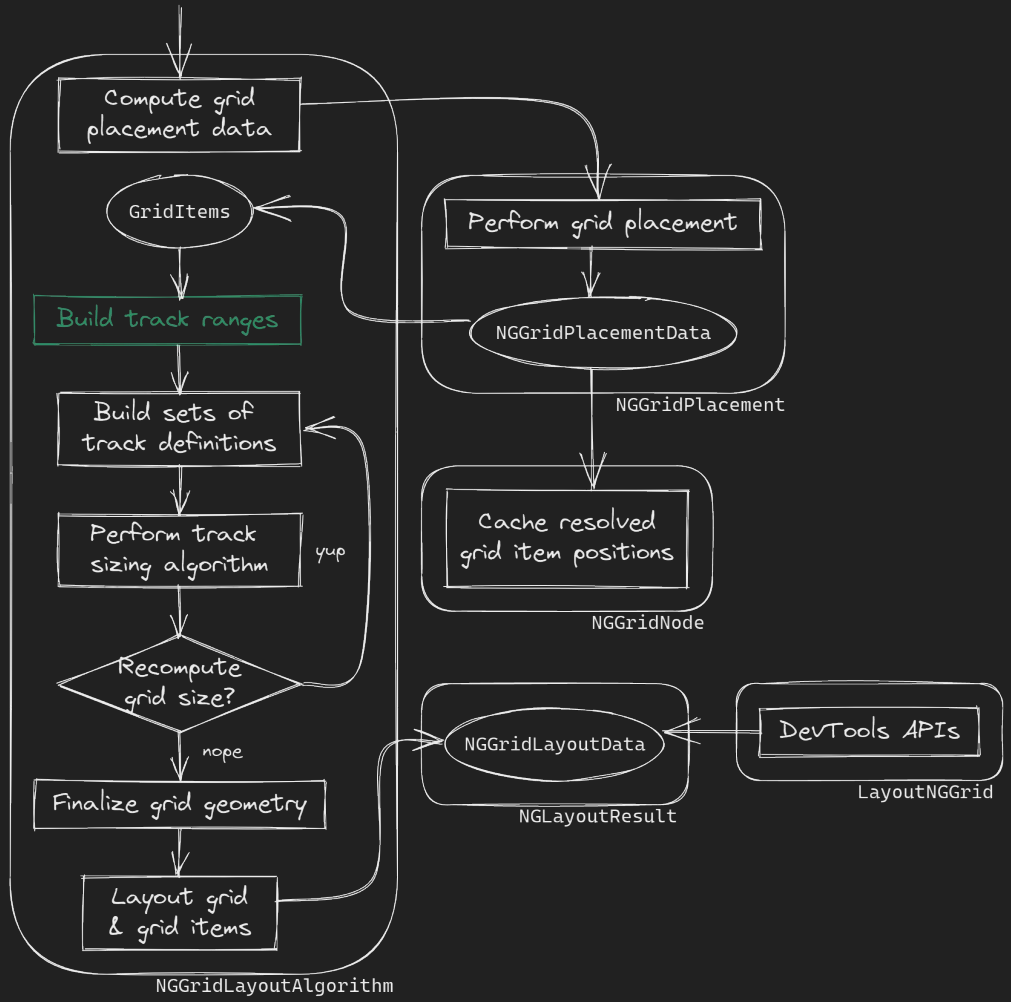
Grid placement

- Auto-placement algorithm is very expensive, we cache its resolved grid item positions.



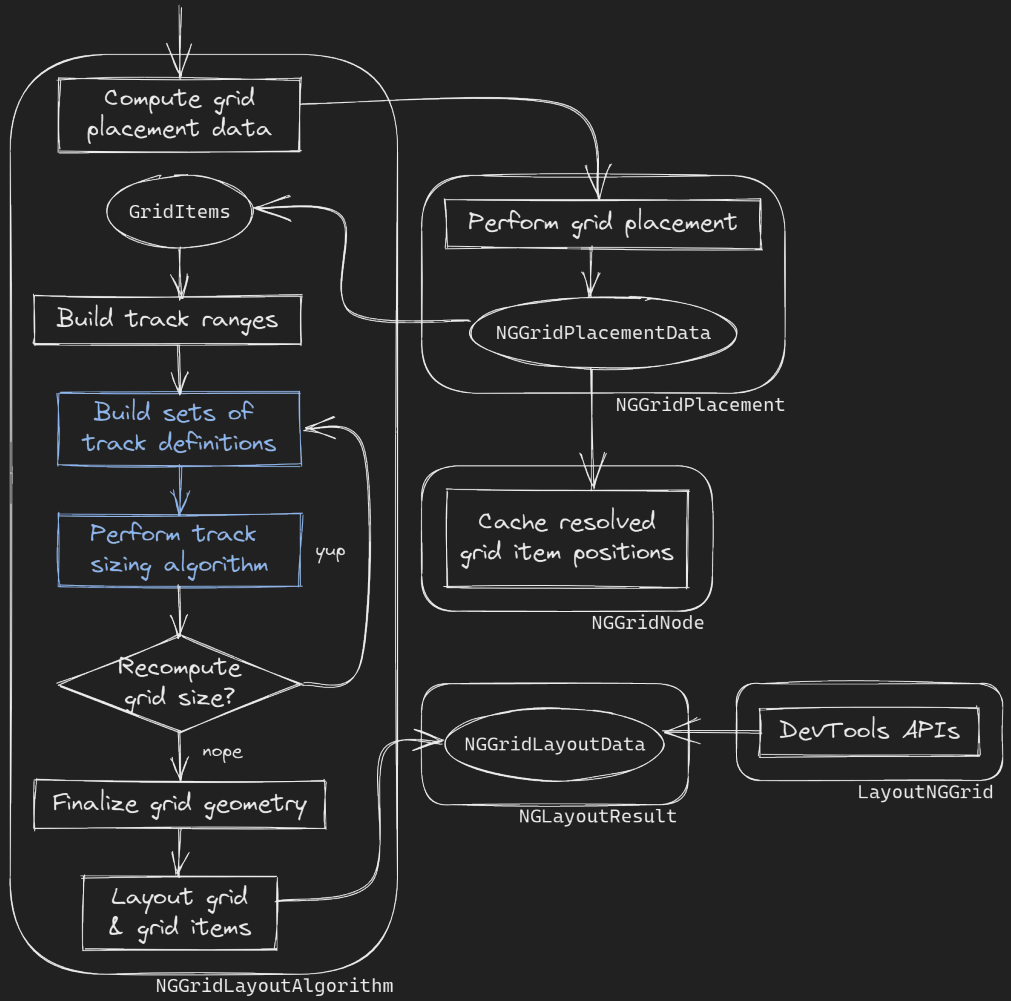
Build track ranges

- NGGridBlockTrackCollection
- Cache properties of each track range, we need to know if they contain intrinsic or flexible tracks, rely on block size, etc.
- Cache the subset of ranges spanned by each grid item.



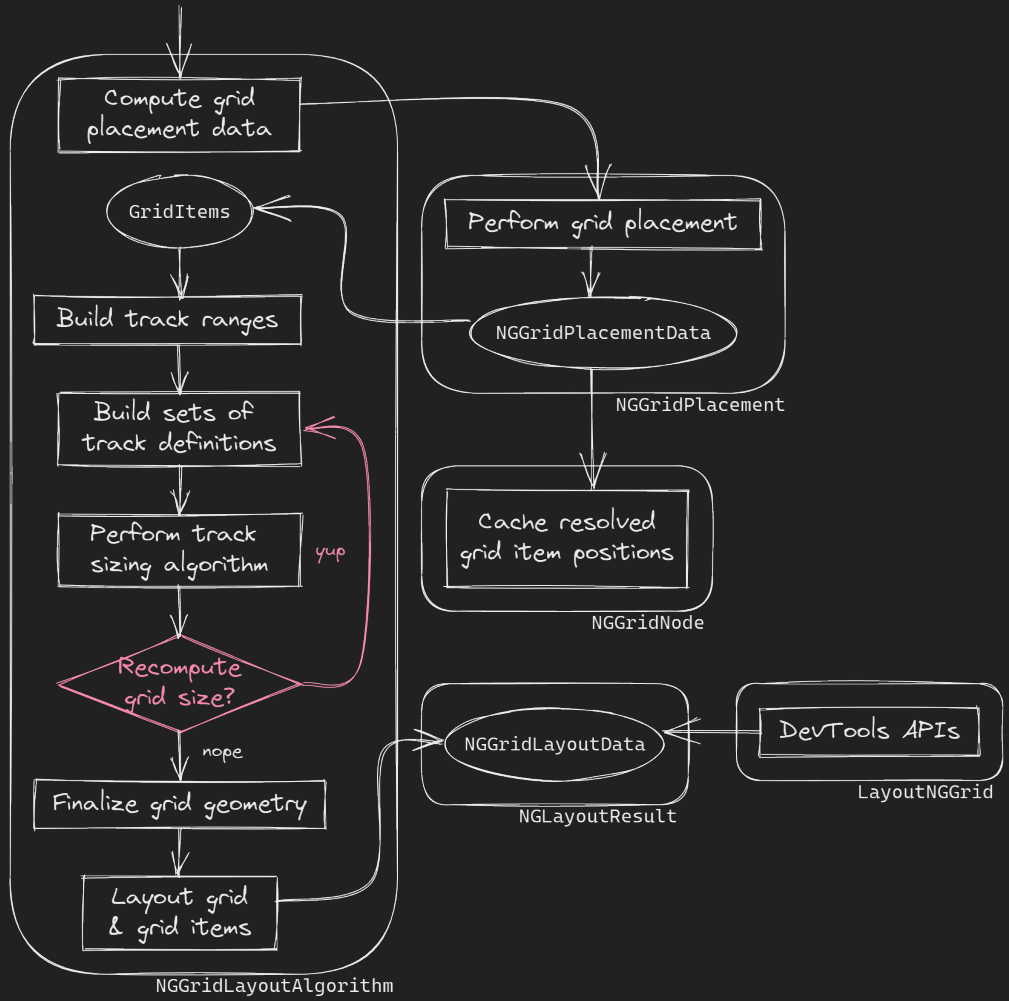
Grid sizing

- Track sizing algorithm works mostly the same, but our implementation sizes sets of tracks instead of single tracks.
- Between passes of the track sizing algorithm, we make use of `NGGridLayoutData` to compute the available space for measuring grid items.



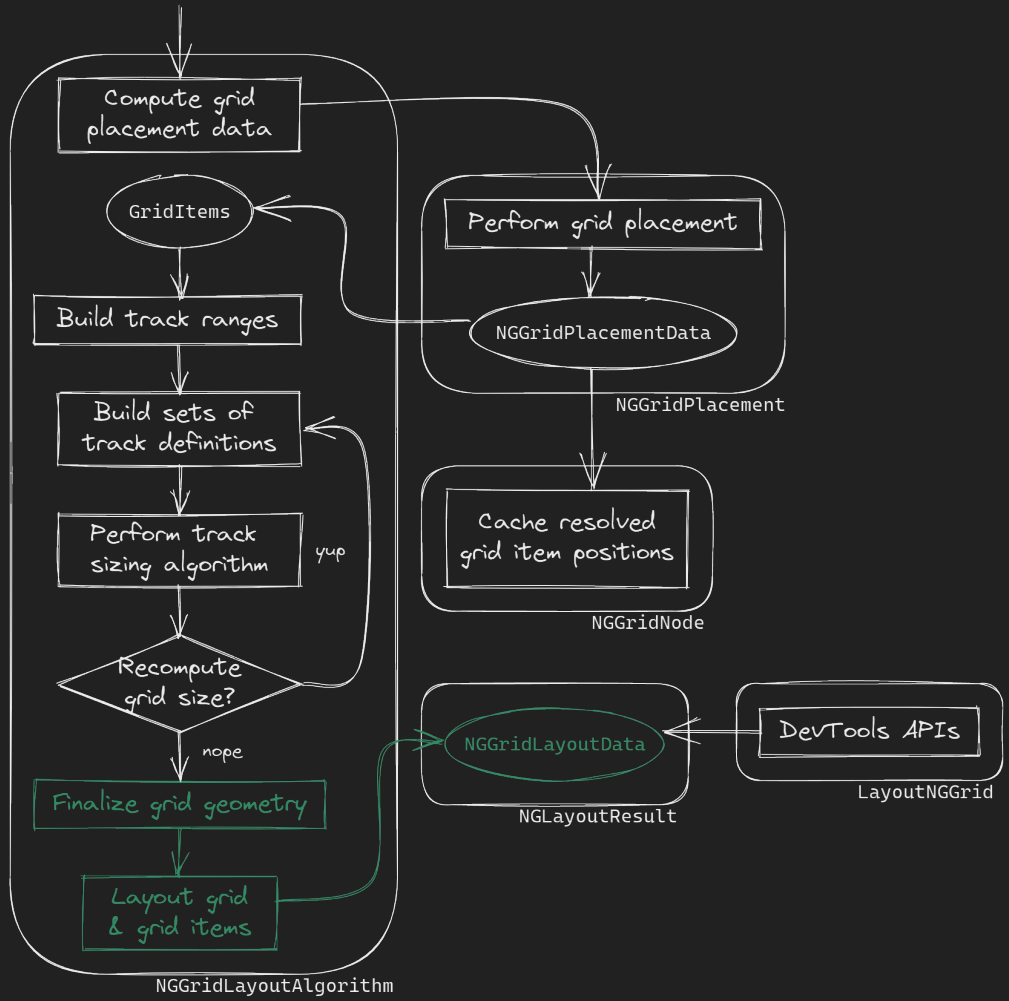
Recompute grid size

- We need to recompute the grid size whenever the available block size was indefinite and we have a dependency on it, e.g., a row has percentage size.



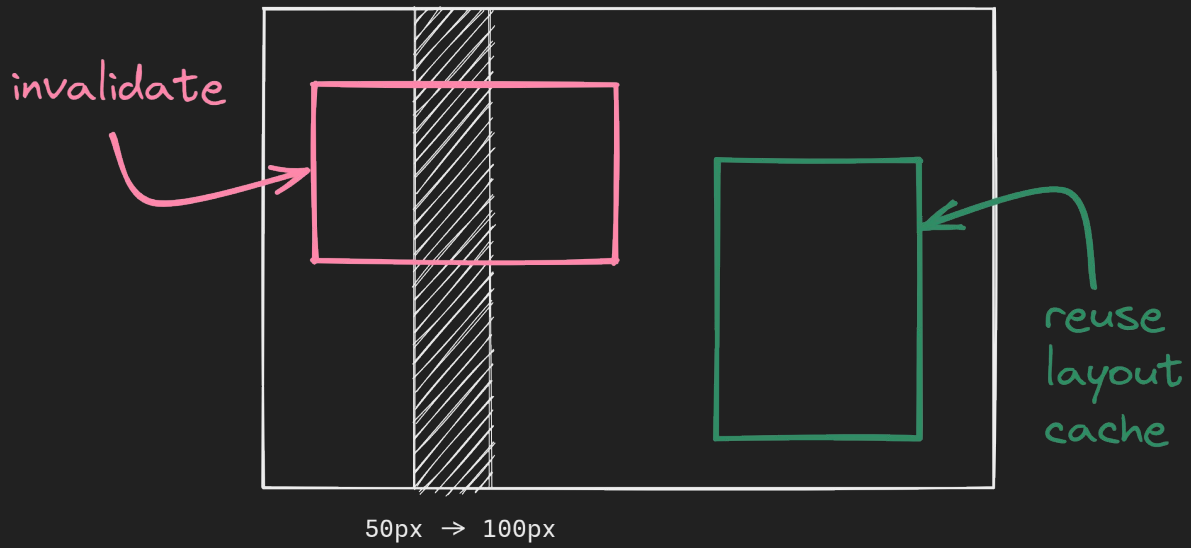
Layout

- After sizing, we cache the start offset of every set in both track collections. These offsets are known as the “grid geometry”.
- At this point, we adjust the grid geometry for fragmentation.



Taking advantage of LayoutNG

- Free caching of `NGLayoutResult`!
  - `NGConstraintSpace` is built from `NGGridLayoutData`.
  - Changes to a track range only invalidate the cache of relevant grid items.



What’s next?

Subgrid will challenge our implementation

- Subgridded items need to be placed before we size the grid.
  - We need to entirely isolate grid placement from `NGGridLayoutAlgorithm`.

Subgrid will challenge our implementation

- Subgridded items need to be placed before we size the grid.
  - We need to entirely isolate grid placement from `NGGridLayoutAlgorithm`.
- A subgrid effectively inherits the grid lines it spans within its parent.
  - We want to pass the geometry of a subgridded axis down to a subgrid in `NGConstraintSpace`, which will give us layout result caching for free too!
  - A full revamp of `GridPositionsResolver` is necessary to inherit grid line names.

Subgrid will challenge our implementation

- Subgridded items need to be placed before we size the grid.
  - We need to entirely isolate grid placement from `NGGridLayoutAlgorithm`.
- A subgrid effectively inherits the grid lines it spans within its parent.
  - We want to pass the geometry of a subgridded axis down to a subgrid in `NGConstraintSpace`, which will give us layout result caching for free too!
  - A full revamp of `GridPositionsResolver` is necessary to inherit grid line names.
- Subgridded items account for a subgrid's margin, border, padding.
  - Rework of track sizing algorithm to accommodate hypothetical items.
  - Introduce accumulated "extra" margin to subgridded items.

Special thanks to the GridNG task force

- Ana Sollano Kim <ana.sollano@microsoft.com>
- Christian Biesinger <cbiesinger@chromium.org>
- Daniel Libby <dlibby@microsoft.com>
- Ian Kilpatrick <ikilpatrick@chromium.com>
- Jacques Newman <jacques.newman@microsoft.com>
- Kurt Catti-Schmidt <kschmi@microsoft.com>

Thank you!

Questions?