

31 May 2018, Utrecht
Front-end United

WORKSHOP

CSS Layout

Hidde de Vries
@hdv

Introductions

hiddedevries.nl

moz://a

Introduction

What's your name?

Where did you travel from?

What's your favourite CSS property?

schedule

- 09.30-09.45** CSS layout in context
- 09.45-10.15** Flexbox 101
- 10.15-12.30** Flexbox practice
- 11.00-12.30** Grid Layout 101, part 1
 - lunch –
- 13.30-14.15** Grid Layout 101, part 2
- 14.15-16.00** Grid Layout practice
- 16.00-17.00** Grid in PROD, fallbacks

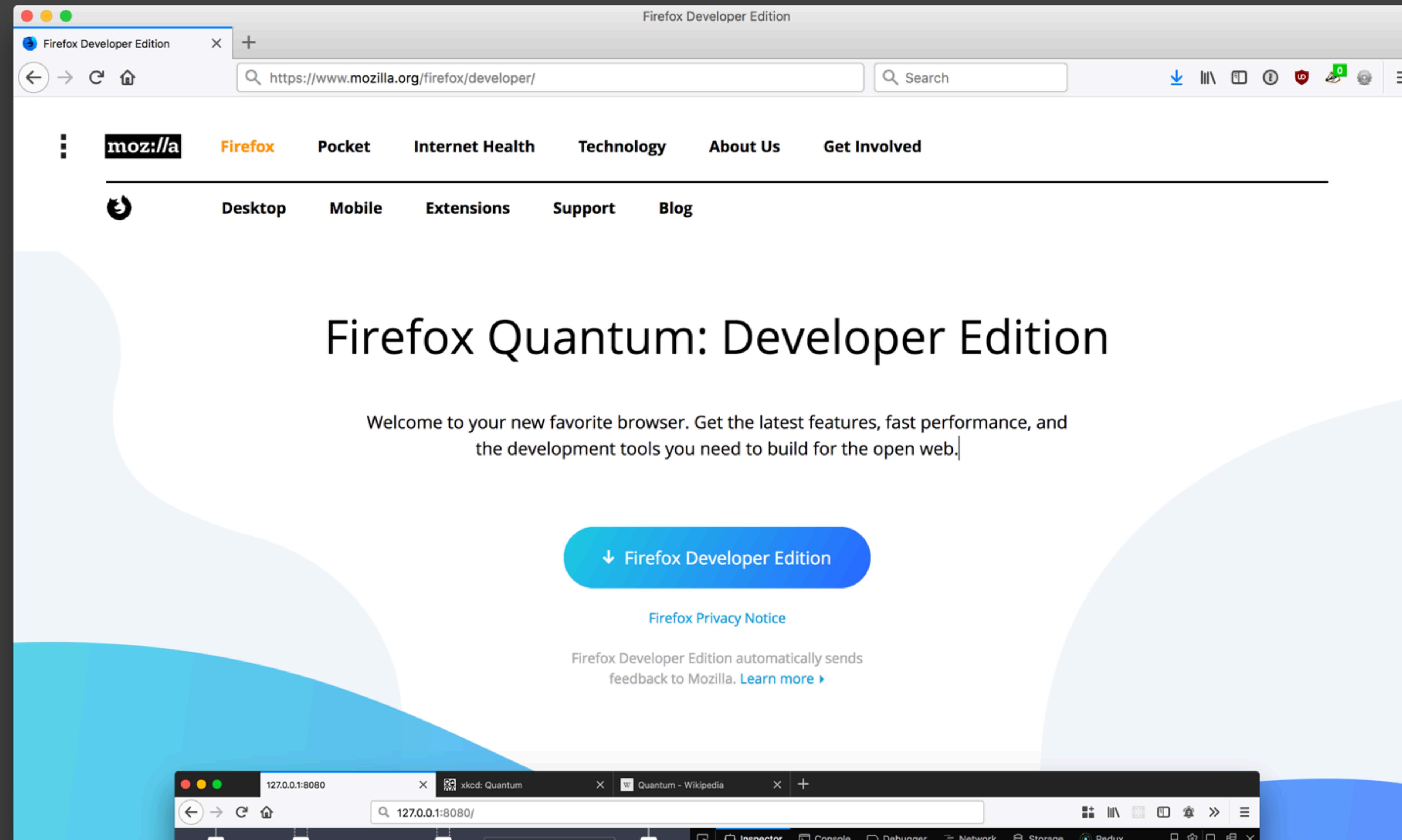
Requirements
for today

Requirements for today

Your favourite text editor; Codepen/JSBin are fine, too

A browser with good layout tools

<https://www.mozilla.org/firefox/developer/>



Requirements for today

Your favourite text editor; Codepen/JSBin are fine, too

A browser with good layout tools

Code examples

<https://github.com/hidde/workshop-css-layout/>

Requirements for today

Your favourite text editor; Codepen/JSBin are fine, too

A browser with good layout tools

Code examples

No conventions. Feel free to use IDs for styling, don't worry about maintainability (just yet).

CSS Layout in context



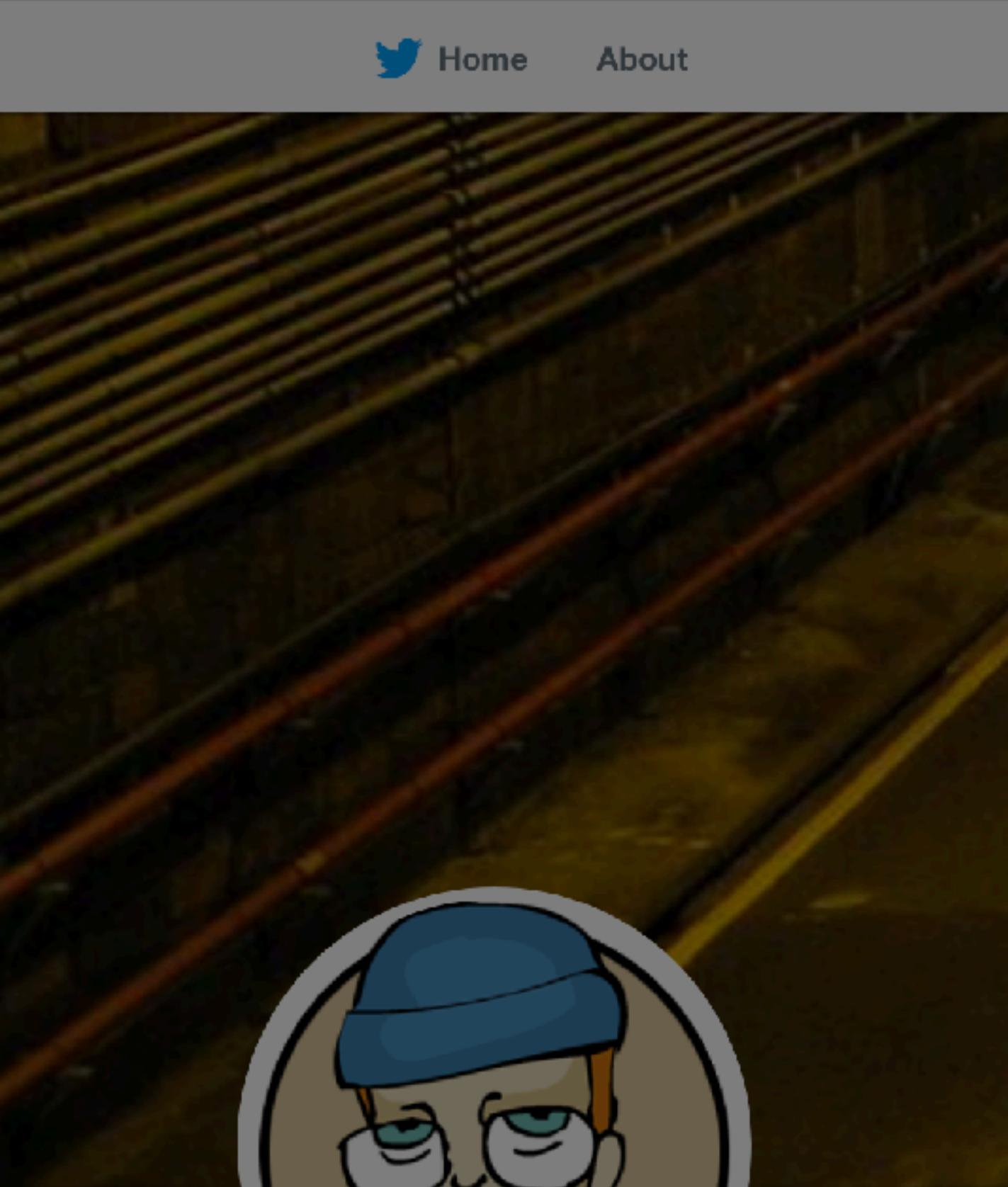
Eric Meyer

@meyerweb

Web standards, HTML/CSS, microformats, community, writing, speaking, signing guy. Husband, father, agnostic in principle, atheist in practice. #663399becca

📍 Cleveland Heights, OH

🔗 meyerweb.com



Eric Meyer
@meyerweb

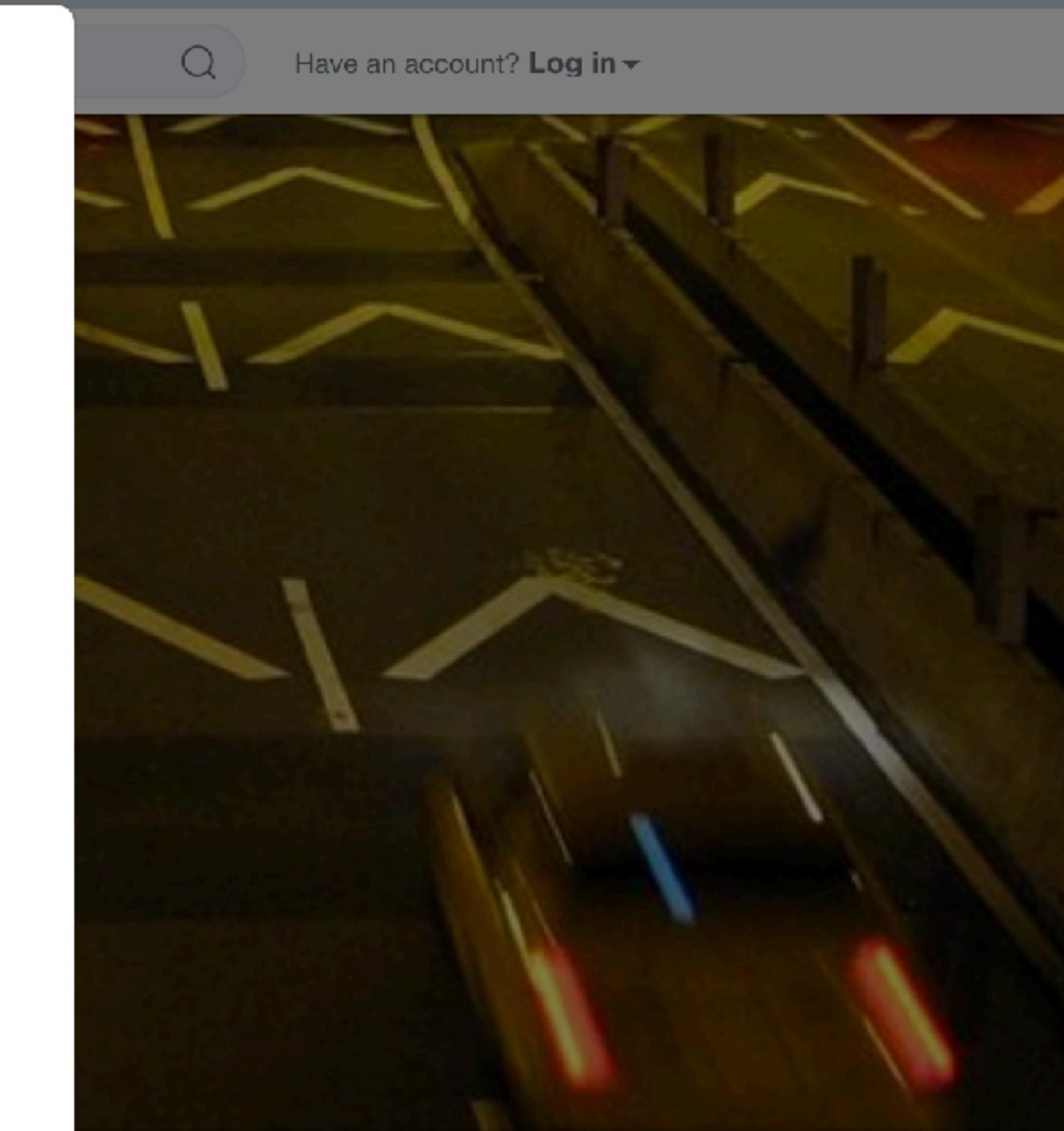
Follow ▾

On the left: the 2nd and 3rd editions of “CSS: The Definitive Guide”. On the right, a single copy of the fourth edition.

?

1:25 PM - 10 Nov 2017

751 Retweets 2,039 Likes



1 spec → many modules

**Specs are implemented
much faster**

We now have dedicated
CSS for lay-out



**Flexibility is built
into the web**

The World Wide Web project.

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#), [Policy](#), November's [W3 news](#), [Frequently Asked Questions](#).

[What's out there?](#) Pointers to the world's online information, [subjects](#), [W3 servers](#), etc.

[Help](#) on the browser you are using

[Software Products](#) A list of W3 project components and their current state. (e.g. Line Mode, X11 Viola, NeXTStep, Servers, Tools, Mail robot, Library)

[Technical](#) Details of protocols, formats, program internals etc

[Bibliography](#) Paper documentation on W3 and references.

[People](#) A list of some people involved in the project.

[History](#) A summary of the history of the project.

[How can I help ?](#) If you would like to support the web..

[Getting code](#) Getting the code by [anonymous FTP](#), etc.

Navigation

< Previous | Back up | Next >

Home | Help

Front Apps | Frequently | Hypertext | Policy -- | Summary -- | 3DII --

CERN Welcome

C.E.R.N.

European Laboratory for Particle Physics, Geneva, Switzerland

- Help[1] on W3 programs. Also: about the World-Wide Web[2]
- About CERN[3] Also phone numbers, offices and e-mail for People[4], Yellow Pages[5], or french Pages Jaunes[6].
- News[7] Public news, e.g. User's Office[8], student news[9]. Also private groups[10] and Internet news[11].
- Computer center Documentation and newsletter index[12], computing news[13] , VMS Help[14].
- Systems/Projects[15] Systems available from CERN, and related projects.
- Experiments[16] and collaborations at CERN.
- H E P[17] Other High-Energy Physics institutes.
- See also: Type of service[18], and OTHER SUBJECTS[19]
1-19, Back, Up, <RETURN> for more, Quit, or Help:



Dan Noyes, CERN - first-website.web.cern.ch

**Flexibility is built
into the web**

**Flexibility is built
into CSS**

Medium-agnostic

“

CSS is a language for describing the rendering of structured documents (such as HTML and XML) on screen, on paper, in speech, etc.

— every CSS spec

Not a programming language

**HTML
describes
what's there**

**CSS describes
what it
looks like**

The screenshot shows the W3C Candidate Recommendation page for the CSS Grid Layout Module Level 1. The page title is "CSS Grid Layout Module Level 1" and it was last updated on 09 May 2017. The left sidebar contains a "TABLE OF CONTENTS" with sections such as Introduction, Overview, Grid Layout Concepts and Terminology, Reordering and Accessibility, Grid Containers, Grid Items, Defining the Grid, and Abstract. The main content area includes sections on Background and Motivation, Declaring the Grid, Placing Items, Sizing the Grid, Grid Areas, Grid Lines, Grid Tracks and Cells, Establishing Grid Containers, Sizing Grid Containers, Clamping Overly Large Grids, Grid Item Display, Grid Item Sizing, Reordered Grid Items, Grid Item Margins and Paddings, Z-axis Ordering, Implied Minimum Size of Grid Items, The Explicit Grid, Explicit Track Sizing, and Named Grid Lines. There are also sections on Disposition of Comments, Inline In Spec, GitHub Issues, Editors, Former Editors, and Copyright information.

The screenshot shows the W3C Candidate Recommendation page for the CSS Flexible Box Layout Module Level 1. The page title is "CSS Flexible Box Layout Module Level 1" and it was last updated on 19 October 2017. The left sidebar contains a "TABLE OF CONTENTS" with sections such as Introduction, Flex Layout Box Model and Terminology, Flex Containers, Flex Items, Ordering and Orientation, Flex Lines, Flexibility, Components of Flexibility, Alignment, and Abstract. The main content area includes sections on Overview, Module Interactions, Flex Container Properties, Flex Item Properties, Flex Direction and Wrap, Display Order, and Flex Lines. It also lists Test Suites, Editors, Former Editors, and Issue Tracking information. A sidebar on the right provides links to previous versions and a test suite.

If in doubt, consult the spec

The image shows two browser windows side-by-side. The left window is titled 'CSS Grid Layout - CSS | MDN' and displays the MDN Web Docs page for CSS Grid Layout. It includes a sidebar with navigation links like 'Related Topics' and 'CSS Reference', and a main content area with a 'Basic example' section containing sample HTML and CSS code. The right window is titled 'A Complete Guide to Flexbox' and shows the CSS-Tricks article by Chris Coyier. It features a sidebar with icons for Home, Snippets, and others, and the main content area has sections for 'Background' and 'Basics & Terminology'. A diagram at the bottom illustrates a flex container with three items, where the container is purple and the items are orange.

CSS Grid Layout

Related Topics
CSS

CSS Reference
CSS Grid Layout
Guides

Basics concepts of grid layout
Relationship to other layout methods
Line-based placement
Grid template areas
Layout using named grid lines
Auto-placement in grid layout
Box alignment in grid layout
Grids, logical values and writing modes
CSS Grid Layout and Accessibility
CSS Grid Layout and Progressive Enhancement
Resizing common layouts using grids

Properties
grid
grid-area
grid-auto-columns

HTML

```
1 <div class="wrapper">
2   <div class="one">One</div>
3   <div class="two">Two</div>
4   <div class="three">Three</div>
5   <div class="four">Four</div>
6   <div class="five">Five</div>
7   <div class="six">Six</div>
8 </div>
```

CSS

CSS Grid Layout excels at dividing a page into major regions, or defining the relationship in terms of size, position, and layer, between parts of a control built from HTML primitives.

Like tables, grid layout enables an author to align elements into columns and rows. However, many more layouts are either possible or easier with CSS grid than they were with tables. For example, a grid container's child elements could position themselves so they actually overlap and layer, similar to CSS positioned elements.

Basic example

The below example shows a three column track grid with new rows created at a minimum of 100 pixels and a maximum of auto. Items have been placed onto the grid using line-based placement.

Background

Basics & Terminology

container

items

Or MDN, CSS Tricks

layout modes

layout modes

inline

block

table

positioned

layout modes

inline

block

table

positioned

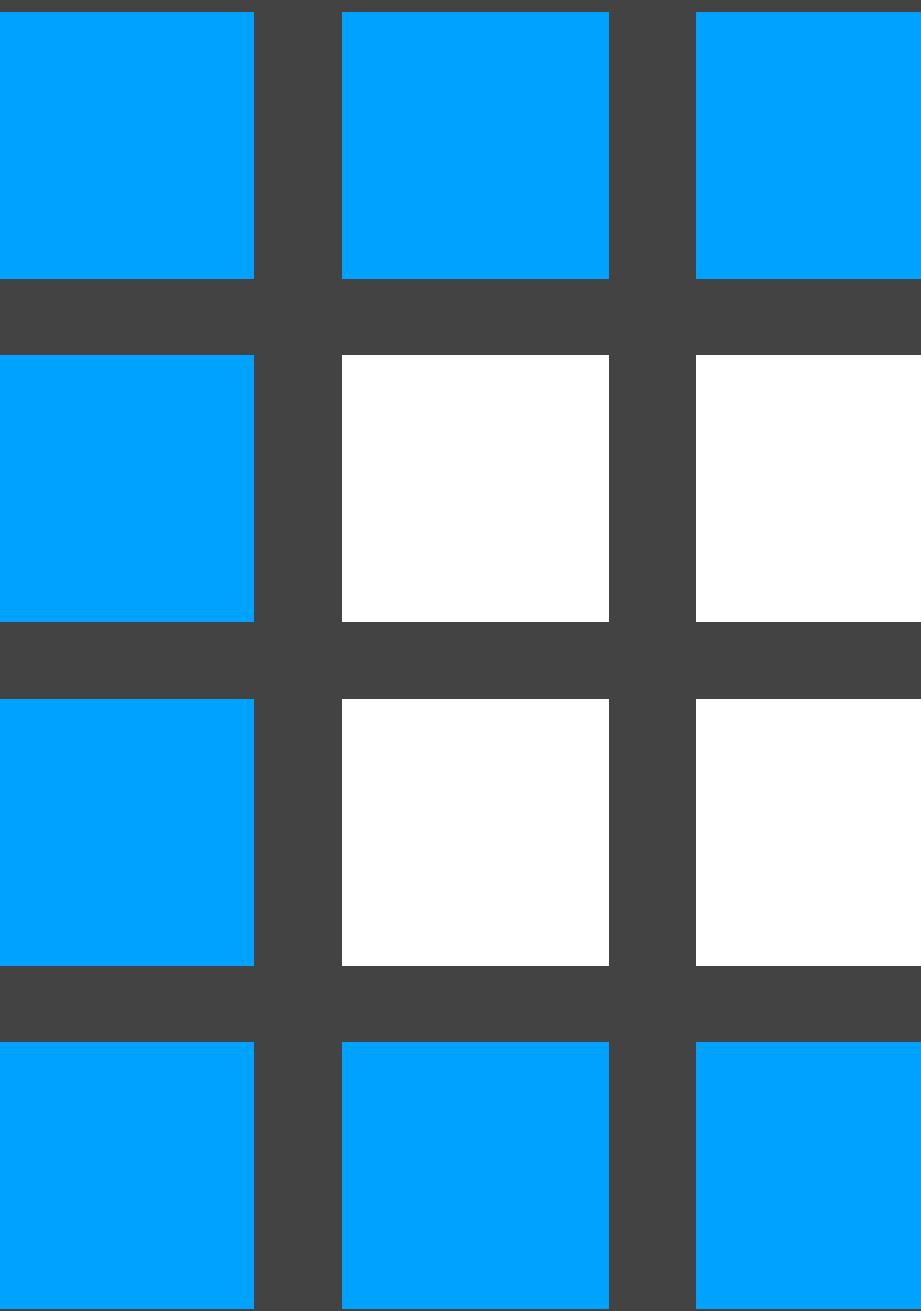
flex

grid

flex



grid



flex

from the
content out

sizes defined
on the item

grid

from the
container

areas defined on
the container

logical vs physical properties

BBC Arabic - الرئيسية

BBC Arabic - الرئيسية

www.bbc.com/arabic

Search

قائمة

BBC

تسجيل الدخول

عربي

الرئيسية | شرق أوسط | عالم | علوم وتكنولوجيا | صحة | اقتصاد | فنون | رياضة | مجلة | مرأة | فيديو | صحافة | صور | براماجنا | ترند | حوارات | المزيد

دخول مساعدات غذائية للغوطة رغم استمرار القتال والقصف

وكالات إغاثية أممية، تعانى دخول قافلة مساعدات إلى عدة مناطق بالغوطة الشرقية التي تشهد قتالاً عنيفاً بين قصائل وقوات الجيش السوري، وذلك رغم استمرار القصف والقارات الجوية للقوات الحكومية.

قبل 2 ساعة

قافلة مساعدات إنسانية تدخل الغوطة الشرقية



القوات التركية "قد تدخل عفرين في أي لحظة"

9 مارس / آذار 2018



الحكم يسجن "أكثر شخص مكره في أمريكا" 7 سنوات

محكمة أمريكية تحكم على قطب صناعة الأدوية السابق مارتن شكريلي، الذي أدين بالاحتيال على مجموعة من المستثمرين، بالسجن سبع سنوات.

قبل 5 ساعات

سجن رجل أعمال أمريكي عرض "مكافأة مقابل خصلة من شعر هيلاري كلينتون"

قظر توقيع اتفاقاً لدوره 24 مقالة "تايفون" من بريطانيا

كيف باع "بي إيه اي سيستمز" البريطانية أنظمة مراقبة منظورة لدول عربية؟

فيديو <

مباشر

ملخص تلفزيون

ملخص راديو

آخرنا لكم



ترامب وكيم جونغ أون: تاريخ من السباب المتبادل

مقامرة القرن: هل خدع زعيم كوريا الشمالية ترامب ومومن؟



ترامب ومومن؟

لماذا تزيد رواتب المغتربين في مومباي عن أي



www.bbc.com/arabic/middleeast-43353539

logical vs physical properties

`align-items: start`

`justify-self: end`

`margin-start / margin-end`

Logical properties and values specification

<https://www.w3.org/TR/css-logical-1/>

The screenshot shows a web browser displaying the 'CSS Logical Properties and Values Level 1' specification. The page is a W3C First Public Working Draft from May 18, 2017. It features a sidebar with a table of contents and a main content area with various sections and links.

Table of Contents (Sidebar):

- 1 Flow-Relative Values: 'block-start', 'block-end', 'inline-start', 'inline-end'
- 1.1 Logical Values for the 'caption-side' Property
- 1.2 Flow-Relative Values for the 'float' and 'clear' Properties
- 1.3 Flow-Relative Values for the 'text-align' Property
- 1.4 Flow-Relative Values for the 'resize' Property
- 2 Flow-Relative Page Classifications
- 3 Flow-Relative Box Model Properties
 - 3.1 Logical Height and Logical Width: the 'block-size' and 'inline-size' properties
 - 3.2 Flow-relative Margins: the 'margin-block-start', 'margin-block-end', 'margin-inline-start', 'margin-inline-end' properties and 'margin-block' and 'margin-inline' shorthands
 - 3.3 Flow-relative Offsets: the 'inset-block-start', 'inset-block-end', 'inset-inline-start', 'inset-inline-end' properties and 'inset-block', 'inset-inline', and 'inset' shorthands
 - 3.4 Flow-relative Padding: the 'padding-block-start', 'padding-block-end', 'padding-inline-start', 'padding-inline-end' properties and 'padding-block' and 'padding-inline' shorthands
 - 3.5 Flow-relative Borders
 - 3.5.1 Flow-relative Border Widths: the 'border-block-start-width', 'border-block-end-width'

Main Content Area:

CSS Logical Properties and Values Level 1

W3C First Public Working Draft, 18 May 2017

This version: <https://www.w3.org/TR/2017/WD-css-logical-1-20170518/>

Latest published version: <https://www.w3.org/TR/css-logical-1/>

Editor's Draft: <https://drafts.csswg.org/css-logical/>

Issue Tracking:
[Inline In Spec](#)
[GitHub Issues](#)

Editors:
[Rossen Atanassov](#) (Microsoft)
[Elika J. Etemad / fantasai](#) (Invited Expert)

Copyright © 2017 W3C® (MIT, ERCIM, Keio, Beihang). W3C liability, trademark and document use rules apply.

Abstract

This module introduces logical properties and values that provide the author with the ability to control layout through logical, rather than physical, direction and dimension mappings. The module defines logical properties and values for the features defined in [CSS21]. These properties are writing-mode relative equivalents of their corresponding physical properties.

CSS is a language for describing the rendering of structured documents (such as HTML and XML) on screen, on paper, in speech, etc.

**Some CSS creates a
block formatting context.**

Some CSS creates a
block formatting context.

This means that from that point onwards, it is responsible for its own lay-out.

Sommige CSS creëert een
block formatting context.

It is a context in which things
are lined out.

Block formatting contexts are created by:

root element

float (behalve float: none)

position: absolute|fixed

display: inline-block

display: table-cell

display: table-caption

overflow (behalve overflow: visible)

Block formatting contexts are created by:

display: grid

display: flex

Takeaways

Flexibility is built into the web and CSS

CSS is not a programming languages

Consult the spec, or sources like MDN, CSS-Tricks

Flex and grid are two new layout modes

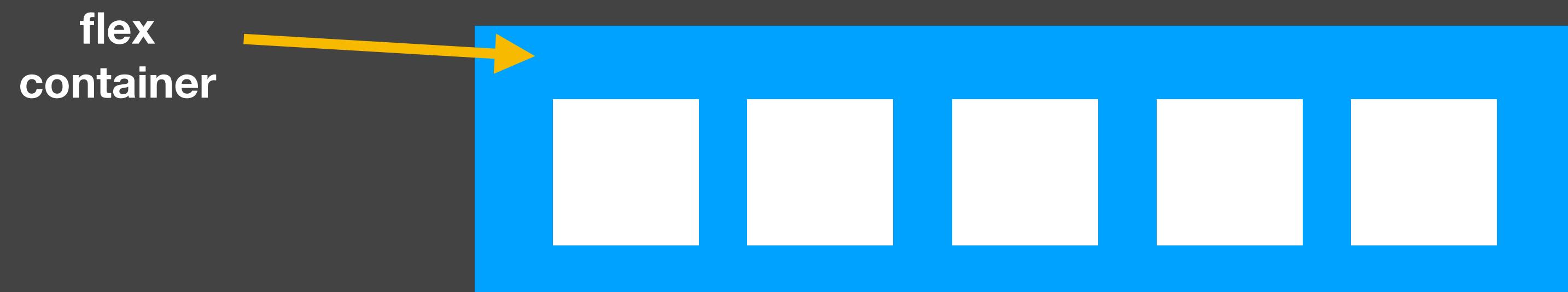
Logical properties replace physical properties

Some CSS creates a new BFC

Flexbox

**flexbox lets you align elements
and use whitespace intentionally**

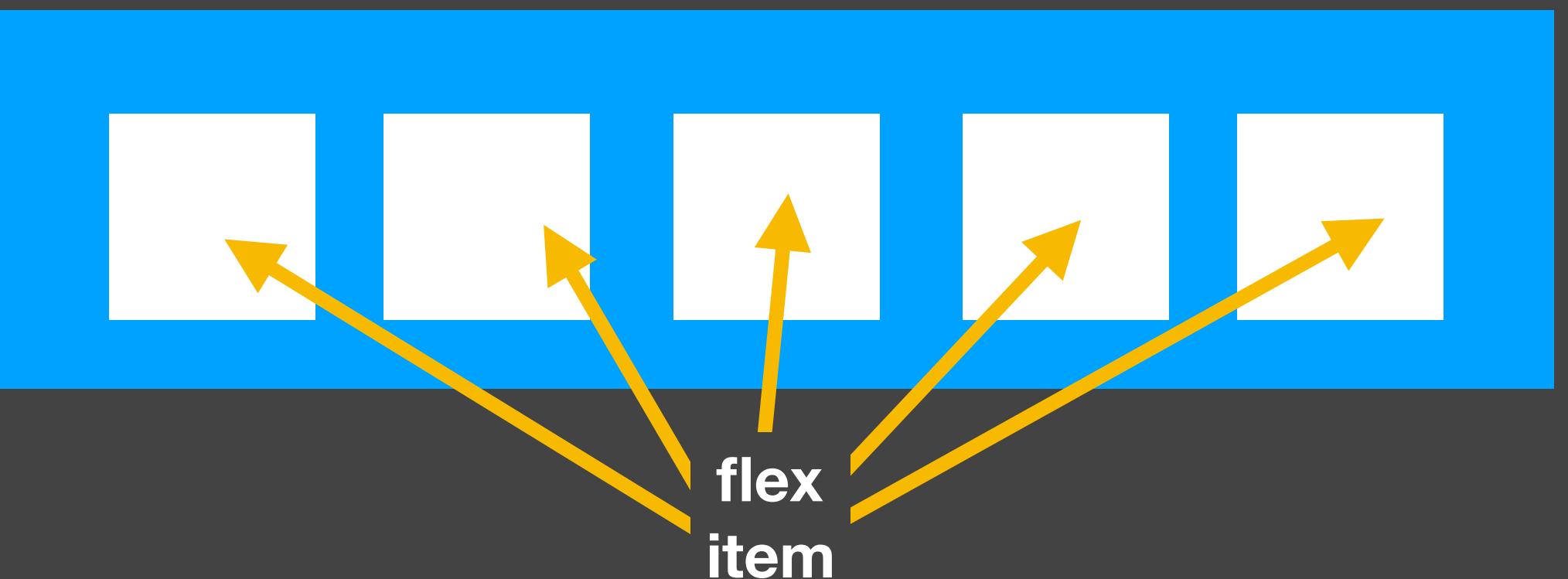
flex container



```
#container { display: flex; }
```

flex container

all direct children become flex items



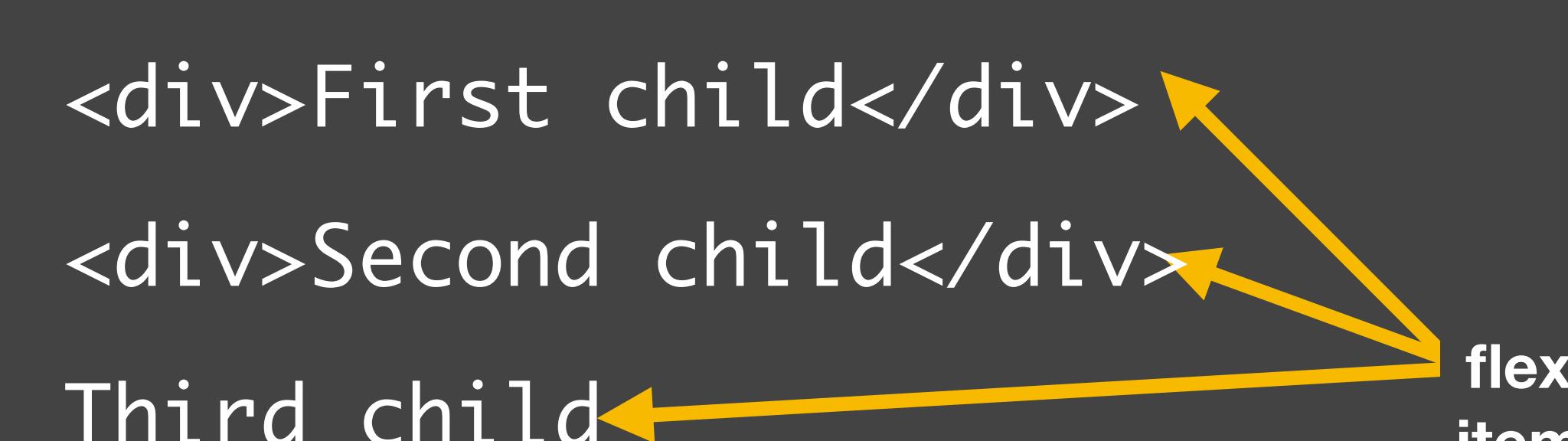
if they're
'in flow'
excludes
whitespace,
includes
loose text

```
#container { display: flex; }
```

flex container

all direct children become flex items

```
<div id="container">  
    <div>First child</div>  
    <div>Second child</div>  
    Third child  
</div>
```



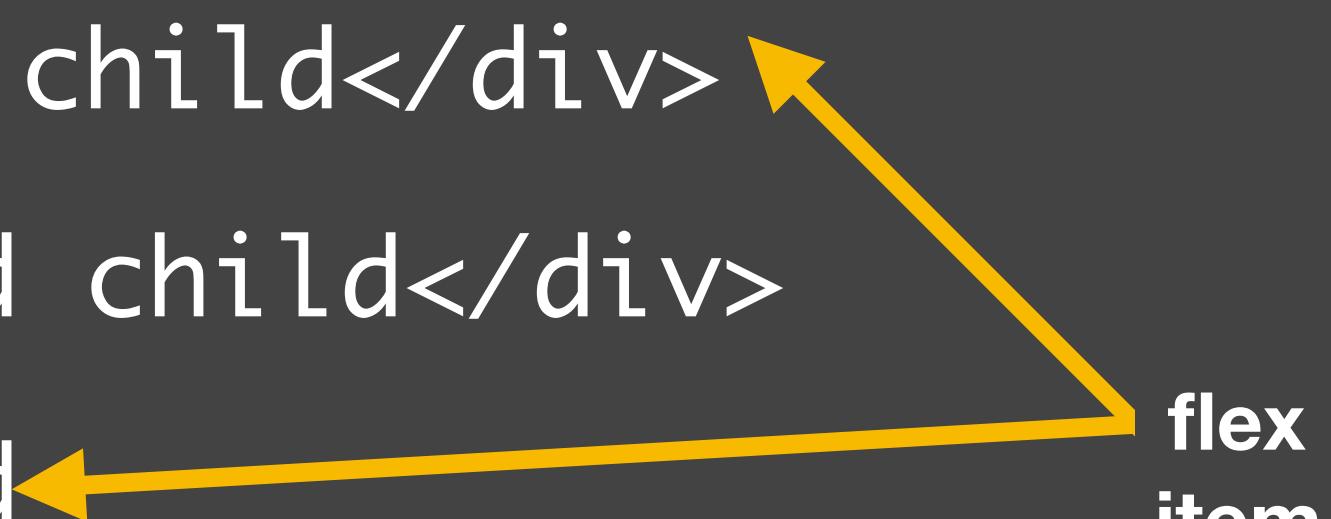
The diagram illustrates a flex container with three direct children: "First child", "Second child", and "Third child". Three yellow arrows point from the text "flex item" to each of these three children respectively.

```
#container { display: flex; }
```

flex container

all direct children become flex items

```
<div id="container">  
    <div>First child</div>  
    <div>Second child</div>  
    Third child  
</div>
```



A yellow arrow originates from the text "flex item" and points to the text "Third child".

```
#container { display: flex; }  
div:nth-child(2) { position: absolute; }
```

**when an element becomes
a flex item, some CSS will
behave differently**

When something becomes a...

flex item

stops working:

- vertical-align
- float

now works:

- flex
- automagical margins
- align-self
- order

flex

```
#item {  
  flex-grow: 0;           maximum growth factor  
  flex-shrink: 1;         maximum shrink factor  
  flex-basis: auto;       initial size  
}
```

flex

```
#item { flex: [grow] [shrink] [basis]; }
```

grow: max growth factor

shrink: maximum shrink factor

basis: initial size

**FLEX
ITEM**

flex

```
#item { flex: 0 1 auto; }
```

**FLEX
ITEM**

flex

```
#item { flex: 1; }
```

EXERCISE 01

Create a page with on it:

- a row of 3 columns
- a row of 4 columns
- a row of 5 columns

EXERCISE

**Lorem ipsum
dolor sit met**

Extra:
**Only do this on
viewports that are
50em or wider**

FLEX ITEM

```
#item { flex: initial; }  
// equivalent to '0 1 auto'
```

let item niet
shrink, but not
grow

```
#item { flex: auto; }  
// equivalent to '1 1 auto'
```

fully flexible -
let item grow as
well as shrink

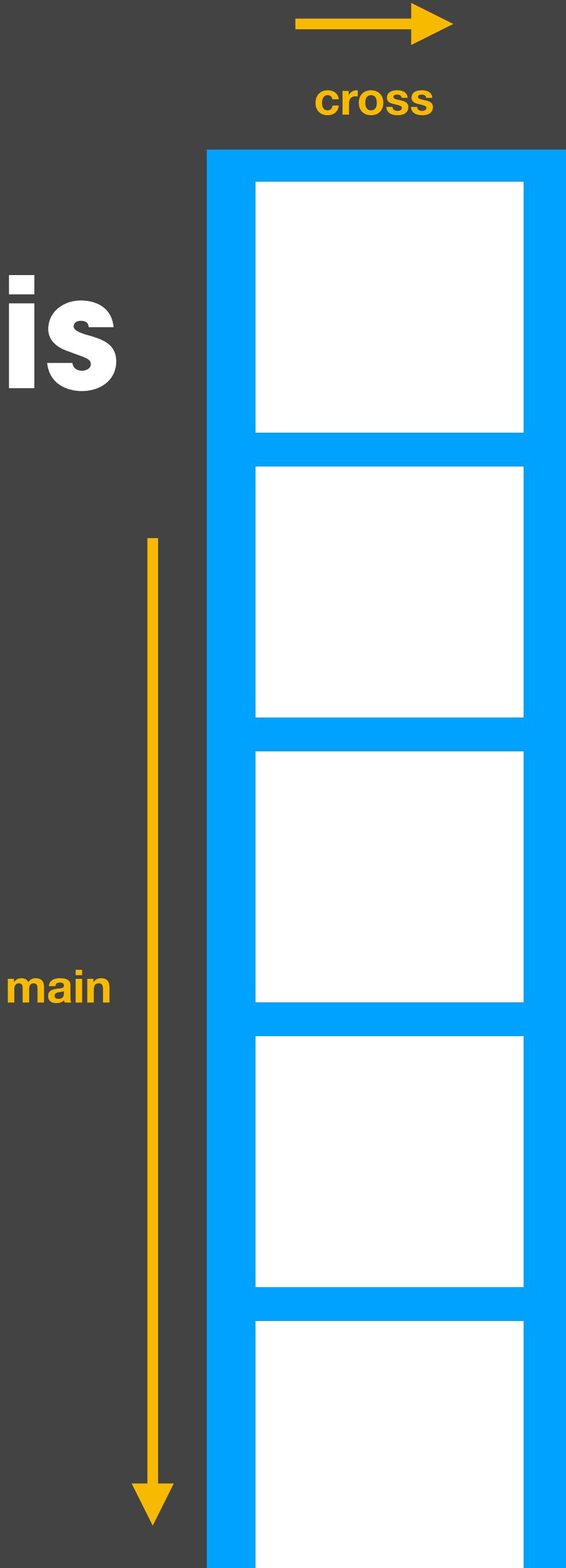
```
#item { flex: none; }  
// equivalent to '0 0 auto'
```

not flexible -
don't let item
shrink or grow

FLEX
CONTAINER

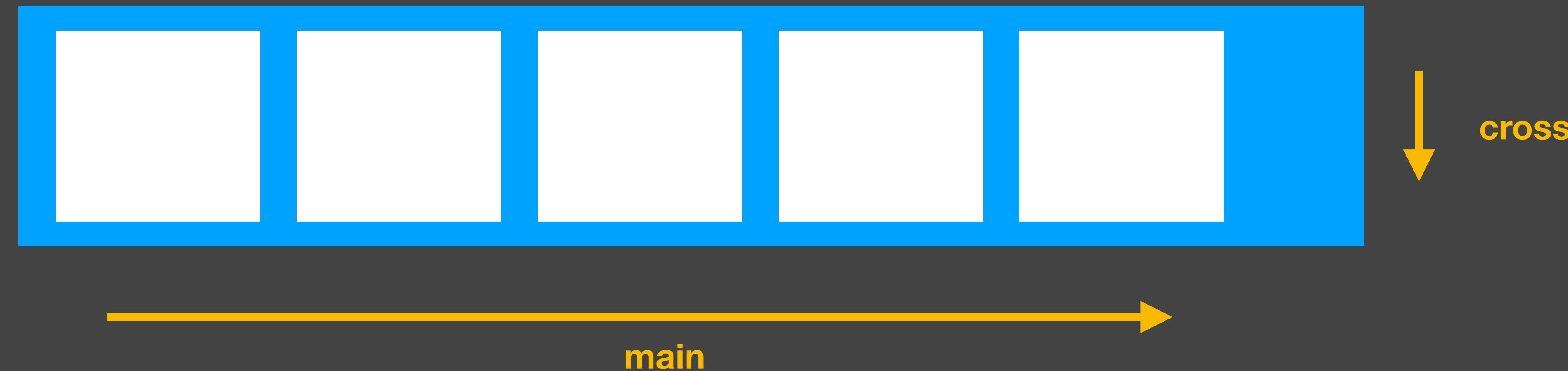
main axis vs cross axis

flex-direction: column;



main axis vs cross axis

flex-direction: row;



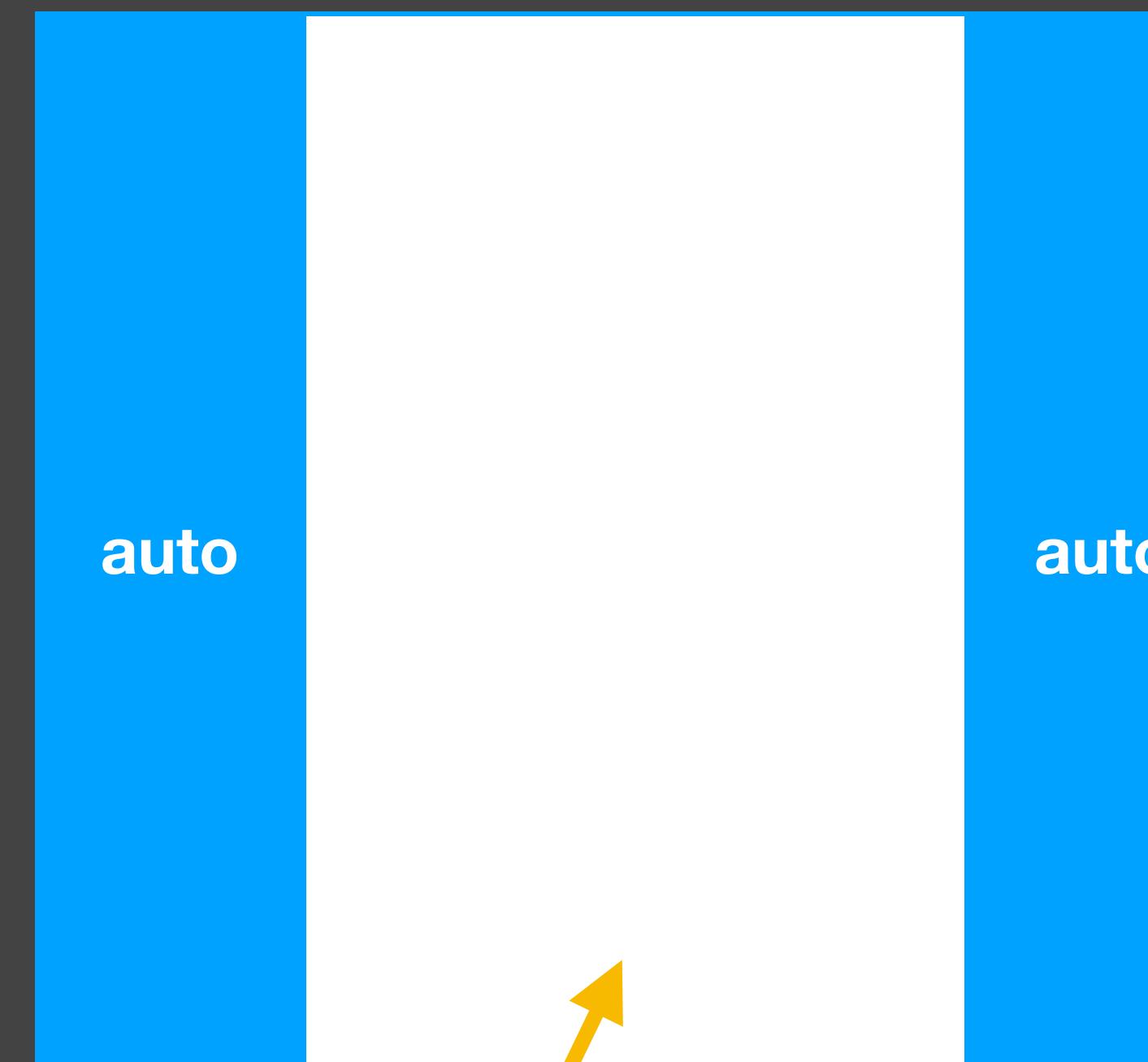
wrapping

**Do you want the items to wrap over
multiple lines or not?**

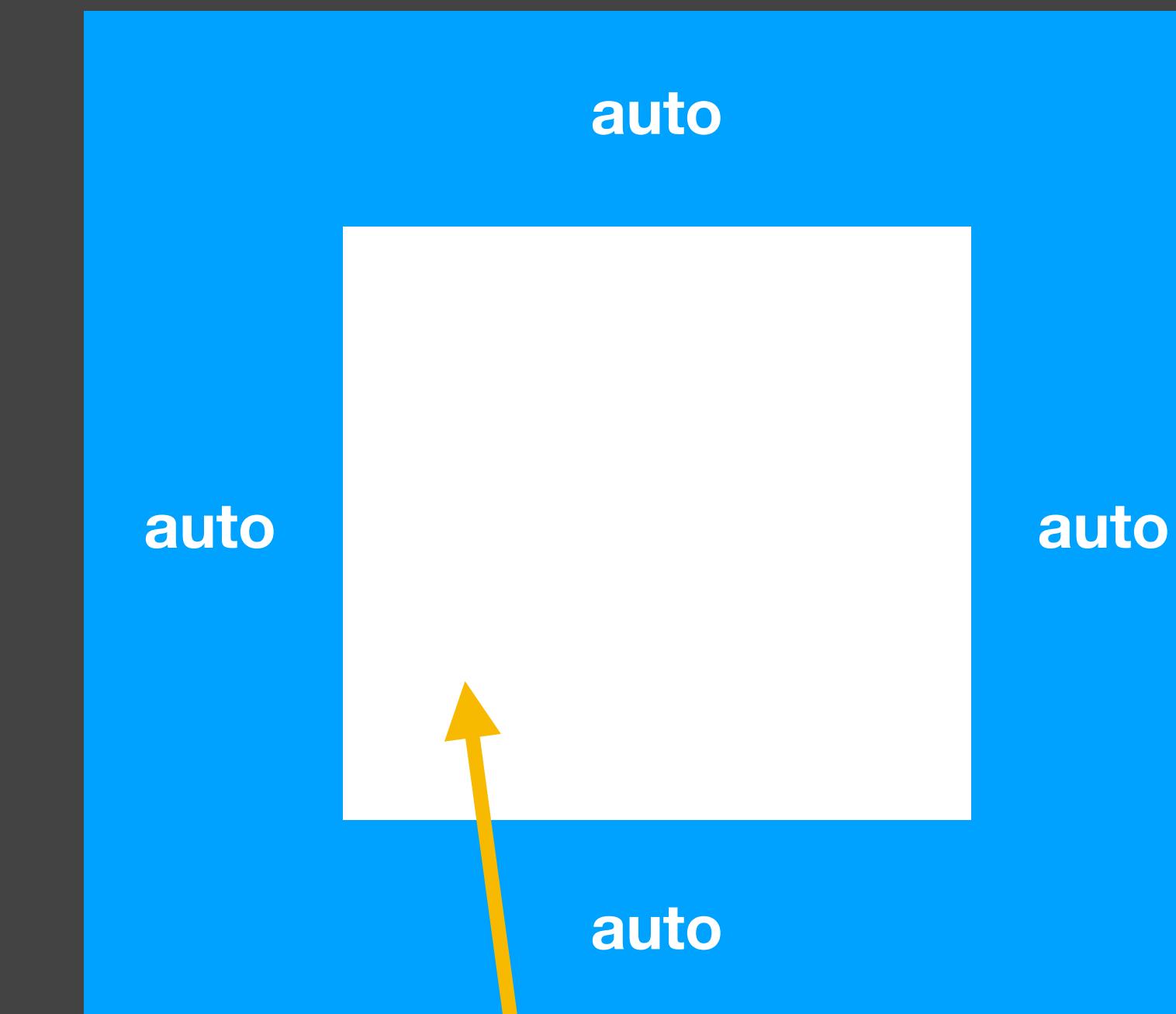
```
#container {  
    flex-wrap: nowrap | wrap;  
}
```

**FLEX
ITEM**

automagical margins



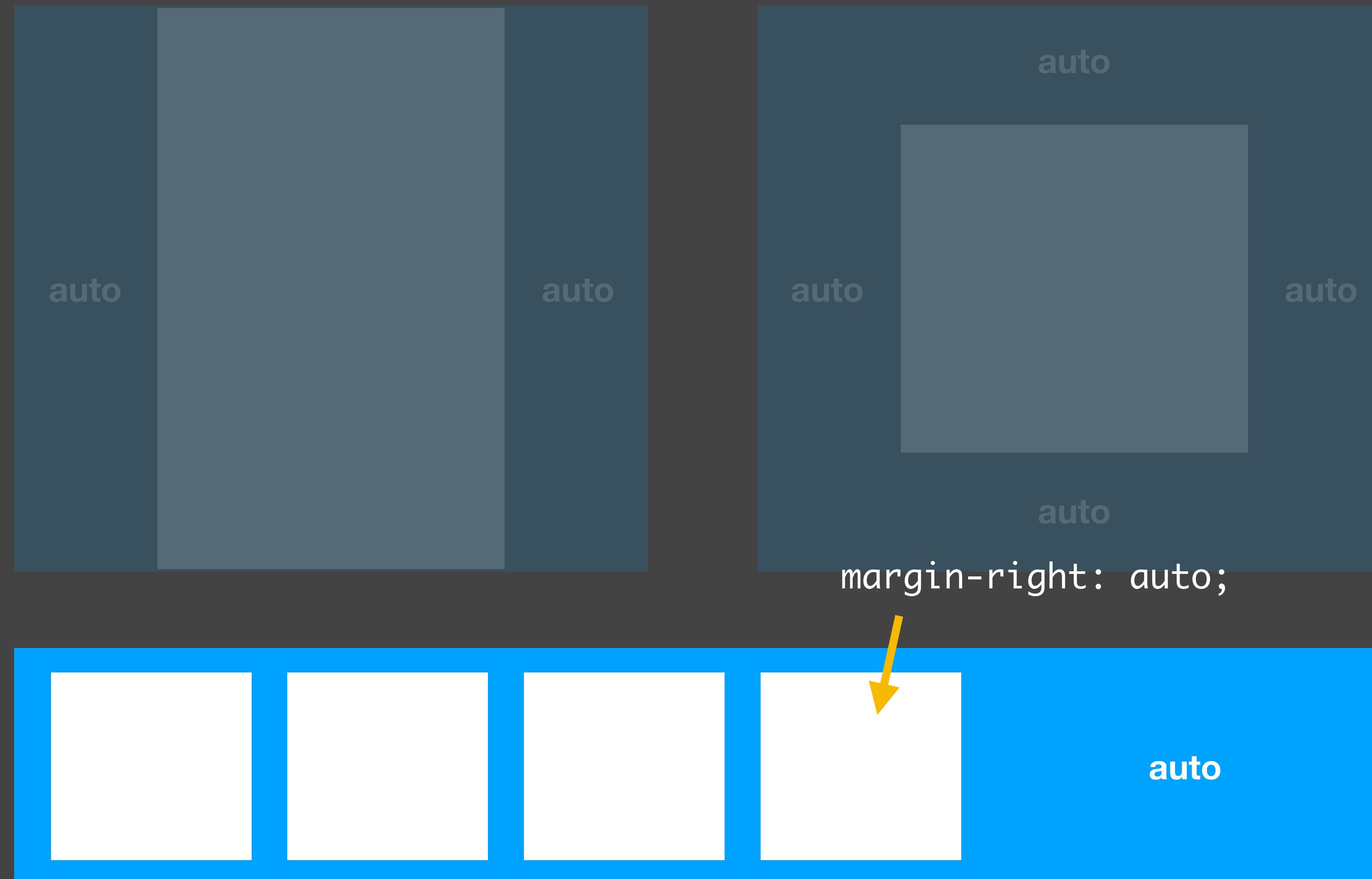
margin: 0 auto;



margin: auto;

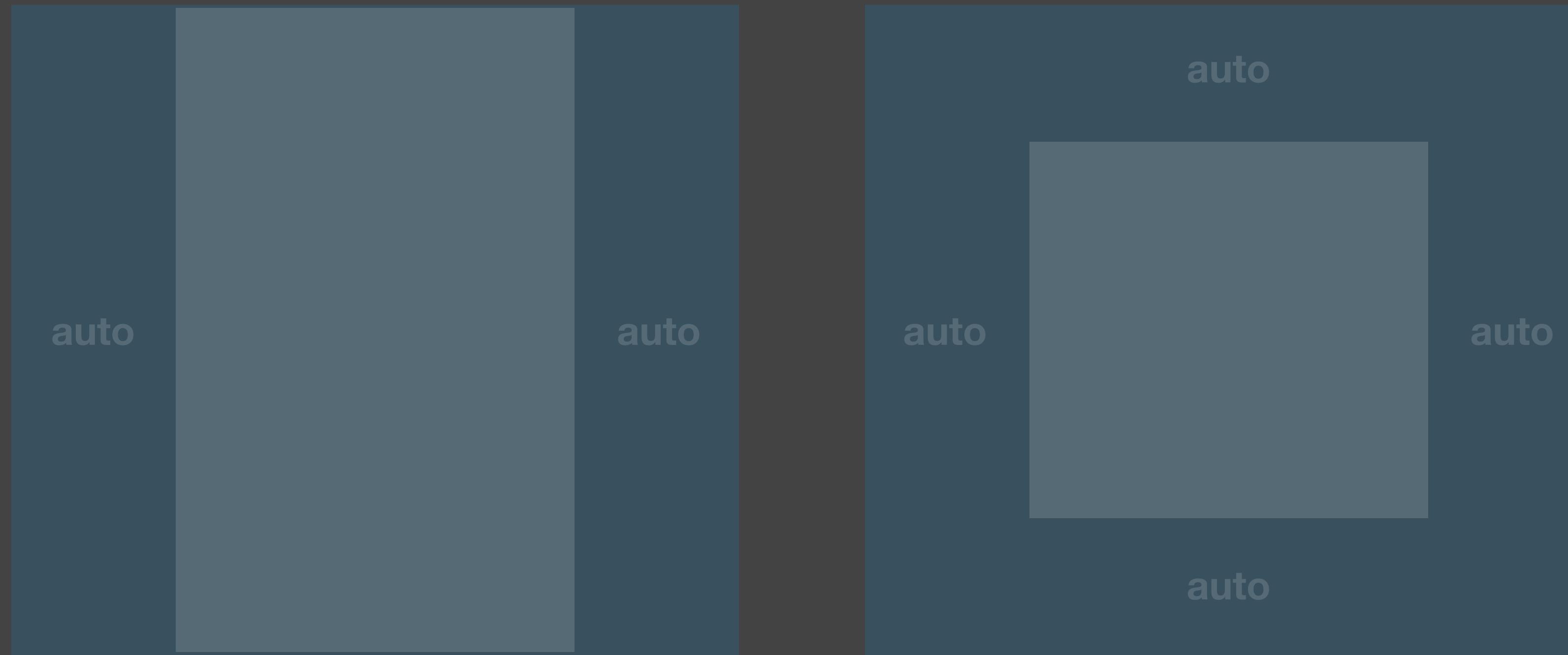
FLEX
ITEM

automagical margins



FLEX
ITEM

automagical margins

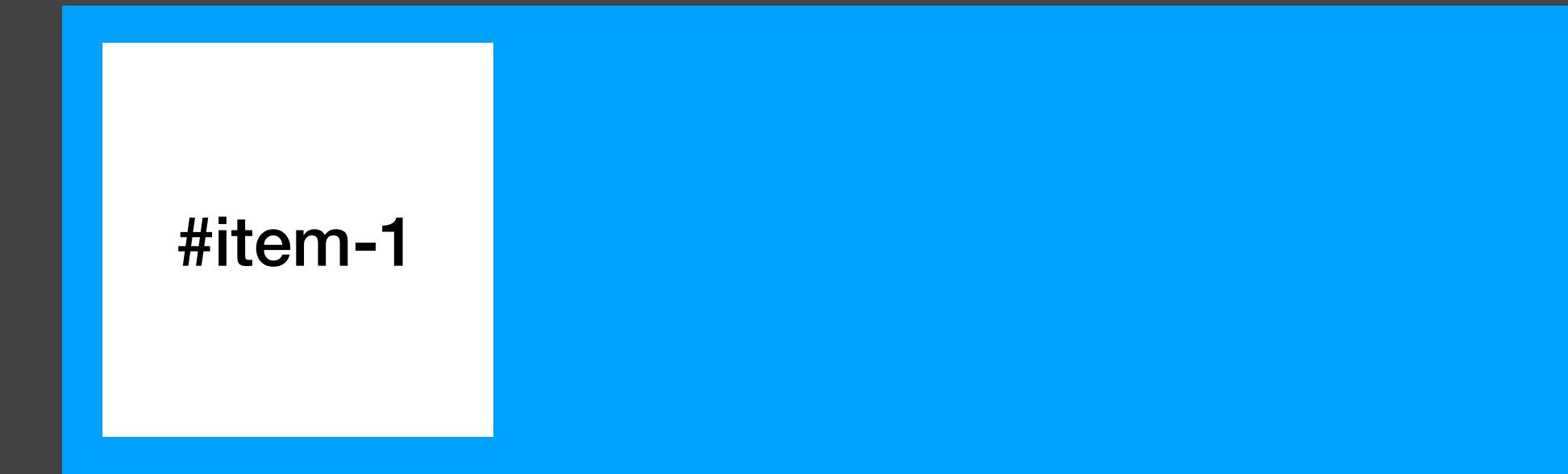


FLEX ITEM

aligning one item on the main axis

On columns if you're creating columns, on rows if you're creating rows.

Lets you align one item in a different way than the others, divides remaining space automatically.



```
#item-1 {  
  margin-left: 0;  
}
```

FLEX ITEM

aligning one item on the main axis

On columns if you're creating columns, on rows if you're creating rows.

Lets you align one item in a different way than the others, divides remaining space automatically.



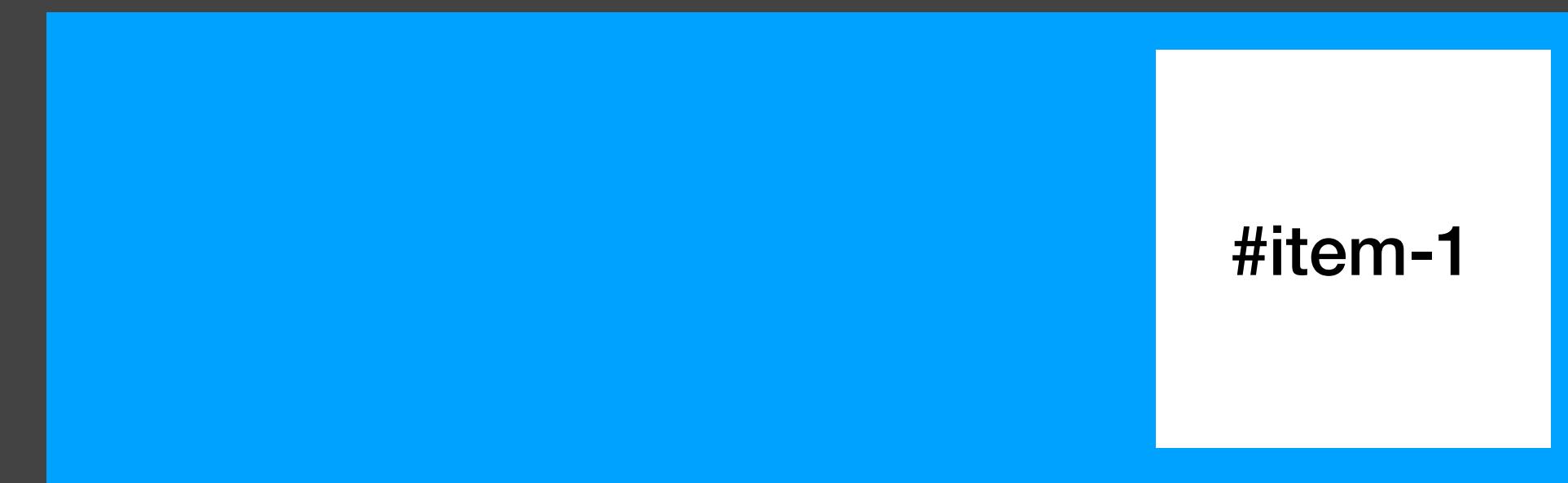
```
#item-1 {  
  margin-left: 1em;  
}
```

FLEX ITEM

aligning one item on the main axis

On columns if you're creating columns, on rows if you're creating rows.

Lets you align one item in a different way than the others, divides remaining space automatically.



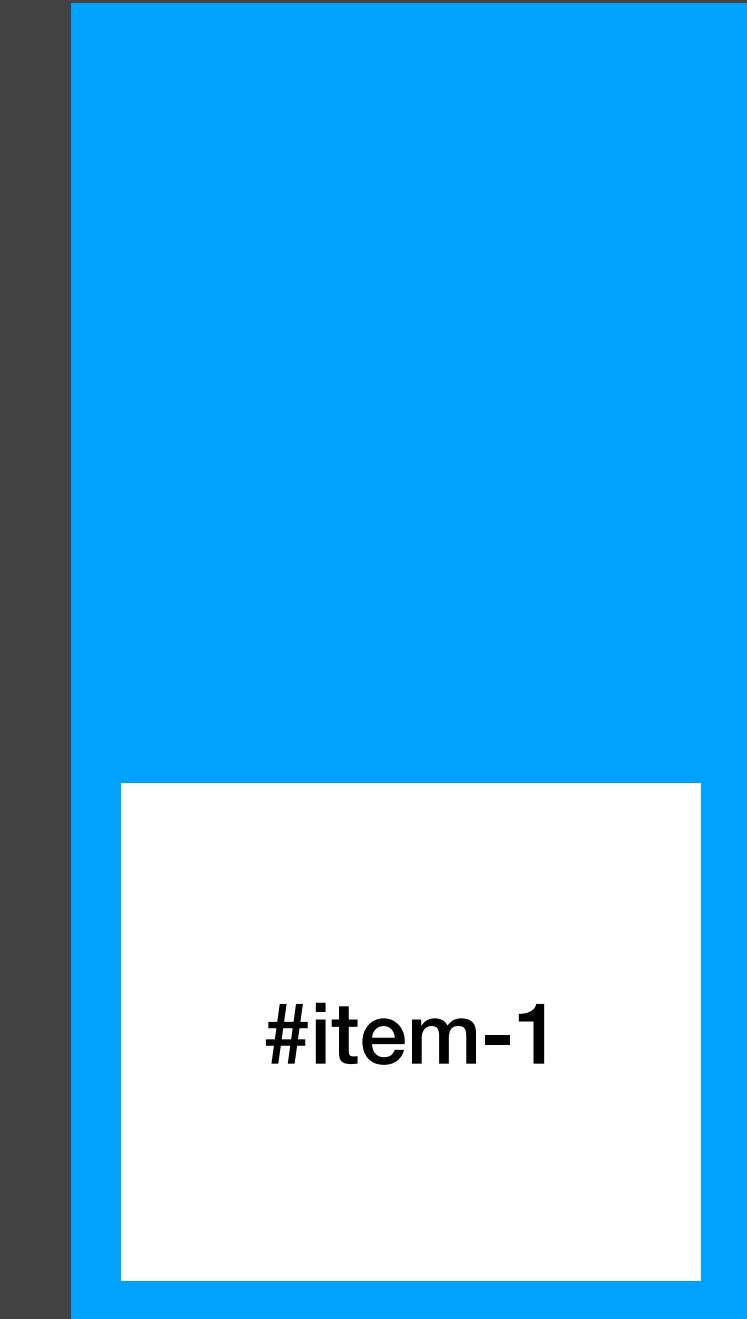
```
#item-1 {  
  margin-left: auto;  
}
```

FLEX ITEM

aligning one item on the main axis

On columns if you're creating columns, on rows if you're creating rows.

Lets you align one item in a different way than the others, divides remaining space automatically.



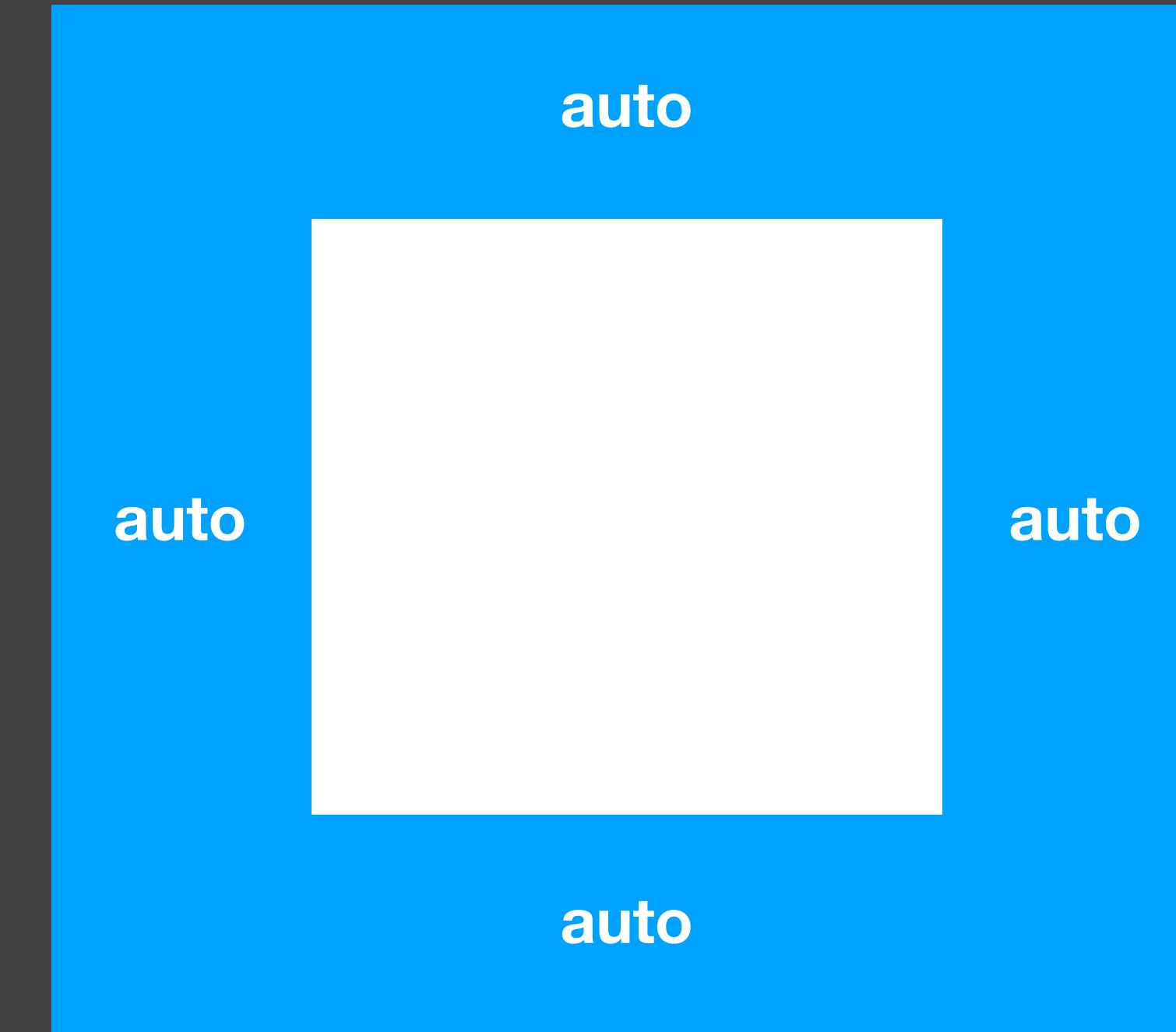
```
#item-1 {  
  margin-top: auto;  
}
```

FLEX ITEM

aligning one item on the main axis

On columns if you're creating columns, on rows if you're creating rows.

Lets you align one item in a different way than the others, divides remaining space automatically.



```
#item {  
  margin: auto;  
}
```

FLEX ITEM

aligning one item on the cross axis

On columns if you're creating rows, on rows
if you're creating columns.

Lets you align one item in a different way
than the others.



#item-1

```
#item {  
  align-self: flex-start  
  | flex-end | center |  
  baseline | stretch;  
}
```

FLEX ITEM

aligning one item on the cross axis

On columns if you're creating rows, on rows
if you're creating columns.

Lets you align one item in a different way
than the others.



#item-1

```
#item {  
  align-self: flex-start  
  | flex-end | center |  
  baseline | stretch;  
}
```

FLEX ITEM

aligning one item on the cross axis

On columns if you're creating rows, on rows
if you're creating columns.

Lets you align one item in a different way
than the others.



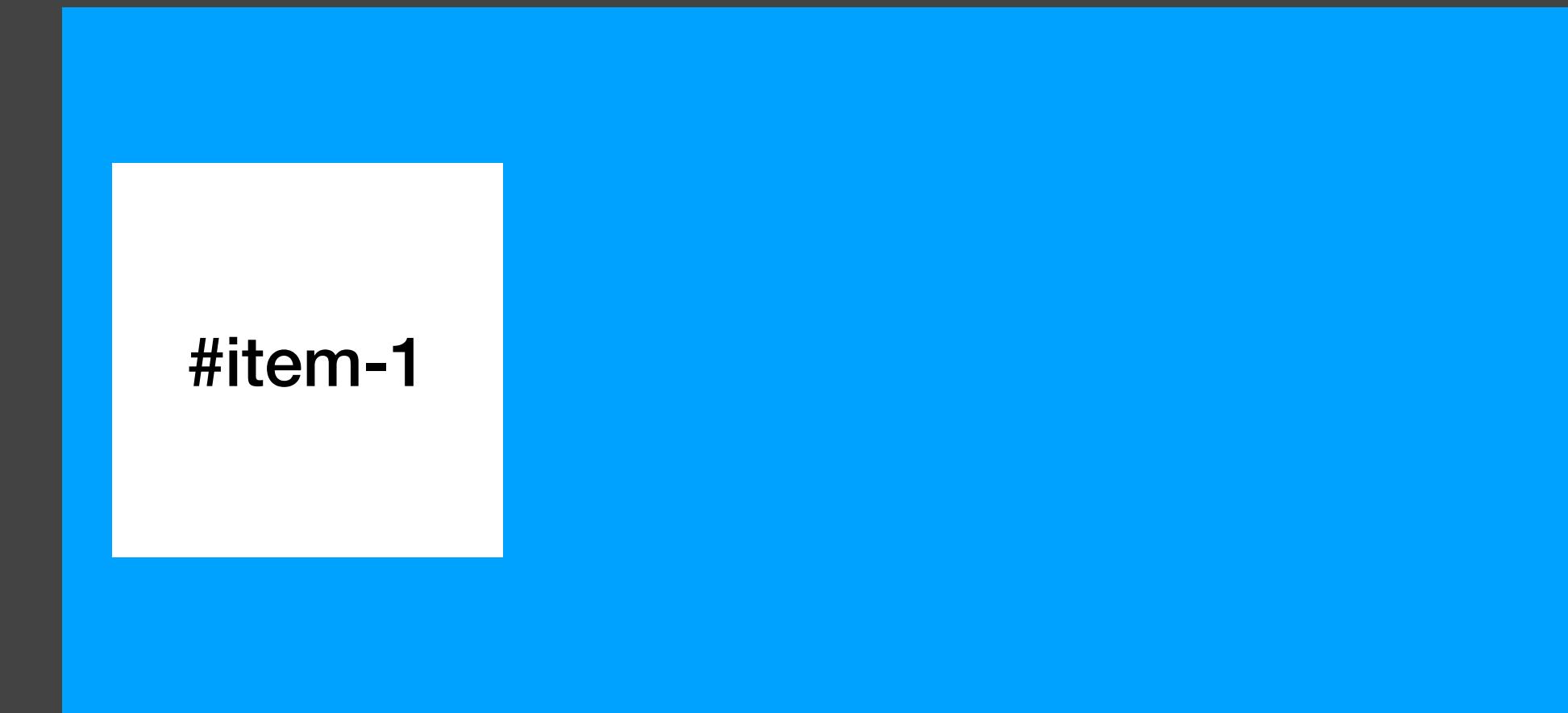
```
#item {  
  align-self: flex-start  
  | flex-end | center |  
  baseline | stretch;  
}
```

FLEX ITEM

aligning one item on the cross axis

On columns if you're creating rows, on rows
if you're creating columns.

Lets you align one item in a different way
than the others.



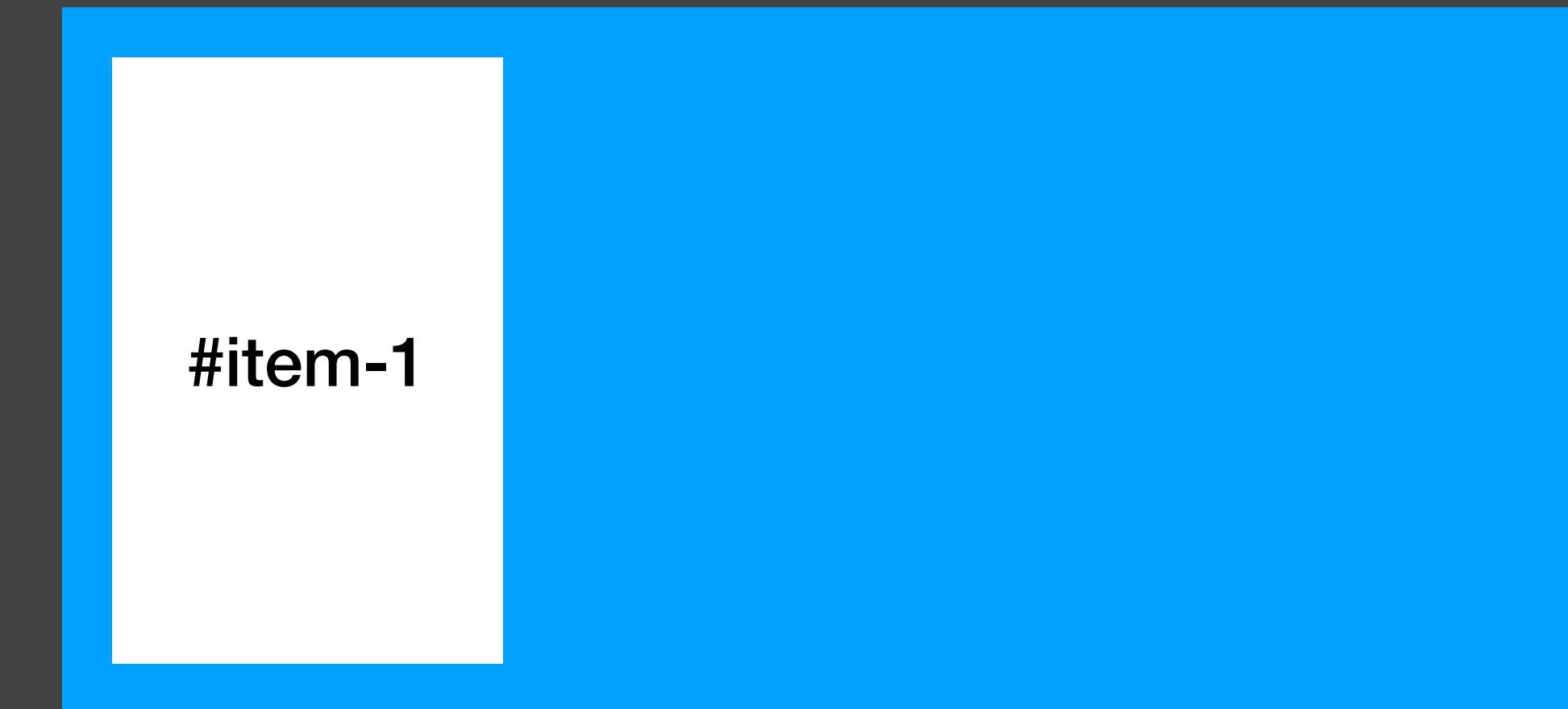
```
#item {  
  align-self: flex-start  
  | flex-end | center |  
  baseline | stretch;  
}
```

FLEX ITEM

aligning one item on the cross axis

On columns if you're creating rows, on rows
if you're creating columns.

Lets you align one item in a different way
than the others.



```
#item {  
  align-self: flex-start  
  | flex-end | center |  
  baseline | stretch;  
}
```

FLEX CONTAINER

FLEX CONTAINER

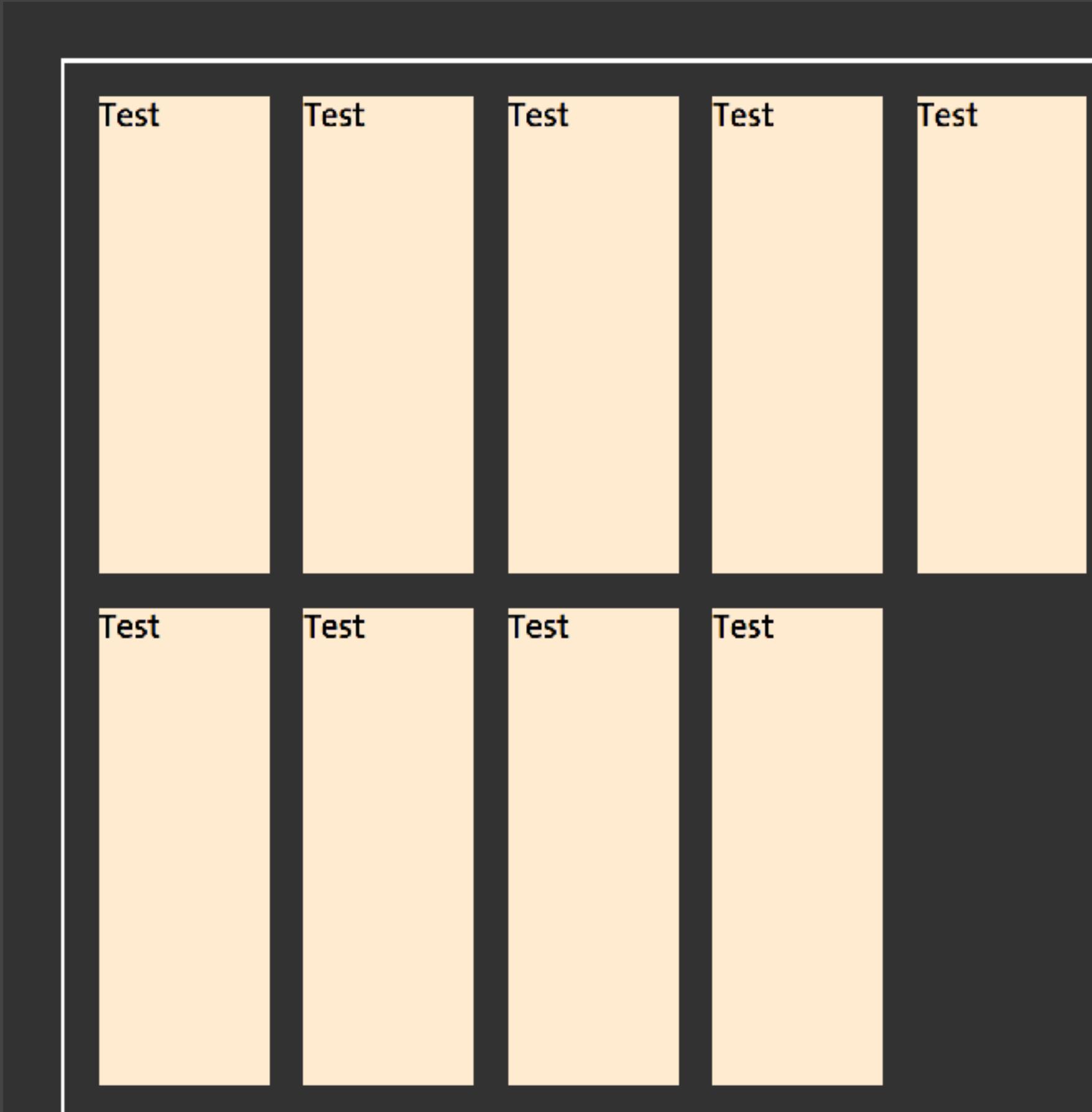
aligning content on the main axis

On columns if you're creating columns, on rows if you're creating rows.

Move all content this container has into one position.

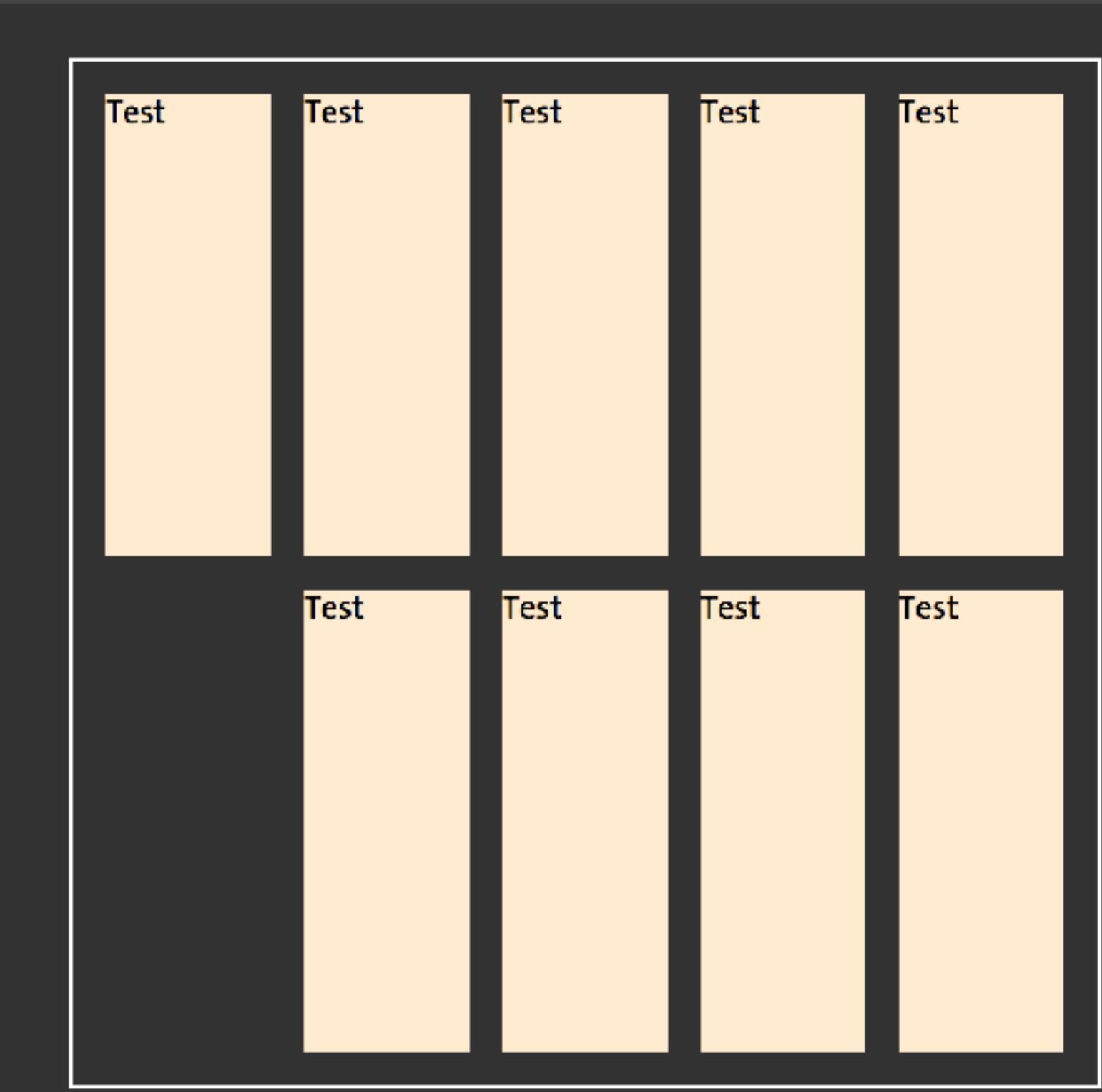
```
#container {  
  justify-content:  
    flex-start | flex-  
    end | center |  
    space-between |  
    space-around;  
}
```

FLEX CONTAINER



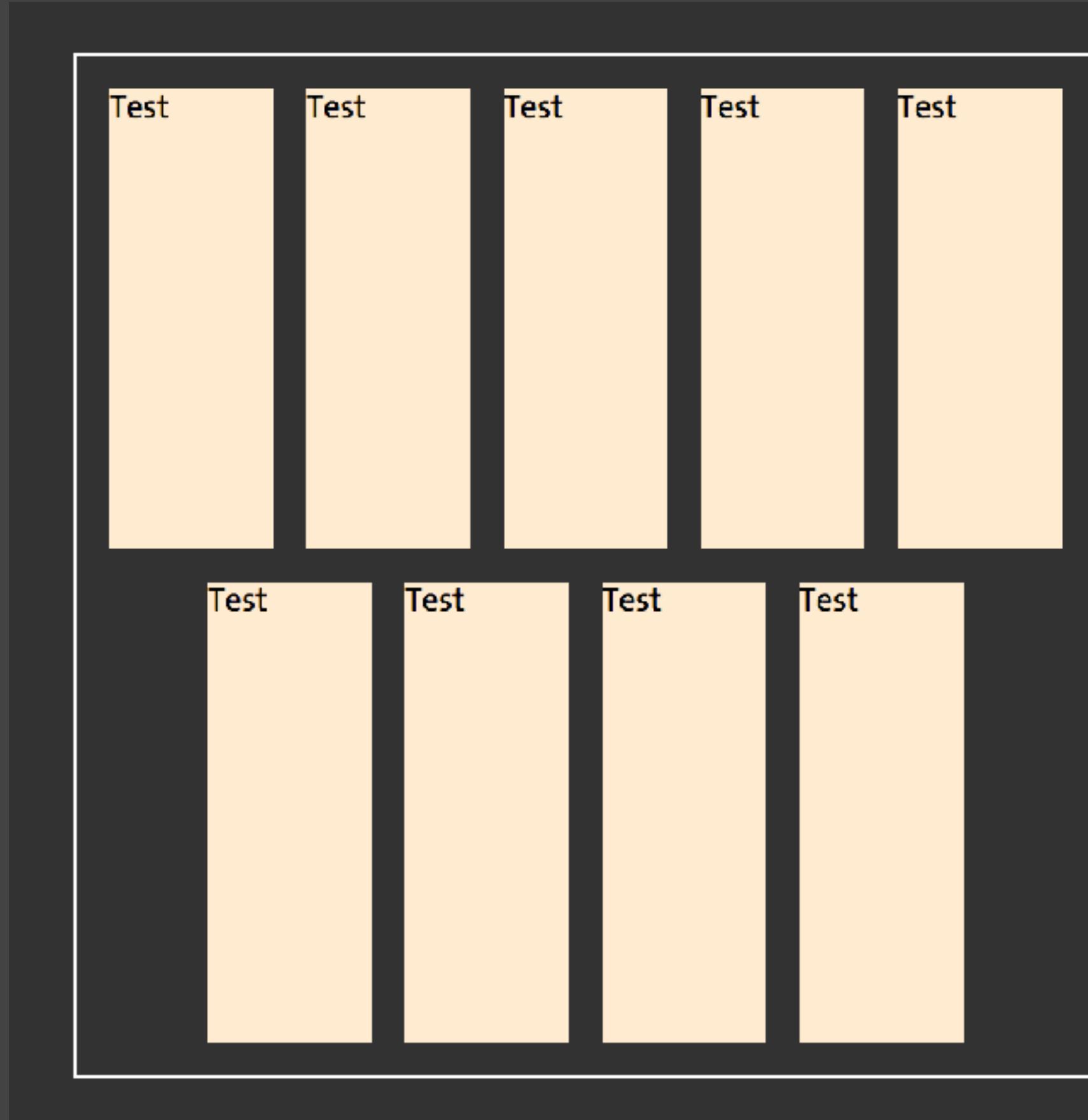
```
#container {  
  justify-content:  
    flex-start | flex-  
    end | center |  
    space-between |  
    space-around;  
}
```

FLEX CONTAINER



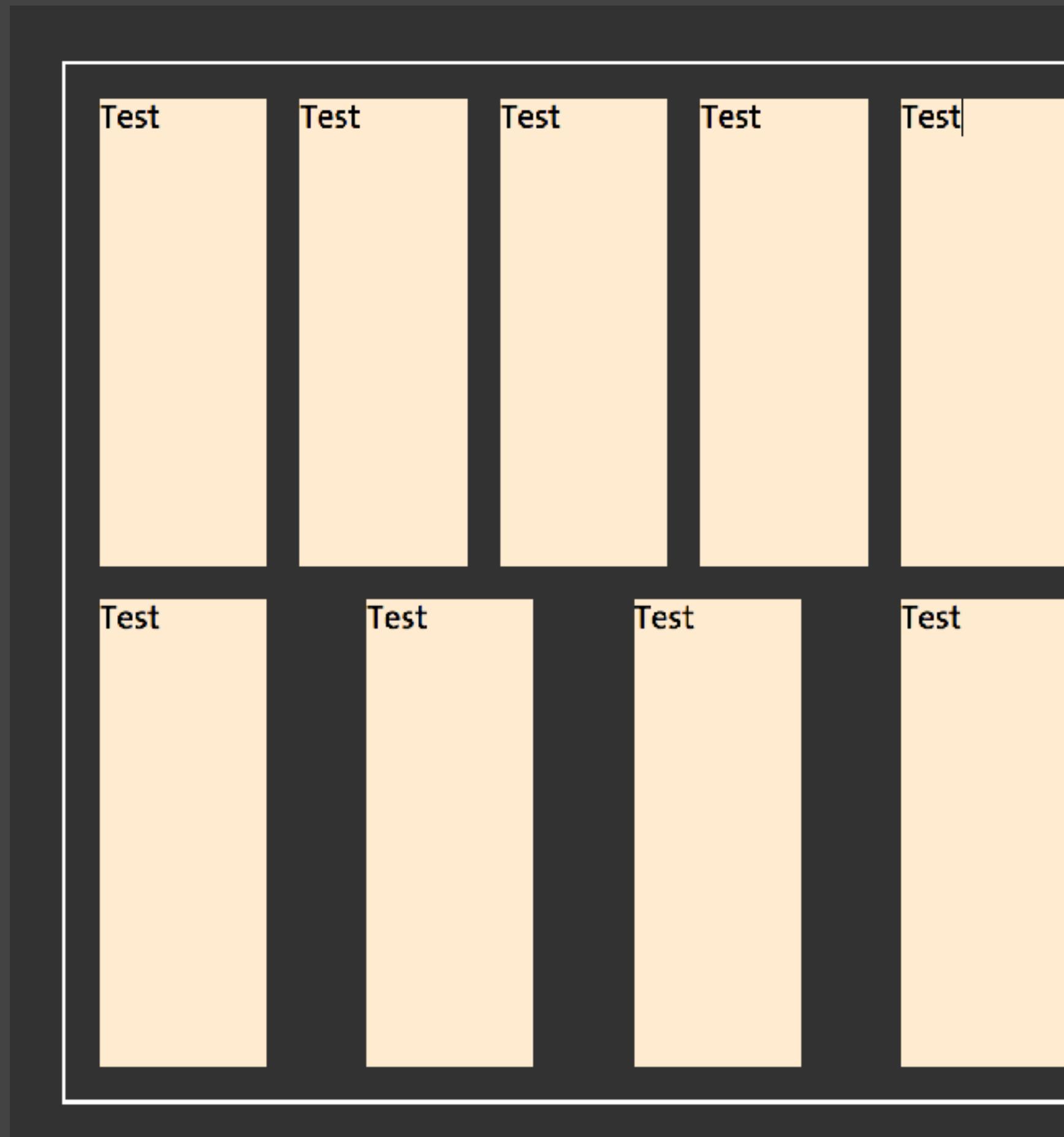
```
#container {  
  justify-content:  
    flex-start | flex-  
    end | center |  
    space-between |  
    space-around;  
}
```

FLEX CONTAINER



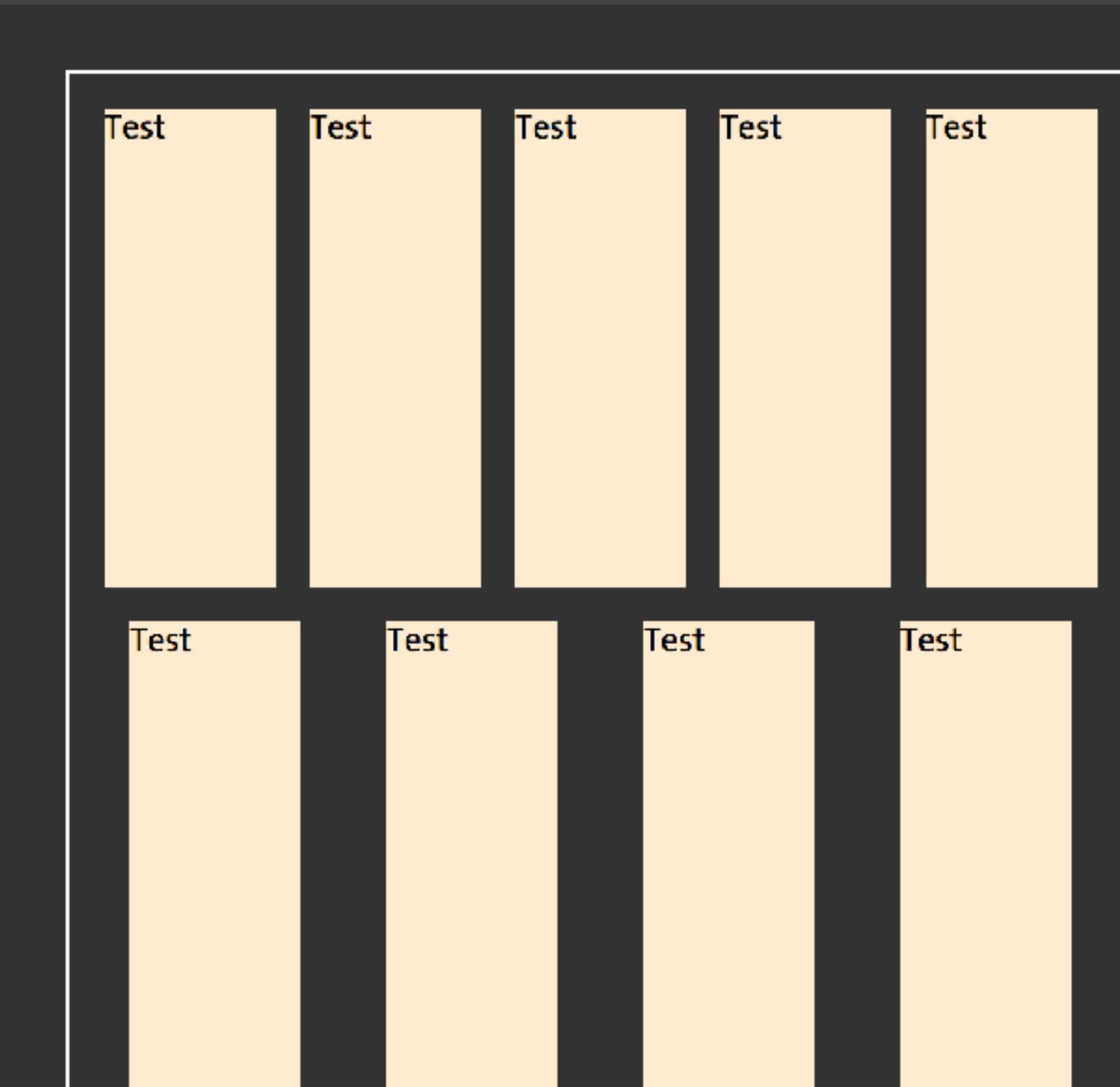
```
#container {  
  justify-content:  
    flex-start | flex-  
    end | center |  
    space-between |  
    space-around;  
}
```

FLEX CONTAINER



```
#container {  
  justify-content:  
    flex-start | flex-  
    end | center |  
    space-between |  
    space-around;  
}
```

FLEX CONTAINER



```
#container {  
  justify-content:  
    flex-start | flex-  
    end | center |  
    space-between |  
    space-around;  
}
```

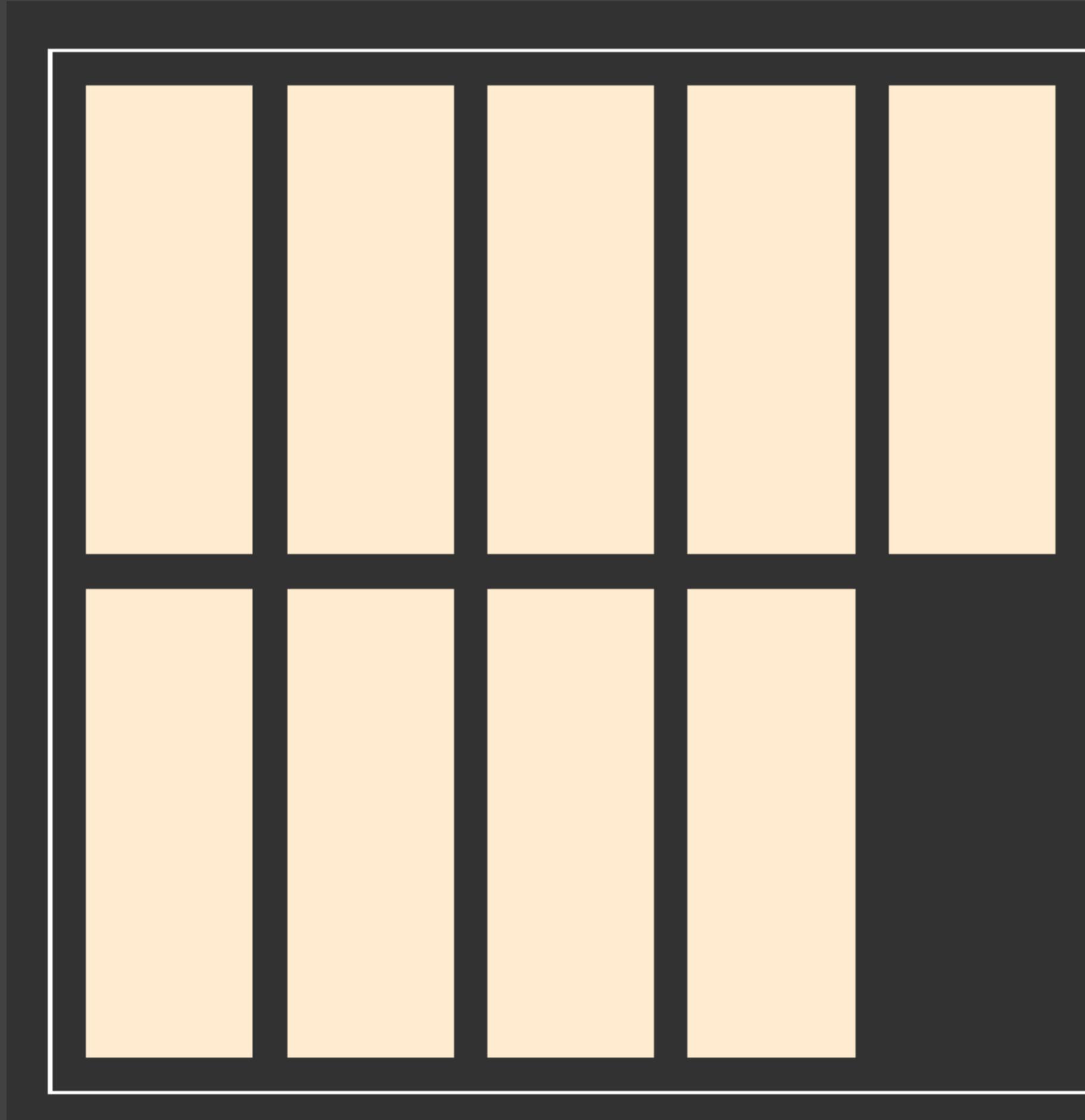
FLEX CONTAINER

aligning content on the cross axis

**Works on columns if you're creating rows,
works on rows if you're creating columns.**

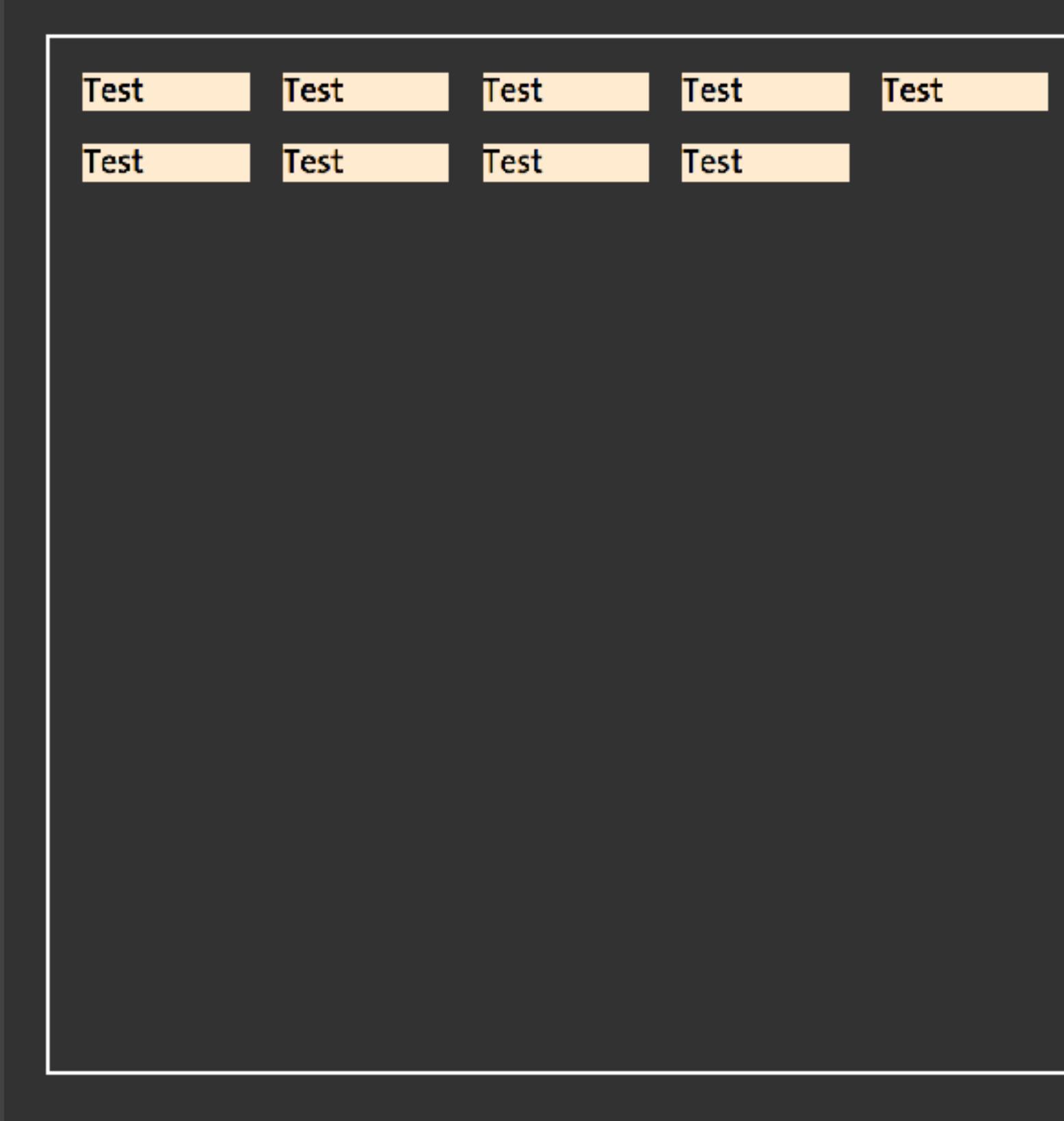
```
#container {  
  align-content: stretch  
  | flex-start | flex-  
  end | center | space-  
  between | space-  
  around;  
}
```

FLEX CONTAINER



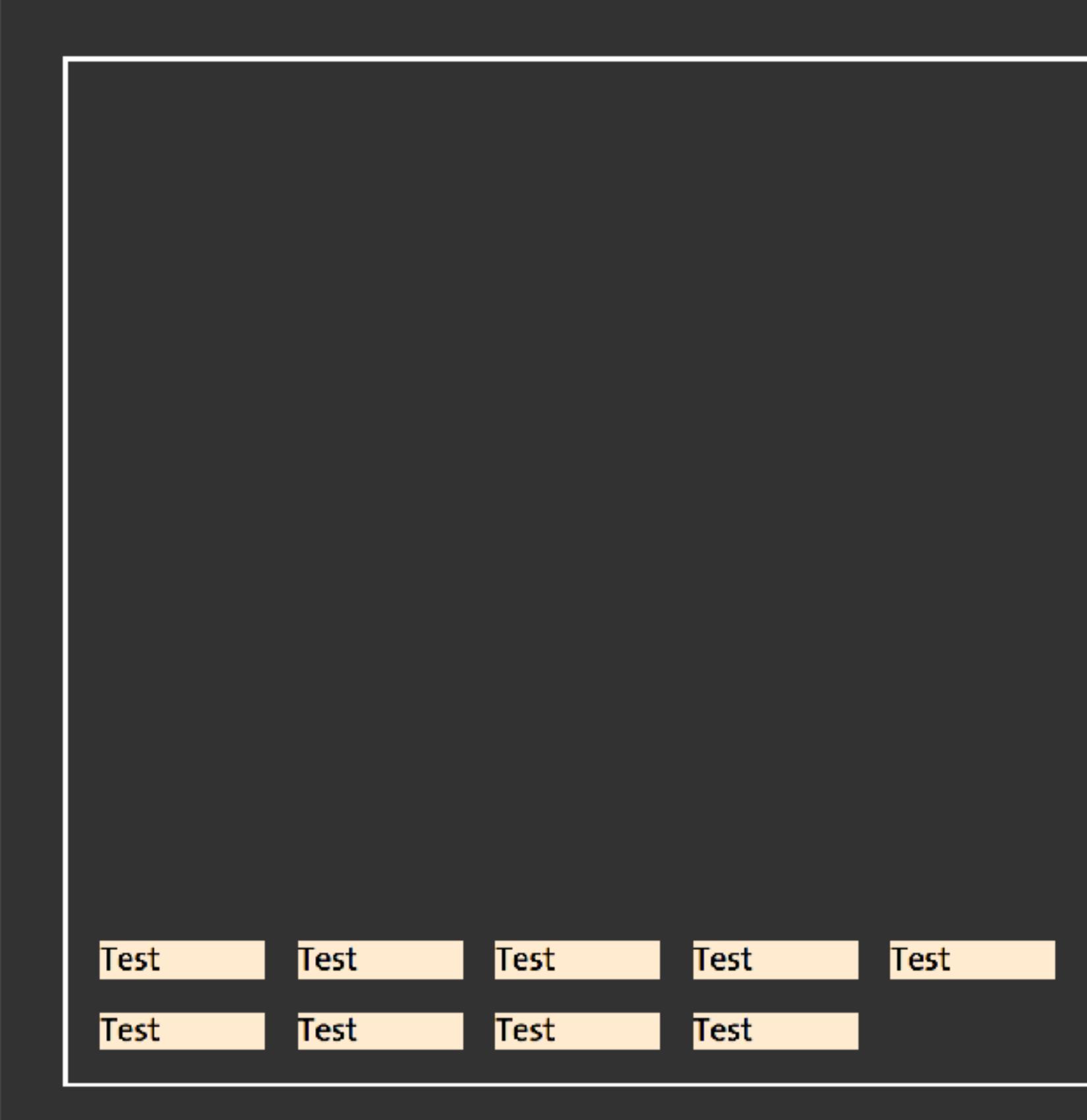
```
#container {  
  align-content: stretch  
  | flex-start | flex-  
  end | center | space-  
  between | space-  
  around;  
}
```

FLEX CONTAINER



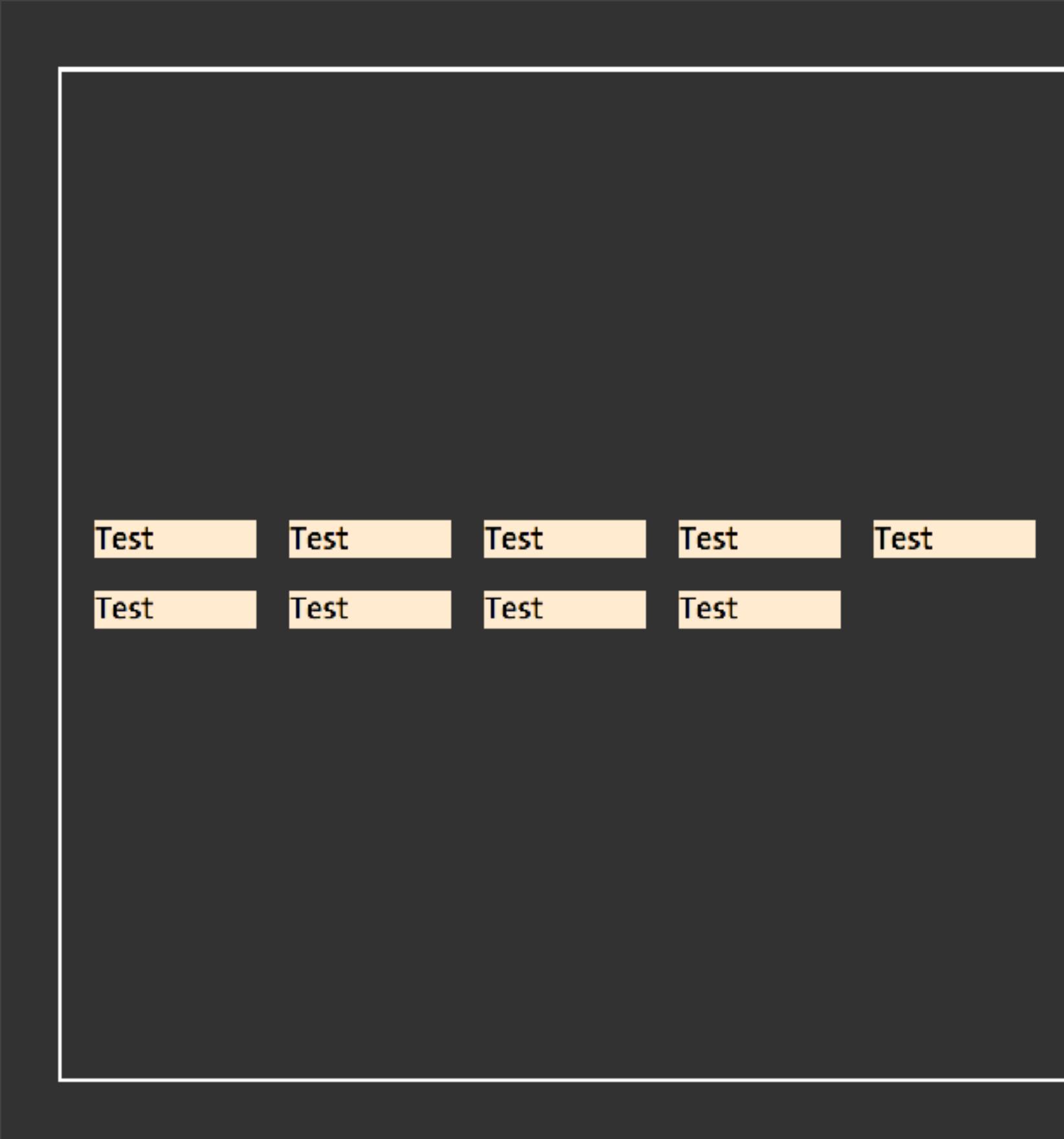
```
#container {  
  align-content: stretch  
  | flex-start | flex-  
  end | center | space-  
  between | space-  
  around;  
}
```

FLEX CONTAINER



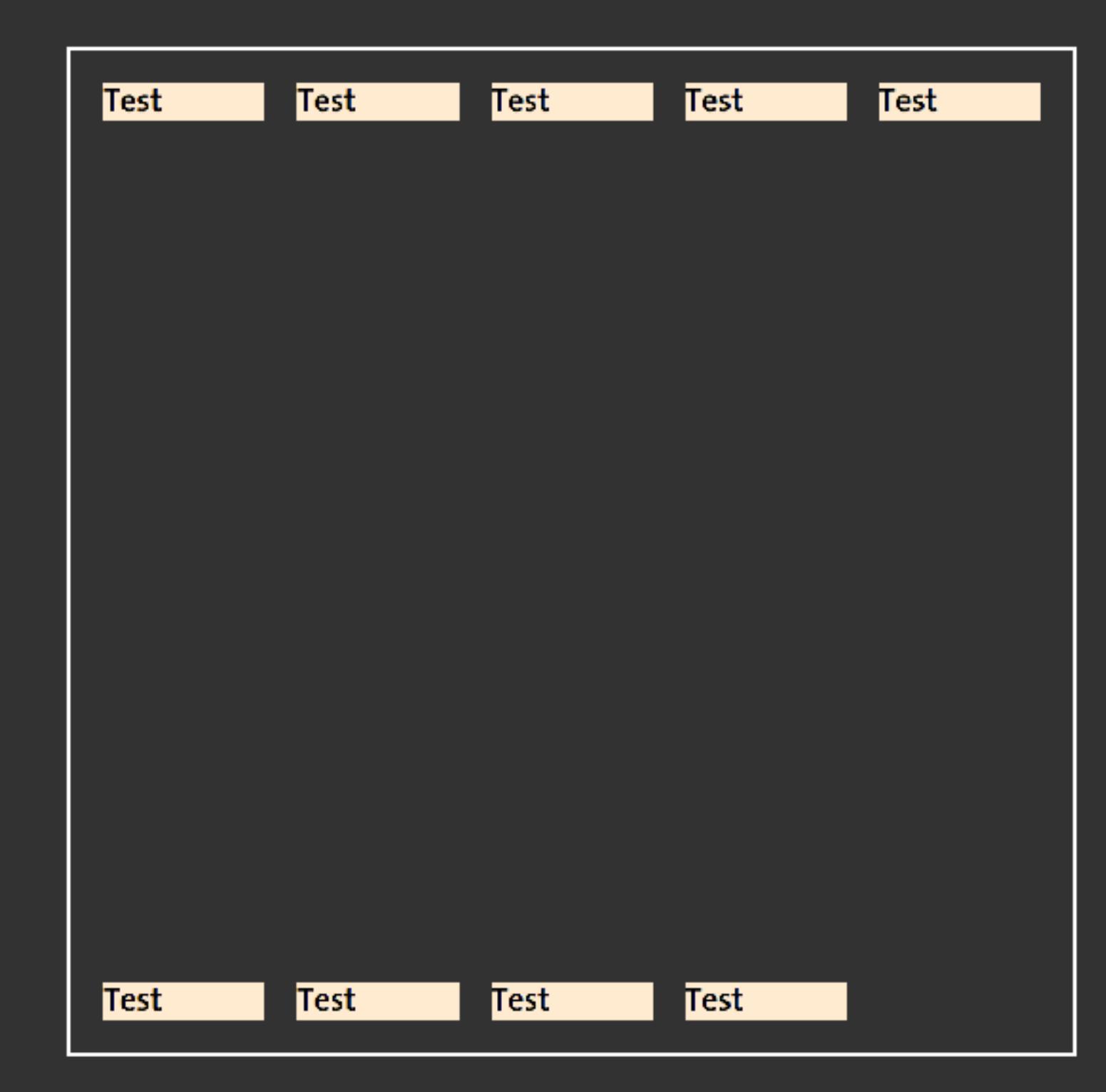
```
#container {  
  align-content: stretch  
  | flex-start | flex-  
  end | center | space-  
  between | space-  
  around;  
}
```

FLEX CONTAINER



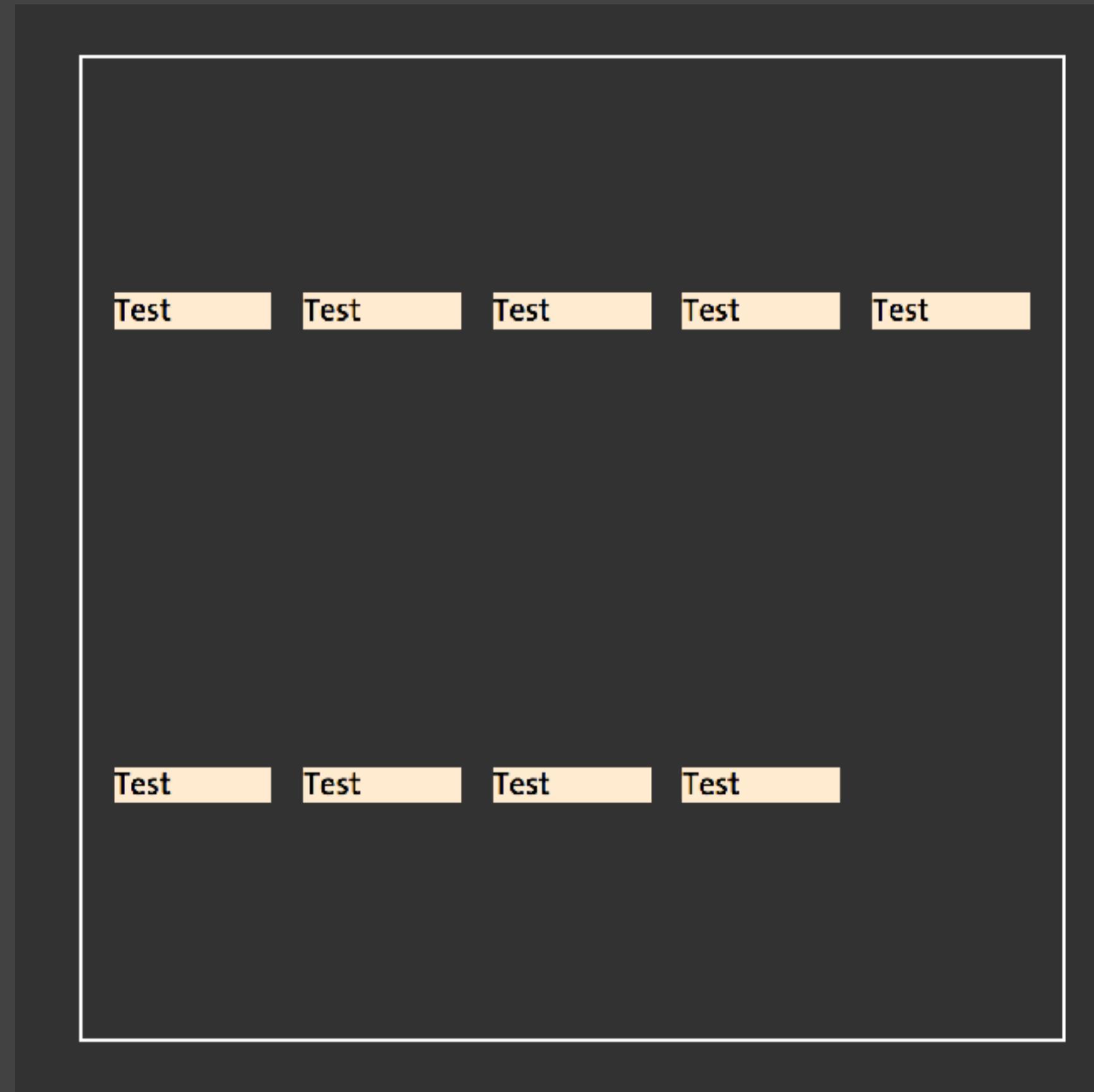
```
#container {  
  align-content: stretch  
  | flex-start | flex-  
  end | center | space-  
  between | space-  
  around;  
}
```

FLEX CONTAINER



```
#container {  
  align-content: stretch  
  | flex-start | flex-  
  end | center | space-  
  between | space-  
  around;  
}
```

FLEX CONTAINER



```
#container {  
  align-content: stretch  
  | flex-start | flex-  
  end | center | space-  
  between | space-  
  around;  
}
```

FLEX CONTAINER

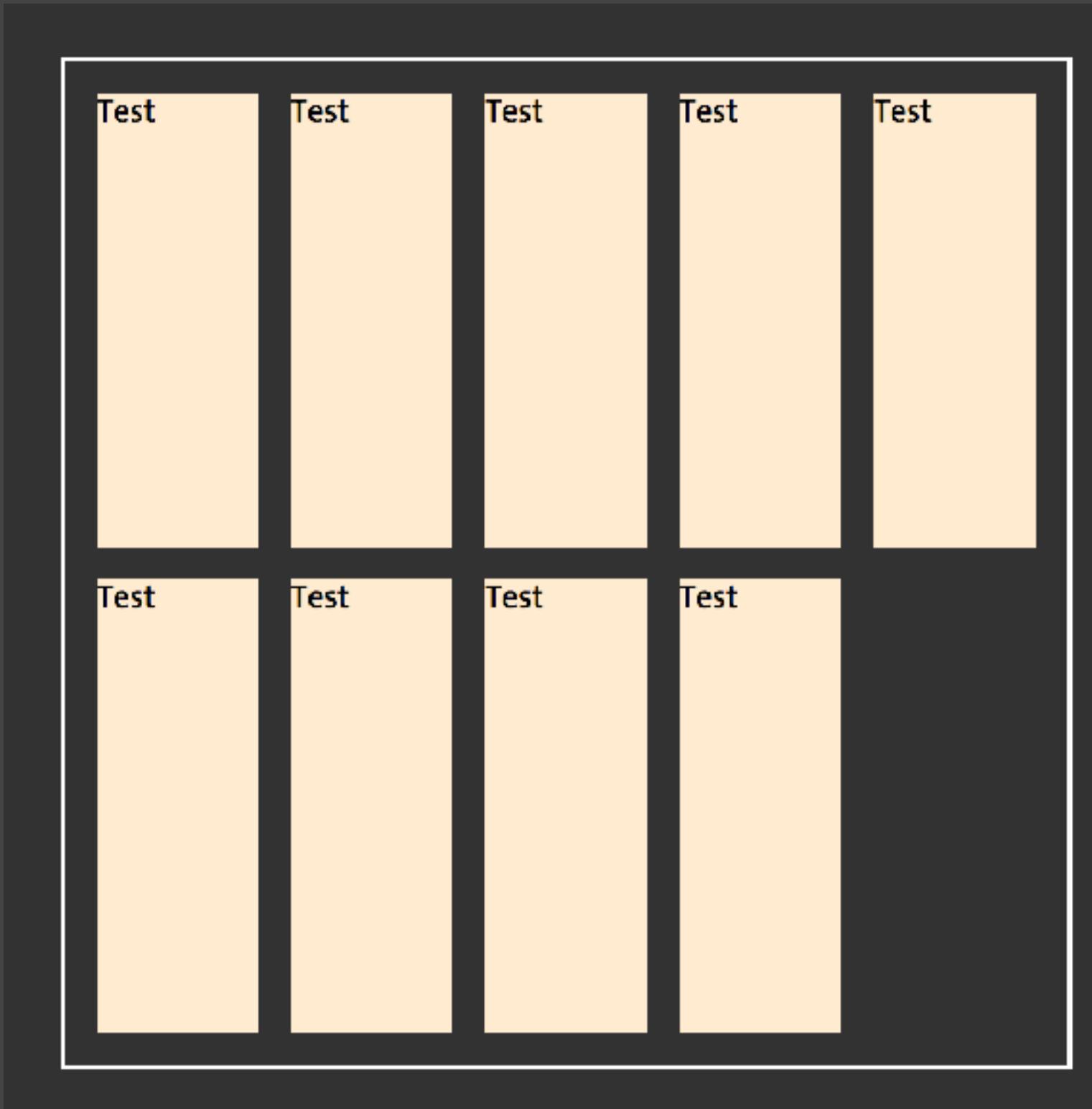
aligning items on the cross axis

**Works on columns if you're creating rows,
works on rows if you're creating columns.**

Aligns items as a whole.

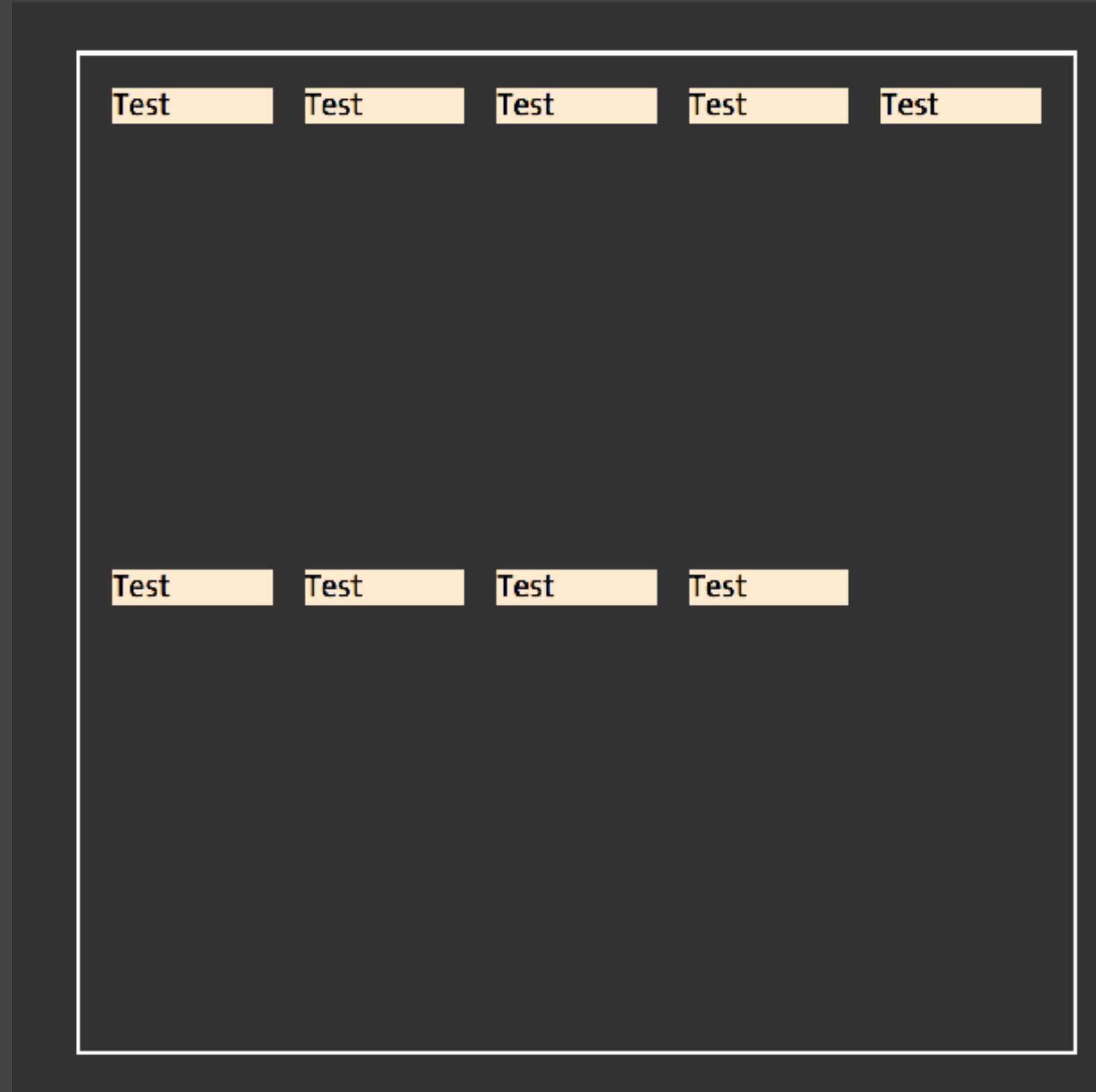
```
#container {  
  align-items: stretch  
  | flex-start | flex-end  
  | center | baseline;  
}
```

FLEX CONTAINER



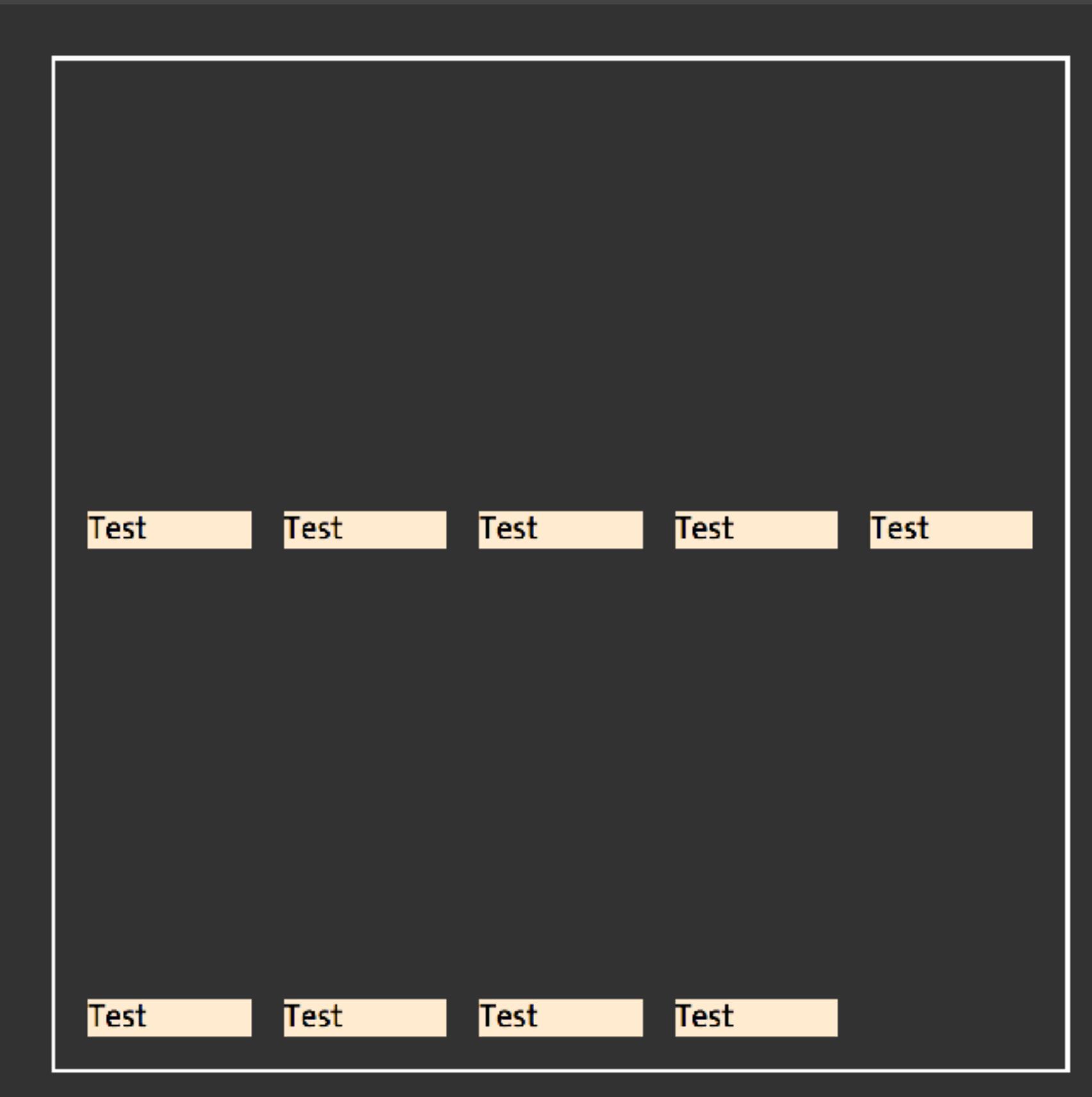
```
#container {  
  align-items: stretch  
  | flex-start | flex-end  
  | center | baseline;  
}
```

FLEX CONTAINER



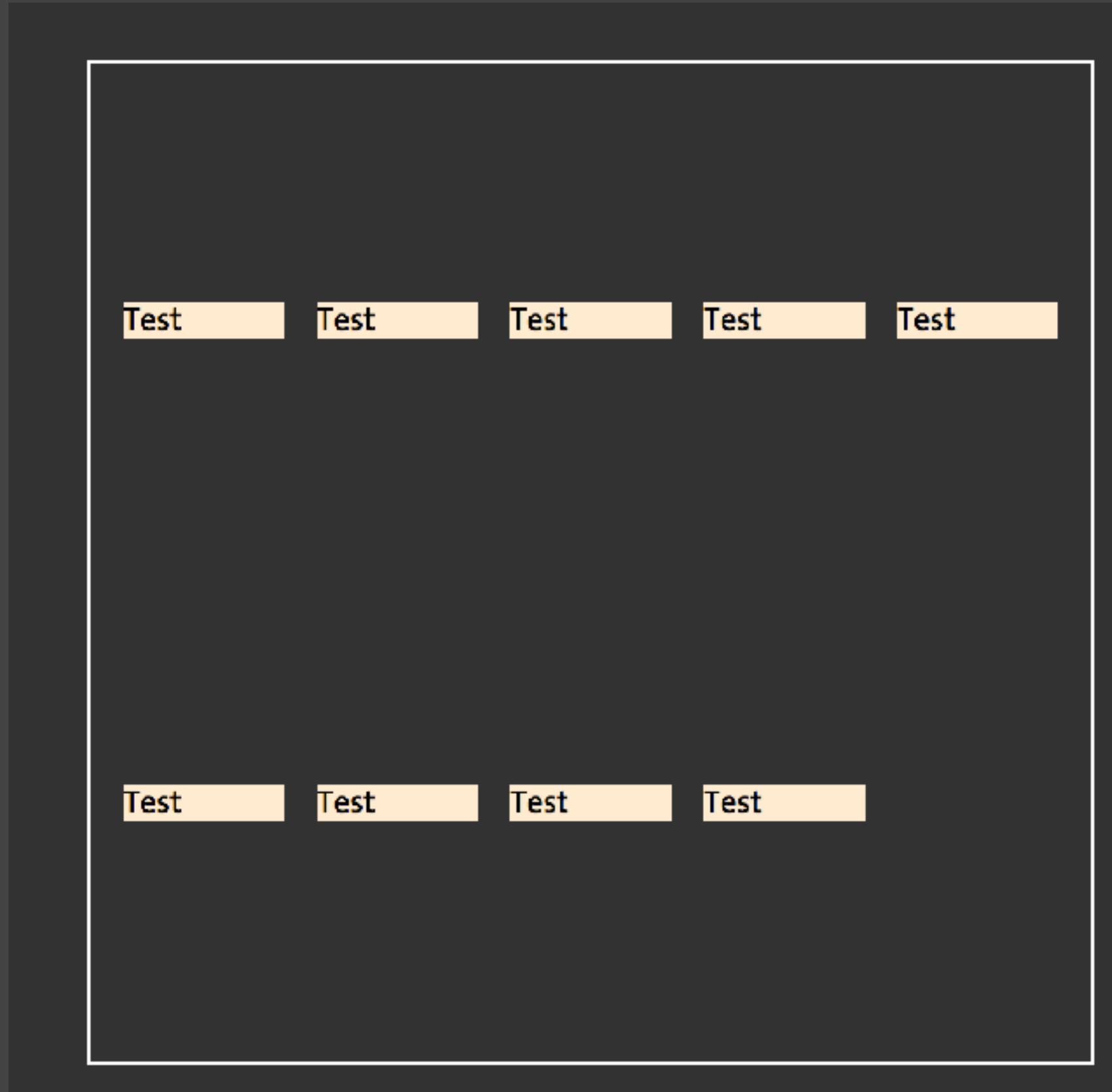
```
#container {  
  align-items: stretch  
  | flex-start | flex-end  
  | center | baseline;  
}
```

FLEX CONTAINER



```
#container {  
  align-items: stretch  
  | flex-start | flex-end  
  | center | baseline;  
}
```

FLEX CONTAINER



```
#container {  
  align-items: stretch  
  | flex-start | flex-end  
  | center | baseline;  
}
```

EXERCISE 02

The Daily Cascade: meta info

THE DAILY  CASCADE

Since 1994 · CSS news for all

24 November 2017

EXERCISE 03

The Daily Cascade: highlights

New box alignment spec in the works

Subgrid and the
need for it

order

#item1

#item2

#item3

#item4

#item3 { order: 2; }

When using order, note that the location of the items can be illogical for keyboard users.

EXERCISE 03

Move the second highlight
to the start of the container

Subgrid and the
need for it

New box alignment spec in the works

Takeaways

With `display:flex`, you can turn an element into a flex container

Auto margins let you make use of remaining whitespace

With alignment properties you can align items

With 'order' you're able to change the visual order

Grid Layout

**CSS Grid Layout lets you
define areas and place
content in them**





neu Japhabet

a possibility
for
the
new
development

een
mogelijkheid
voor
de
nieuwe
ontwikkeling

une
possibilité
pour
le
nouveau
développement

eine
Gelegenheit
für
die
neue
Entwicklung

In
Introduction
for
the
programmed
typography

abcdeFGH
ijklmnop
qrstvwxyz
ijklmnopqrstuvwxyz





CSS Grid Layout Module Level 1 X +

<https://www.w3.org/TR/css-grid-1/>

TABLE OF CONTENTS

- 1 Introduction**
 - 1.1 Background and Motivation
 - 1.1.1 Adapting Layouts to Available Space
 - 1.1.2 Source-Order Independence
- 2 Overview**
 - 2.1 Declaring the Grid
 - 2.2 Placing Items
 - 2.3 Sizing the Grid
- 3 Grid Layout Concepts and Terminology**
 - 3.1 Grid Lines
 - 3.2 Grid Tracks and Cells
 - 3.3 Grid Areas
- 4 Reordering and Accessibility**
- 5 Grid Containers**
 - 5.1 Establishing Grid Containers: the 'grid' and 'inline-grid' 'display' values
 - 5.2 Sizing Grid Containers
 - 5.3 Clamping Overly Large Grids
- 6 Grid Items**
 - 6.1 Grid Item Display
 - 6.2 Grid Item Sizing
 - 6.3 Reordered Grid Items: the 'order' property
 - 6.4 Grid Item Margins and Paddings
 - 6.5 Z-axis Ordering: the 'z-index' property
 - 6.6 Implied Minimum Size of Grid Items
- 7 Defining the Grid**
 - 7.1 The Explicit Grid
 - 7.2 Explicit Track Sizing: the 'grid-template-rows' and 'grid-template-columns' properties
 - 7.2.1 Named Grid Lines: the '<custom-ident>*' syntax

W3C Candidate Recommendation

CSS Grid Layout Module Level 1

W3C Candidate Recommendation, 09 May 2017

This version: <https://www.w3.org/TR/2017/CR-css-grid-1-20170509/>

Latest published version: <https://www.w3.org/TR/css-grid-1/>

Editor's Draft: <https://drafts.csswg.org/css-grid/>

Previous Versions:

- <https://www.w3.org/TR/2017/CR-css-grid-1-20170209/>
- <https://www.w3.org/TR/2016/WD-css-grid-1-20160519/>
- <https://www.w3.org/TR/2015/WD-css-grid-1-20150917/>
- <https://www.w3.org/TR/2015/WD-css-grid-1-20150806/>
- <https://www.w3.org/TR/2015/WD-css-grid-1-20150317/>
- <https://www.w3.org/TR/2014/WD-css-grid-1-20140513/>
- <https://www.w3.org/TR/2014/WD-css-grid-1-20140123/>
- <https://www.w3.org/TR/2013/WD-css3-grid-layout-20130402/>
- <https://www.w3.org/TR/2012/WD-css3-grid-layout-20121106/>

Test Suite: http://test.csswg.org/suites/css-grid-1_dev/nightly-unstable/

Issue Tracking:

- [Disposition of Comments](#)
- [Inline In Spec](#)
- [GitHub Issues](#)

Editors:

- [Tab Atkins Jr. \(Google\)](#)
- [Elika J. Etemad / fantasai \(Invited Expert\)](#)
- [Rossen Atanassov \(Microsoft\)](#)

Former Editors:

- [Alex Mogilevsky \(Microsoft Corporation\)](#)
- [Phil Cupp \(Microsoft Corporation\)](#)

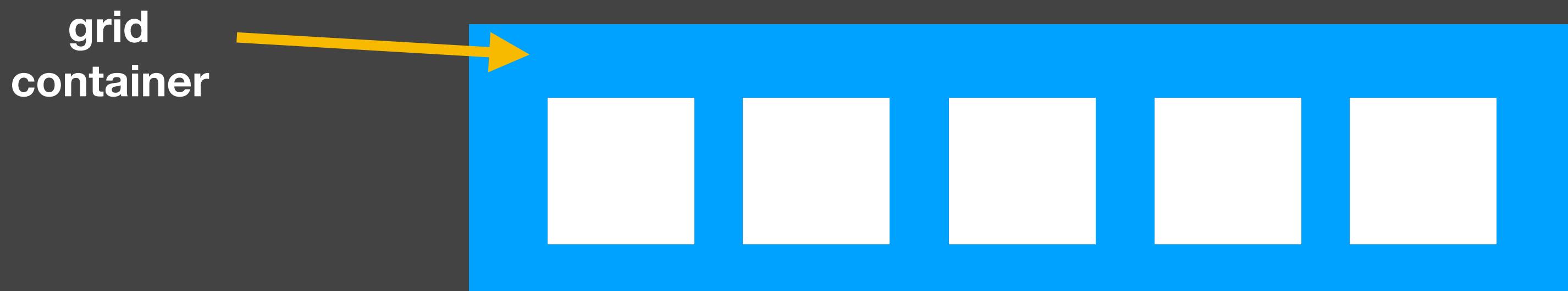
Copyright © 2017 W3C® ([MIT](#), [ERCIM](#), [Keio](#), [Beihang](#)). W3C liability, trademark and license terms apply.

**The beauty of CSS Grid is that
grid cells can grow with the
content that lives in them**

defining a grid

```
#container { display: grid; }
```

grid container



```
#container { display: grid; }
```

When something becomes a...

grid container

stops working:

- column-*
- float
- clear
- vertical-align
- ::first-[item|letter]

can now be used:

- grid-template-columns
- grid-template-rows
- grid-auto-flow
- justify-content
- align-content

When something becomes a...

grid item

stops working:

- vertical-align
- ::first-[item|letter]

can now be used:

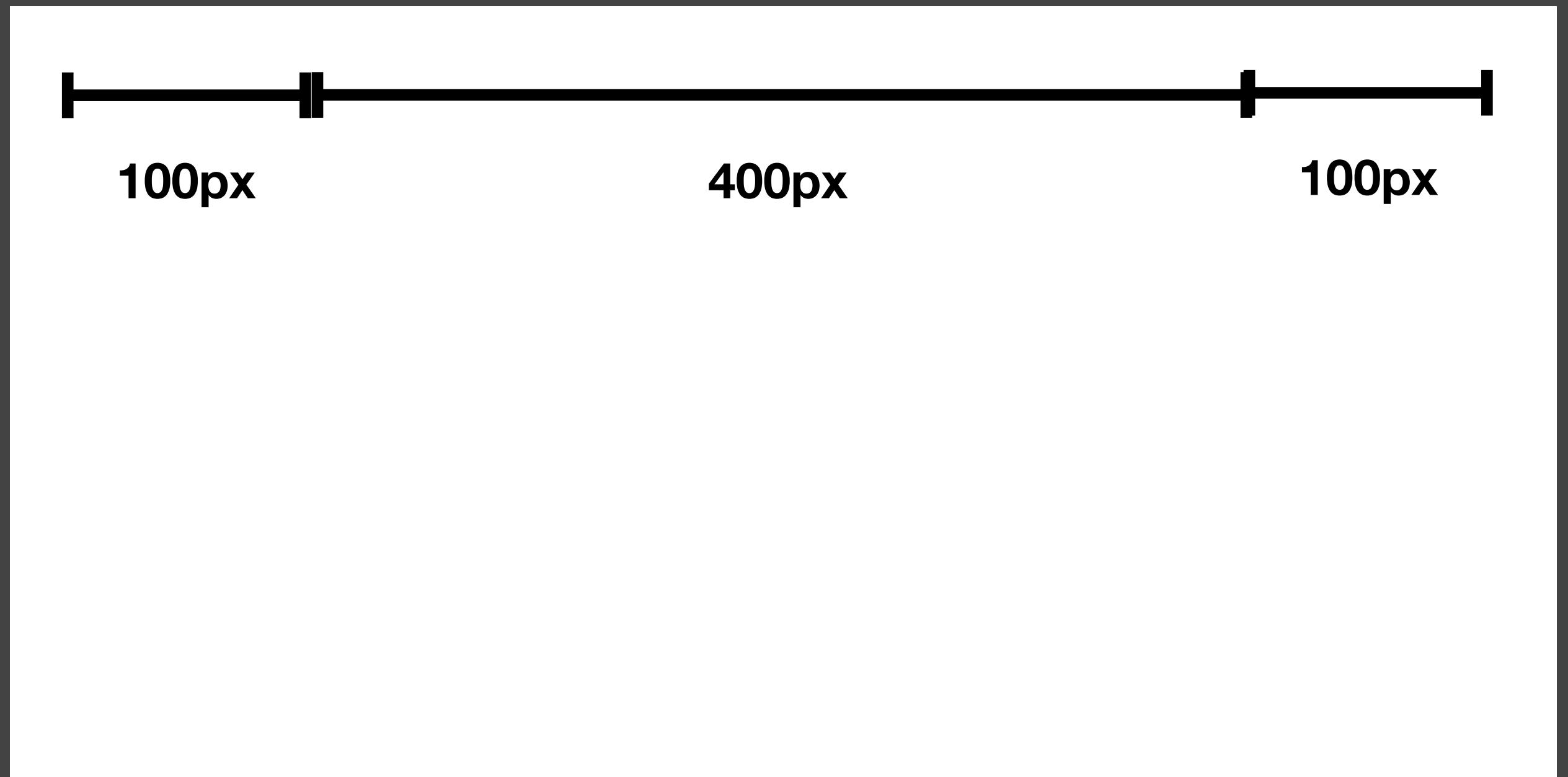
- grid-area
- grid-[column|row]
- order
- align-self
- justify-self

In Grid Layout you can
use ~ the same alignment
properties as in flexbox



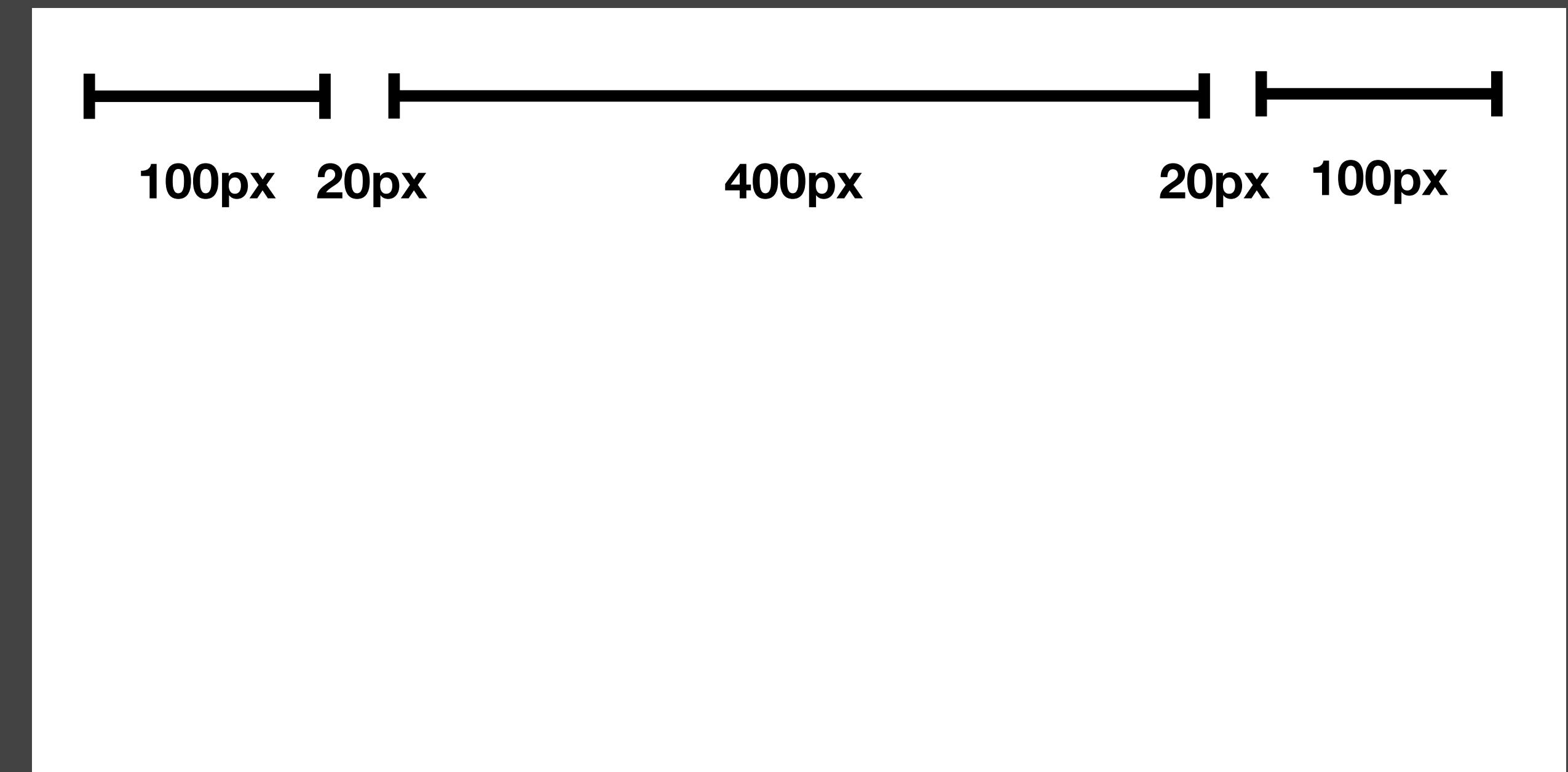
defining a grid

```
#container {  
  display: grid;  
  grid-template-columns:  
    100px 400px 100px;  
}
```



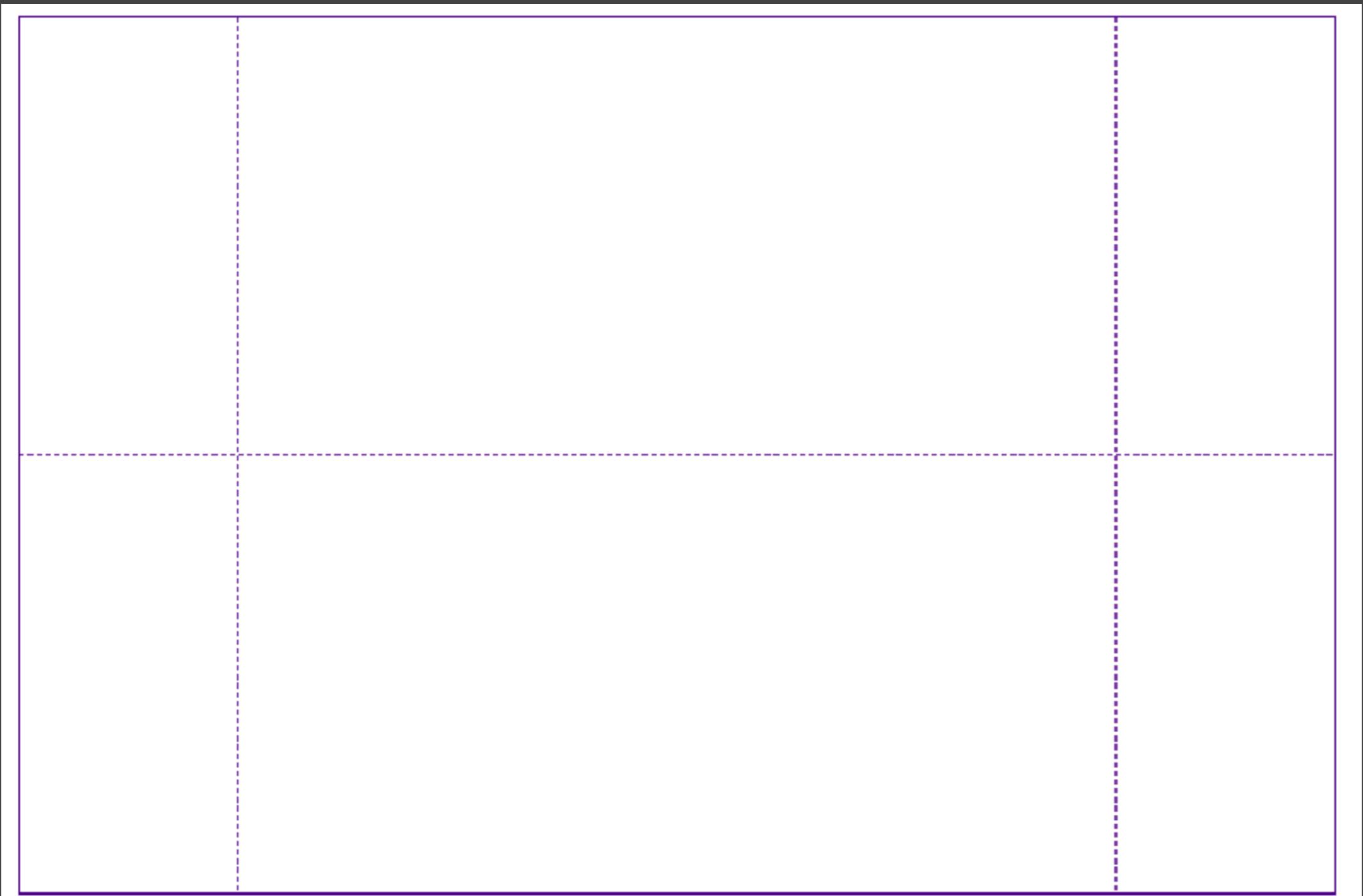
defining a grid

```
#container {  
  display: grid;  
  grid-template-columns:  
    100px 400px 100px;  
  grid-gap: 20px;  
}
```



defining a grid

```
#container {  
  display: grid;  
  grid-template-columns:  
    100px 400px 100px;  
  grid-template-rows:  
    200px 200px;  
}
```

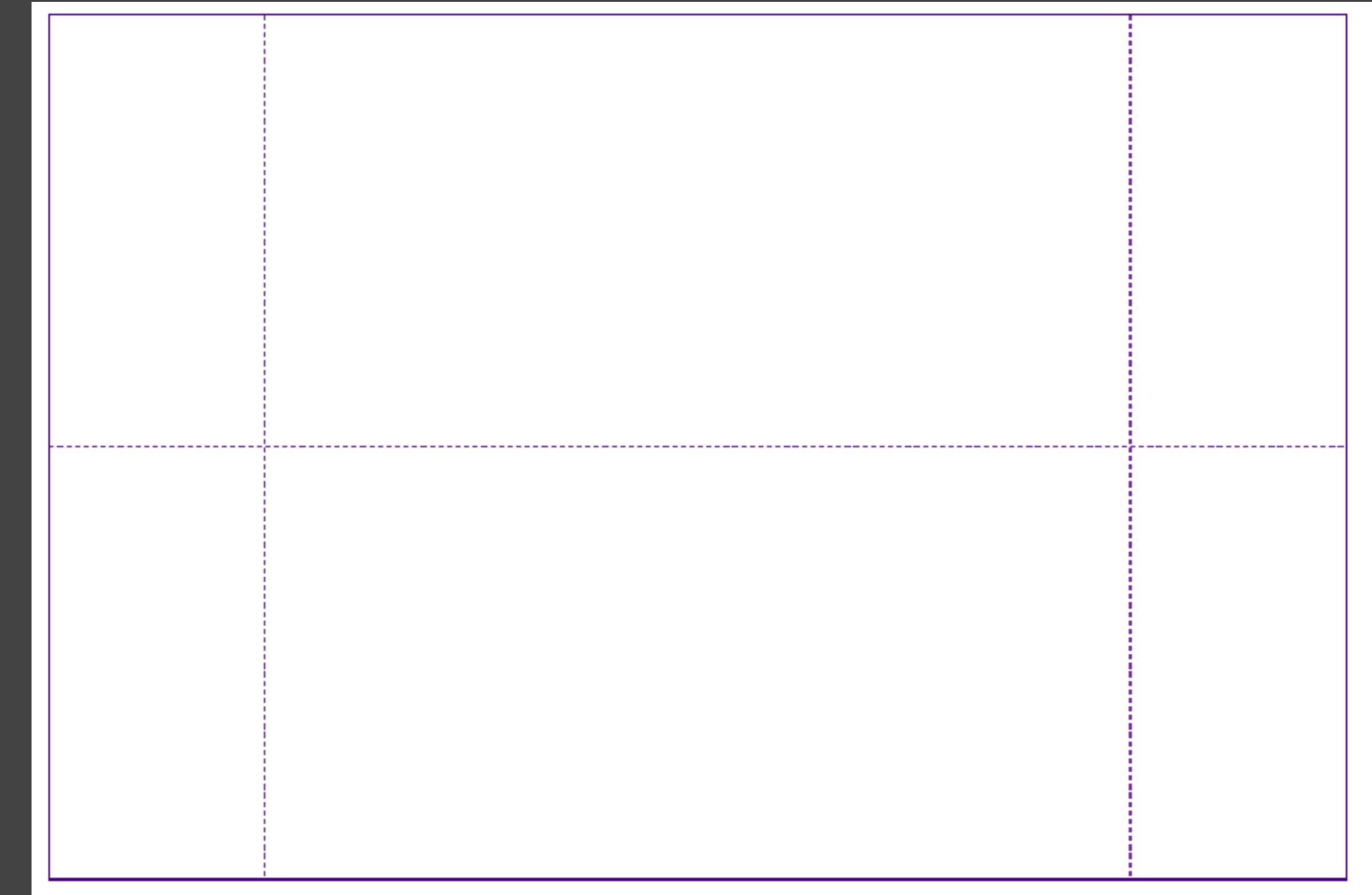


**With flexible units, you can
leave the details to the browser**

**The less precise your style hints
are, the more a browser can
make your content fit.**

flexible units for grid tracks: fr

```
#container {  
  display: grid;  
}
```

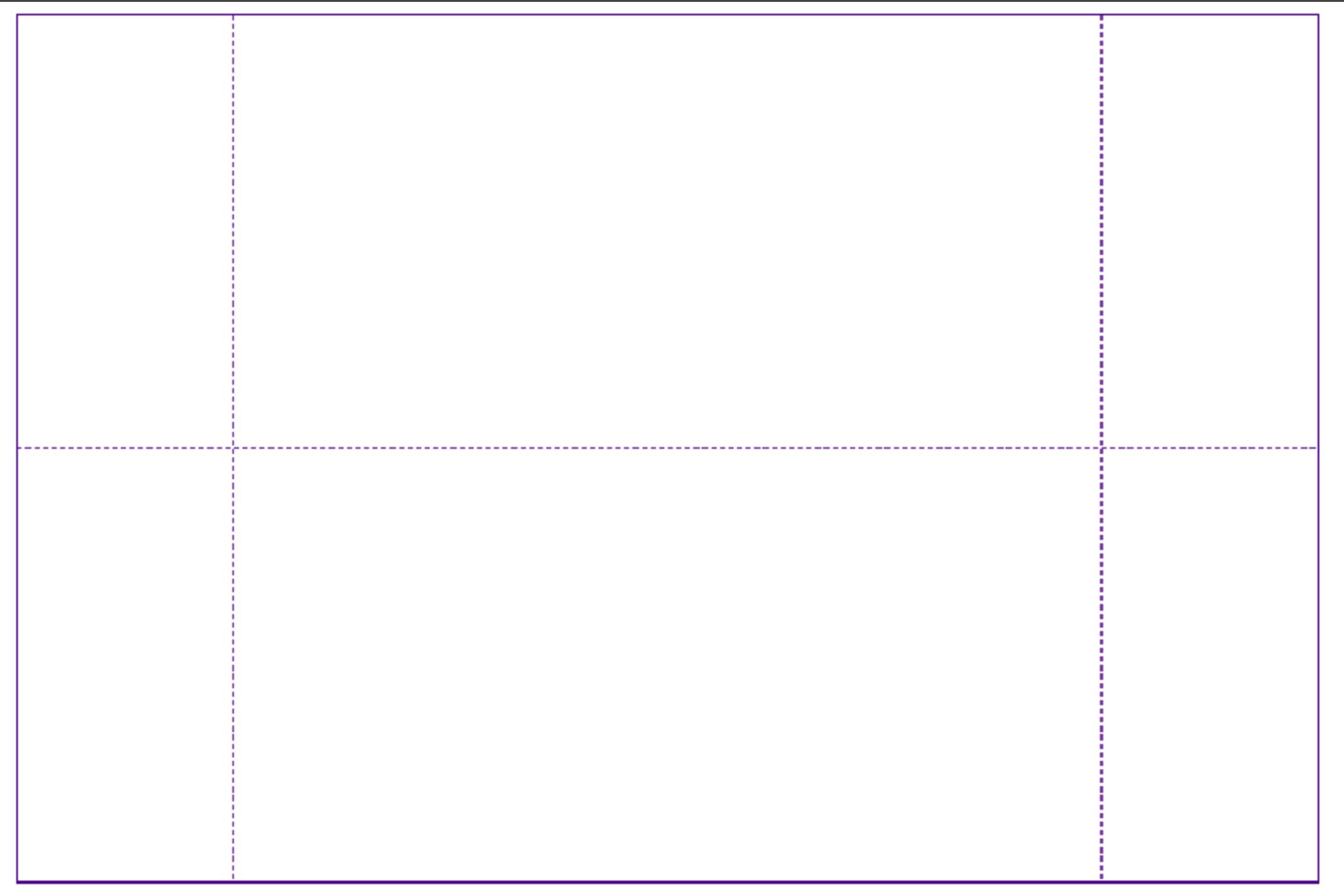


flexible units for grid tracks: fr



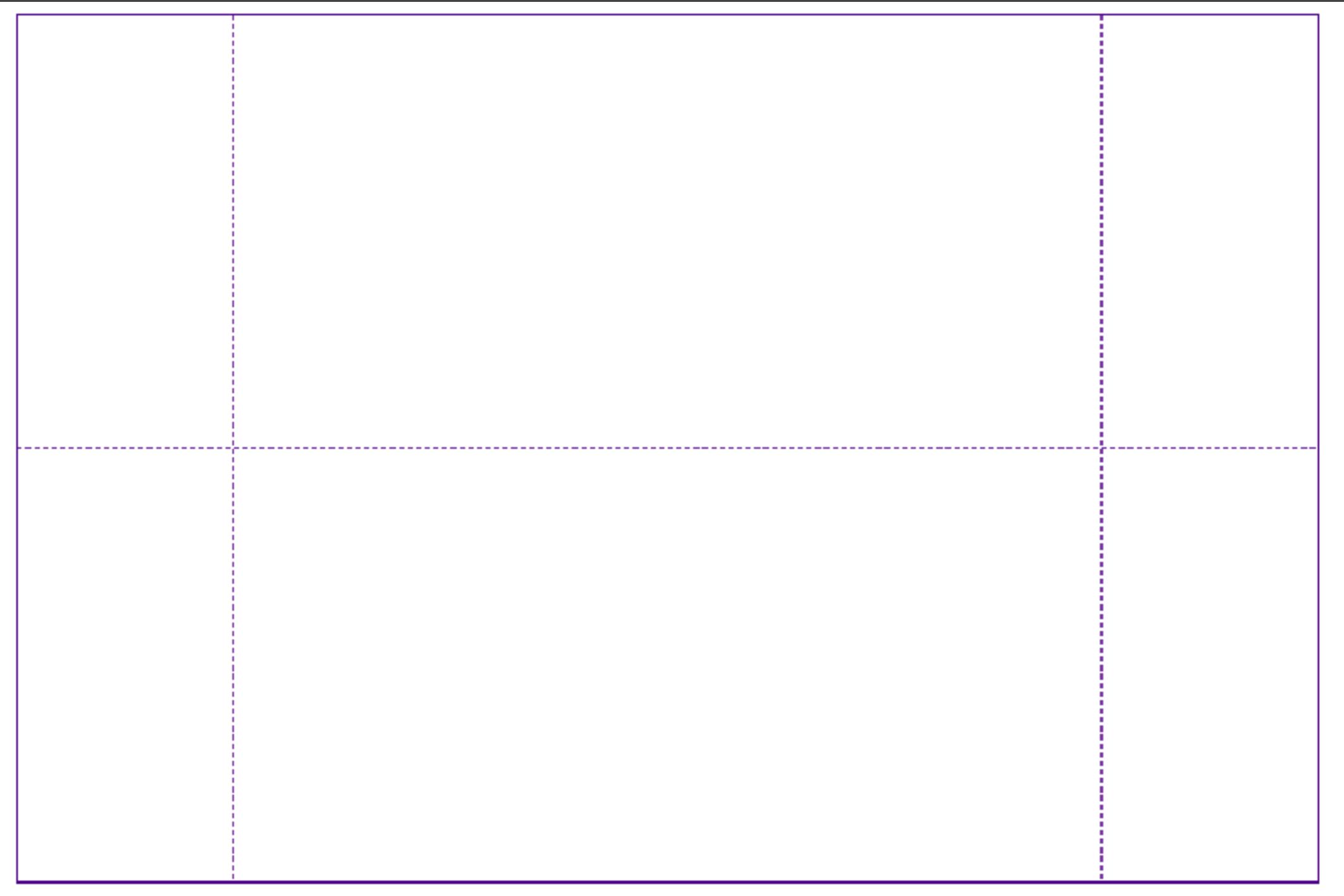
flexible units for grid tracks: fr

```
#container {  
  display: grid;  
  grid-template-columns:  
    1fr 3fr 1fr;  
  grid-template-rows:  
    1fr 1fr;  
}
```



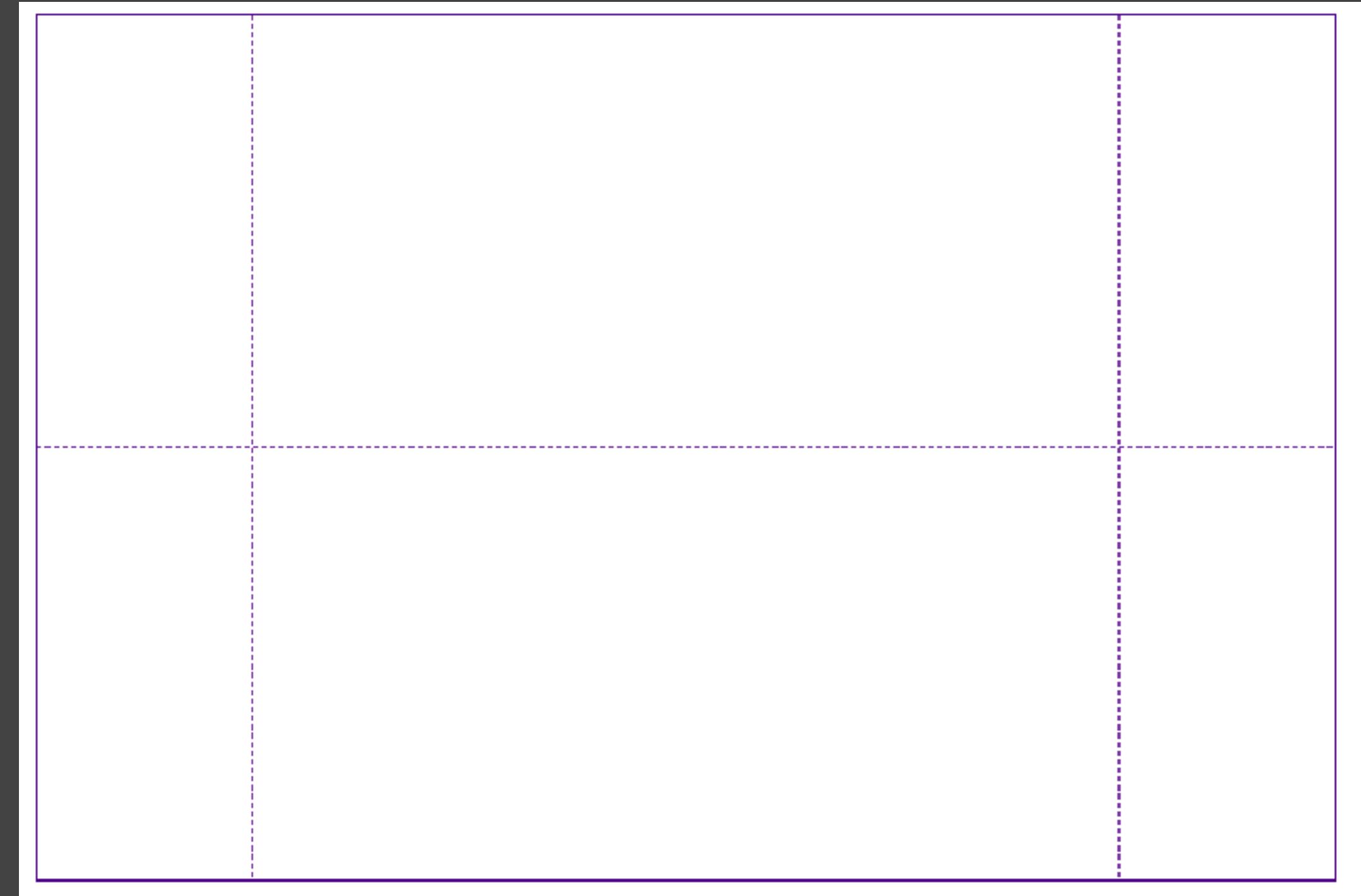
flexible units for grid tracks: ch

```
#container {  
  display: grid;  
  grid-template-columns:  
    1fr 60ch 1fr;  
  grid-template-rows:  
    1fr 1fr;  
}
```



flexible units for grid tracks: auto

```
#container {  
  display: grid;  
  grid-template-columns:  
    auto 60ch auto;  
}
```



grids in Firefox Dev Tools



From: [Jen Simmons' layout experiments](#)

grids in Firefox Dev Tools

line numbers

grid area names

overlay

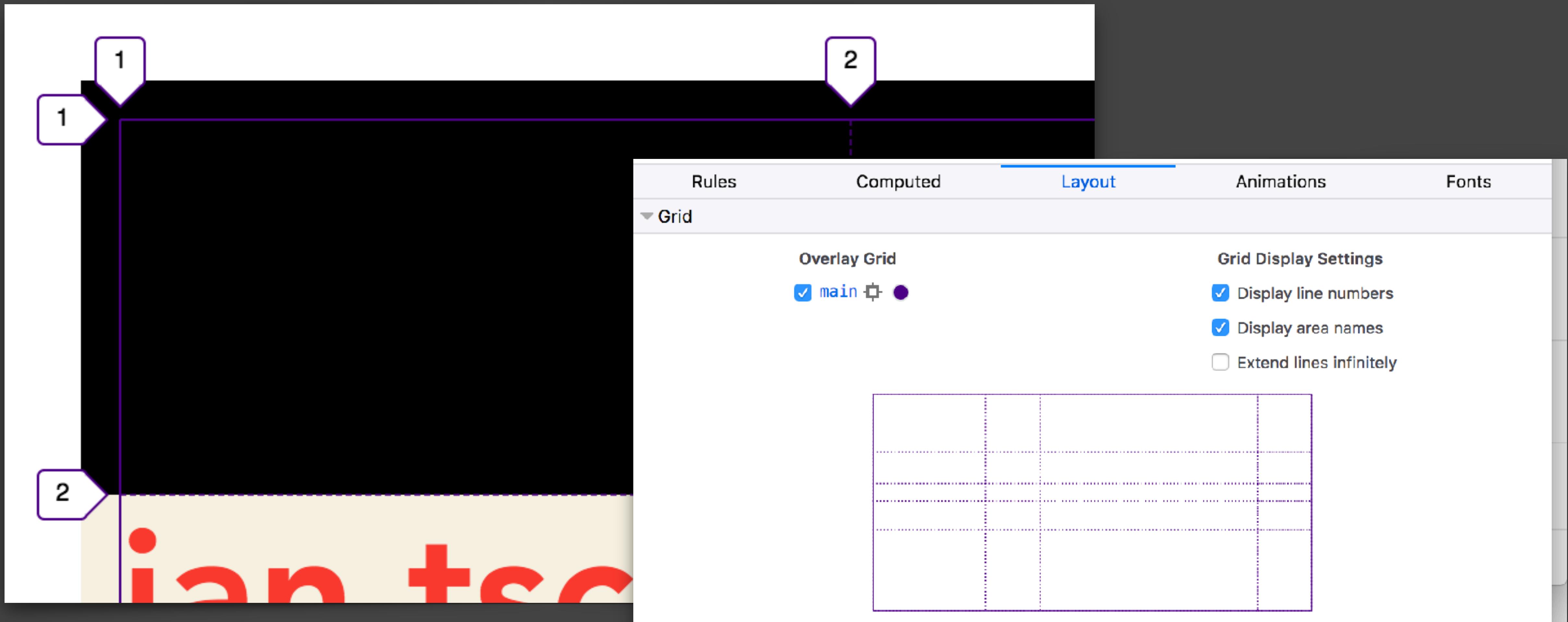
tweak line colours

implicit vs explicit grid

flexible units for grid tracks: minmax()

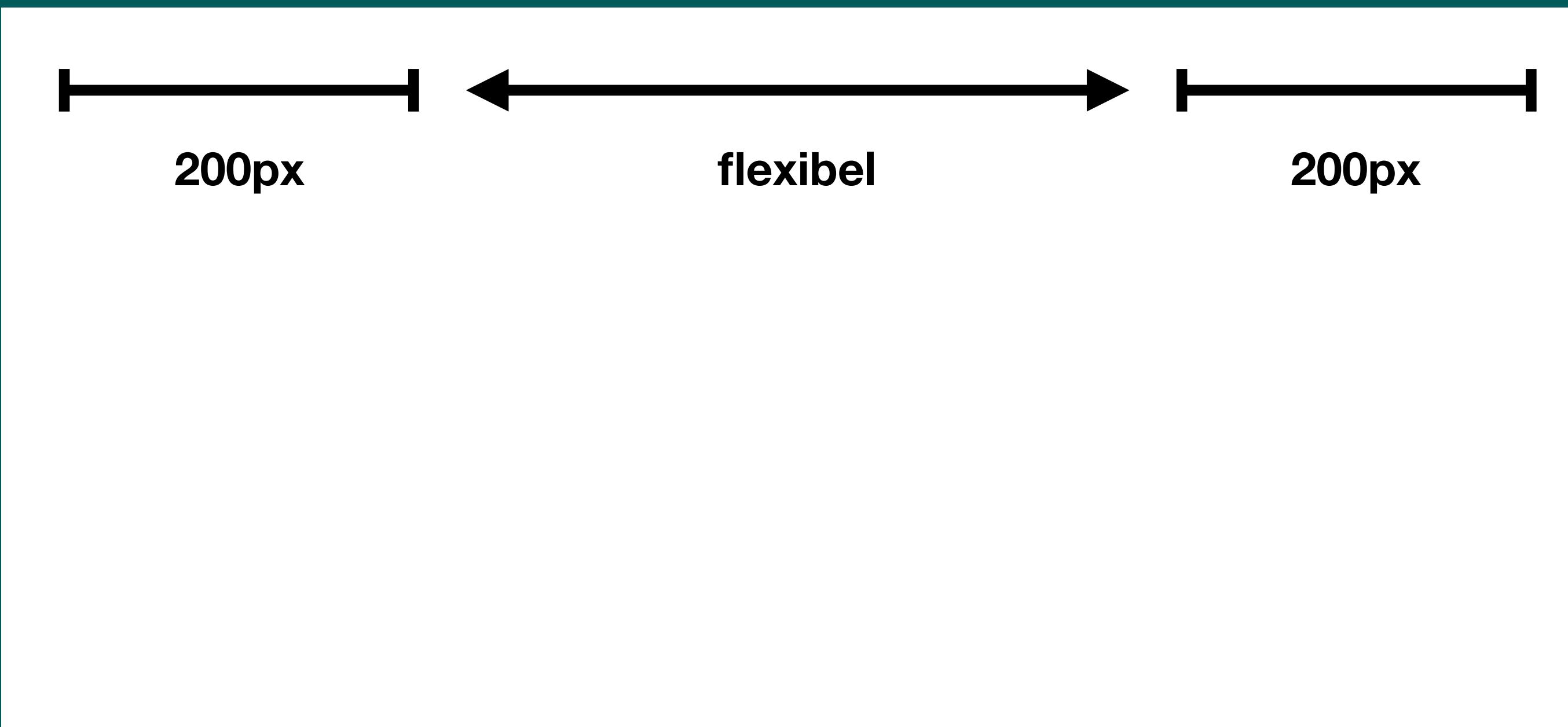
```
#container {  
  display: grid;  
  grid-template-columns:  
    1fr minmax(40em, 50em) 1fr;  
  grid-template-rows:  
    1fr 1fr;  
}
```

grids in Firefox Dev Tools



EXERCISE 04

The Daily Cascade:
articles in "Holy Grail" layout



EXERCISE 05

THE DAILY CASCADE

Since 1994 · CSS news for all

24 November 2017

This daily is made with
love.

The Story of CSS Grid, from Its Creators

by Aaron Gustafson · October 19, 2017

Designers have used grids for centuries. And after more than 20 years of waiting, they are finally here for the browser. This is the story of CSS Grid. It took a lot of people in the right place and at the right time to make it happen.

Please donate.

Practical CSS Grid: Adding Grid to an Existing Design

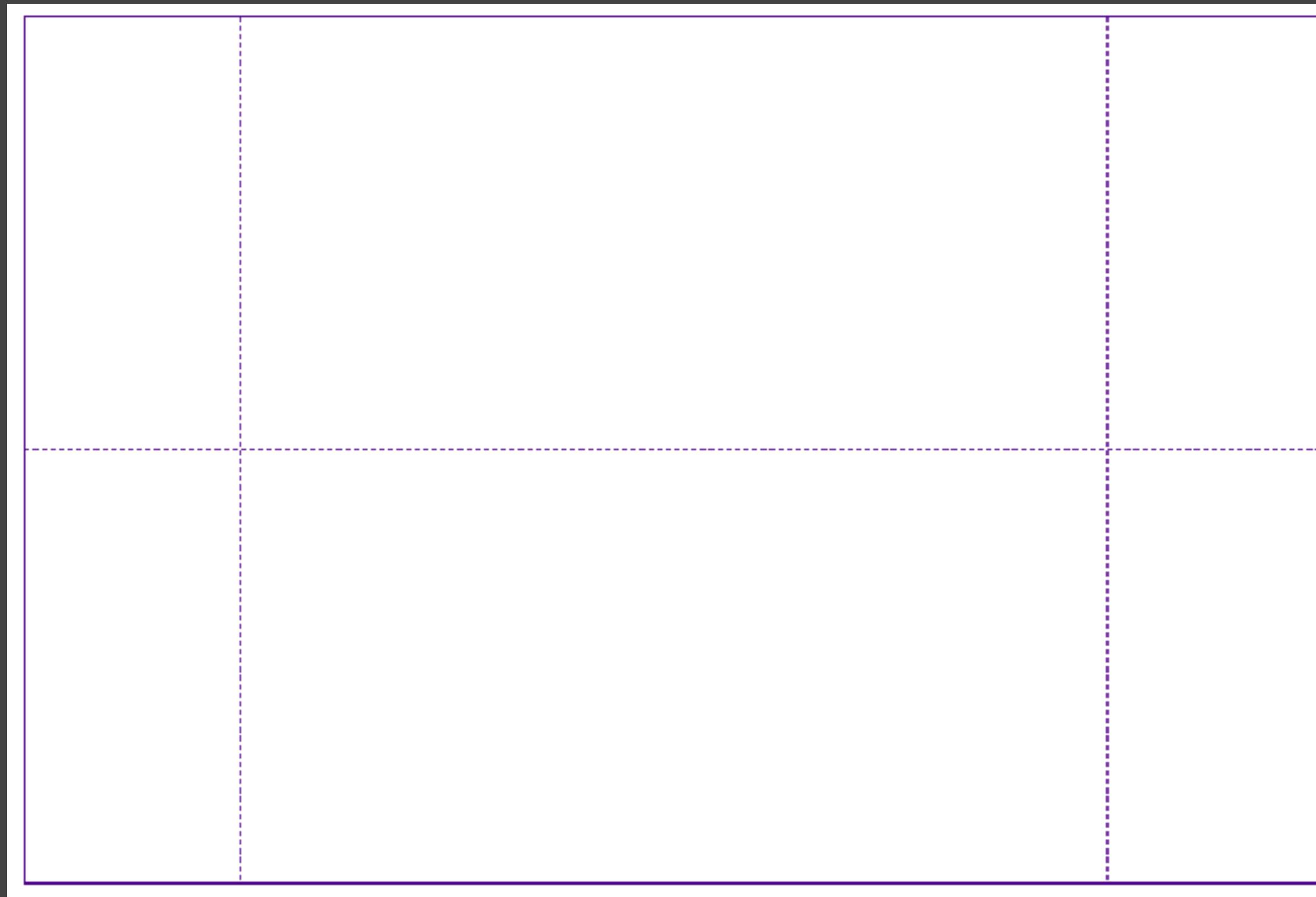
by Eric Meyer · March 23, 2017

CSS Grid is here—and easier than you might expect. Eric Meyer shows us how to use Grid on an existing design without breaking things for non-grid browsers. With pictures! Also a couple of gotchas.

Learning from Lego: A Step Forward in Modular Web Design

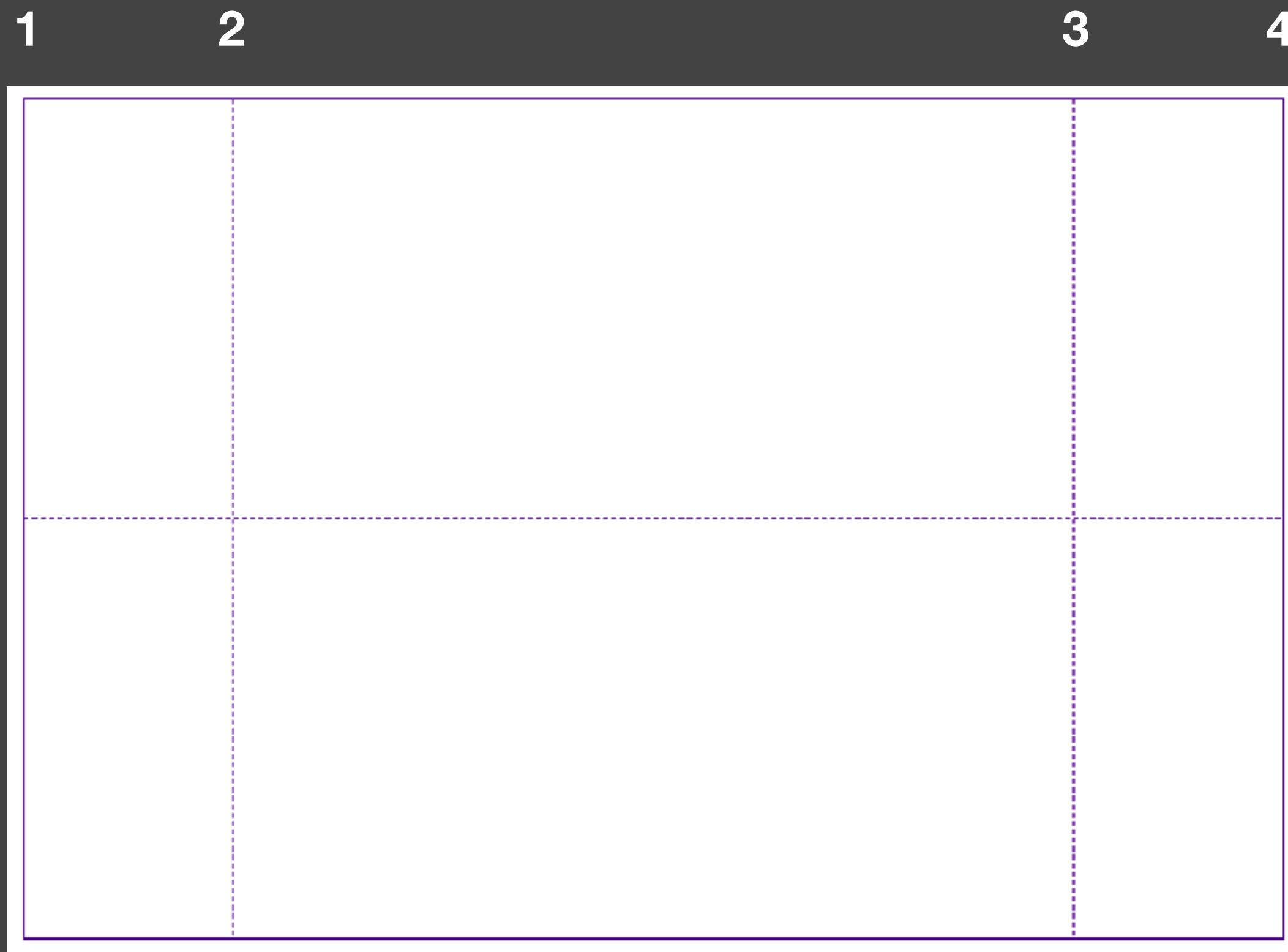
by Samantha Zhang · December 22, 2016

placing items on the grid



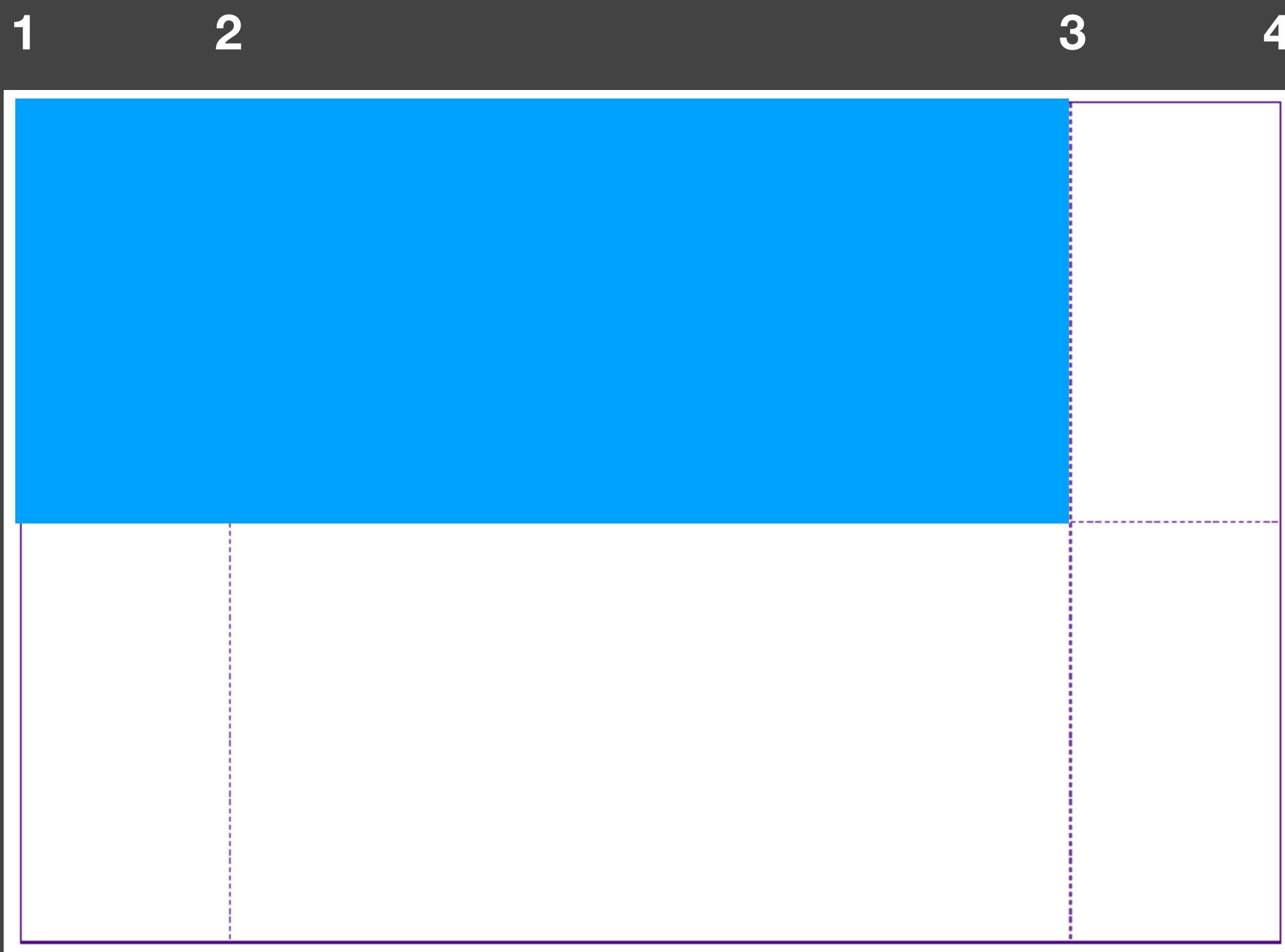
```
#item {  
  grid-column-start: 1;  
  grid-column-end: 3;  
}
```

placing items on the grid



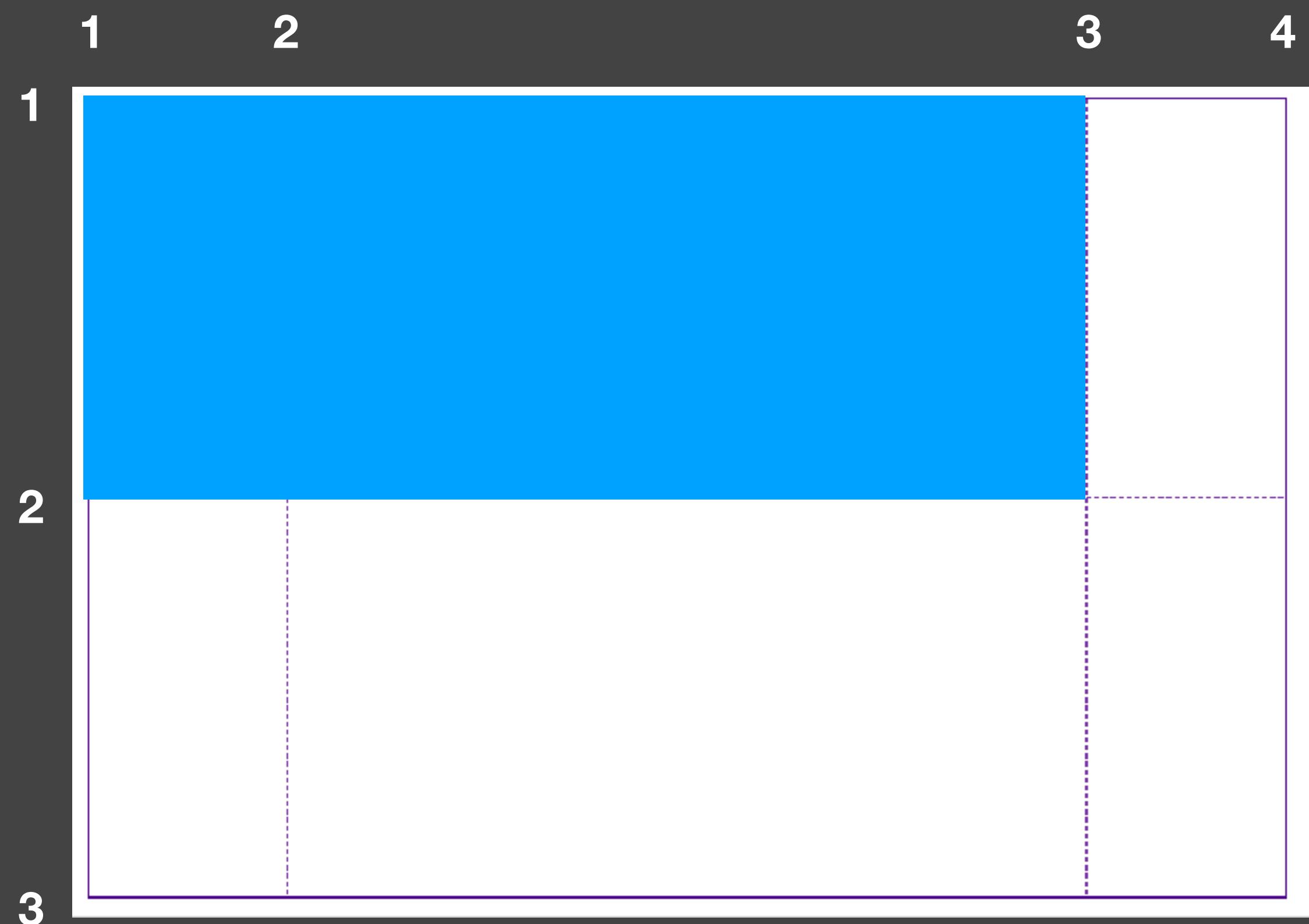
```
#item {  
  grid-column-start: 1;  
  grid-column-end: 3;  
}
```

placing items on the grid



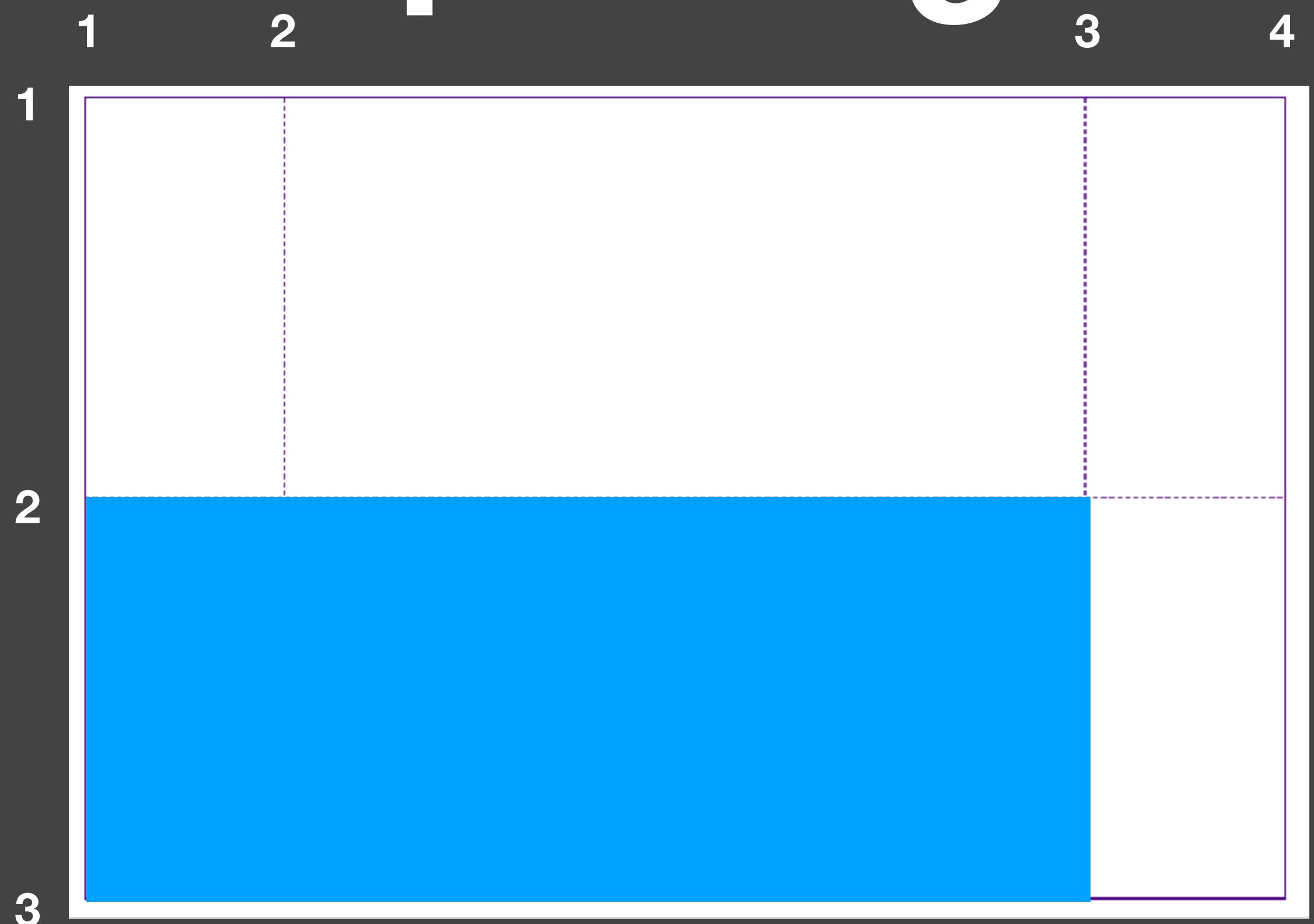
```
#item {  
  grid-column-start: 1;  
  grid-column-end: 3;  
}
```

placing items on the grid



```
#item {  
  grid-column-start: 1;  
  grid-column-end: 3;  
}
```

placing items on the grid



```
#item {  
    grid-column-start: 1;  
    grid-column-end: 3;  
    grid-row-start: 2;  
    grid-row-end: 3;  
}
```

EXERCISE

Wim Crouwel's Grids

Georg Baselitz

stedelijk van abbemuseum eindhoven

dagelijks geopend
van 10-17 uur
zon- en feestdagen
van 13-17 uur
dinsdag- en
donderdagavond
van 20-22 uur

hipogima

panelen van iri maruki en toshiko akamatsu

30 maart tot
15 april 1957

Baselitz

1960-83

Schilderijen / Paintings

Lunch

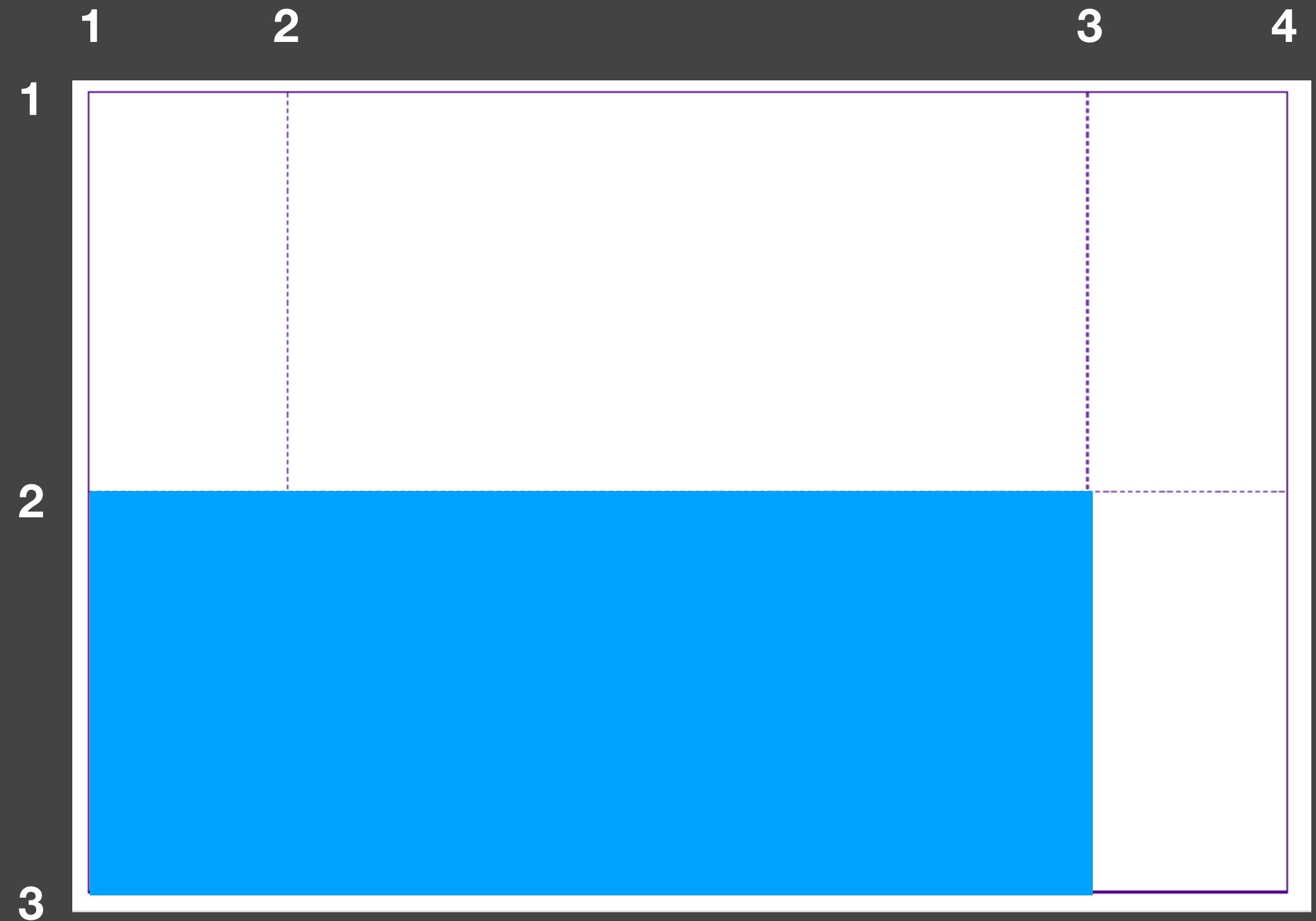
you can also name grid lines

```
#container {  
  grid-template-columns:  
    [sidebar-start] 1fr [sidebar-end] 4fr;  
}
```

```
#item {  
  grid-column-start: sidebar-start;  
  grid-column-end: sidebar-end;  
}
```

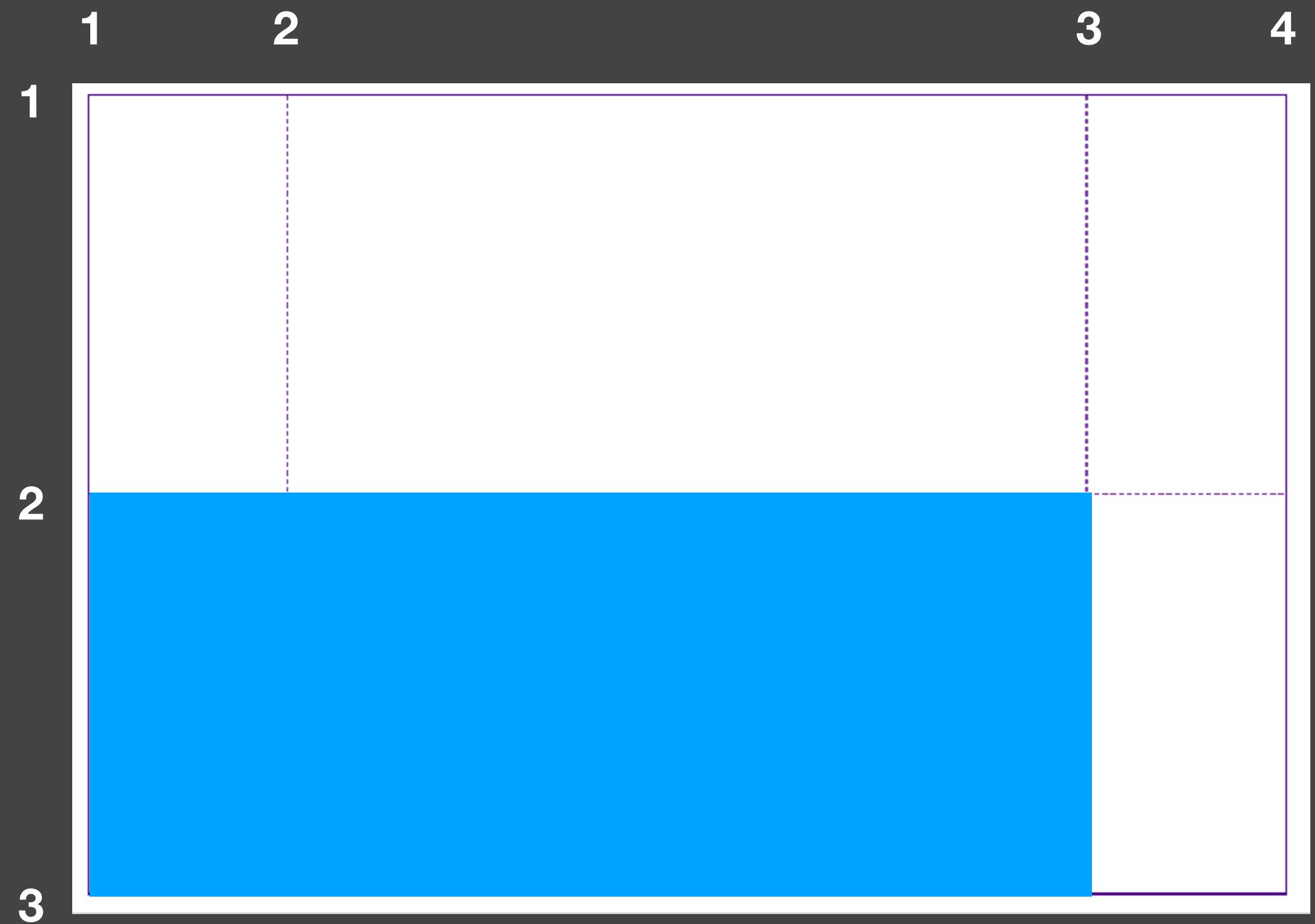
```
#container {  
  grid-template-columns:  
    [sidebar-start] 1fr [sidebar-end] 4fr;  
}  
  
#item {  
  grid-column: sidebar;  
}
```

placing items on the grid (2)



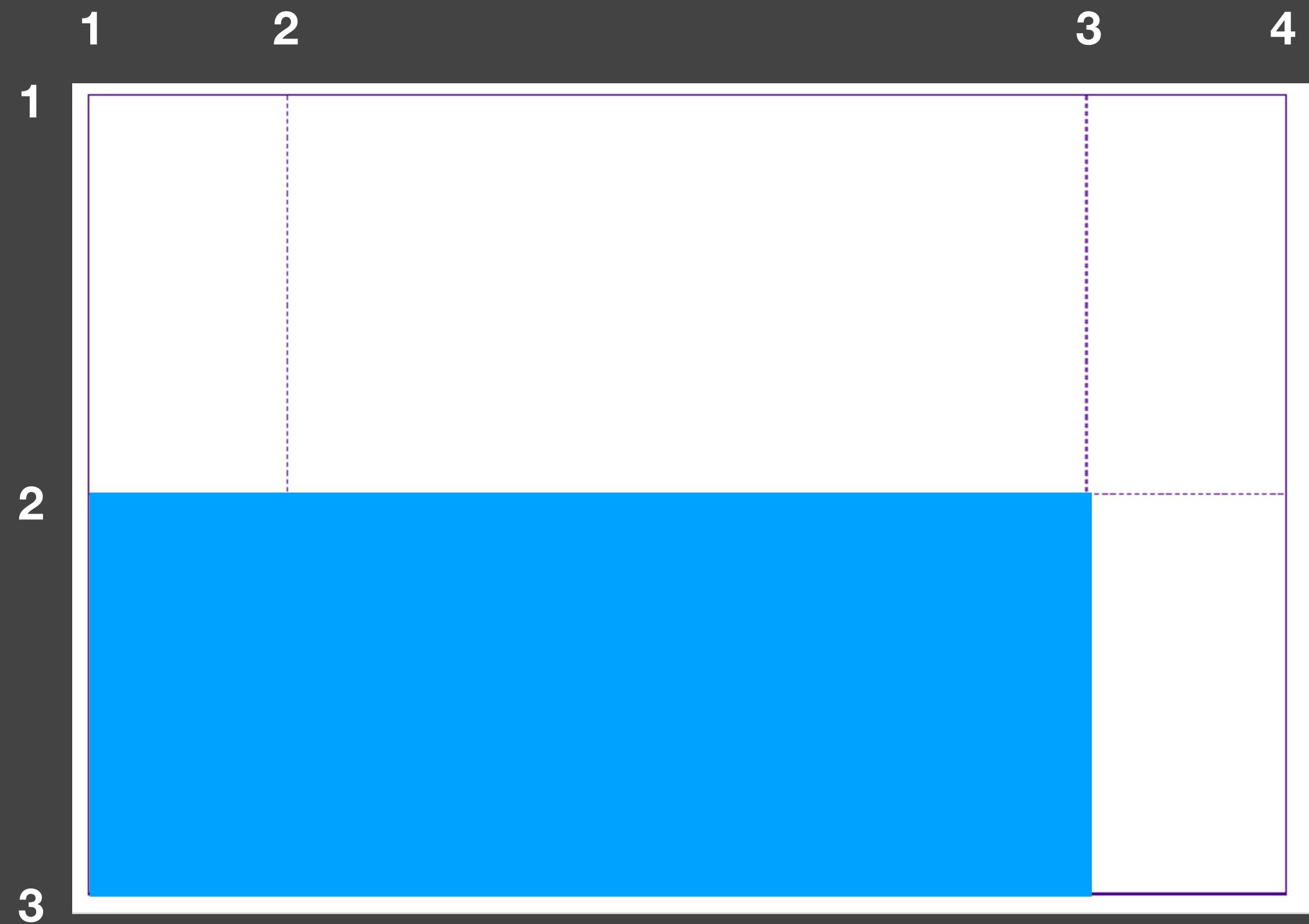
```
#item {  
  grid-column-start: 1;  
  grid-column-end: 3;  
  grid-row-start: 2;  
  grid-row-end: 3;  
}
```

placing items on the grid (2)



```
#item {  
  grid-column: 1 / 3;  
  grid-row: 2 / 3;  
}
```

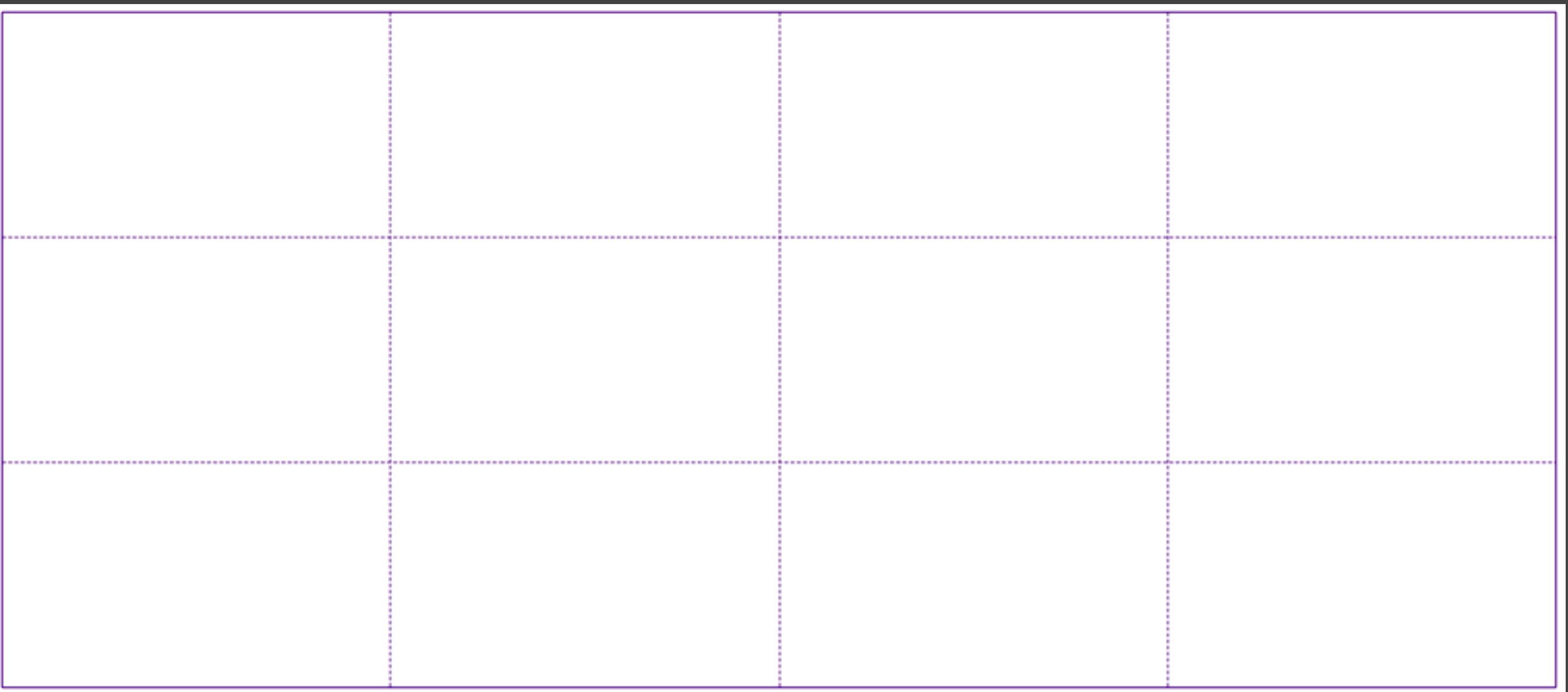
The span keyword



```
#item {  
  grid-column: 1 / span 3;  
  grid-row: 2 / span 1;  
}
```

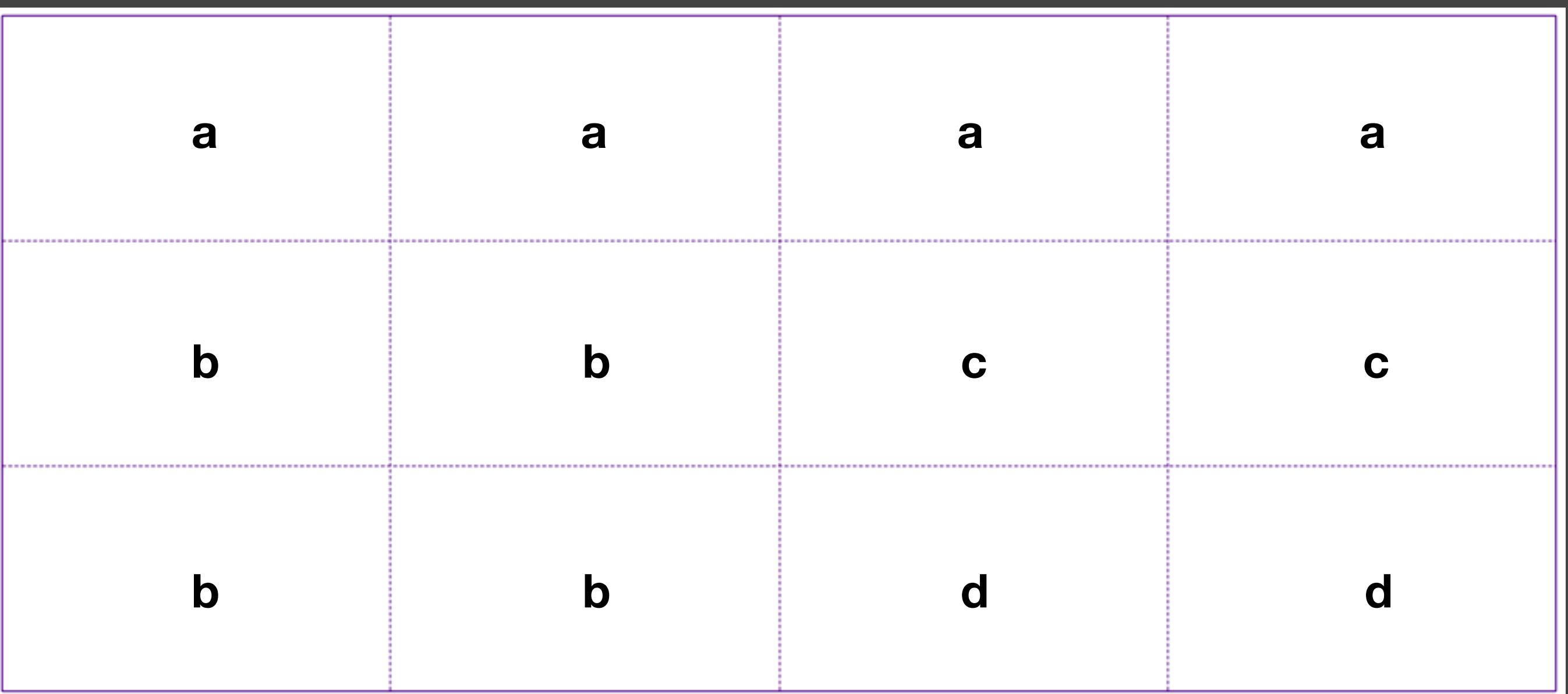
grids with named areas

```
#container {  
  display: grid;  
  grid-template-areas:  
    "a a a a"  
    "b b c c"  
    "b b d d";  
}
```



grids with named areas

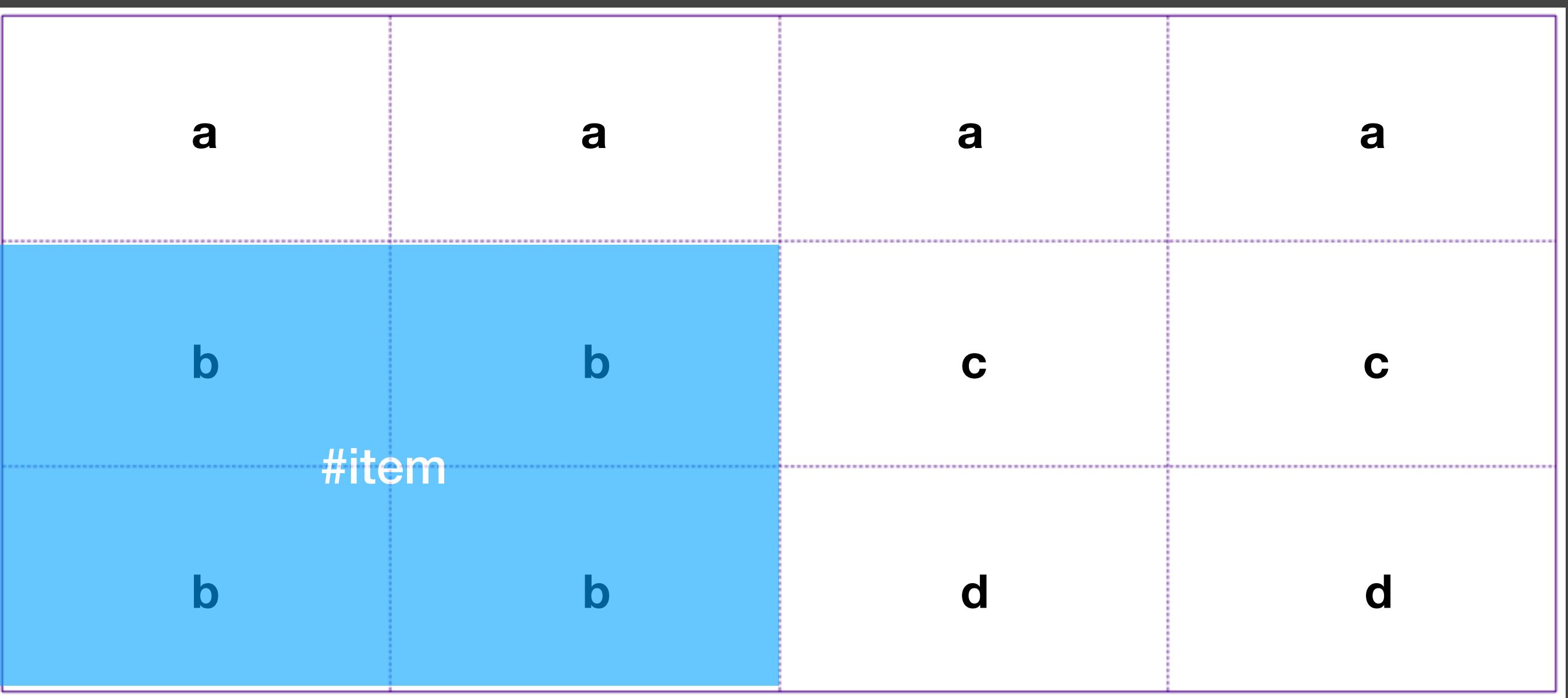
```
#container {  
  display: grid;  
  grid-template-areas:  
    "a a a a"  
    "b b c c"  
    "b b d d";  
}
```



Note: areas should form squares

grids with named areas

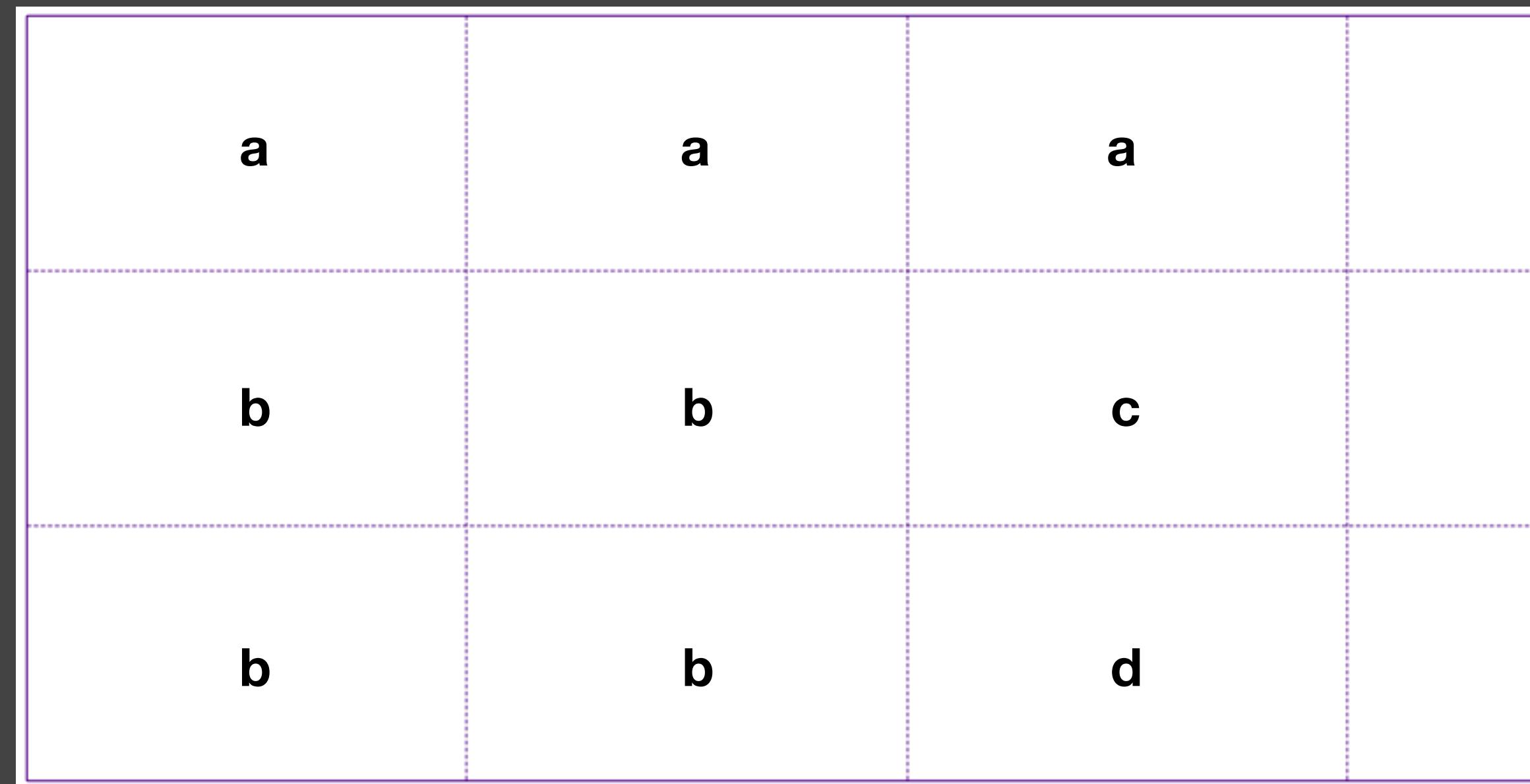
```
#item {  
  grid-area: b;  
}
```



grid lines are
automatically named

grids with named areas

```
#item {  
  
  grid-column-start: b-start;  
  
  grid-column-end: b-end;  
}
```



```
#container {  
    grid-template-columns:  
        [sidebar-start] 1fr [sidebar-end] 4fr;  
}
```

```
#item {  
    grid-column-start: sidebar-start;  
    grid-column-end: sidebar-end;  
}
```

```
#container {  
    grid-template-columns:  
        [sidebar-start] 1fr [sidebar-end] 4fr;  
}
```

```
#item {  
    grid-area: sidebar;  
}
```

EXERCISE

The Daily Cascade: invent your own grid and place the articles on it

Note that in this exercise, the articles are wrapped in a div

repeating grid definitions
using repeat()

repeating grid definitions

```
#container {  
  grid-template-columns: repeat([count], [size]);  
}
```

repeating grid definitions

```
#container {  
  grid-template-columns: repeat(4, 1fr);  
}
```

repeating with auto-fill

```
#container {  
  grid-template-columns: repeat(auto-fill, 10em);  
}
```

EXERCISE

Repeating grid items

picasso

braque

chagall

leger

miro

ernst

mondriaan

lipchitz

kandinsky

kokoschka

zadkine

appel

bazaine

corneille

delaunay

dubuffet

dagelijks geopend van 10.00 tot zondag van 18.00 uur

**van
abbemuseum
eindhoven**

autoplacement

Autoplacement is what the browser will do with grid items that are not explicitly placed on the grid.

If there aren't enough rows or columns, the browser will create them for you ('implicit grid')

sizing the implicit grid

`grid-auto-rows: [size];`

`grid-auto-columns: [size];`

direction of the implicit grid

grid-auto-flow: row | column;

filling up empty spaces automatically

grid-auto-flow: [row | column] dense;

**Note: auto-flow: dense can
make the location of your items
harder to understand for
keyboard users.**

EXERCISE

Rebuild a section of your
company (or personal) websites
using Grid Layout!

Grid in production

Best Practices With CSS Grid Layout — Smashing Magazine

s With CSS Grid Layout X + https://www.smashingmagazine.com/2018/04/best-practices-grid-layout/ Search

SMASHING MAGAZINE

Articles Books Events Jobs Membership Topics

Design & development Physical & digital books Conferences & workshops Find work & employees Webinars & early-birds

 **ABOUT THE AUTHOR**
Rachel Andrew is not only editor-in-chief of Smashing Magazine, but also a web developer, writer and speaker. She is the author of a number of books, including ... [More about Rachel...](#)

APRIL 16, 2018 • 4 COMMENTS

Best Practices With CSS Grid Layout

CSS 243 # Layouts 41 # Browsers 34

 An increasingly common question – now that people are using CSS Grid Layout in production – seems to be “What are the best practices?”
The short answer to this question is to use the layout method as defined in the specification. The particular parts of the spec you choose to use, and indeed how you combine Grid with other layout methods such as Flexbox, is down to what works for the patterns you are trying to build and how you and your team want to work.

 Members support

be consistent between visual and source order

which grid properties to choose?

using grid in combination with other layout methods

use as many grids as you like

“Is there a polyfill?”

Simple fallbacks using the cascade

Feature queries

Feature queries

```
@supports ( display: grid ) {  
  #item {  
    margin: 0;  
    width: auto;  
  }  
}
```

**“Is there a Grid
Layout framework?”**

“What to use Grid Layout for?”

Show what you
have on offer

Aardappel, groente, fruit online bestellen | AH.nl

Aardappel, groente, fruit online X +

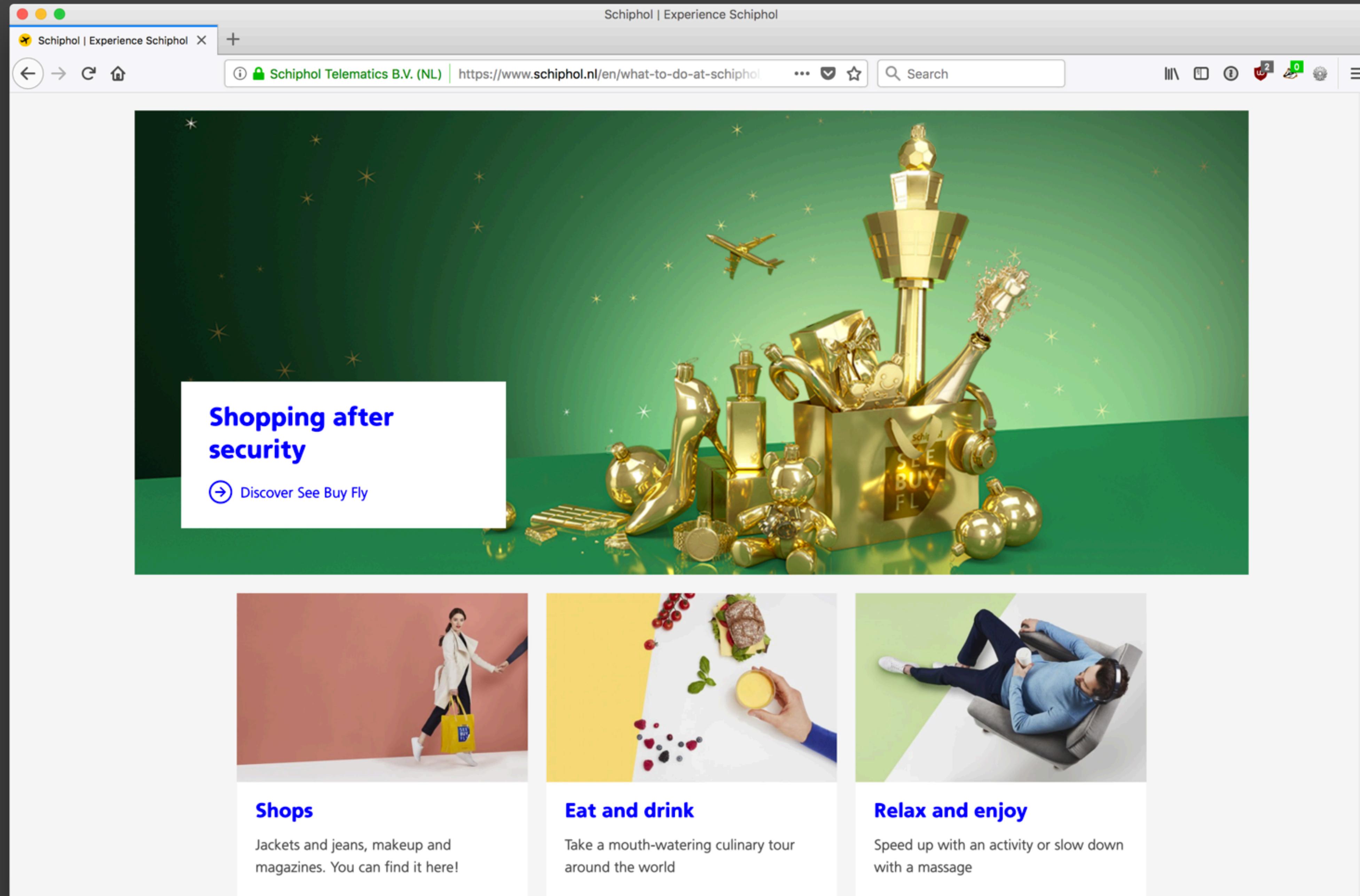
https://www.ah.nl/producten/aardappel-groente-fruit ... Search

Producten Bonus Allerhande box Recepten Winkels Acties Meer... Online bestellen 0

Proef heerlijk herfstfruit Bekijk Allerhande maaltijdkit Eet een salade

Stevige soep na een wandeling Maak zelf de lekkerste soep De lekkerste producten voor de herfst >

 AH Verspakket pompoensoep 1,2 kg	 AH Verse pureersoep pompoen-wortel 528 g	 AH Verspakket erwtensoep 900 g	 AH Verspakket tomatensoep 1,1 kg	
 AH Verse pureersoep tomaat-oregano 638 g	 AH Romatomaten 750 g	 AH Verspakket groentesoep 850 g	 AH Fijne soepgroente 250 g	 AH Erwtensoepgroente 500 g
				



Collecties

Collecties

https://www.bibliotheek.nl/onze-collectie.html

Search

Hoog contrast | Zet voorlezen aan

de Bibliotheek

home collecties in de Bibliotheek jeugd & jongeren meer van de bieb word lid

Waar ben je naar op zoek? Catalogus Vind Q

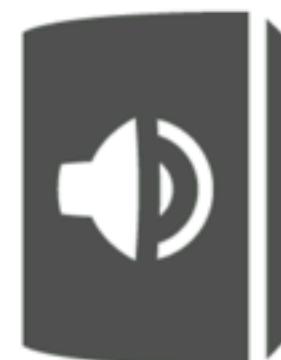


Collecties

Meer dan 4,3 miljoen items

Hier kan je door het totale bezit van alle Nederlandse Bibliotheken bladeren. Van boeken tot films, van games tot e-books. Verbaas je over de rijkdom van onze collecties. Wil je iets lenen? Dan verwijzen we je graag door naar de website van jouw eigen Bibliotheek.

Zoeken op materiaal



For programmes

Lock X CodePen - CSS Grid #6: Codev +

https://codepen.io/Rumyra/full/ZaoegM/ ... ⌂ ⌂ Search Fork Change View Log In Sign Up

CSS Grid #6: Codevember '17

A PEN BY Rumyra

Columbia University Graduate School of Architecture, Planning and Preservation

Lectures and Exhibitions Fall 1985

Wednesday Lecture Series

6:00 PM
Wood Auditorium
Avery Hall

100 Level Avery Hall

Exhibitions

Oct 2: Werner Seligman
Architect, Dean
School of
Architecture
Syracuse University.
"Frank Lloyd Wright:
The Evolution of the
Pararie House"

9: Sam Bass Warner Jr
William Edwards
Huntington Professor
of University Boston
University The Awful
History and Fresh
Promise of Urban
Gardens

SEP 23- OCT 18 Tianjin University
China "Student Work"

Want to learn more?

The New CSS Layout, Rachel Andrew

<https://abookapart.com>



Håkon Wium Lie's PhD thesis

<http://www.wiumlie.no/2006/phd/css.pdf>

The screenshot shows a Mac OS X-style PDF viewer window titled "css.pdf (page 1 of 306) — Locked". The window contains two pages of a thesis.

Page 1: The title page features the text "CASCADING STYLE SHEETS" and "Håkon Wium Lie". It includes a small circular logo and the text "Thesis submitted for the degree of Doctor Philosophiae Faculty of Mathematics and Natural Sciences University of Oslo Norway 2005". A blue number "1" is overlaid at the bottom center of this page.

Page 2: The second page contains the text "Håkon Wium Lie" and a large circular seal of the University of Oslo. The seal features a figure holding a book and a staff, surrounded by the text "UNIVERSITAS OSLOENSIS" and "MDCCCCXI". A blue number "2" is overlaid at the bottom center of this page.

Page Content:

CASCADING STYLE SHEETS
Håkon Wium Lie
Thesis submitted for the degree of Doctor Philosophiae
Faculty of Mathematics and Natural Sciences
University of Oslo
Norway
2005

© Håkon Wium Lie 2005
Published by the author under the terms of the Creative Commons License (CC-BY-NC-ND).
Sjølyst 1, Oslo 2005. All rights reserved.
This is a pre-publication version of the thesis.
An electronic copy of the thesis can be found at:
http://urn.nbn.no/URN:NBN:no-111111
URN: urn:nbn:no-111111
A digital version of the thesis can be found at:
http://urn.nbn.no/URN:NBN:no-111111
URN: urn:nbn:no-111111
This is a pre-publication version of the thesis.
Published in Monographie de l'Ac. Oslo 2006.
It is illegal to reproduce or distribute this thesis without the author's explicit permission. Any such unauthorized reproduction or distribution is a violation of copyright law.
Copyright © University of Oslo 2006. This thesis is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike license.
http://creativecommons.org/licenses/by-nc-sa/2.0/no/

ABSTRACT

Jen Simmons | Labs

Jen Simmons | Labs [x](#) [+](#)

[labs.jensimmons.com](#)

Intro to CSS Grid

5 Basic Examples of how CSS Grid Works

12 Variations of Card Layouts

Example of Nesting Flexbox and Grid

AEA Boston 2017 2016 demos

The Experimental Layout Lab of Jen Simmons

See how these demos work by inspecting them with the [Firefox Grid Inspector Tool](#). Learn how to use [CSS Grid](#).

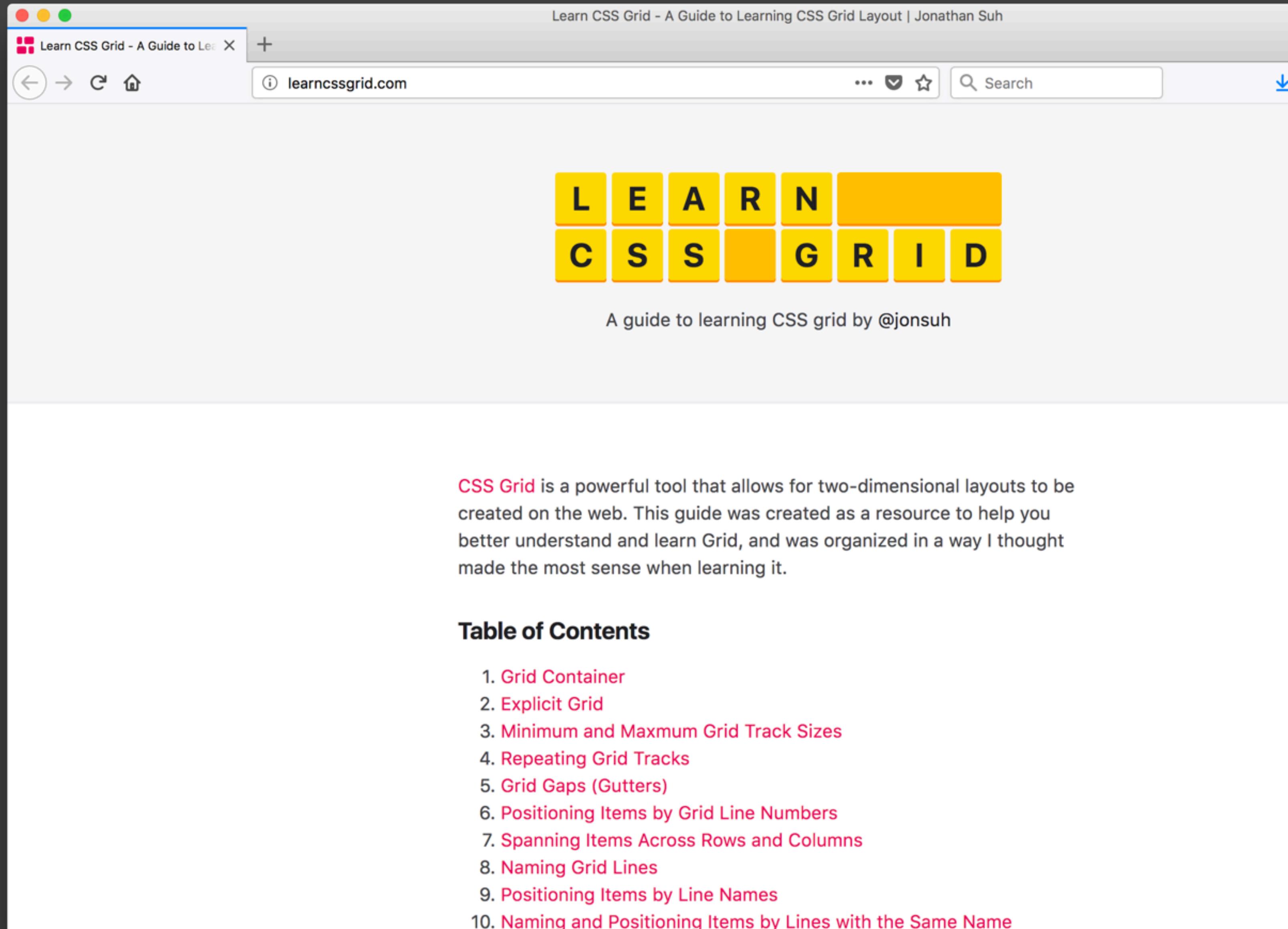
on Codepen

The Experimental Layout Lab of Jen Simmons

<http://labs.jensimmons.com/>

Learn CSS Grid

<http://learncssgrid.com>



The screenshot shows a web browser window with the title bar "Learn CSS Grid - A Guide to Learning CSS Grid Layout | Jonathan Suh". The address bar shows the URL "learncssgrid.com". The main content area features a large graphic of the words "LEARN CSS GRID" composed of yellow squares. Below the graphic is the text "A guide to learning CSS grid by @jonsuh". To the right of the graphic, there is a paragraph about CSS Grid. At the bottom, there is a "Table of Contents" section with a numbered list of 10 items.

Learn CSS Grid - A Guide to Learning CSS Grid Layout | Jonathan Suh

learncssgrid.com

Search

L E A R N
C S S G R I D

A guide to learning CSS grid by @jonsuh

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

Table of Contents

1. Grid Container
2. Explicit Grid
3. Minimum and Maximum Grid Track Sizes
4. Repeating Grid Tracks
5. Grid Gaps (Gutters)
6. Positioning Items by Grid Line Numbers
7. Spanning Items Across Rows and Columns
8. Naming Grid Lines
9. Positioning Items by Line Names
10. Naming and Positioning Items by Lines with the Same Name

CSS Day 2018, 14th and 15th of June, Amsterdam

CSS Day 2018, 14th and 15th of June, Amsterdam X +

CSS Day 2018, 14th and 15th of June, Amsterdam https://cssday.nl/2018 Search

2018 2017 2016 2015 2014 2013



```
#ux-special + .css-day {  
    June 14 & 15, 2018;  
}  
  
Compagnietheater, Amsterdam
```

Subscribe for updates!

On Thursday 14th and Friday 15th of June, 2018, CSS Day is returning with two conference days: a UX Special on Thursday, and a regular CSS Day on Friday.

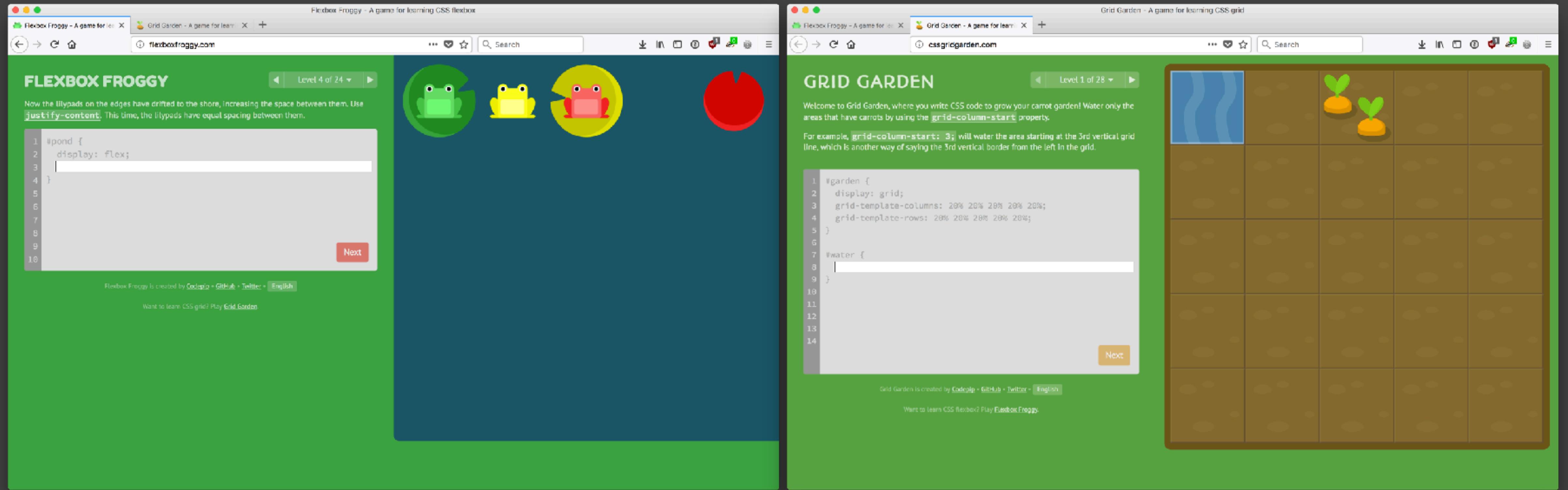
See the [coverage of the 2017 edition](#) for an idea of what you can expect. (Or [2016](#), for that matter.)

As always the conference will take place in [Het Compagnietheater](#). Ticket sales haven't started yet, but you can [subscribe for updates](#).

We hope to see you then!

CSS Day

<https://cssday.nl/2018>



Flexbox Froggy / CSS Grid Garden

<http://flexboxfroggy.com/>

<http://cssgridgarden.com/>



Jen Simmons / @jensimmons

Rachel Andrew / @rachelandrew

Elika J. Etemad / @fantasai

Tab Atkins / @tabatkins

Hui Jing Chen / @hj_chen

Mary Lou / @crnacura

Eric Meyer / @ericmeyer

CSS Working Group / @csswg

WORKSHOP

CSS Layout

Thanks for being here
today! Any questions?

Feel free to contact me on:

@hdv
hidde@hiddedevries.nl