 Lecture plan

- ✏️ CSS Cascade. CSS Specificity
- ✏️ CSS Custom Properties
- ✏️ CSS Units
- ✏️ CSS Layouts — Flexbox
- ✏️ CSS Layouts — Grid
- ✏️ CSS Modules & CSS-in-JS
- ✏️ Technologies

CSS Cascade

🤓 CSS Cascade definition

> *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.*
>
> ***CSS WG***

Cascade sorting order

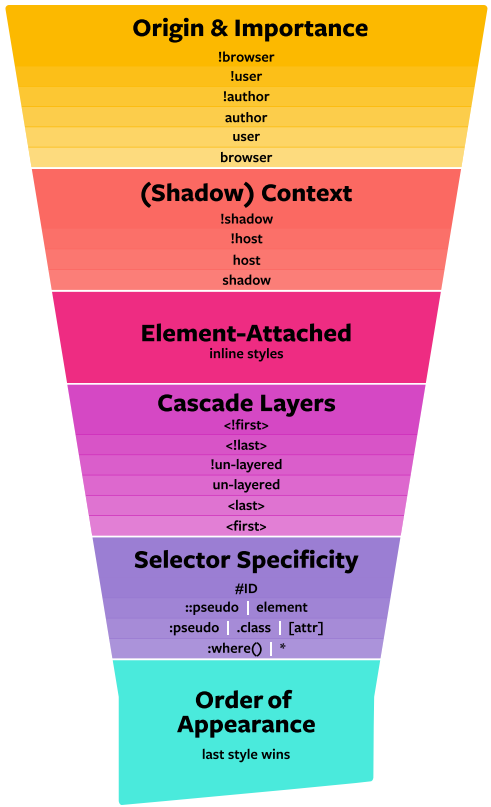
- 👉 Origin and Importance
- 👉 Context
- 👉 Specificity
- 👉 Order of Appearance

Cascade Sorting Order

Origin and Importance

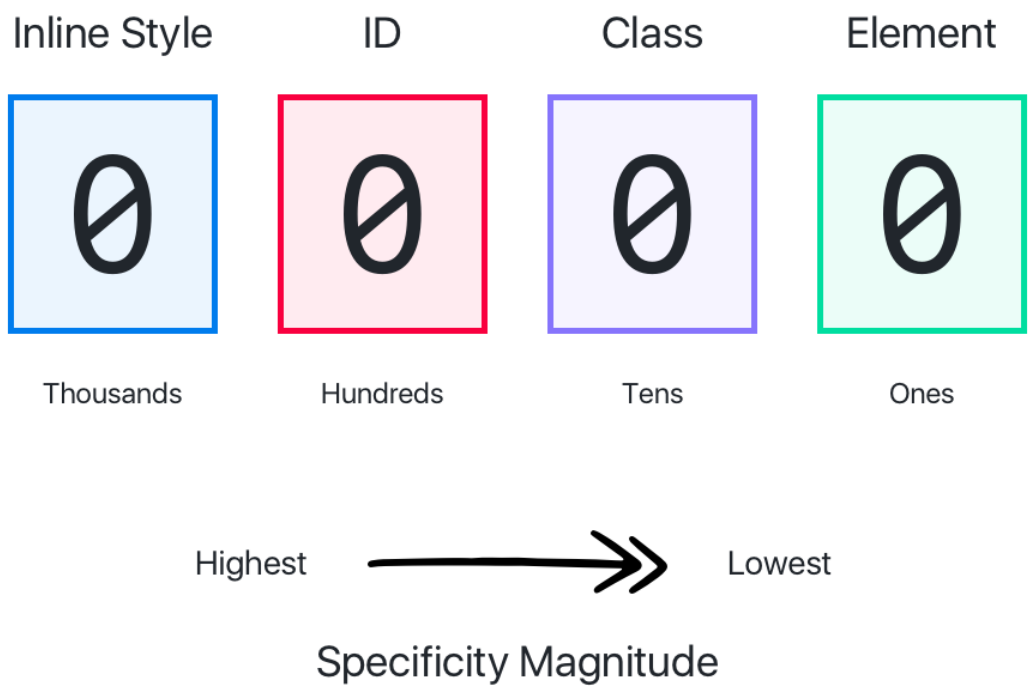
1. Transition declarations
2. !important user agent declarations
3. !important user declarations
4. !important author declarations
5. Animation declarations
6. Normal author declarations
7. Normal user declarations
8. Normal user agent declarations

Summary



Source

Specificity



[Specificity Calculator]

CSS Specificity Quiz 1/8

```
section p {
  color: red;
}
.p1 {
  color: green;
}
```

```
<section class="section" id="section">
  <h1>Title</h1>
  <p class="p1">Paragraph 1</p>
  <p class="p2">Paragraph 2</p>
</section>
```

💡 The specificity of section p is 0.0.2 and .p1 is 0.1.0

CSS Specificity Quiz 2/8

```
.section p {
  color: red;
}
.p1 {
  color: green;
}
```

```
<section class="section" id="section">
  <h1>Title</h1>
  <p class="p1">Paragraph 1</p>
  <p class="p2">Paragraph 2</p>
</section>
```

💡 The specificity of .section1 p is 0.1.1 and .p1 is 0.1.0

CSS Specificity Quiz 3/8

```
#section {
  color: red;
}
.p1 {
  color: green;
}
```

```
<section class="section" id="section">
  <h1>Title</h1>
  <p class="p1">Paragraph 1</p>
  <p class="p2">Paragraph 2</p>
</section>
```

💡 p would inherit its parent's color only if there is no color applied directly to paragraphs

CSS Specificity Quiz 4/8

```
#section * {
  color: red;
}
.p1 {
  color: green;
}
```

```
<section class="section" id="section">
  <h1>Title</h1>
  <p class="p1">Paragraph 1</p>
  <p class="p2">Paragraph 2</p>
</section>
```

💡 The specificity of #section1 * is 1.0.0 and .p1 is 0.1.0

CSS Specificity Quiz 5/8

```
01. section * {
02.   color: 🟥 red;
03. }
04. p {
05.   color: 🟩 green;
06. }
```

```
01. <section class="section" id="section">
02.   <h1>Title</h1>
03.   <p class="p1">Paragraph 1</p>
04.   <p class="p2">Paragraph 2</p>
05. </section>
```

💡 The specificity of section * is 0.0.1. The specificity of p is also 0.0.1 but it comes later in the stylesheet .

CSS Specificity Quiz 6/8

```
section .p1 {
  color: red;
}
section :nth-child(2) {
  color: green;
}
```

```
<section class="section" id="section">
  <h1>Title</h1>
  <p class="p1">Paragraph 1</p>
  <p class="p2">Paragraph 2</p>
</section>
```

💡 The specificity of section .p1 is 0.1.1. The specificity of section :nth-child(2) is also 0.1.1 but it comes later in the stylesheet.

CSS Specificity Quiz 7/8

```
01. section > p {
02.   color: red;
03. }
04. section p {
05.   color: green;
06. }
```

```
01. <section class="section" id="section">
02.   <h1>Title</h1>
03.   <p class="p1">Paragraph 1</p>
04.   <p class="p2">Paragraph 2</p>
05. </section>
```

💡 The specificity of section > p is 0.0.2. The specificity of section p is also 0.0.2 but it comes later in the stylesheet.

CSS Specificity Quiz 8/8

```
.red {
  color: red;
}
.green {
  color: green;
}
```

```
<section class="section" id="section">
  <p class="green red">Paragraph</p>
</section>
```

💡 The specificity of .red is 0.1.0. The specificity of .green is also 0.1.0 but it comes later in the stylesheet.

CSS Custom Properties

CSS Custom Properties

Property names that are prefixed with --, like --example-name, represent custom properties that contain a value that can be used in other declarations using the var() function.

Custom properties are scoped to the element(s) they are declared on, and participate in the cascade: the value of such a custom property is that from the declaration decided by the cascading algorithm.

- 👉 Create complex CSS properties
- 👉 Create color themes
- 👉 Use it with JavaScript (e.g. animation)
- 👉 --env browser properties

CSS Custom Properties

```
01. :root {
02.   --bg-color: plum; // declare a custom variable
03.   --base-padding: 0.5rem;
04. }
05. .Component {
06.   background-color: var(--bg-color); // use the custom variable
07.   padding: calc(2 * var(--base-padding));
08. }
```

Why do we need --env?

Designing Websites for iPhone X

Browser --env properties

```
01. .container {
02.     padding:  env(safe-area-inset-top)
                  env(safe-area-inset-right)
                  env(safe-area-inset-bottom)
                  env(safe-area-inset-left);
03. }
```

```
getComputedStyle(element).getPropertyValue('env(safe-area-inset-top)')
```

👨‍💻 [Practice] CSS Custom Properties

1. Convert it to CSS Custom Properties
2. (optional) Make it responsive - change base font size on different screen sizes

   type-scale.com

   Codesandbox

Comparison to CSS preprocessors

Preprocessors variables are...

- 👉 Both global & local
- 👉 Static
- 👉 Not flexible

CSS Custom Properties are...

- 👉 Global
- 👉 Dynamic
- 👉 Accessible from JS land
- 👉 CSS as API (exaple)

JavaScript API

```
01. let element = document.querySelector("#element")
02. let styles = getComputedStyle(element)
03. let size = +styles.getPropertyValue("--base-size")
04. element.style.setProperty("--base-size", size + 4)
```

CSS Custom Properties

- 👍 Native solution
- 👍 Obeyed to CSS Cascade
- 👍 Dynamic nature
- 👍 Accessible via JS
- 👍 Could be pollyfilled in some base cases
- 👍 Good compilation target

- 🤔 Performance

CSS units

Absolute and Relative units

Unit	Name	Equivalent to
`cm`	Centimeters	1cm = 37.8px = 25.2/64in
`mm`	Millimeters	1mm = 1/10th of 1cm
`Q`	Quarter-millimeters	1Q = 1/40th of 1cm
`in`	Inches	1in = 2.54cm = 96px
`pc`	Picas	1pc = 1/6th of 1in
`pt`	Points	1pt = 1/72nd of 1in
`px`	Pixels	1px = 1/96th of 1in

Docs, Viewports

Unit	Relative to
`em`	Font size of the parent, in the case of typographical properties like `font-size`, and font size of the element itself, in the case of other properties like `width`.
`ex`	x-height of the element's font.
`ch`	The advance measure (width) of the glyph "0" of the element's font.
`rem`	Font size of the root element.
`lh`	Line height of the element.
`rlh`	Line height of the root element. When used on the `font-size` or `line-height` properties of the root element, it refers to the properties' initial value.
`vw`	1% of the viewport's width.
`vh`	1% of the viewport's height.
`vmin`	1% of the viewport's smaller dimension.
`vmax`	1% of the viewport's larger dimension.
`vb`	1% of the size of the initial containing block in the direction of the root element's block axis.
`vi`	1% of the size of the initial containing block in the direction of the root element's inline axis.
`svw`, `svh`	1% of the small viewport's width and height, respectively.
`lvw`, `lvh`	1% of the large viewport's width and height, respectively.
`dvw`, `dvh`	1% of the dynamic viewport's width and height, respectively.

CSS Layouts
Flexbox & Grid

Responsive Design could be achived with...

- 👉 Typography
- 👉 Media
- 👉 Meta tag viewport
- 👉 Media Queries
- 👉 **CSS Flexbox**
- 👉 **CSS Grids**
- 👉 CSS Container Queries

CSS Flexbox API

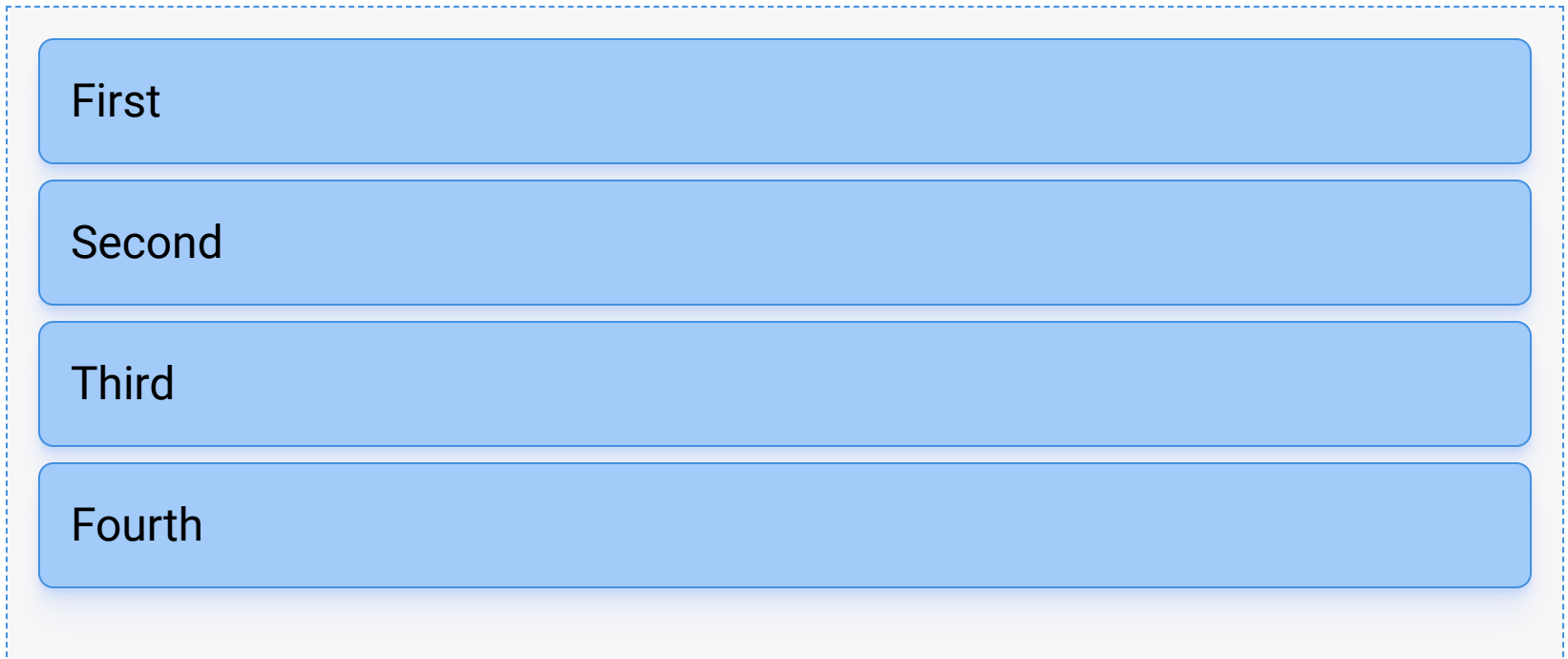
- flex-direction
- justify-content
- align-items
- order
- flex-grow, flex-shrink, flex-basis
- flex-wrap
- align-content

A Guide to Flexbox

initial state

```
01. <div class="container">
02.     <div class="item">First</div>
03.     <div class="item">Second</div>
04.     <div class="item">Third</div>
05.     <div class="item">Fourth</div>
06. </div>
```

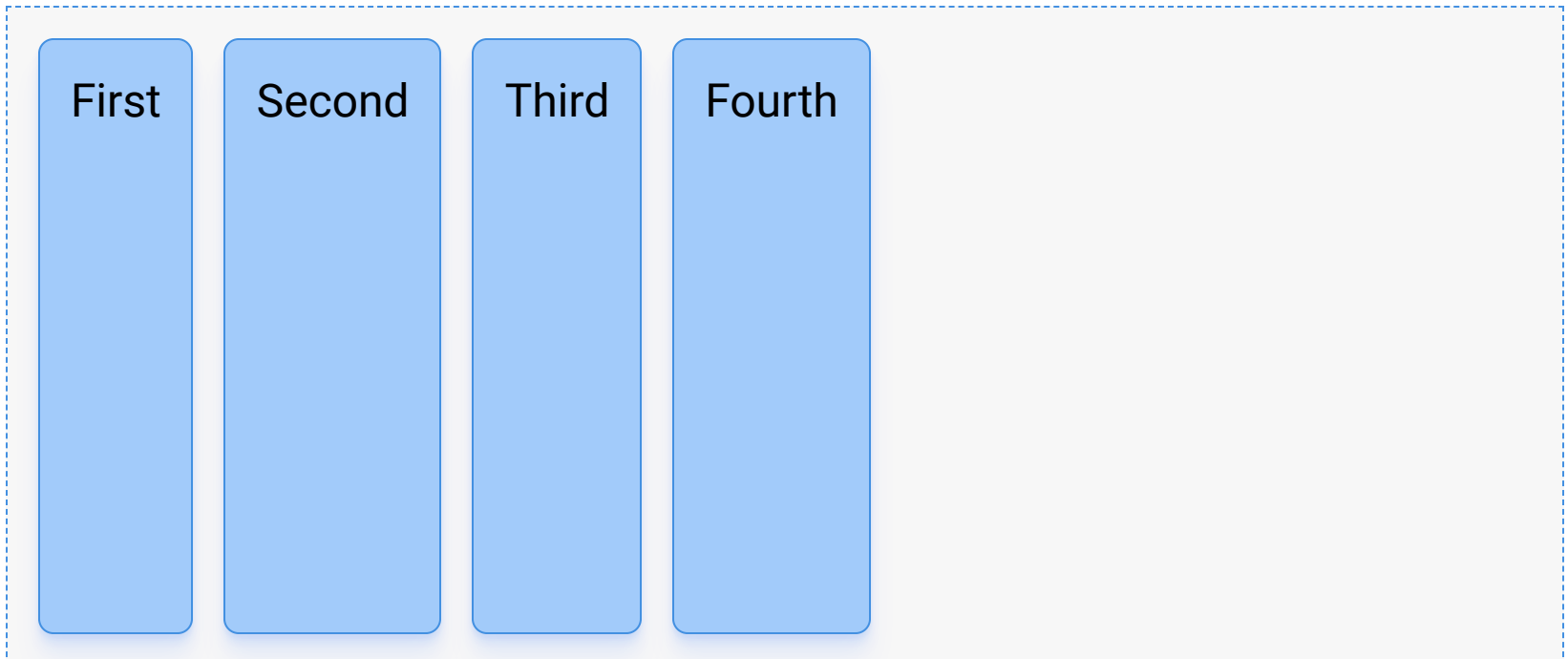
initial state



display: flex

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

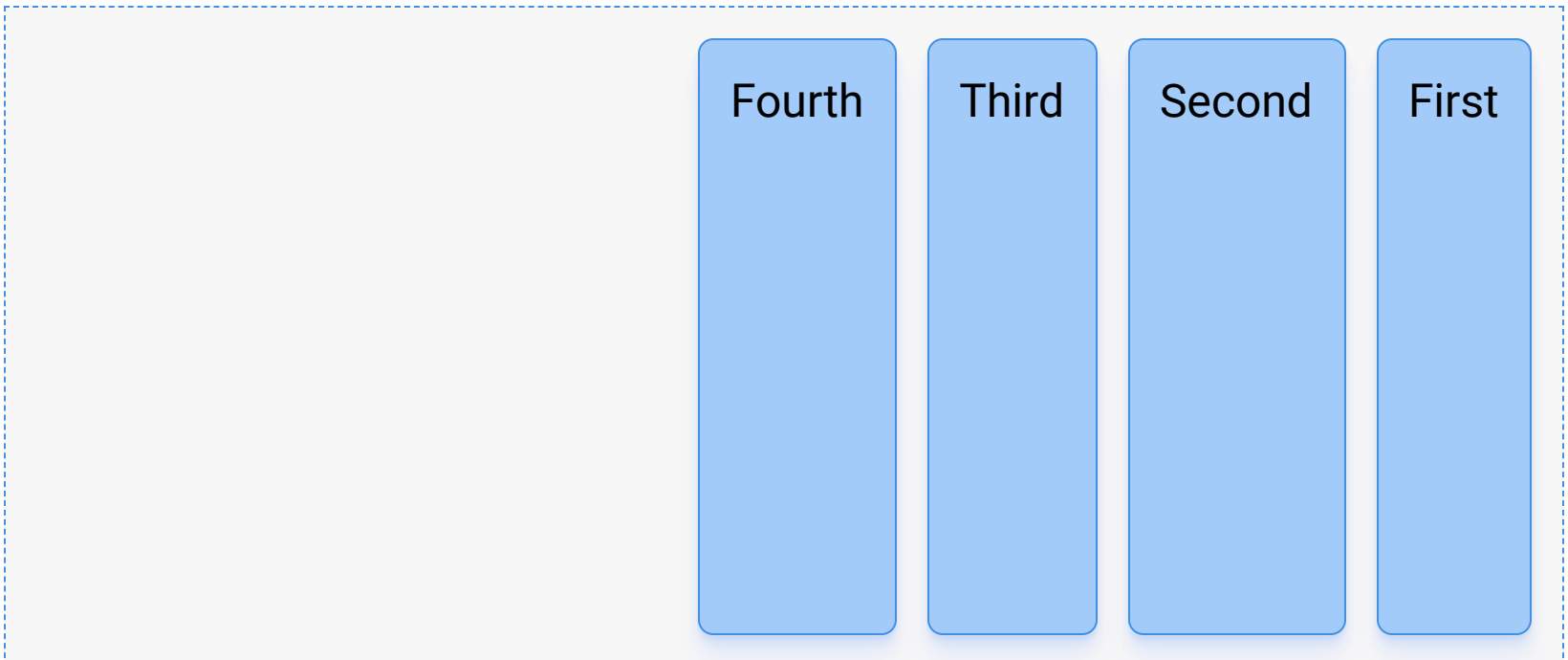
display: flex



flex-direction: row-reverse

```
01. .container {
02.     display: flex;
03.     flex-direction: row-reverse;
04. }
```

flex-direction: row-reverse



flex-direction: column

```
01. .container {
02.     display: flex;
03.     flex-direction: column;
04. }
```

flex-direction: column

flex-direction: column-reverse

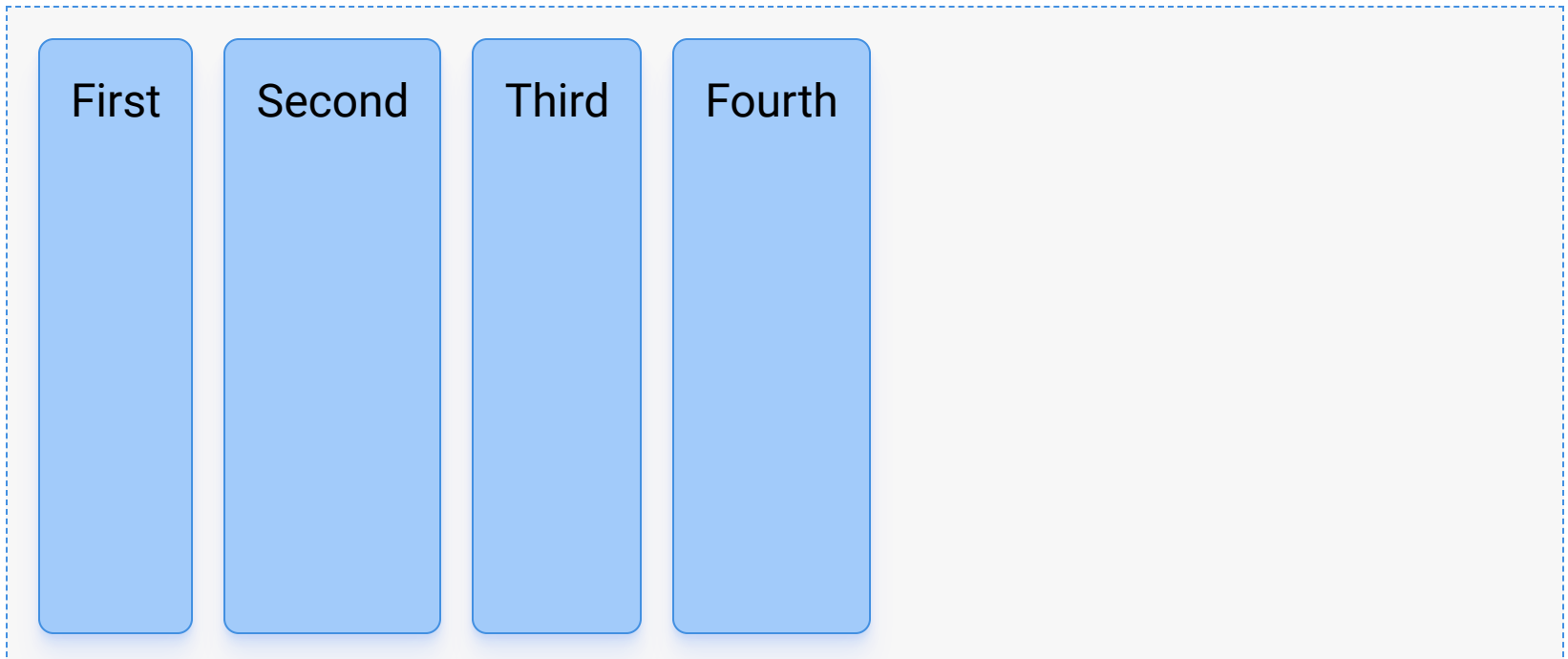
```
01. .container {
02.     display: flex;
03.     flex-direction: column-reverse;
04. }
```

flex-direction: column-reverse

justify-content: flex-start (default)

```
.container {
    display: flex;
    justify-content: flex-start;
}
```

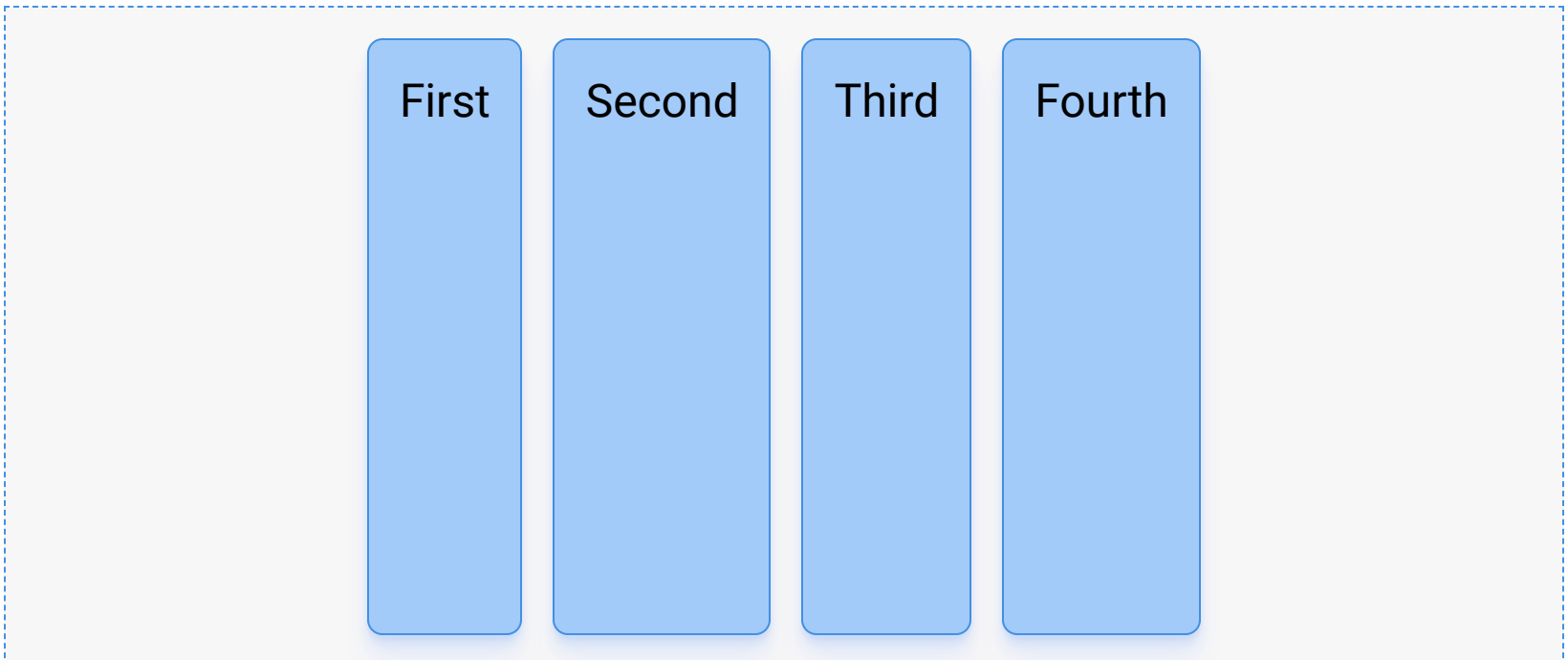
justify-content: flex-start (default)



justify-content: center

```
.container {
    display: flex;
    justify-content: center;
}
```

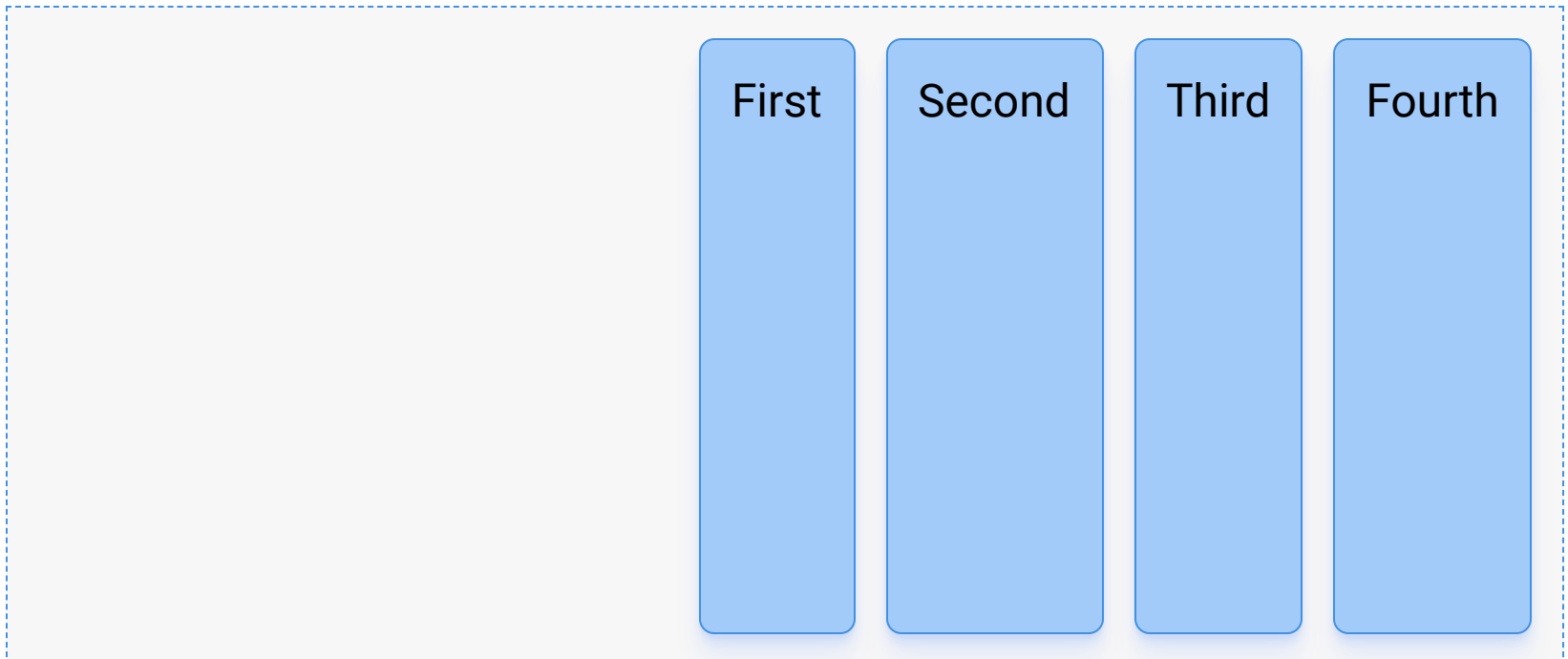
justify-content: center



justify-content: flex-end

```
.container {
    display: flex;
    justify-content: flex-end;
}
```

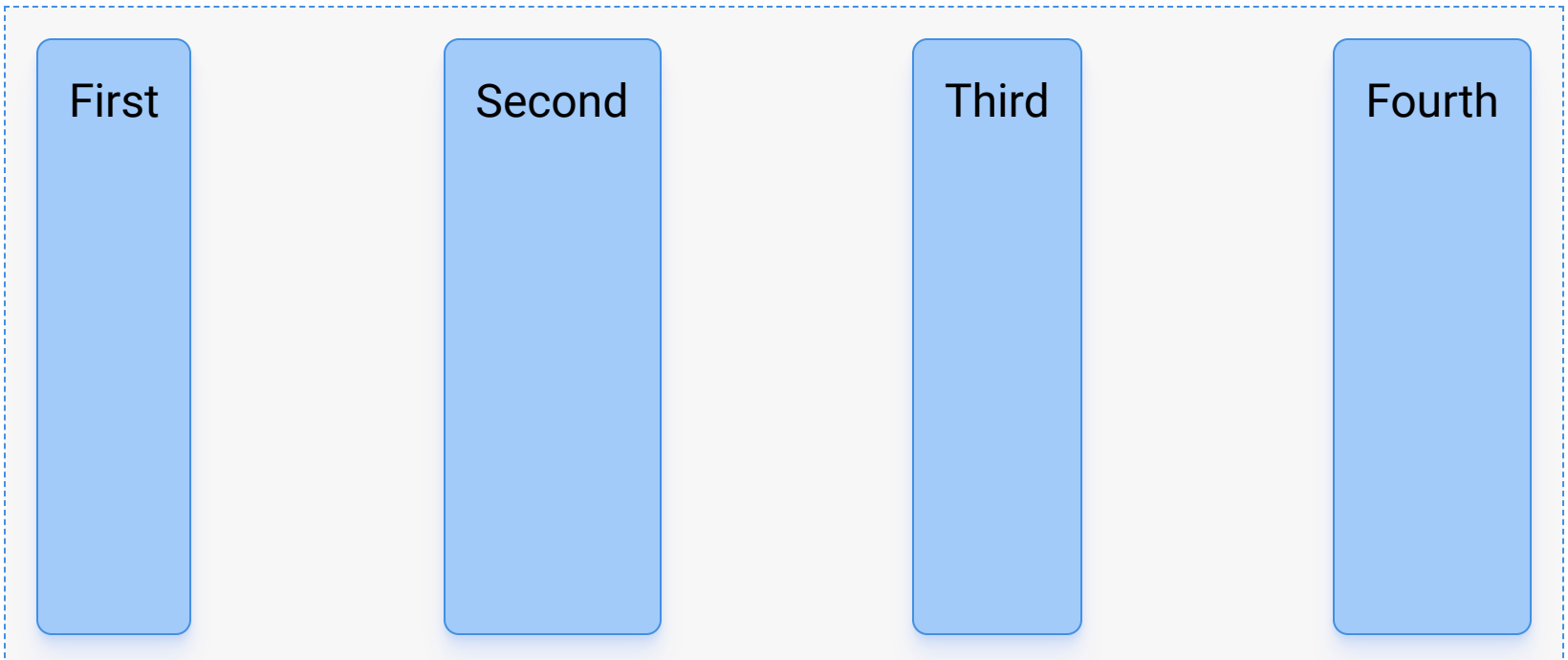
justify-content: flex-end



justify-content: space-between

```
.container {
    display: flex;
    justify-content: space-between;
}
```

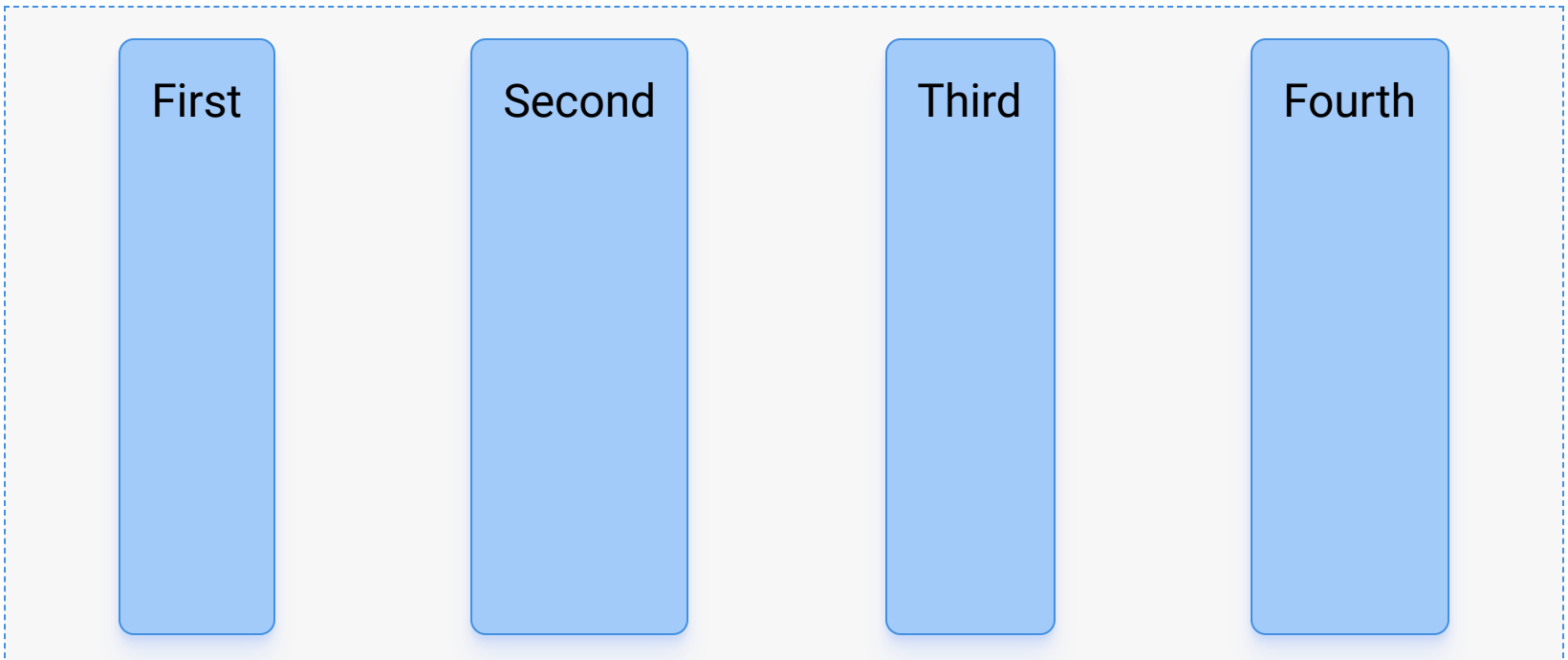
justify-content: space-between



justify-content: space-around

```
.container {
    display: flex;
    justify-content: space-around;
}
```

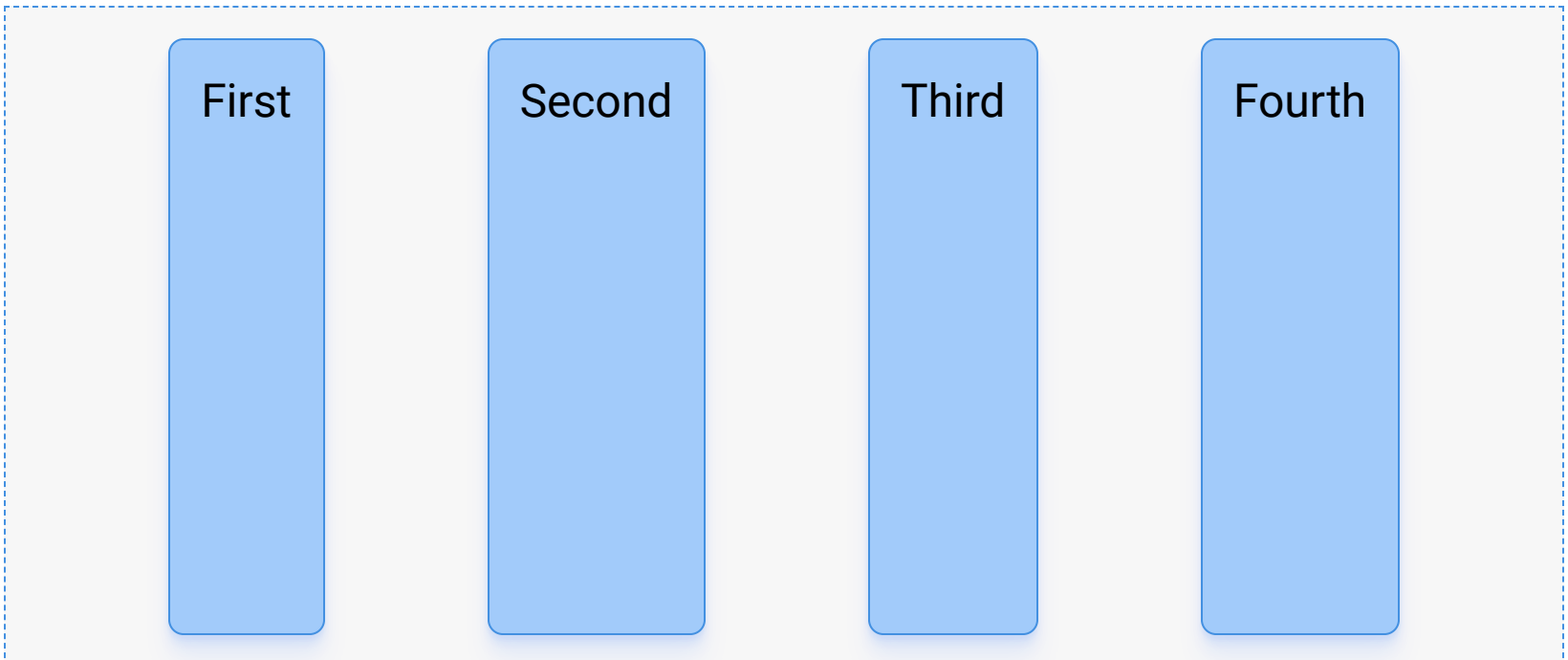
justify-content: space-around



justify-content: space-evenly

```
.container {
    display: flex;
    justify-content: space-evenly;
}
```

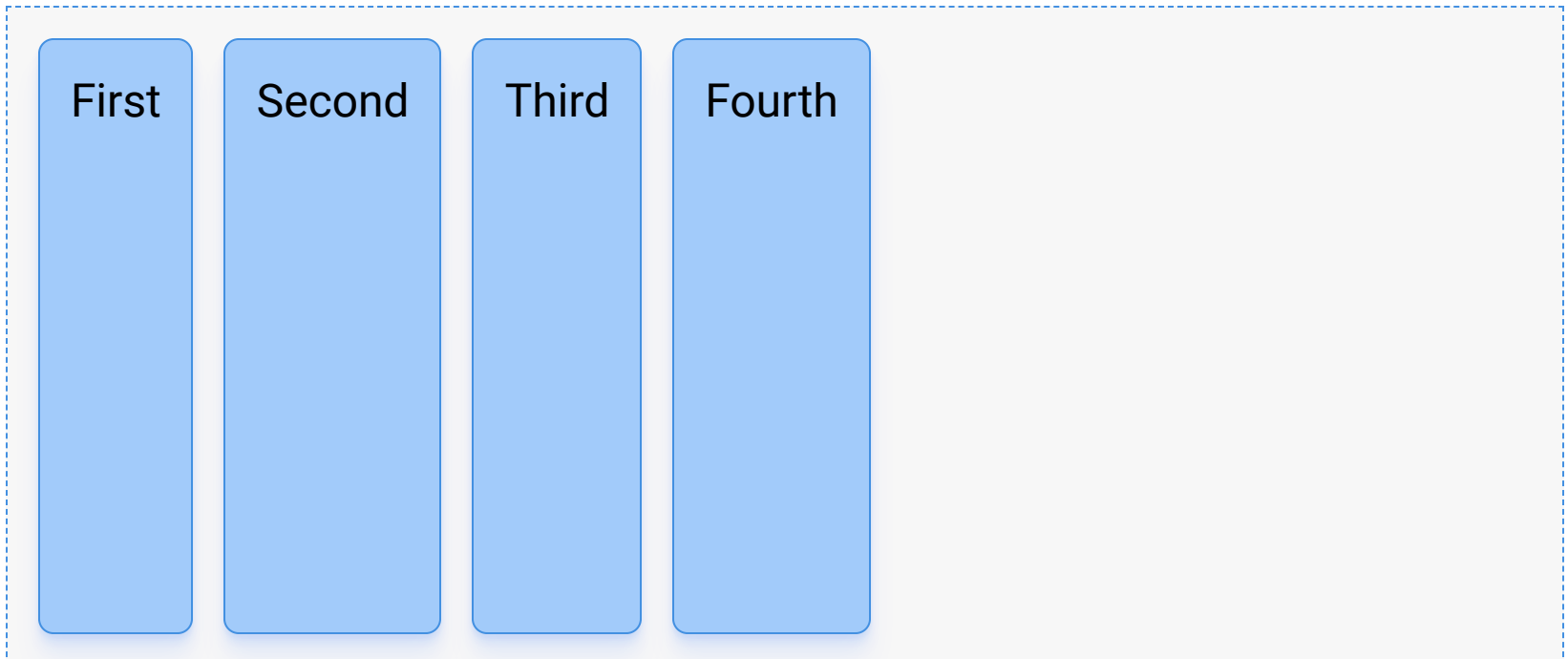
justify-content: space-evenly



align-items: stretch (default)

```
.container {
    display: flex;
    align-items: stretch;
}
```

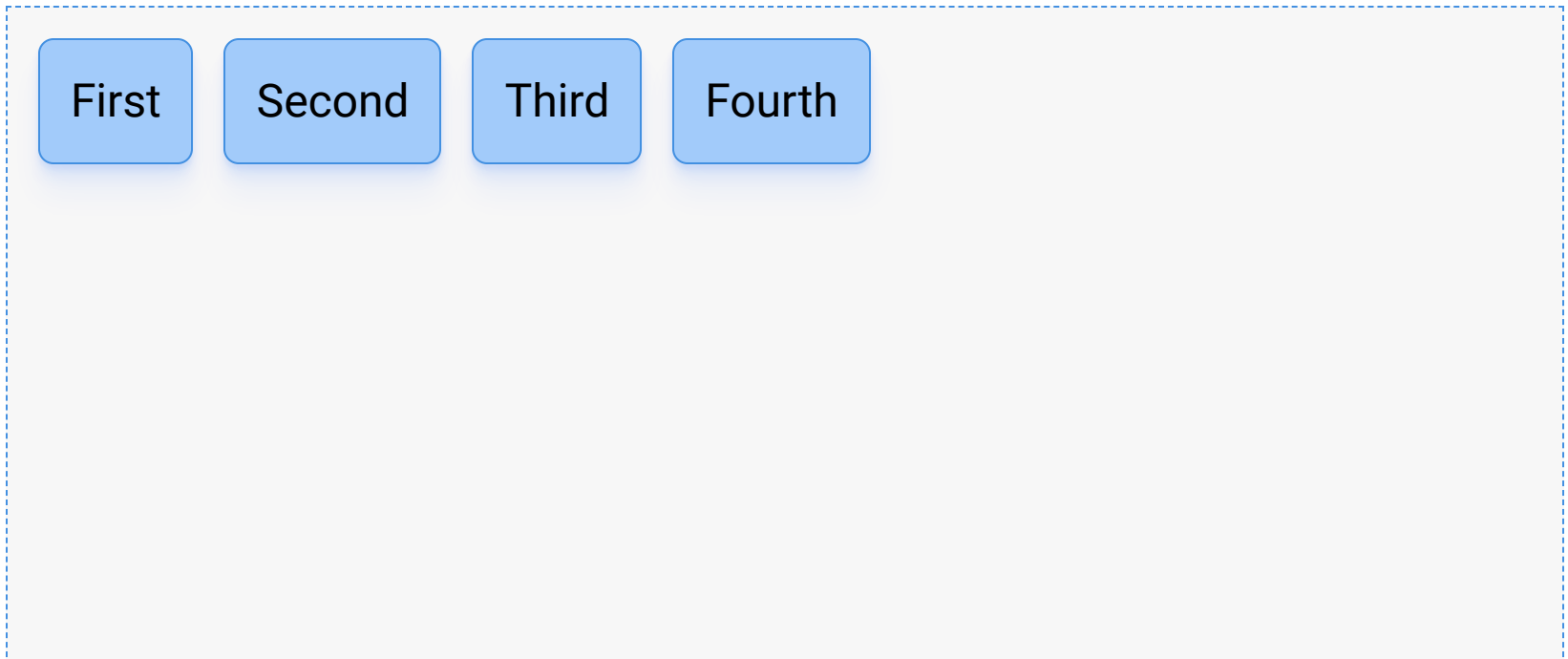
align-items: stretch (default)



align-items: flex-start

```
01. .container {
02.     display: flex;
03.     align-items: flex-start;
04. }
```

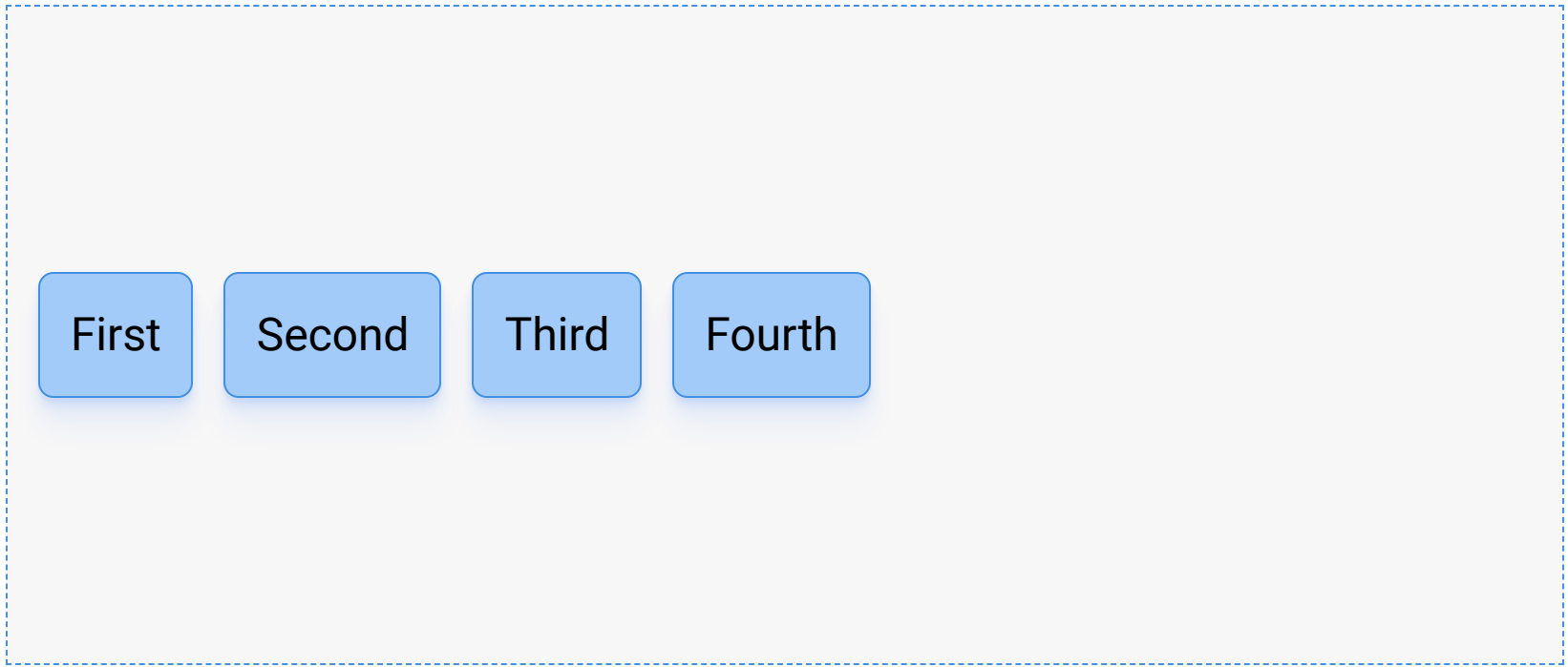
align-items: flex-start



align-items: center

```
.container {
    display: flex;
    align-items: center;
}
```

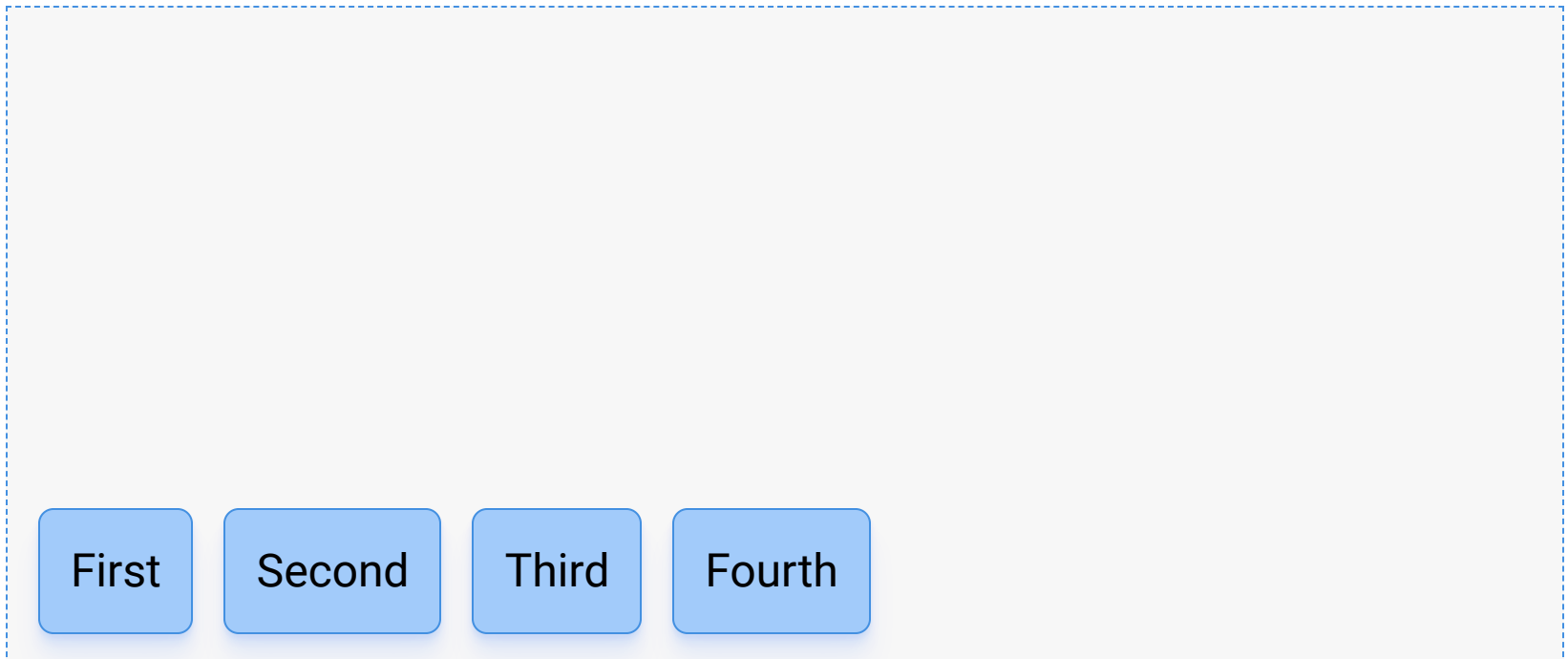
align-items: center



align-items: flex-end

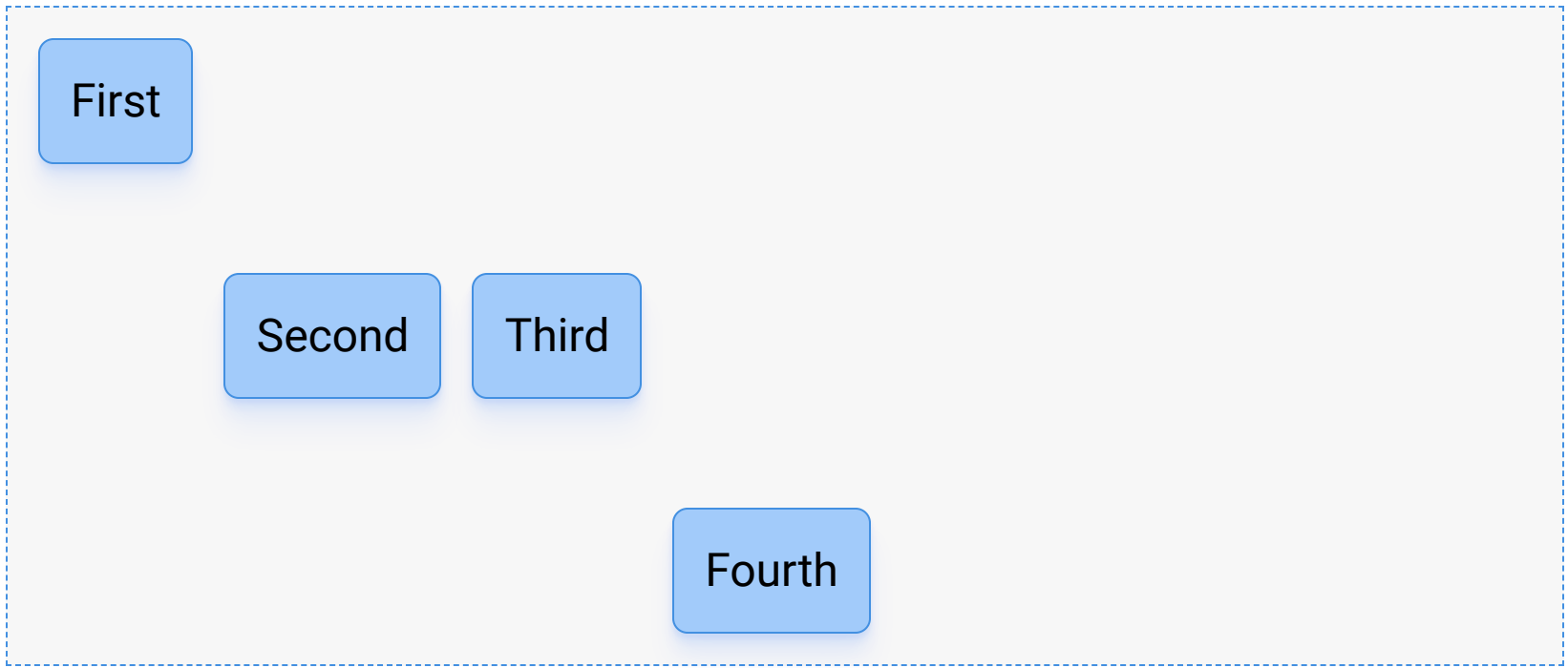
```
01. .container {
02.     display: flex;
03.     align-items: flex-end;
04. }
```

align-items: flex-end



```css
.container {
    display: flex;
    align-items: center;
}
.item:nth-child(1) {
    align-self: flex-start;
}
.item:nth-child(4) {
    align-self: flex-end;
}
```

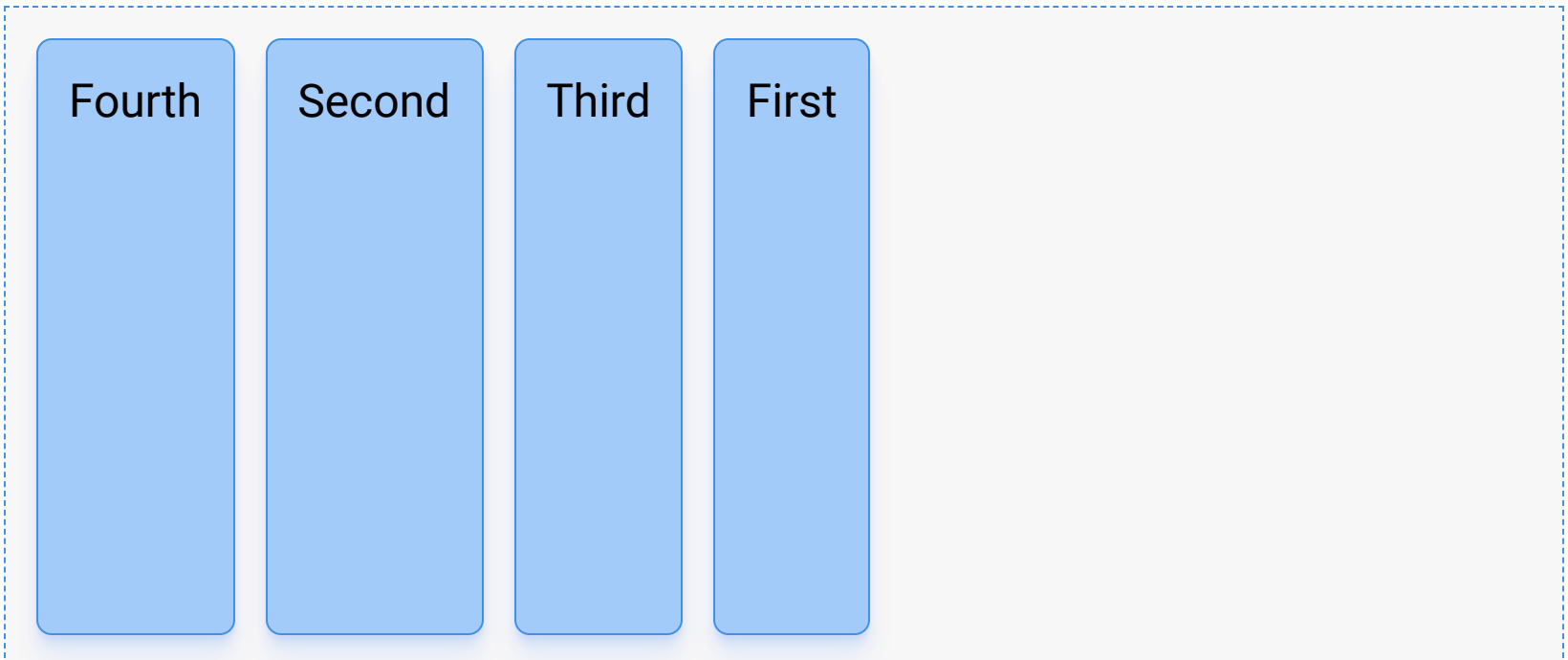
align-self



Order

```
.item {
    order: 1;
}
.item:nth-child(1) {
    order: 2;
}
.item:nth-child(4) {
    order: -1;
}
```

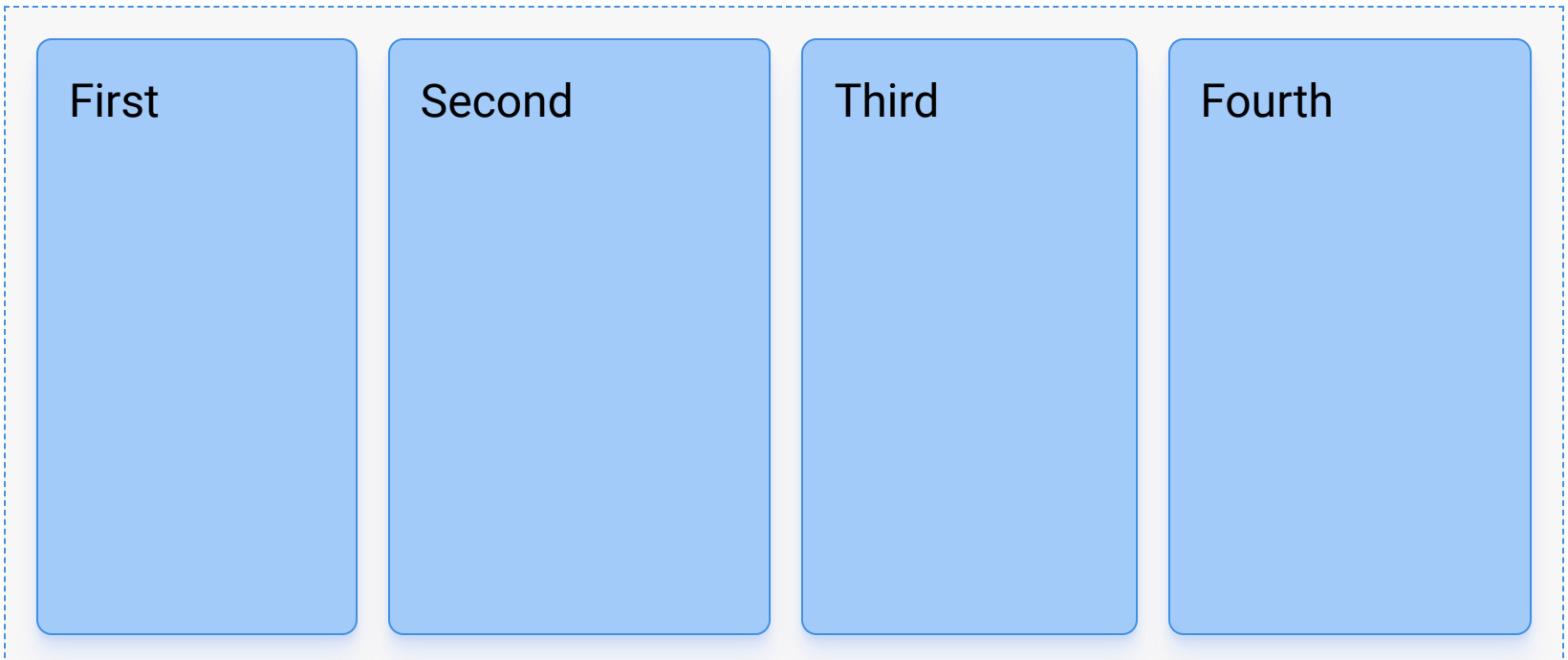
Order



flex-grow

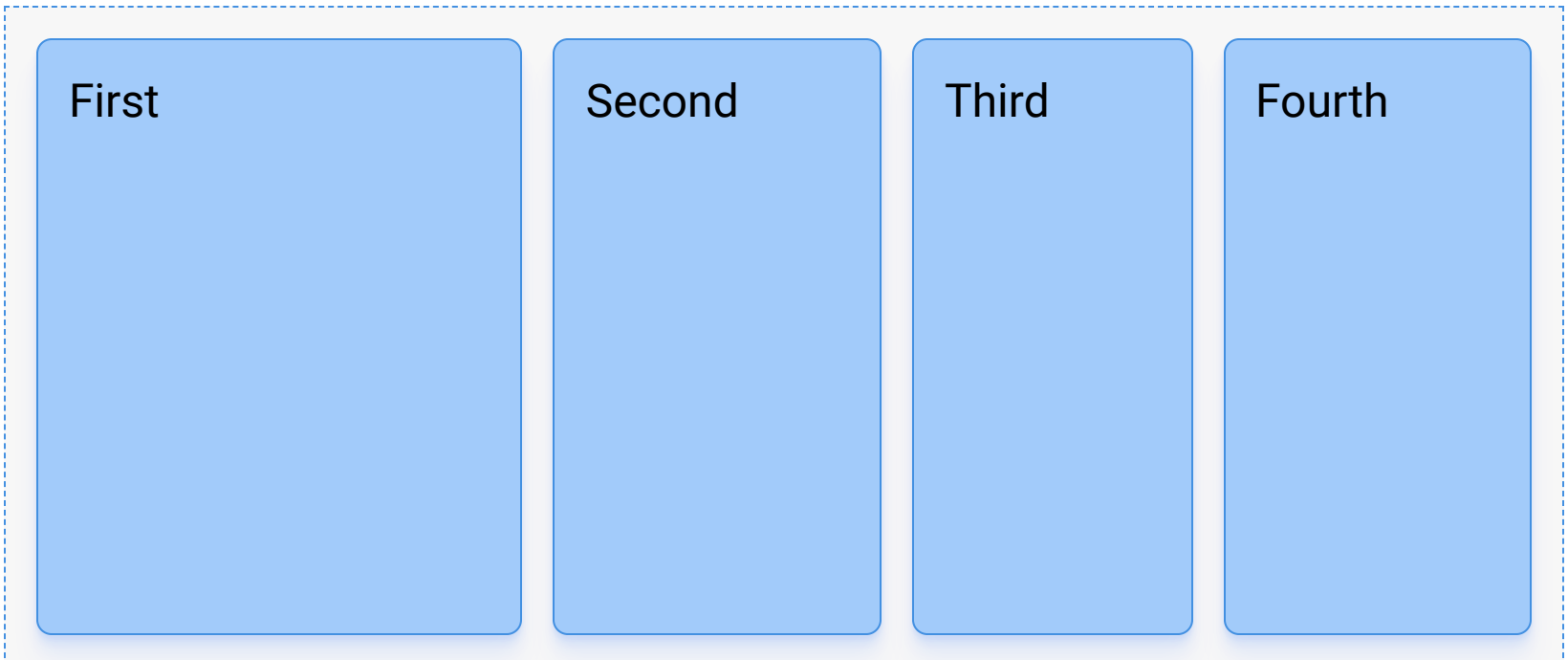
```
.container {
    display: flex;
}
.item {
    flex-grow: 0; /* By default */
    flex-grow: 1;
}
```

flex-grow



```
01. .container {
02.     display: flex;
03. }
04. .item {
05.     flex-grow: 1;
06. }
07. .item:nth-child(1) {
08.     flex-grow: 3;
09. }
```

flex-grow



flex-shrink

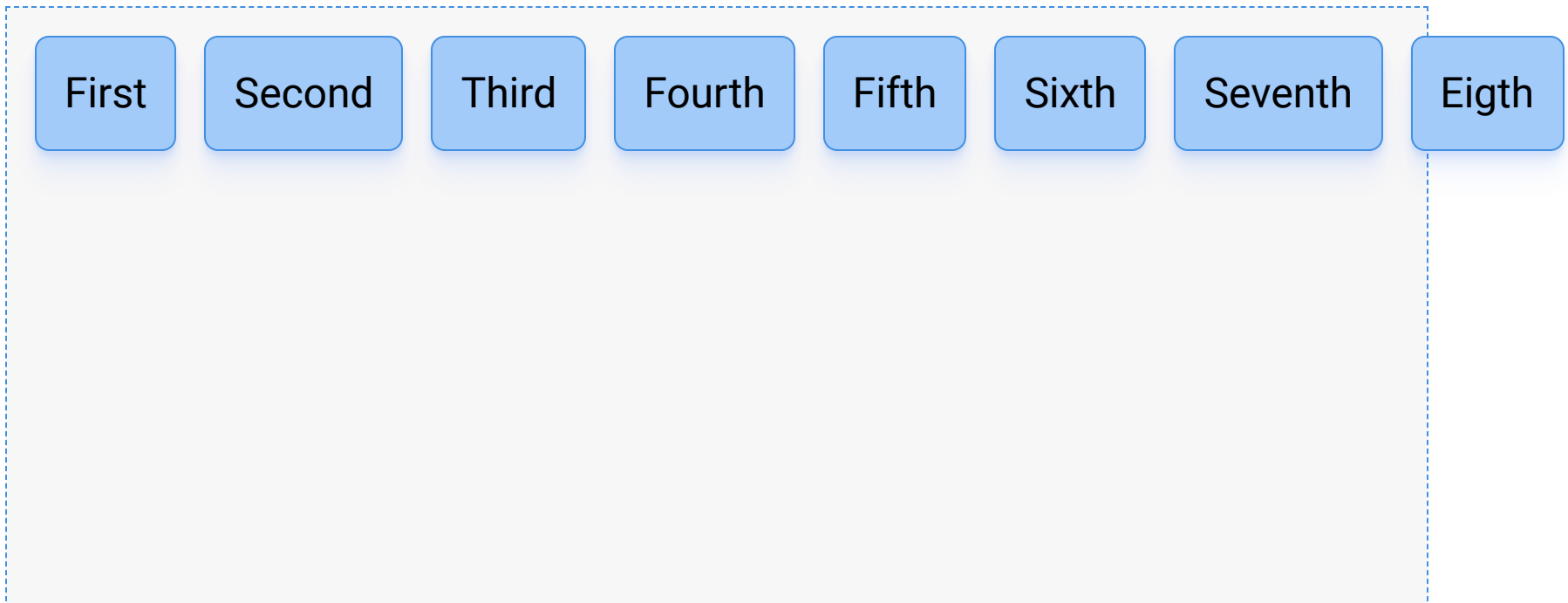
Example

flex-basis

Codesandbox

Difference between flex-basis and width

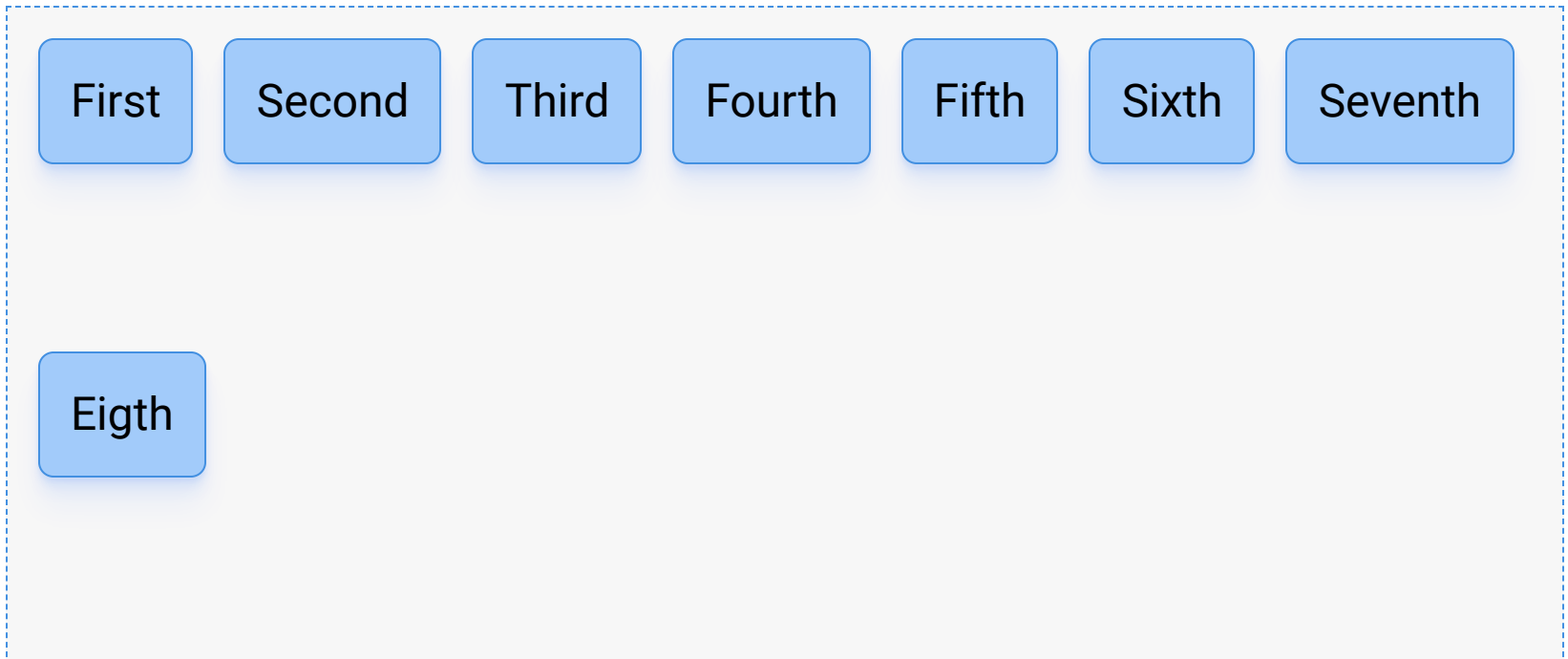
flex-wrap: nowrap



flex-wrap: wrap

```
.container {
    display: flex;
    align-items: flex-start;
    flex-wrap: wrap;
}
```

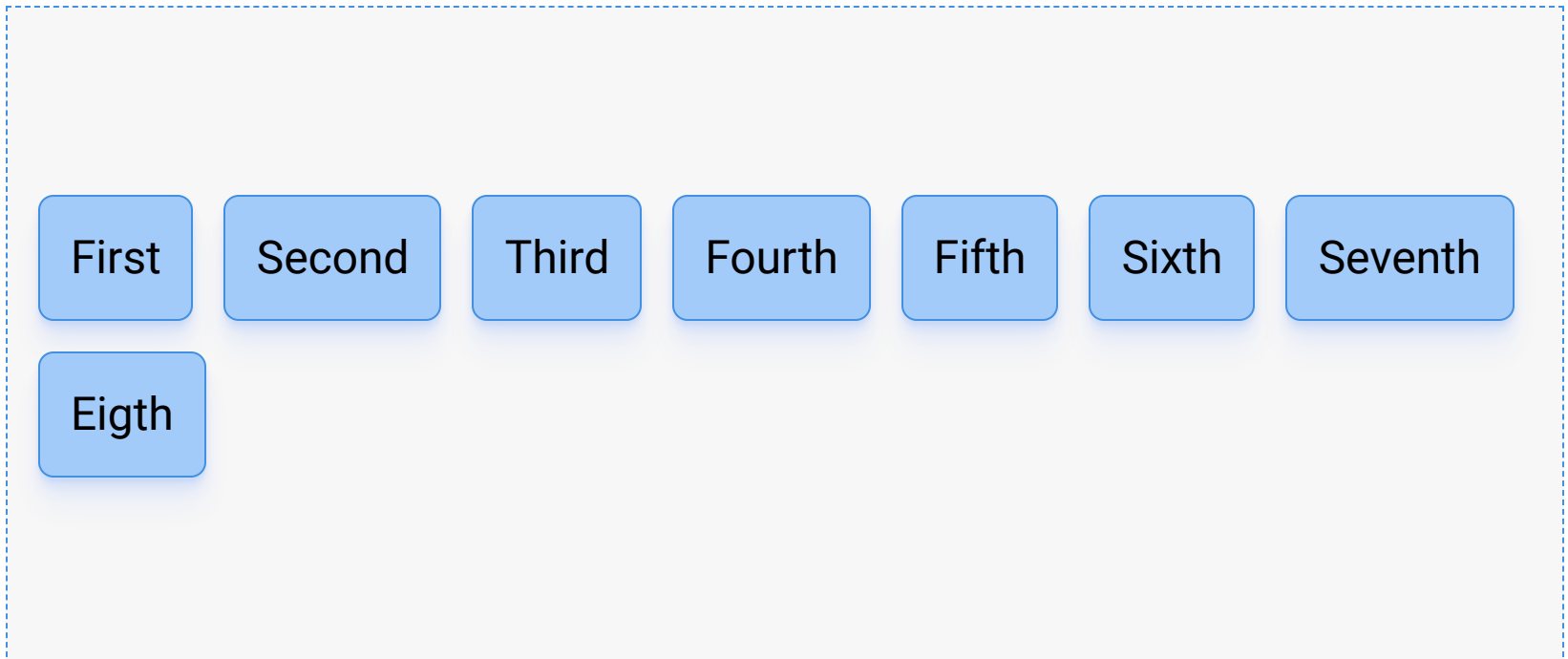
flex-wrap: wrap



align-content

```
.container {
    display: flex;
    align-items: flex-start;
    flex-wrap: wrap;
    align-content: center;
}
```

align-content



Flexbox bugs

⚠️ **Not all is well in flexbox land**

Flexbugs

CSS Grid

CSS Grid API

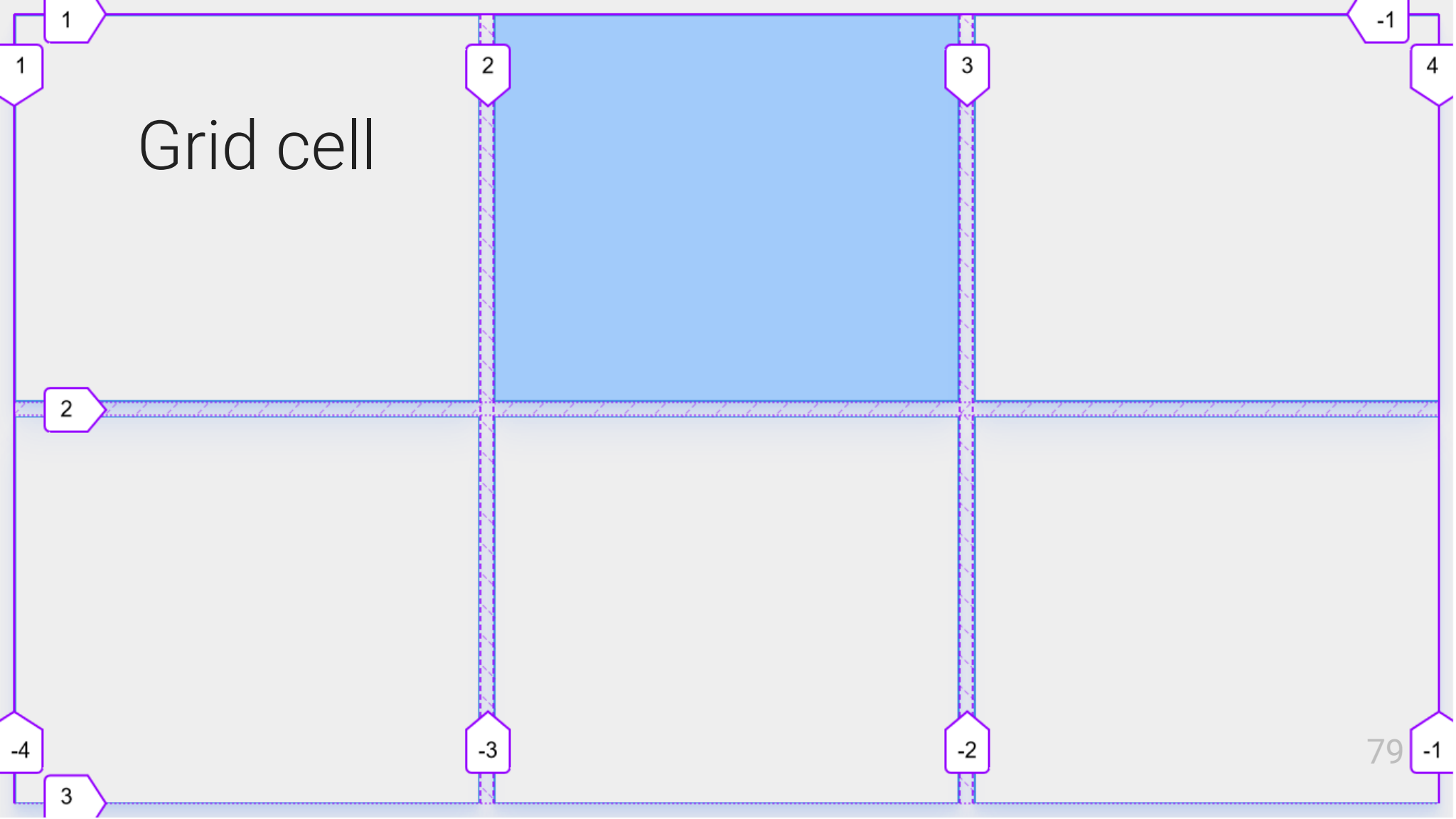
- grid-template-columns
- grid-template-rows
- grid-column
- grid-row
- grid-template-area / grid-area
- gap
- align-items
- justify-content
- align-content
- grid-auto-columns
- grid-auto-rows
- grid-auto-flow

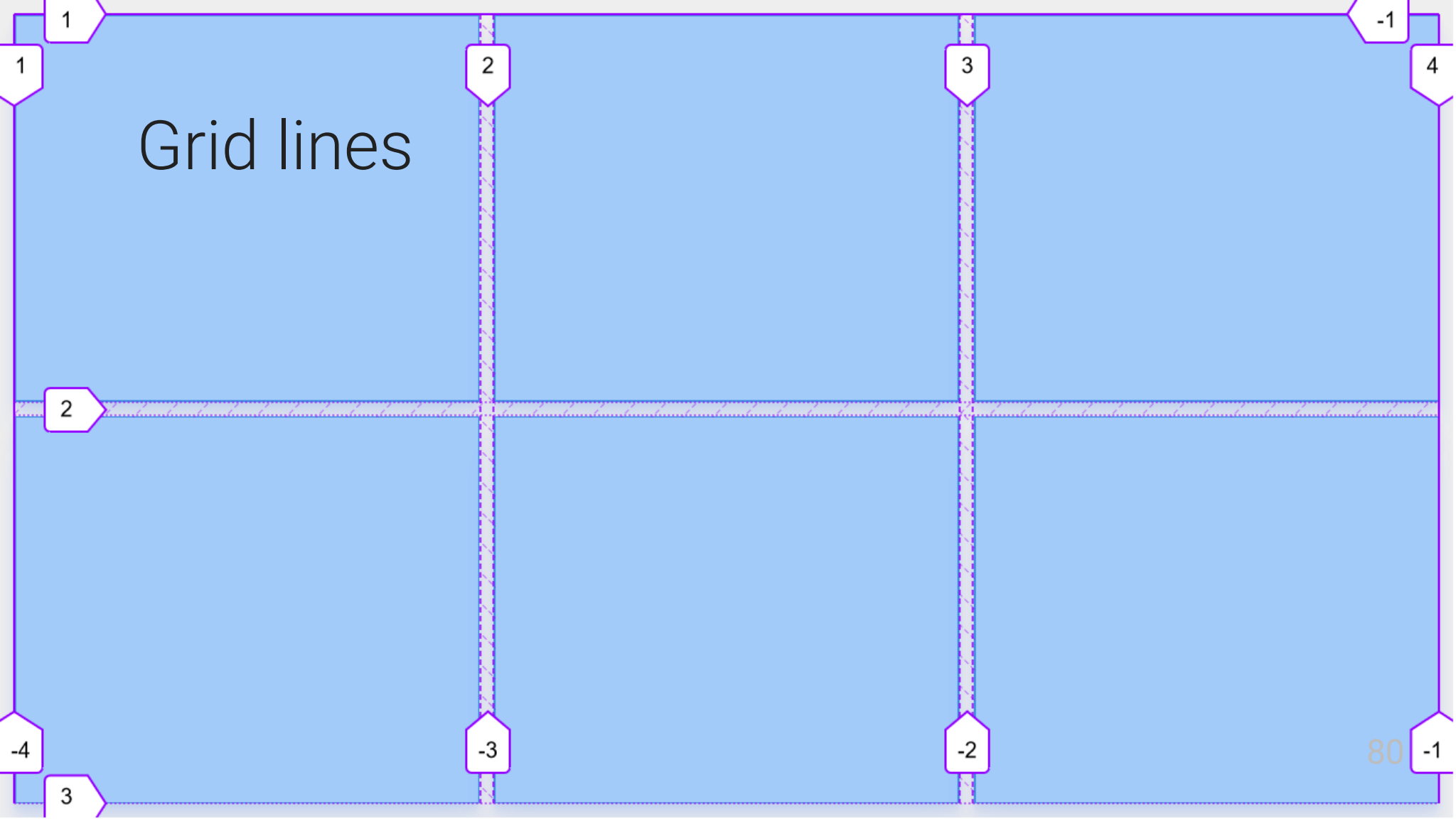
Complete guide to Grid

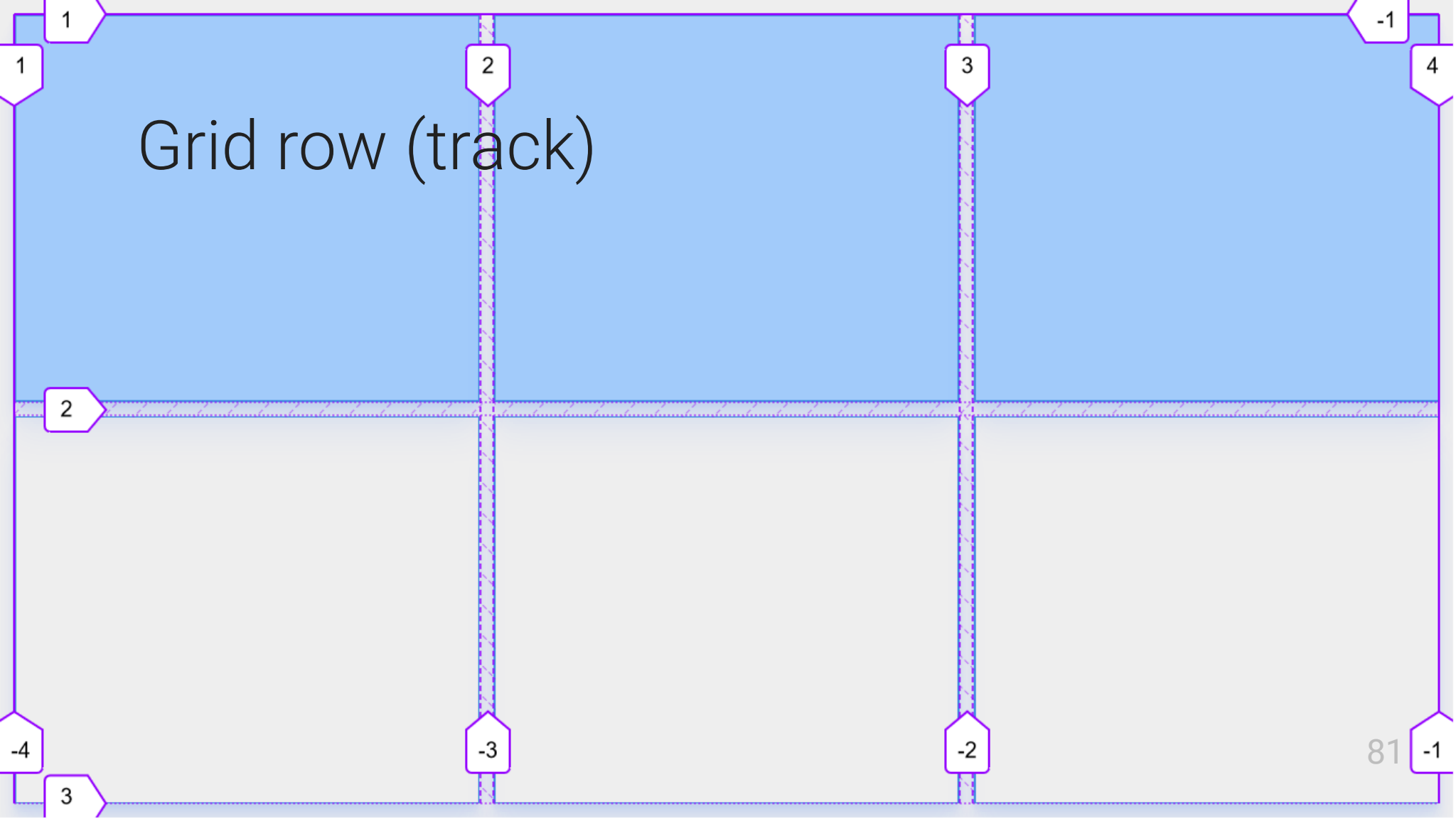
CSS Grid Terminology

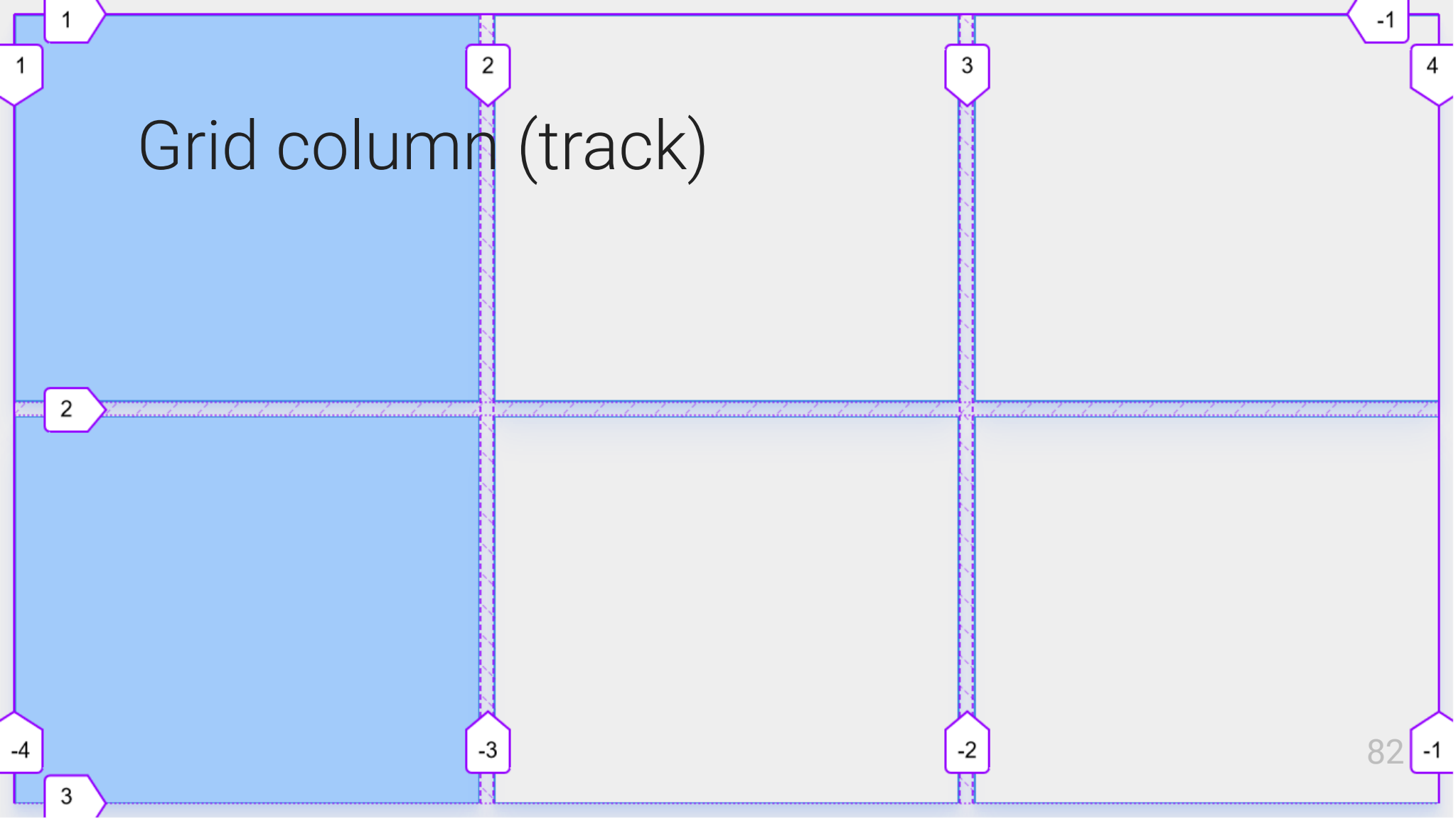
- Grid lines
- Grid tracks
- Grid areas
- Grid cell

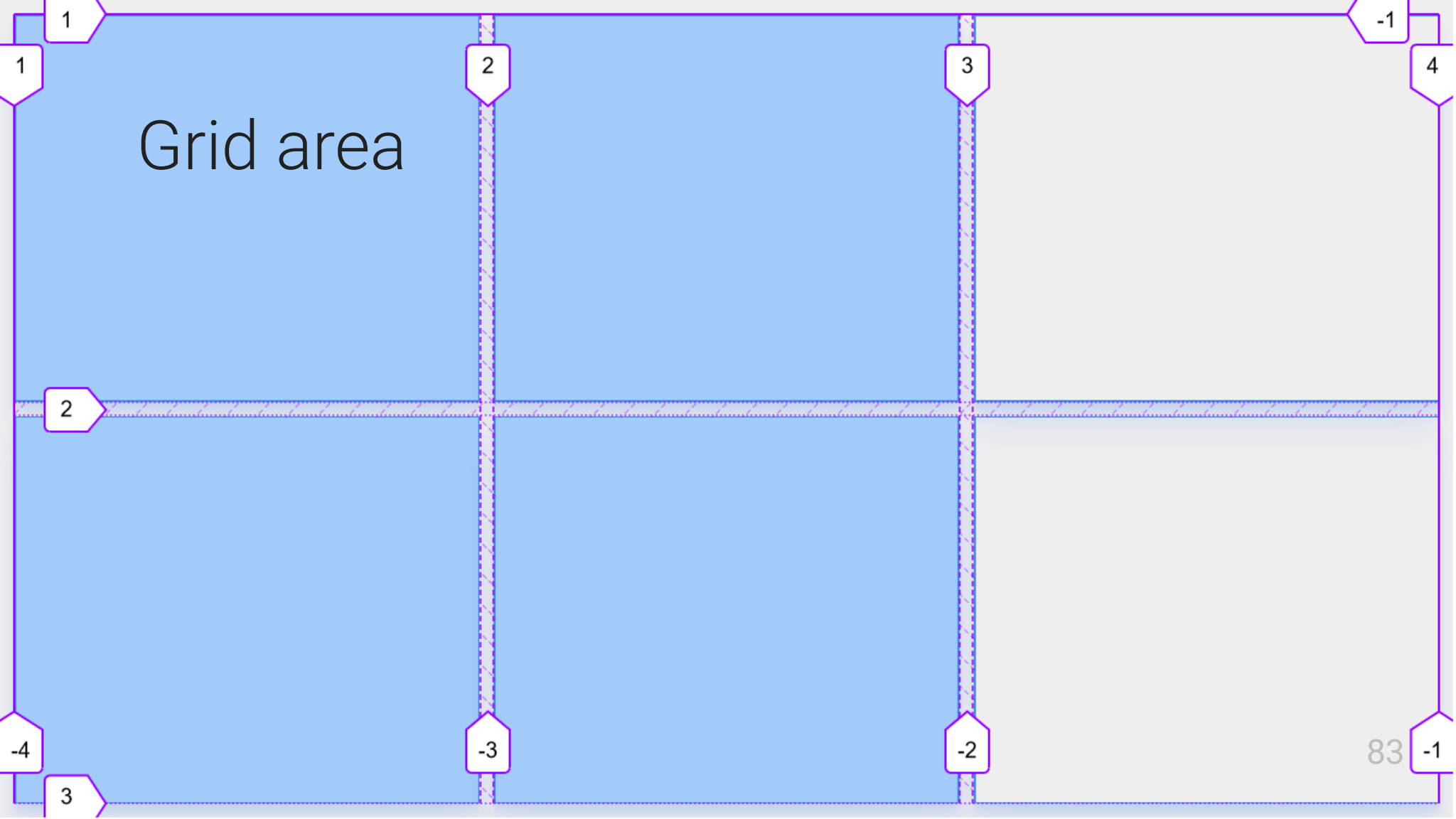
- Explicit grid
- Implicit grid


Grid cell
1
2
3
-1
1
2
4
2
-4
-3
-2
-1
3


Grid lines
1
2
3
-1
1
2
4
2
-4
-3
-2
-1
3
80


Grid row (track)
1
2
3
-1
1
2
4
2
-4
-3
-2
-1
3

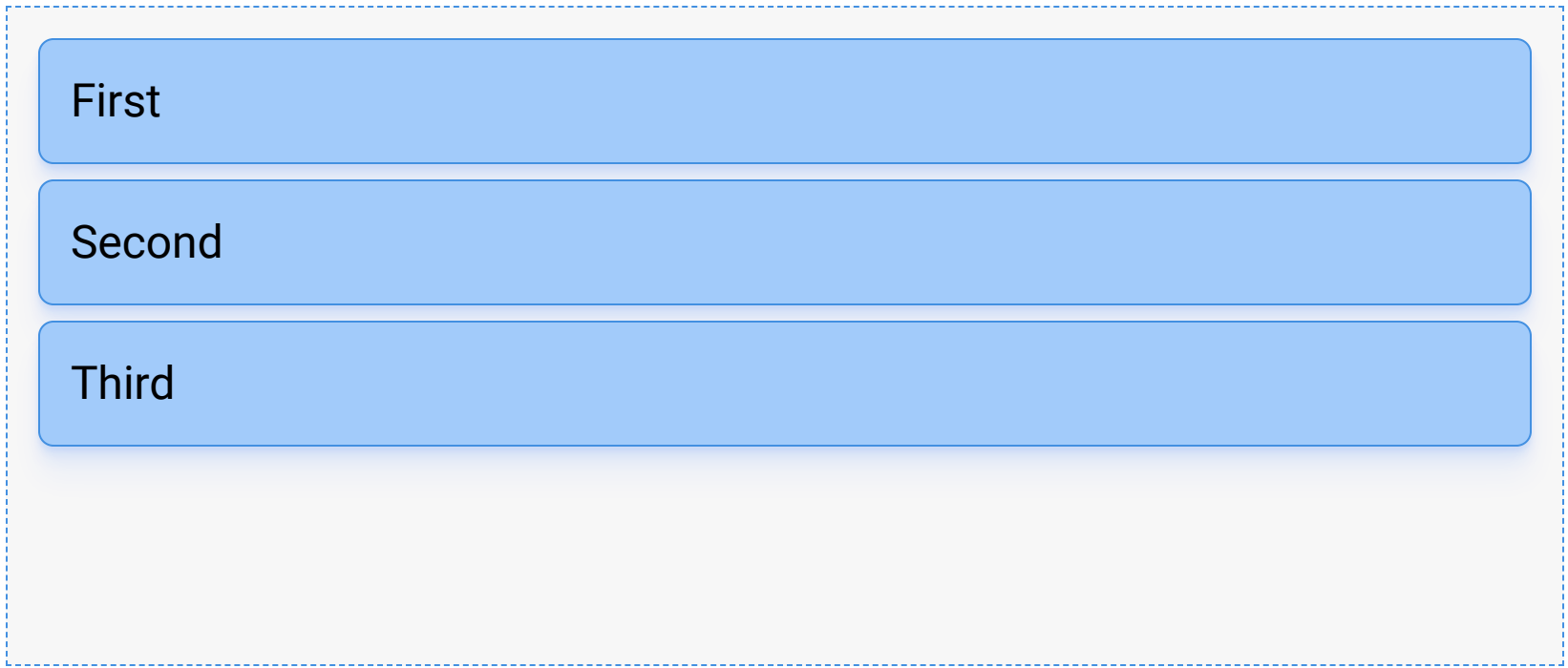

Grid column (track)
1
2
3
-1
1
2
3
4
-4
-3
-2
-1
3


1
-1
1
2
3
4
Grid area
2
-4
-3
-2
-1
3

initial state

```
01. <div class="container">
02.     <div class="item">First</div>
03.     <div class="item">Second</div>
04.     <div class="item">Third</div>
05. </div>
```

initial state



display: grid

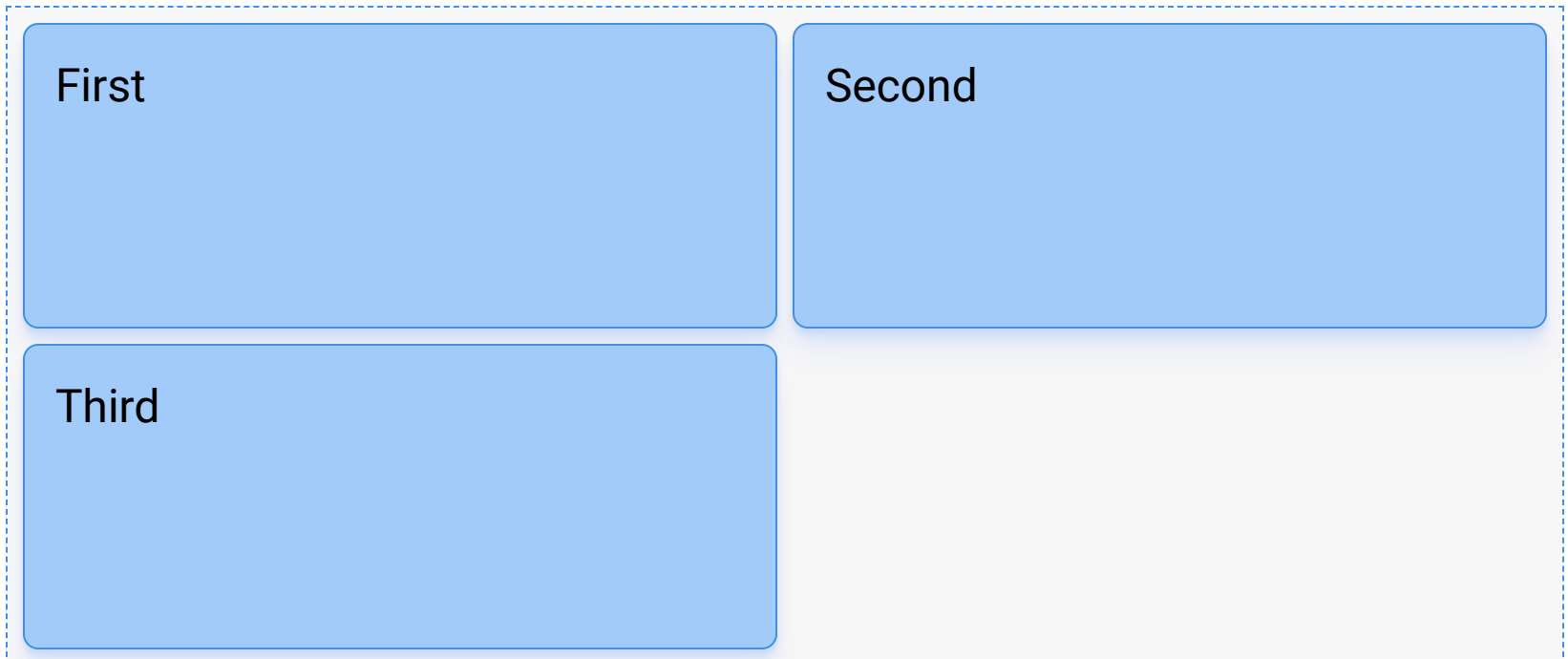
```
.container {
    display: grid;
    gap: 0.5rem; /* row-gap column-gap */
}
```

display: grid

grid-template-columns: 1fr 1fr

```
01. .container {
02.     display: grid;
03.     grid-template-columns: 1fr 1fr;
04. }
```

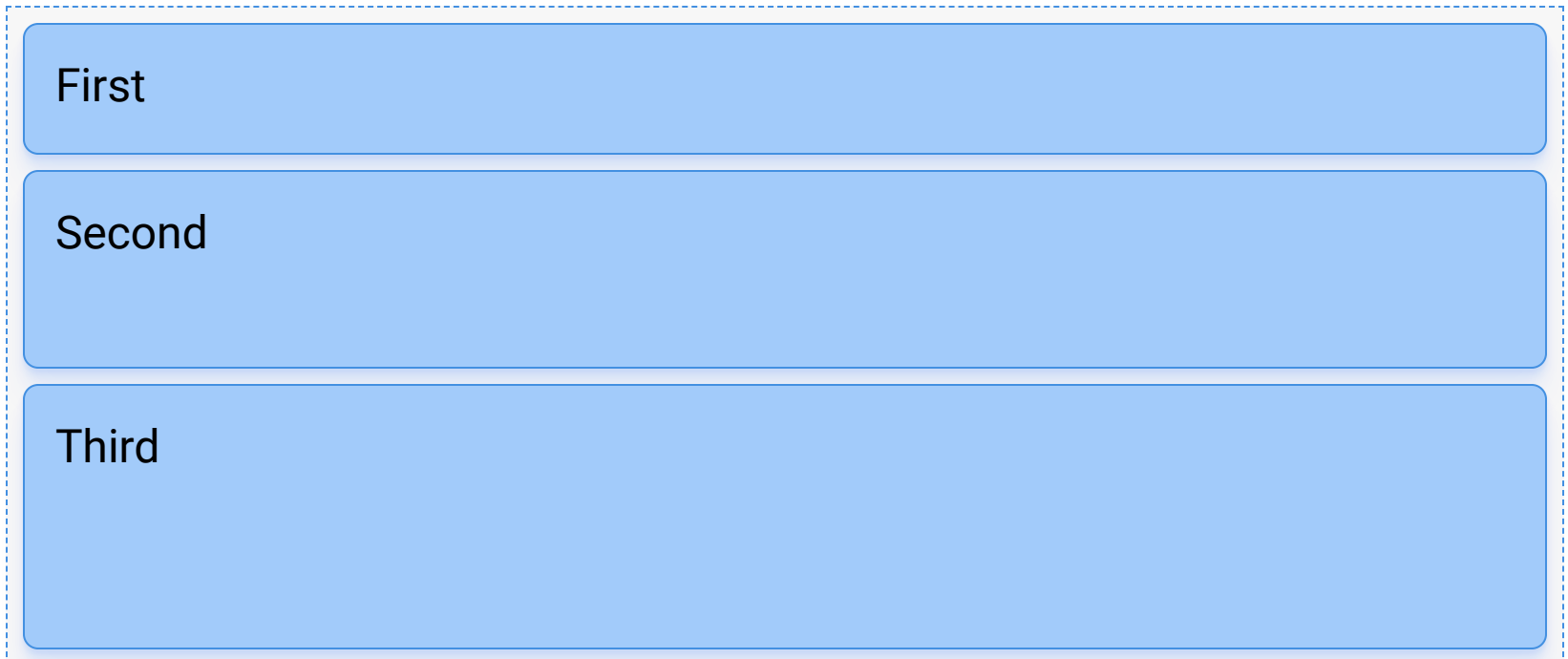
grid-template-columns: 1fr 1fr



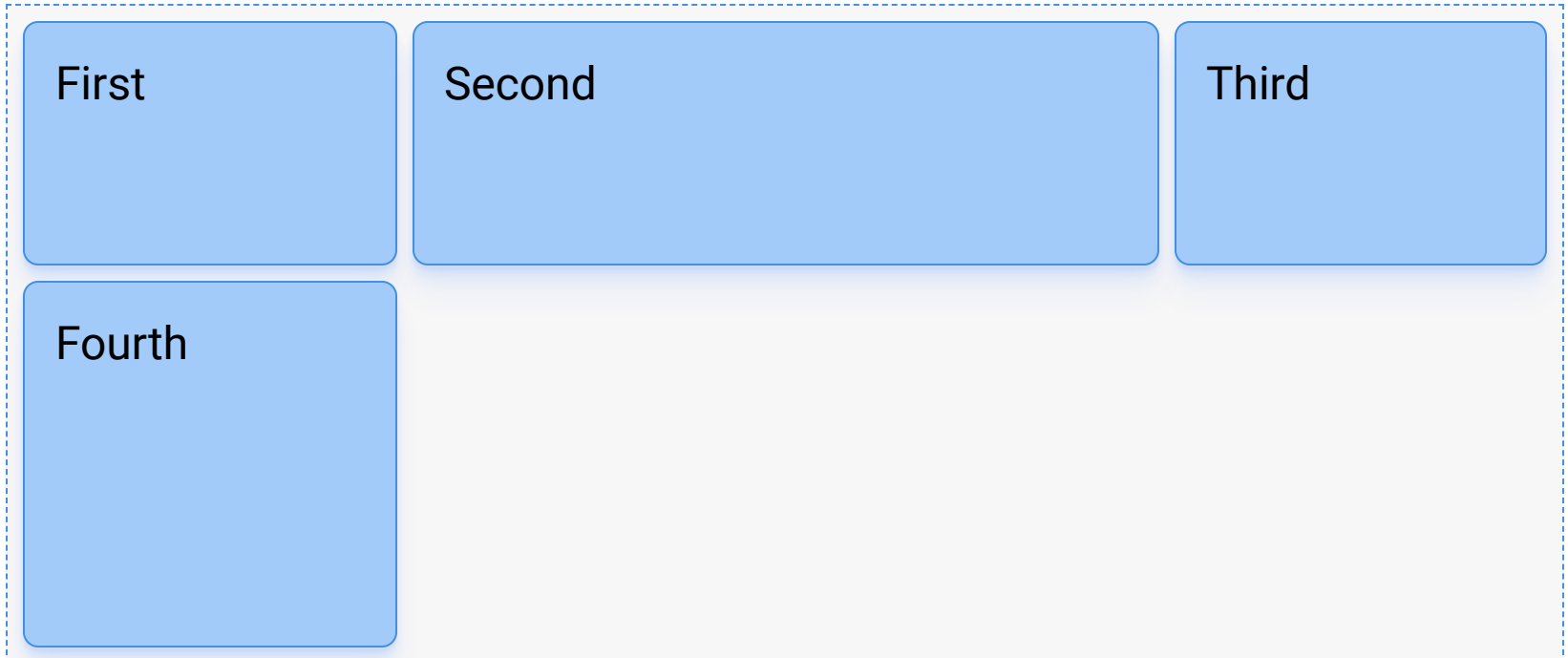
grid-template-rows: 1fr 1.5fr 2fr

```
01. .container {
02.     display: grid;
03.     grid-template-rows: 1fr 1.5fr 2fr;
04. }
```

grid-template-rows: 1fr 1.5fr 2fr



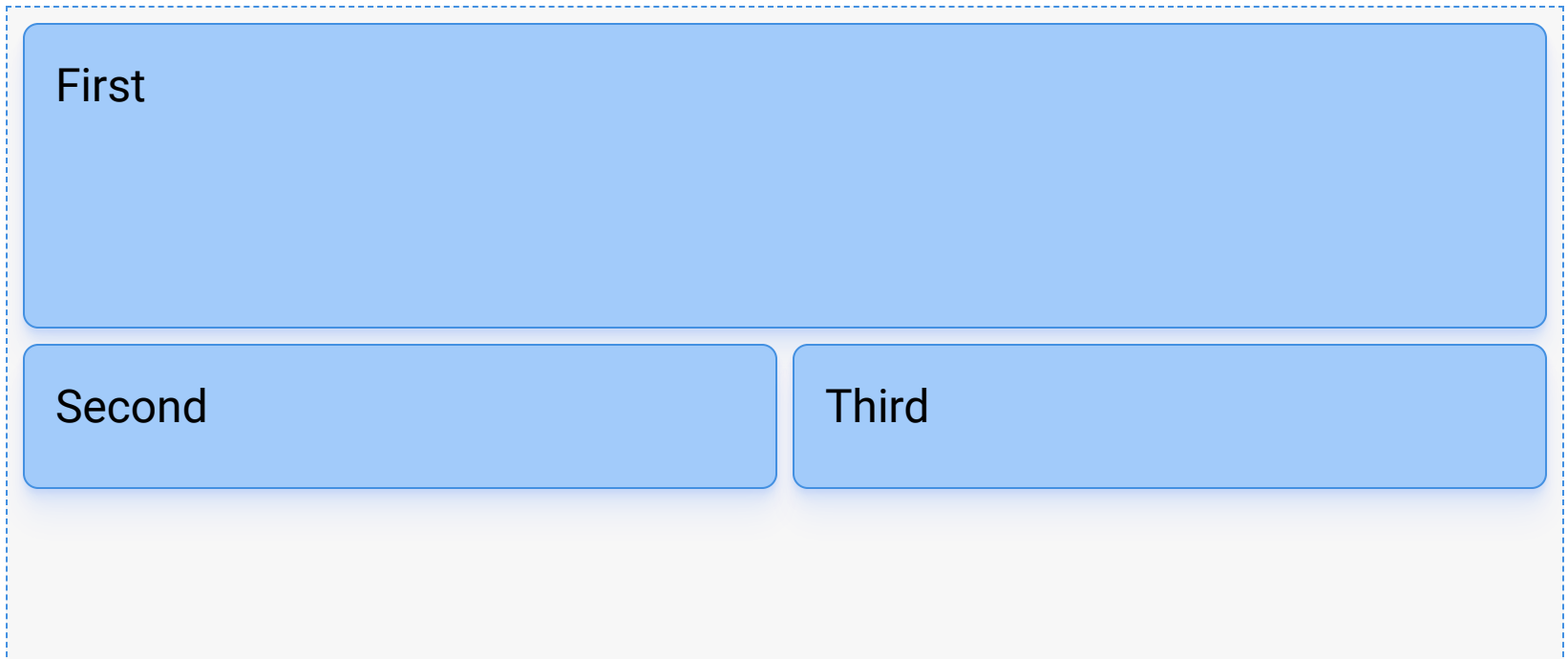
```
.container {
    display: grid;
    grid-template-rows: 1fr 1.5fr;
    grid-template-columns: 1fr 2fr 1fr;
}
```


First
Second
Third
Fourth

grid-rows / grid-columns

```
.container {
    display: grid;
    grid-template-rows: repeat(4, 1fr);
    grid-template-columns: 1fr 1fr;
}
.item:nth-child(1) {
    grid-row: 1 / 3; /* grid-row-start / grid-row-end */
    grid-column: span 2; /* grid-column-start / grid-column-end */
}
```

grid-rows / grid-columns

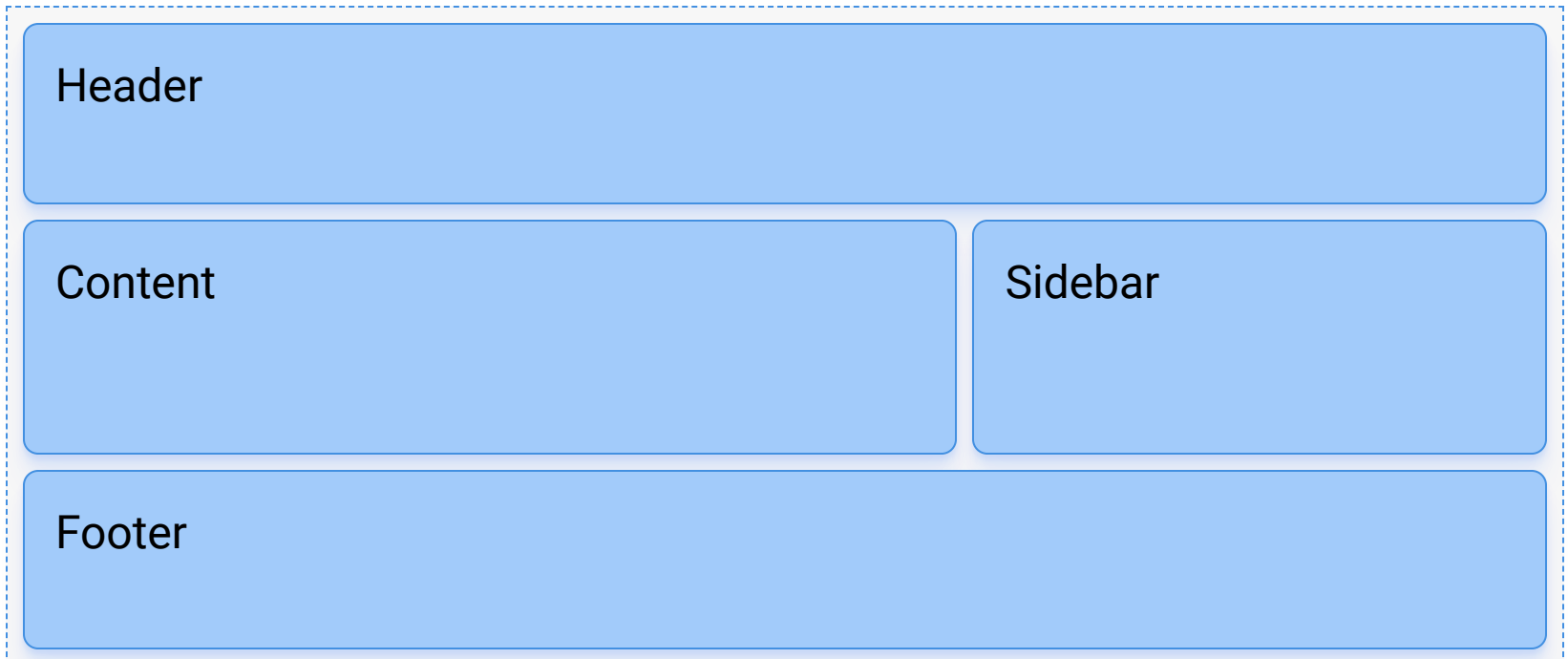


grid-template-area

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

```
.header {
    grid-area: header;
}
.sidebar {
    grid-area: sidebar;
}
.content {
    grid-area: content;
}
```

grid-template-area



Implicit grid

> *When grid items are positioned outside of these bounds, the grid container generates implicit grid tracks by adding implicit grid lines to the grid. These lines together with the explicit grid form the implicit grid. The grid-auto-rows and grid-auto-columns properties size these implicit grid tracks, as well as any explicit grid tracks created by grid-template-areas but not explicitly sized by grid-template-rows or grid-template-columns*
>
> ***CSS WG***

Implicit grid

Codesandbox 1

Codesandbox 2

The Explicit and Implicit grid explained

👨‍💻 [Practice] Grid

👉 **Implement Calendar component with CSS Grid**

Codesandbox

CSS Grid vs CSS Flexbox

Flexbox layout

CSS Grid layout

Useful links

Games

1. Flexbox Froggy
2. Grid Garden
3. Guess-css.app

Articles

1. Grid by Example by Rachel Andrew
2. Understanding CSS Grid Container
3. Masonry with CSS by Tobias Ahlin Bjerrome
4. Blog by Ahmad Shadeed
5. Grid bugs

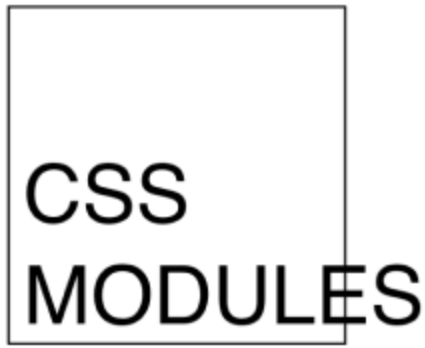
CSS Modules

```
01. body { ... }
02. body div.content div.container { ... }
03. body div.content div.container div.articles { ... }
04. body div.content div.container div.articles > div.post { ... }
05. body div.content div.container div.articles > div.post div.title { ... }
06. body div.content div.container div.articles > div.post div.title h1 { ... }
07. body div.content div.container div.articles > div.post div.content { ... }
```

🔥 Existing solutions

- 👉 Naming conventions & frameworks
  - BEM
    - Block__Element-Modifier
  - Atomic CSS
    - .bg-gray-100.py-6.flex
- 👉 W3C CSS Scoping Module (Shadow DOM)
- 👉 CSS-in-JS
- 👉 ... and many more

CSS Modules & CSS-in-JS



Emotion

CSS Modules

... are local by default 📦

```
.Block { background-color: plum; } // declare css
import styles from "./styles.module.css" // import a css module
function Block() { // apply the class name
  return <div class={styles.Block}>Component</div>
}
<div class="Block-9gbnd5">Component</div>// result
```

CSS Modules and intellisense

You can have css class autocomplete when using css modules with Intellisense.

A simple example on how to get it working on VS Code with CRA

CSS Modules

- 👍 Solution to global scoping
- 👍 "Native" CSS
- 👍 Server-side rendering

- 👎 Building step is required (and some tweaking may be needed)

Styled Components

```
01. import styled from "styled-components"
02. let Block = styled.div`
03.   background-color:  plum;
04. `
05. function Component() {
06.   return <Block>Component</Block>
07. }
08. <div class="Block-9gbnd5">Component</div>// result
```

... all might of JS 💪

```
01. import styled, { css } from "styled-components"
02. let Button = styled.button`
03.   background: ■ black;
04.   color: □ white;
05.   ${props => props.primary && css`
06.     background: □ white;
07.     color: ■ black;
08.   `}
```

Styled Components

- 👍 Solution to global scoping
- 👍 Variables, functions, all might of JS
- 👍 Easy to make Critical Path CSS
- 👍 Server-side rendering
- 👍 Compilation step is optional

- 🤔 If you skip the compilation step, the css will be generated right at runtime.

PostCSS

PostCSS is a tool for transforming CSS

- 👉 Polyfills
- 👉 Code Minification
- 👉 Linting
- 👉 Mixins
- 👉 Code isolation (scoping)

- 👉 Many CSS tools are built on top of it
- 👉 Has a huge ecosystem of plugins

A searchable catalog of PostCSS plugins

Tools built with PostCSS

... and many more!

Autoprefixer

Adds vendor prefixes to CSS rules using values from Can I Use depending on the configured supported browser version range.

CSS input

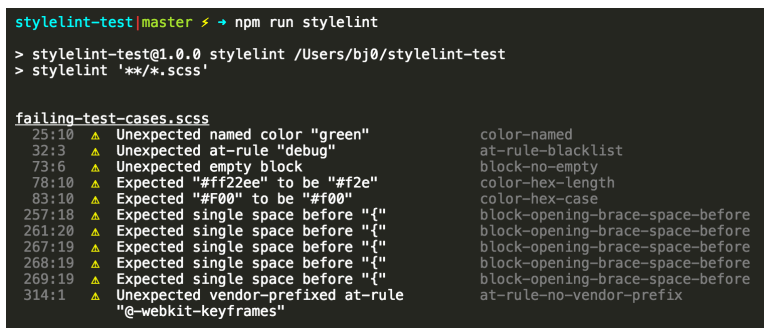
```
:fullscreen {
}
```

CSS output

```
:-webkit-full-screen {
}
:-ms-fullscreen {
}
:fullscreen {
}
```

Stylelint

A mighty, modern linter that helps you avoid errors and enforce conventions in your styles.



Taking CSS Linting to the Next Level with Stylelint

TailwindCSS

```
<figure class="bg-gray-100 rounded-xl p-8">
  <img class="w-32 h-32 rounded-full mx-auto" src="/sarah-dayan.jpg" alt="" width="384" height="512">
  <div class="pt-6 space-y-4">
    <blockquote>
      <p class="text-lg">
        “Tailwind CSS is the only framework that I've seen scale
        on large teams. It’s easy to customize, adapts to any design,
        and the build size is tiny.”
      </p>
    </blockquote>
    <figcaption>
      <div>
        Sarah Dayan
      </div>
      <div>
        Staff Engineer, Algolia
      </div>
    </figcaption>
  </div>
</figure>
```

PostCSS Preset Env

Lets you convert modern CSS into something most browsers can understand, determining the polyfills you need based on your targeted browsers or runtime environments, using cssdb

- Enables polyfilling
- Extends CSS syntax

Extra stuff

Extra stuff

- Can I use ...?
- CSS Masks
- CSS @supports

Homework


320
675
(1x)
Mobile
A
B
C
1
2
3
4
5
SPIN
$ 234623


1024
765
(1x)
Tablet
A
B
C
1
2
3
4
5
SPIN
$ 234623

Thank you!