ANGULAR
ARCHITECTS

Modern CSS

Alexander Thalhammer | @LX_T

Modern CSS

- Flexbox
- Grid
- Flexbox vs Grid
- Transforms
- Transitions & Animations
- Custom Properties
- Frameworks

Agenda

ANGULAR
ARCHITECTS

Flexbox

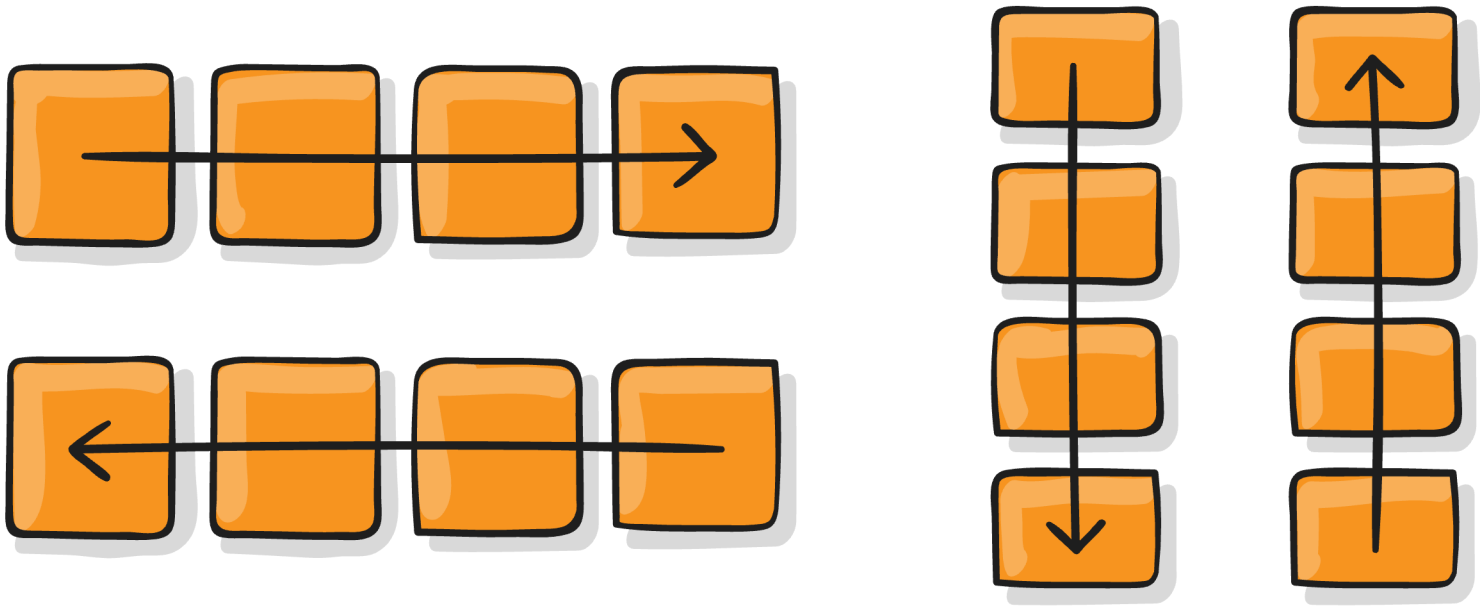
- display: flex; → flex container
  - flex-flow:
  - justify-content:
  - align-items:
  - align-content:
  - gap:
- child elements → flex items
  - order:
  - flex:
  - align-self:

https://css-tricks.com/snippets/css/a-guide-to-flexbox/


flex container

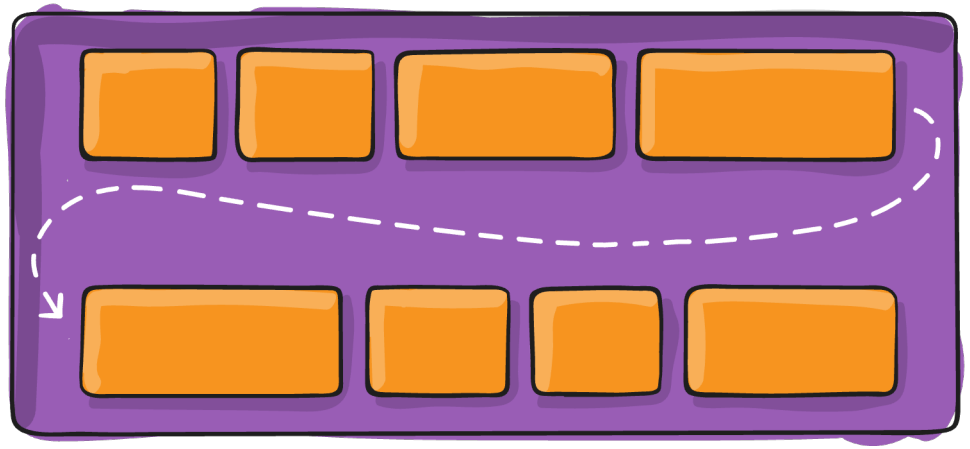
flex-direction:

– **row** | row-reverse | column | column-reverse; // main-axis



flex-wrap:

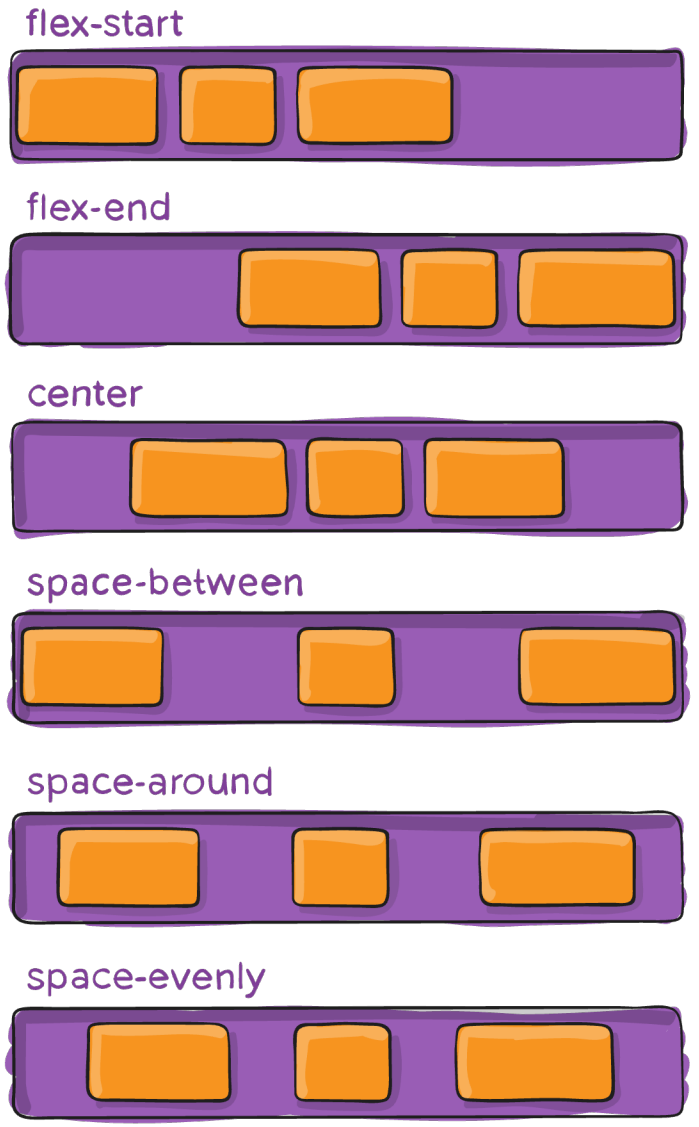
– **nowrap** | wrap | wrap-reverse;



– flex-flow: shorthand for direction & wrap ☺

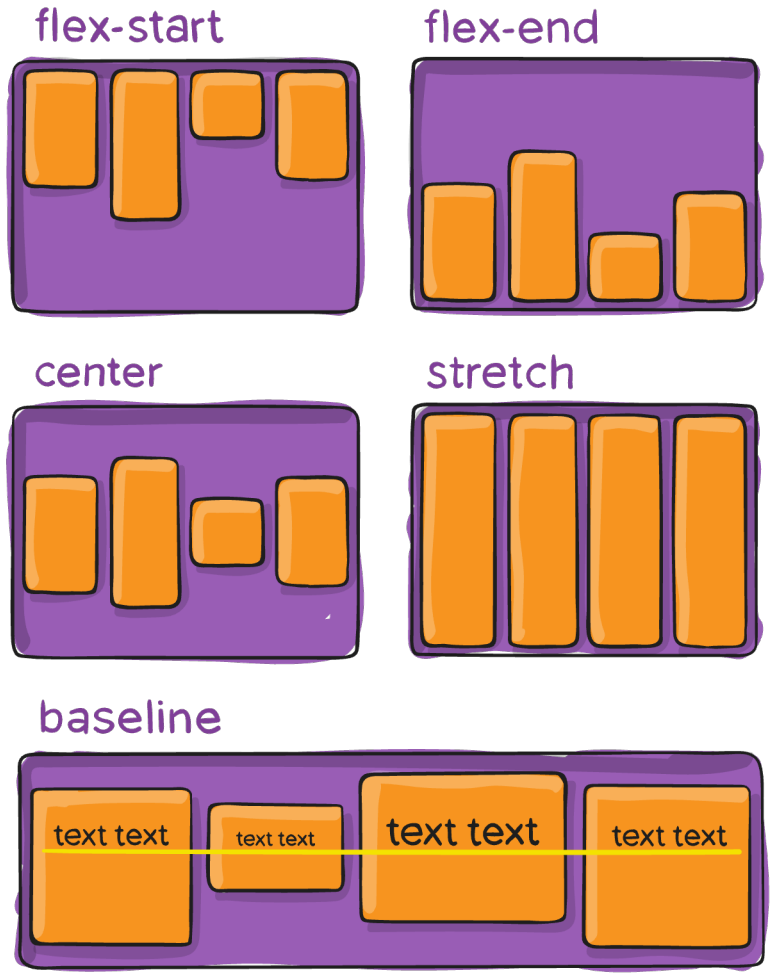
justify-content:

- **flex-start**
- flex-end
- center
- space-between
- space-around
- space-evenly



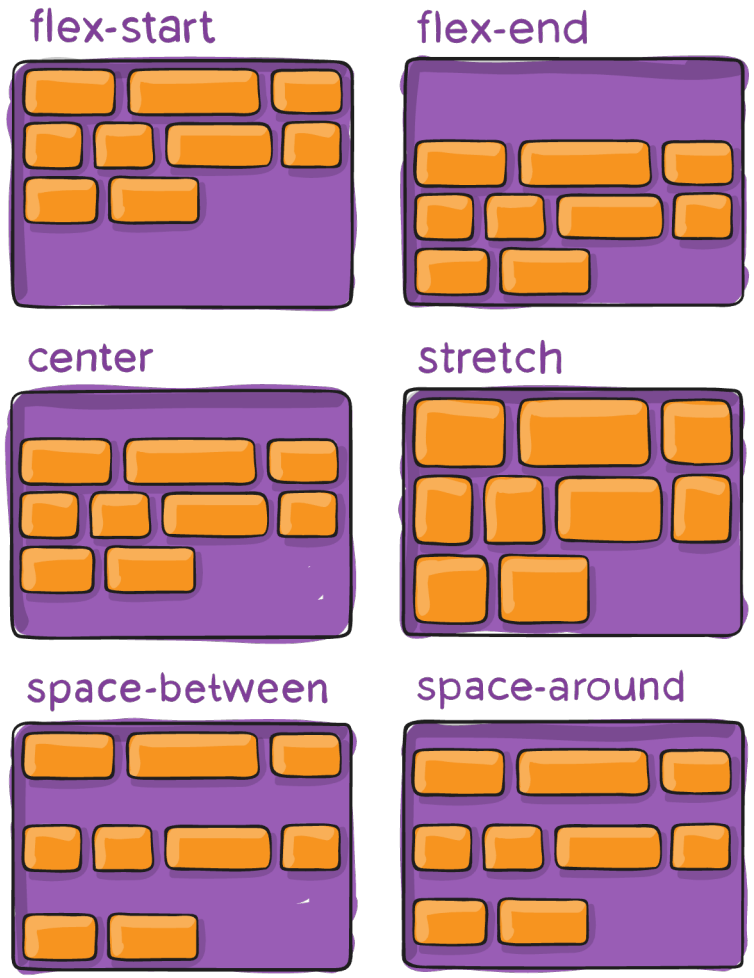
align-items:

- flex-start
- flex-end
- center
- **stretch**
- baseline



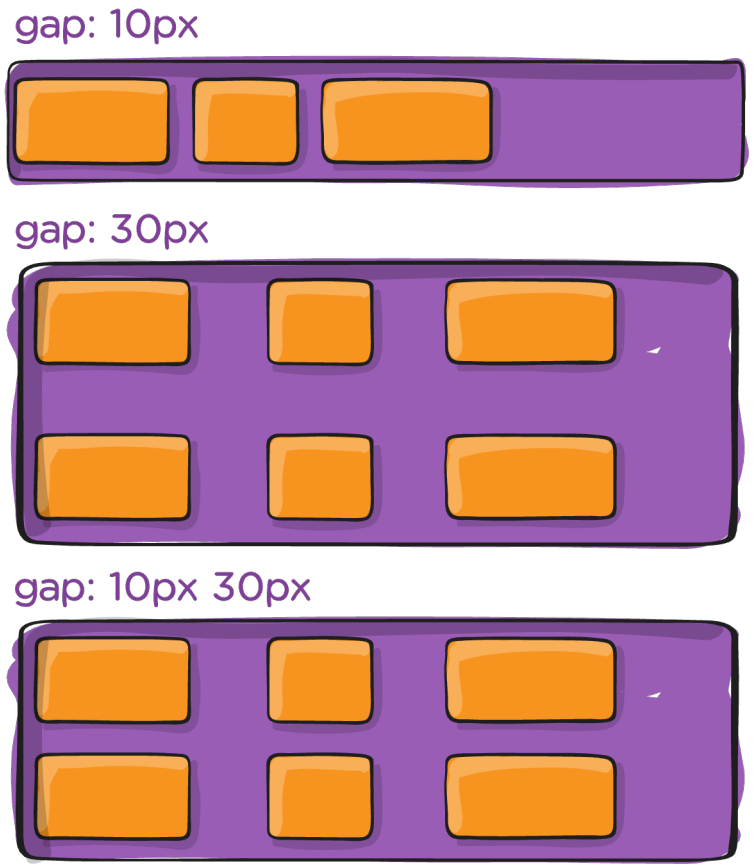
align-content (only if wrap):

- **normal**
- flex-start
- flex-end
- center
- stretch
- space-between
- space-around



gap, row-gap, column-gap:

```
.flex-container {
    display: flex;
    ...
    gap: 10px;
    gap: 10px 20px;
    row-gap: 10px;
    column-gap: 20px;
}
```

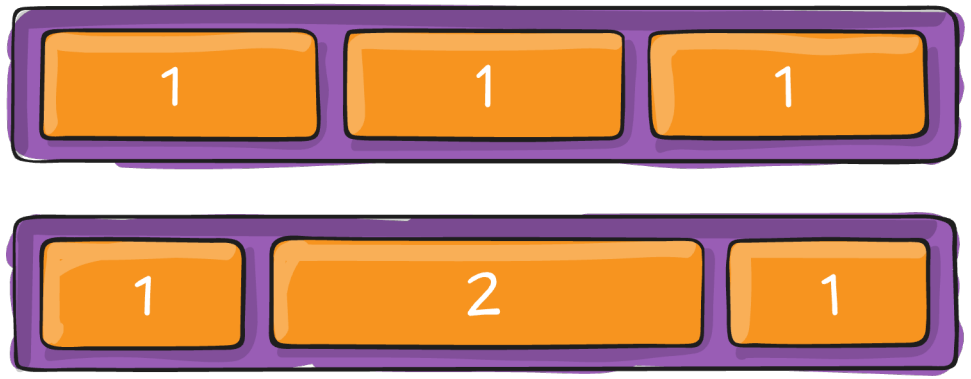


ANGULAR
ARCHITECTS

flex: (distribution of space, if nowrap)

– flex-grow: 0 | **1** | 2;

– flex-shrink: **0** | 1 | 2;

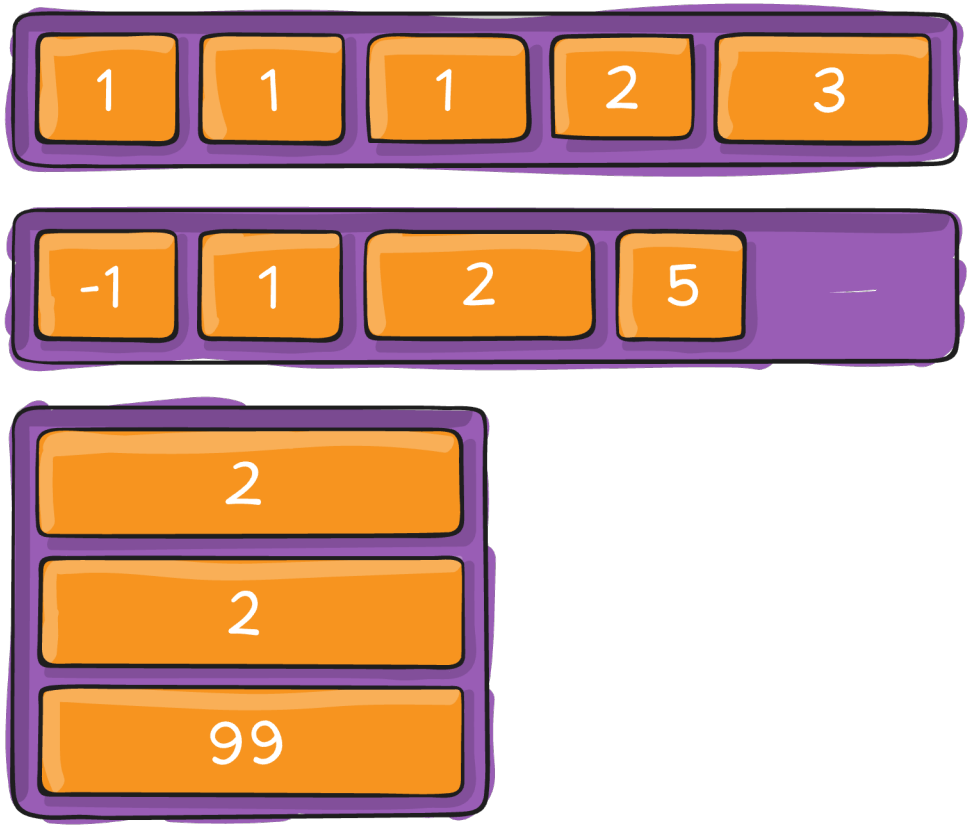


– flex-basis: **auto**; // can be used to override main-axis

– flex: shorthand, default is "1 0 auto"

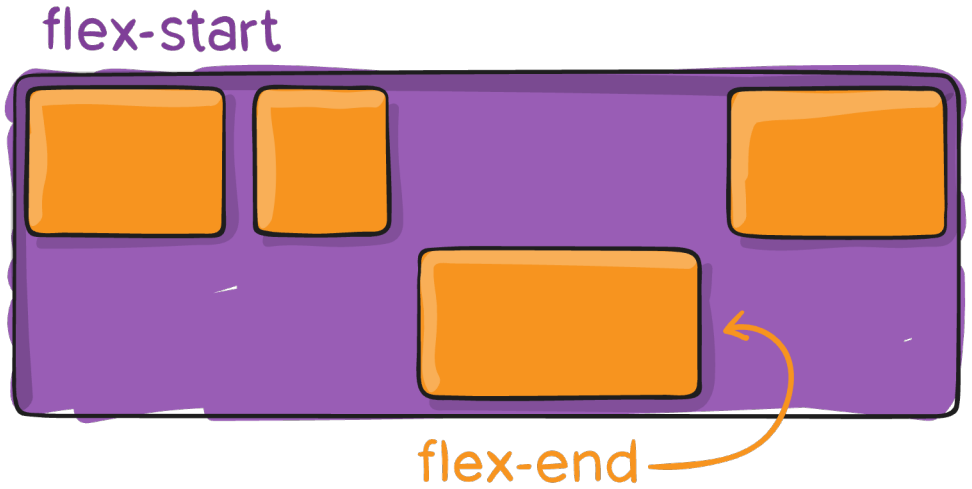
order:

– manually set order of items



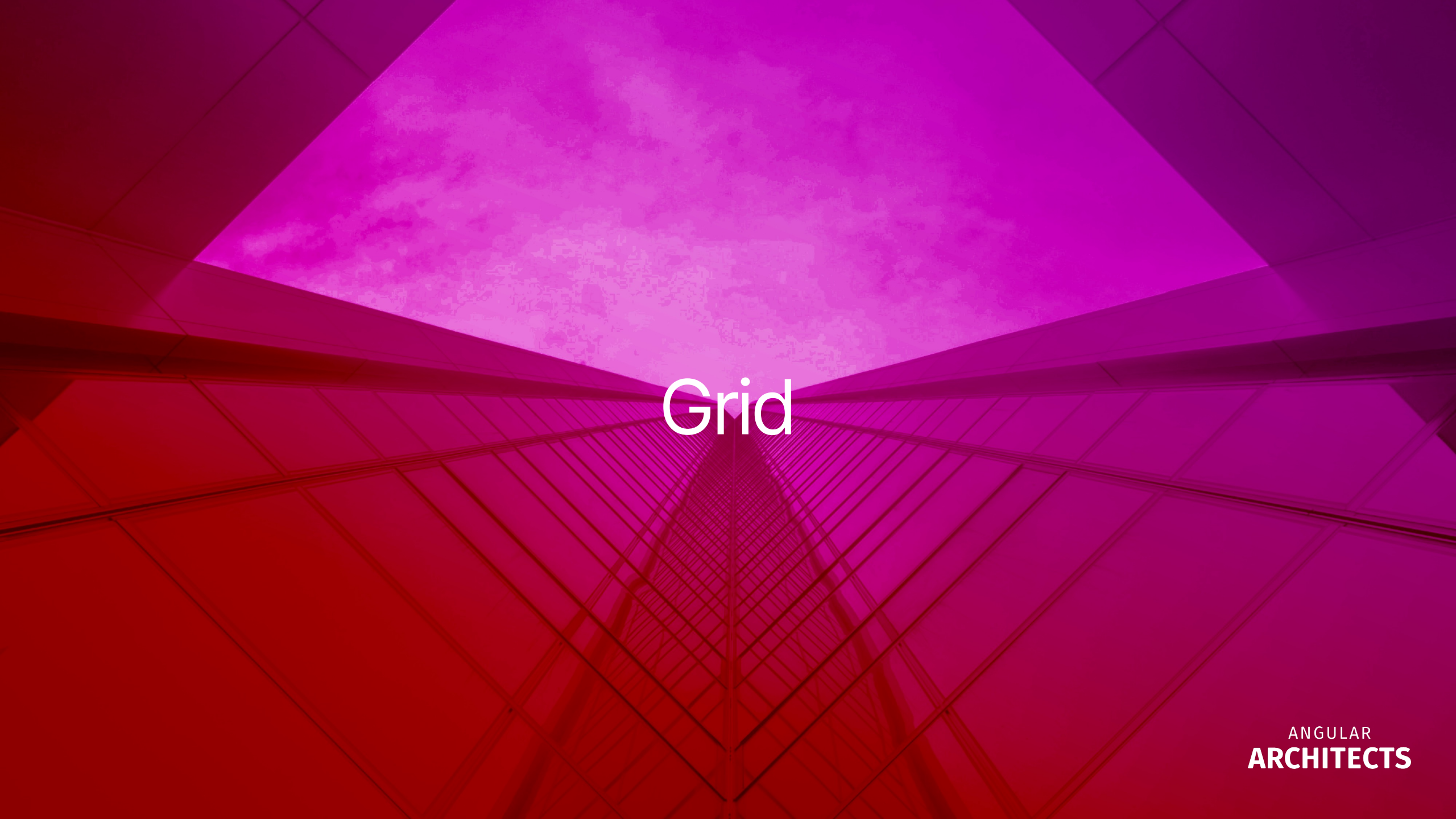
align-self: (related to align-items)

- **auto**
- flex-start
- flex-end
- center
- baseline
- stretch



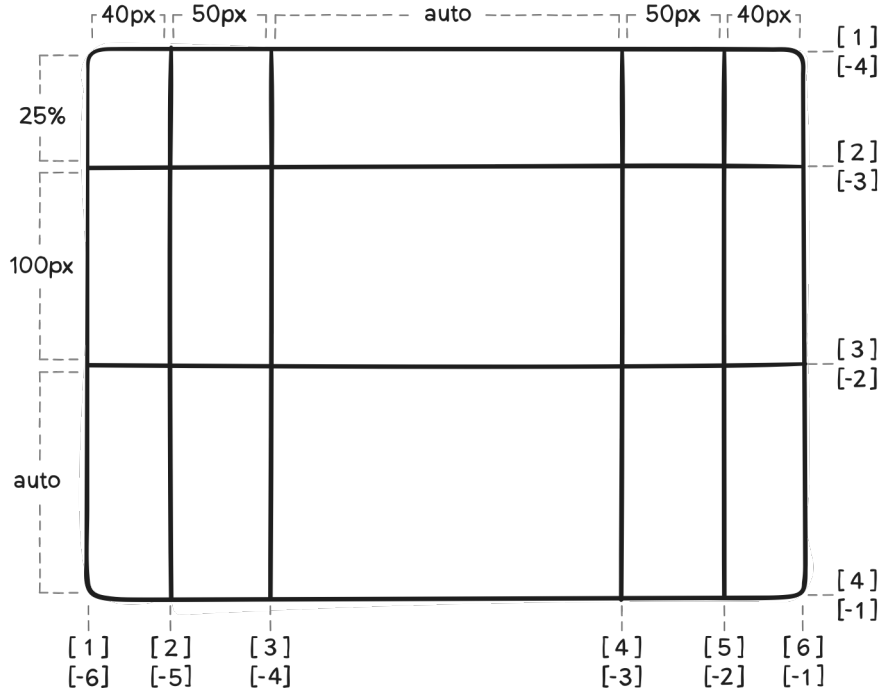

DEMO
https://www.webmart.de/web/css-tools-flexbox

ANGULAR
ARCHITECTS



ANGULAR
ARCHITECTS

Grid

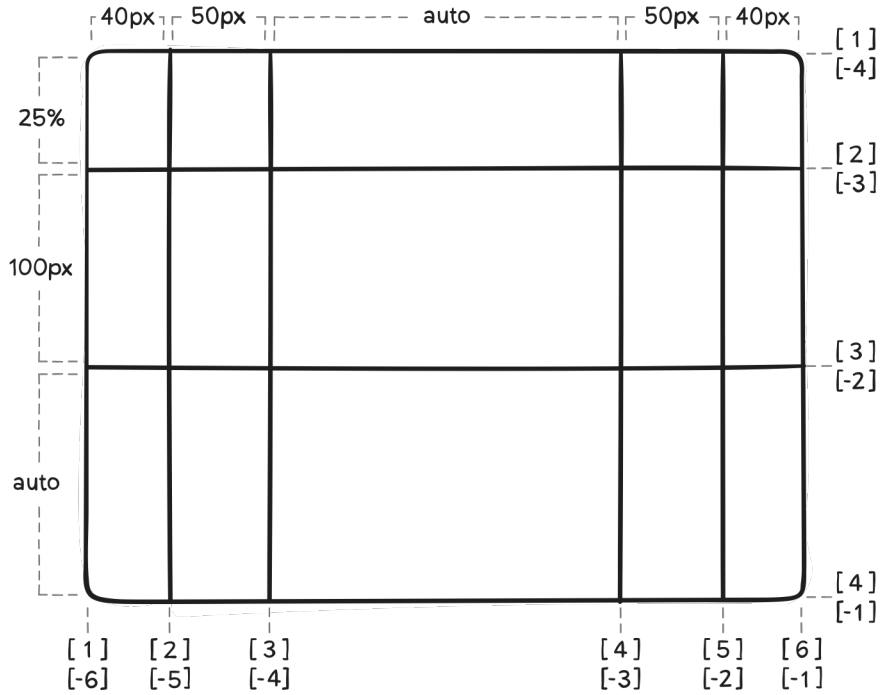


- display: grid; → grid cnt
  - grid-template-columns
  - grid-template-rows
  - grid-template-areas
- child elements → grid items
  - grid-column-start: <number> | <name> | span <number|name> | auto;
  - grid-column-end: <number> | <name> | span <number|name> | auto;
  - grid-row-start: <number> | <name> | span <number|name> | auto;
  - grid-row-end: <number> | <name> | span <number|name> | auto;
  - grid-column: <start-line> / <end-line> | <start-line> / span <value>;
  - grid-row: <start-line> / <end-line> | <start-line> / span <value>;

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Grid Template Columns & Rows

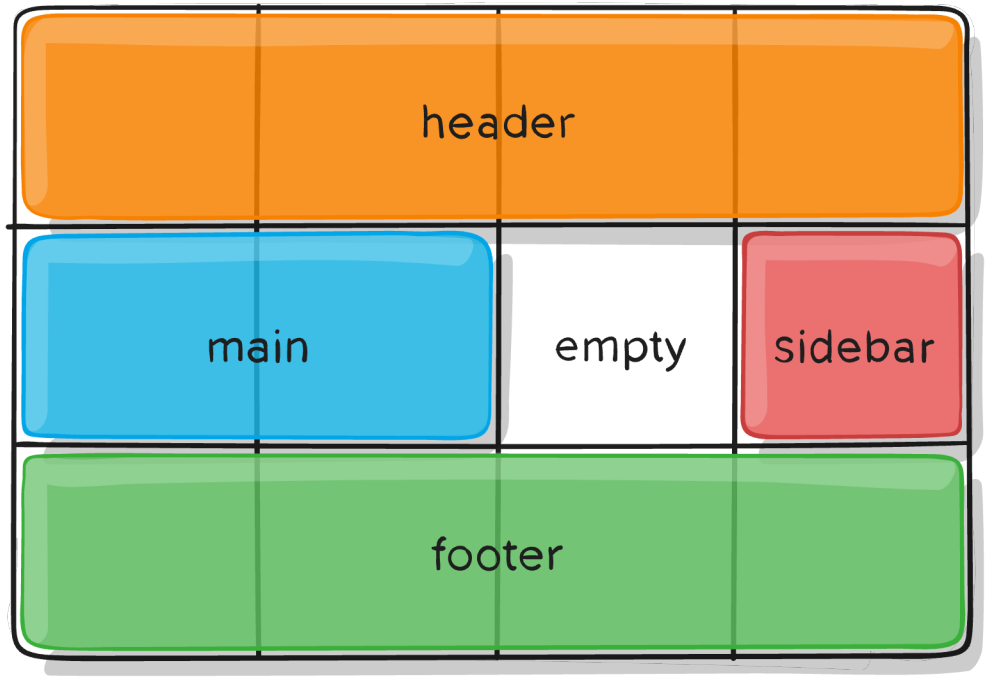
```
.grid-container {
    grid-template-columns: 40px 50px auto 50px 40px;
    grid-template-rows: 25% 100px auto;
}
```



Grid Template Areas

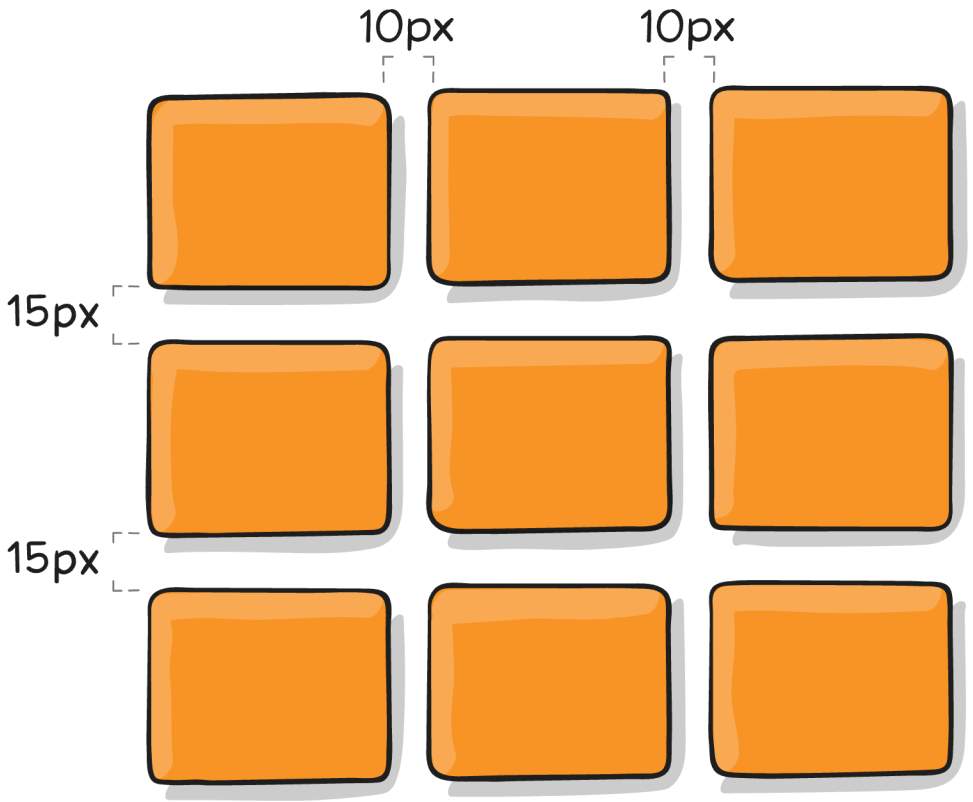
```
.grid-container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    grid-template-rows: 1fr 1fr 1fr;
    grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}

.item-a { grid-area: header }
.item-b { grid-area: main }
.item-c { grid-area: sidebar }
.item-d { grid-area: footer }
```



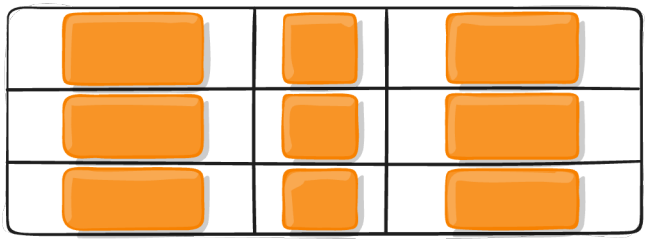
Grid Gap

```
.grid-container {
    grid-template-columns: 100px 50px 100px;
    grid-template-rows: 80px auto 80px;
    column-gap: 10px;
    row-gap: 15px;
}
```

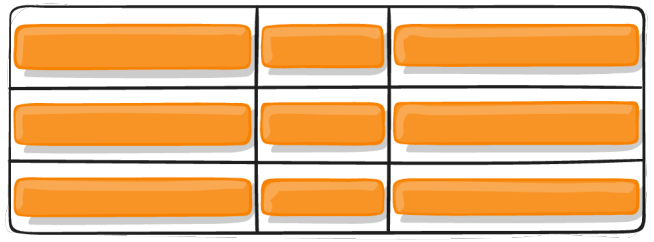


Grid justify / align

- justify-items: start | end | center | **stretch**;
  - justify grid items along the inline (row) axis



- align-items: start | end | center | **stretch**;
  - aligns grid items along the block (column) axis

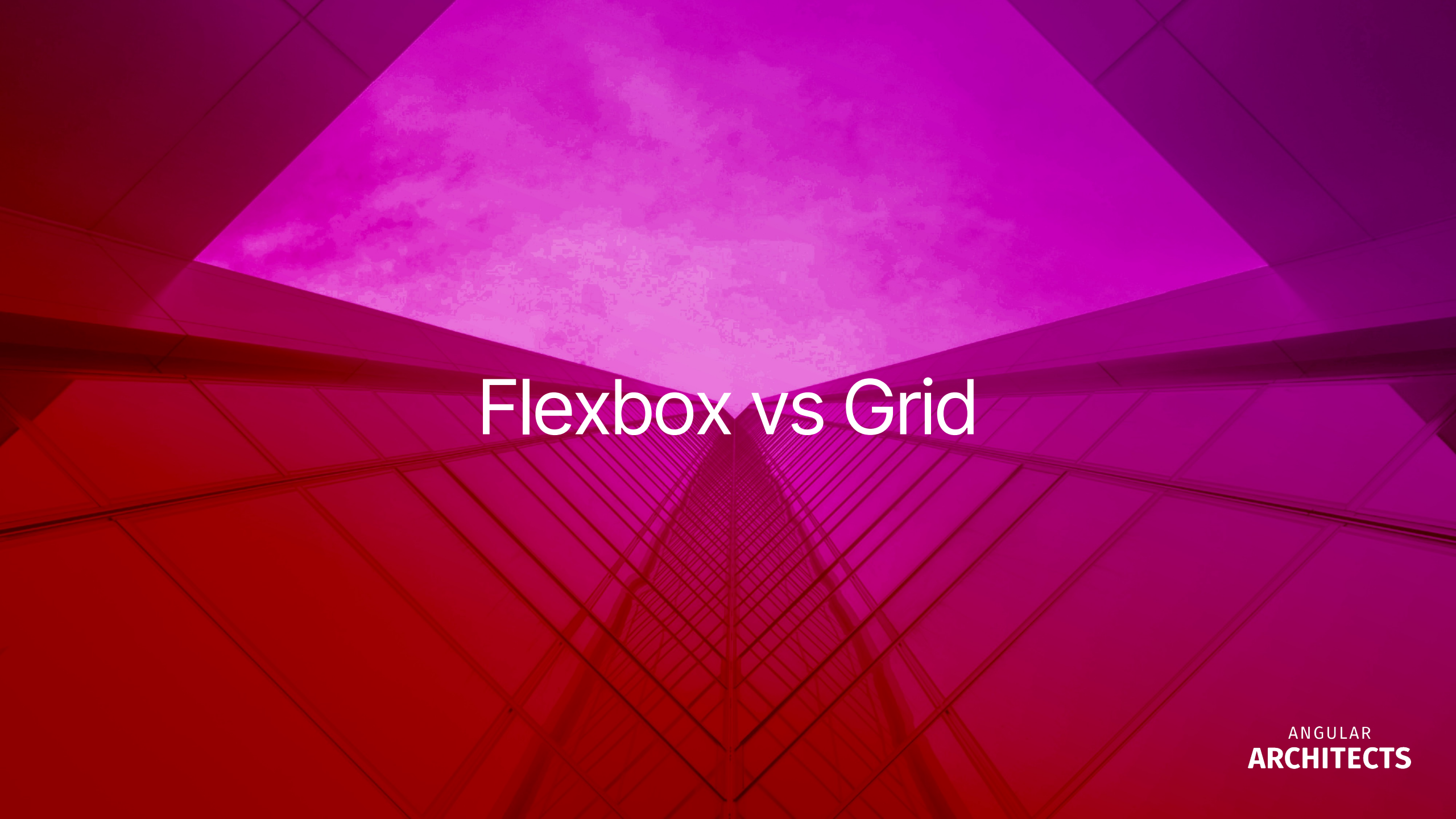


- place-items: shorthand for both

Grid ...

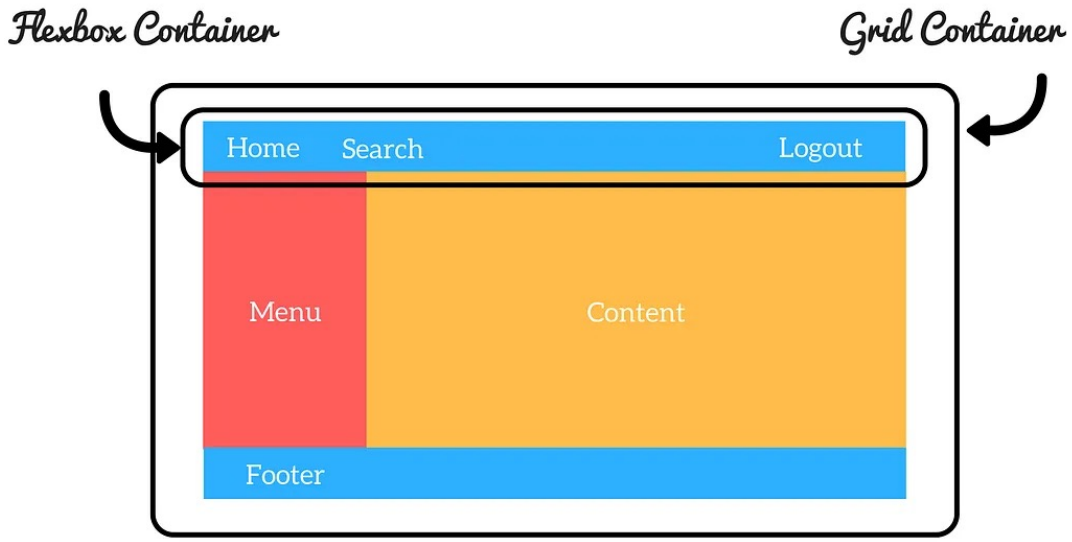
– ... there are many more settings, but we'll just stop here ☺

ANGULAR
ARCHITECTS


Flexbox vs Grid
ANGULAR
ARCHITECTS

Flexbox vs Grid

- Flexbox
  - 1 dimensional (row | column)
- Grid
  - 2 dimensional (rows & columns)



https://medium.com/youstart-labs/beginners-guide-to-choose-between-css-grid-and-flexbox-783005dd2412

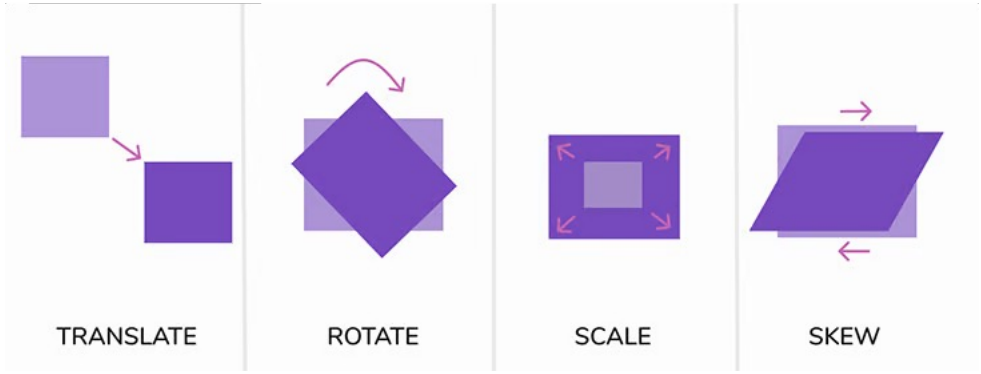

Transforms

Transform

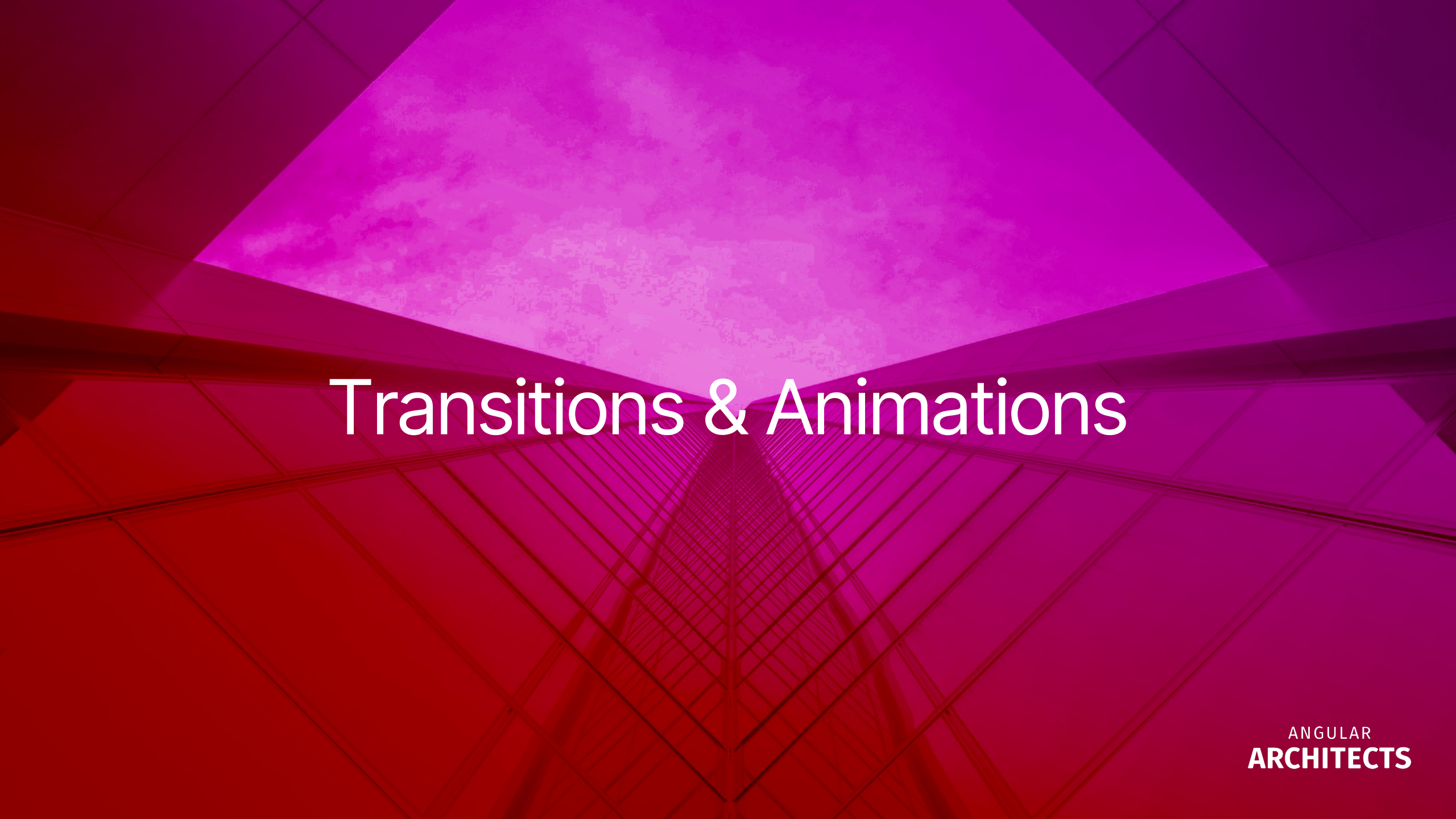
– transform: translateX(1rem) translateY(1rem);
– transform: translate(1rem, 1rem);

– transform: rotate(45deg);

– transform: scale(2);



https://blog.logrocket.com/css-before-after-custom-animations-transitions/


Transitions & Animations
ANGULAR
ARCHITECTS

Transitions

– transition: width 2s linear 1s;

```
.transition-container {
    transition-property: width;
    transition-duration: 2s;
    transition-timing-function: ease-in-out;
    transition-delay: 1s;
}
```

– typically used for :hover, :focus, errors and so on

https://developer.mozilla.org/en-US/docs/Web/CSS/transition

Animations

– define 2-n keyframes

```
div {
    width: 100px; height: 100px; background-color: red;
    animation-name: example; animation-duration: 4s;
}

@keyframes example {
    0%   {background-color:red; left:0; top:0;}
    25%  {background-color:yellow; left:200px; top:0;}
    50%  {background-color:blue; left:200px; top:200px;}
    75%  {background-color:green; left:0; top:200px;}
    100% {background-color:red; left:0; top:0;}
}
```

Note: When an animation is finished, it goes back to its original style.

https://developer.mozilla.org/en-US/docs/Web/CSS/animation

ANGULAR
ARCHITECTS

Custom Properties (Variables)

- :root { --my-color: value; }
- element { color: var(--my-color); }
- reuse same value and change it easily in 1 place
- also very good for switching / theming


CSS Frameworks / Design Kits
ANGULAR
ARCHITECTS

Vanilla vs Component vs Utility Framework

- Vanilla CSS
  - full control
  - good performance
  - no naming conflicts
  - but a lot of work
- Component frameworks
  - rapid development
  - best practices
  - easy
  - but little control & big overhead
- Utility frameworks (Tailwind)
  - faster development
  - best practices
  - also easier
  - medium control
  - but still some overhead

Best Practices

- avoid float, learn & use Flexbox & Grid
- enhance UX with transitions for
  - :hover
  - :focus
- use CSS custom properties

Tips

Modern CSS

- Flexbox
- Grid
- Flexbox vs Grid
- Transforms
- Transitions & Animations
- Custom Properties
- CSS Frameworks

Summary

ANGULAR
ARCHITECTS