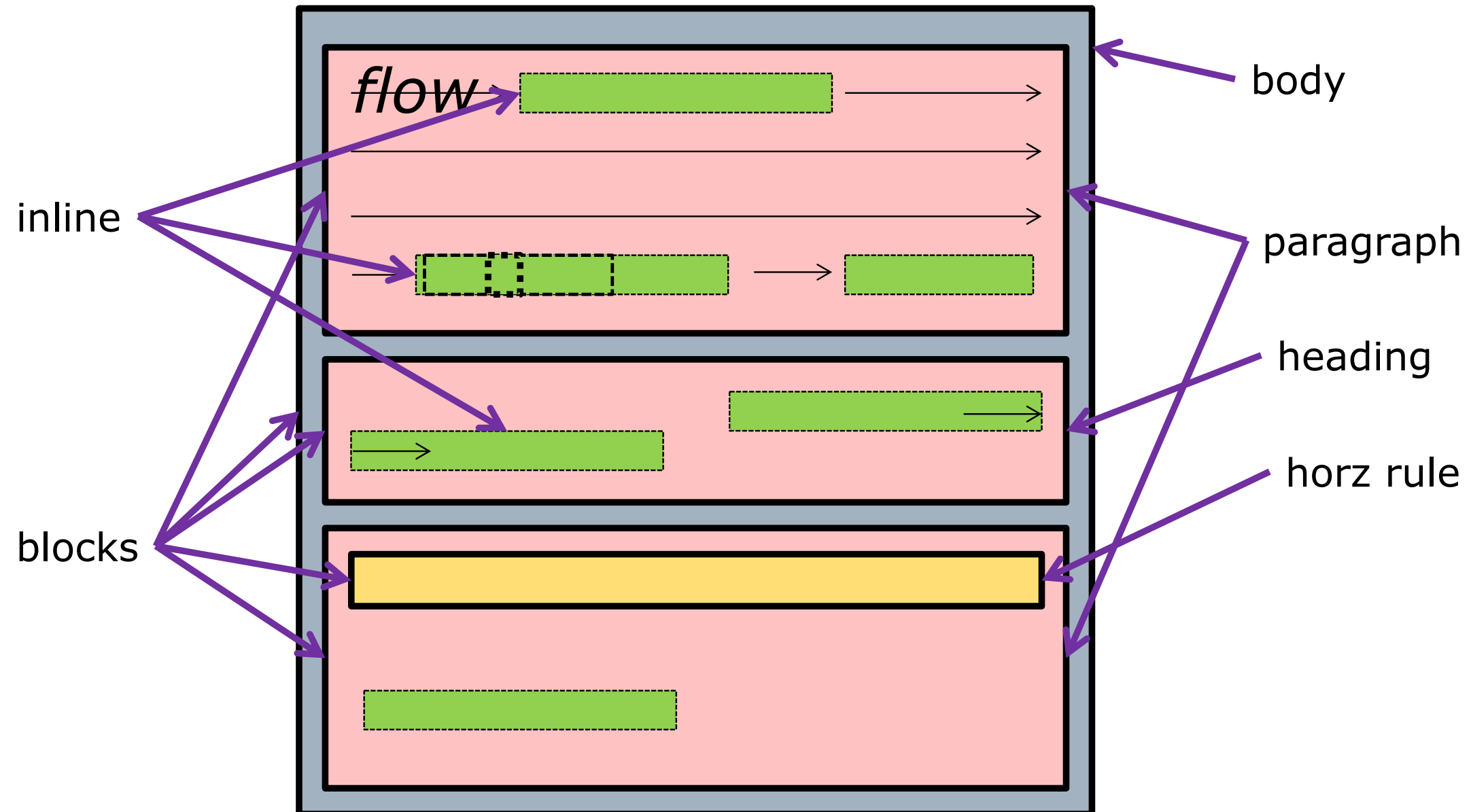


Style: Flow, Flex, Grid, and Fonts

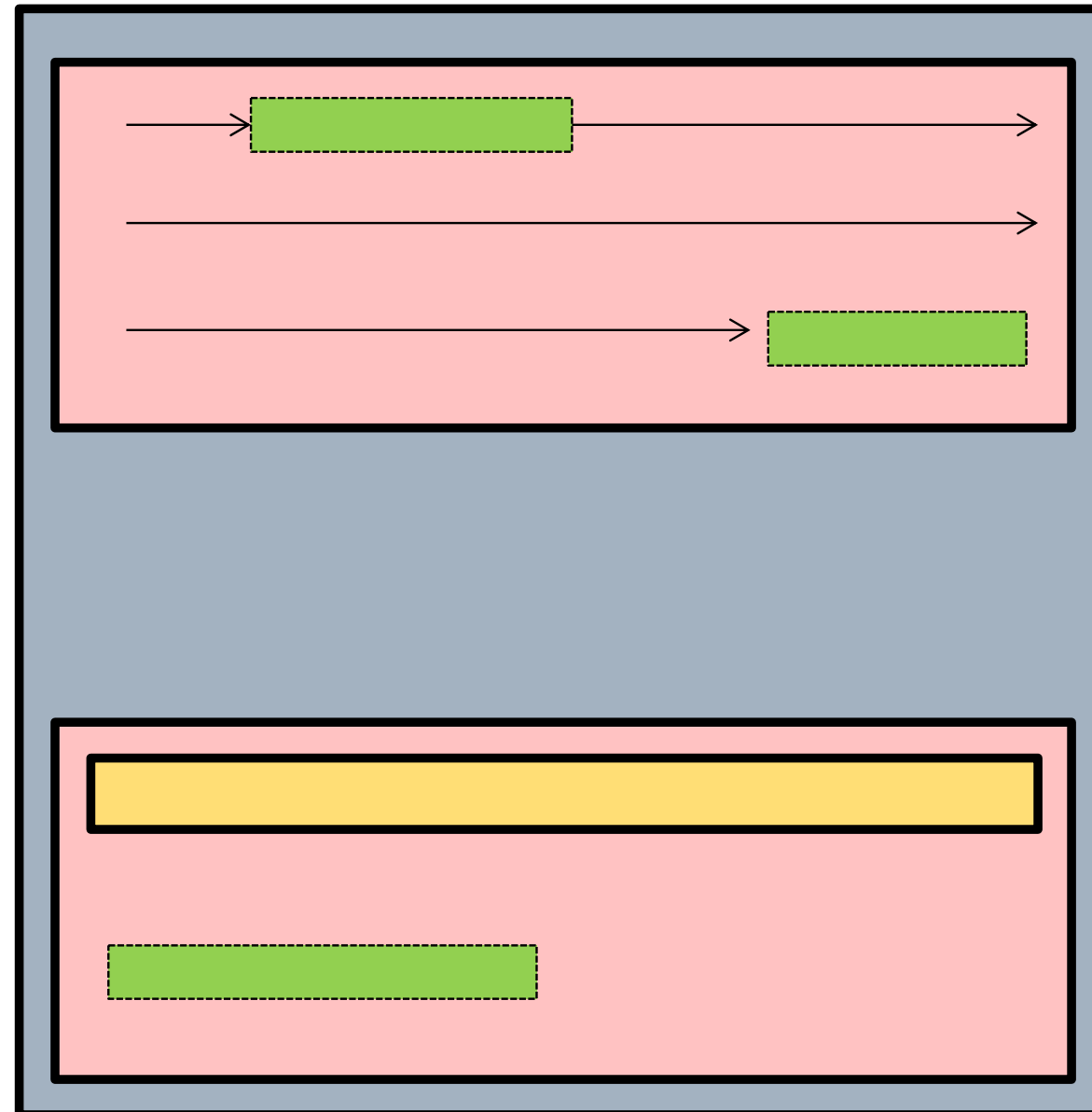
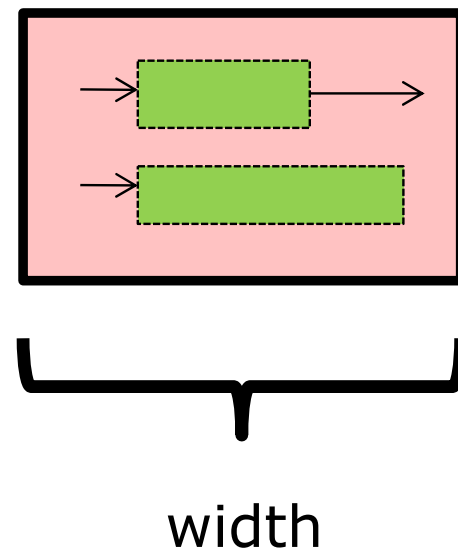
Computer Science and Engineering ■ College of Engineering ■ The Ohio State University

Lecture 19

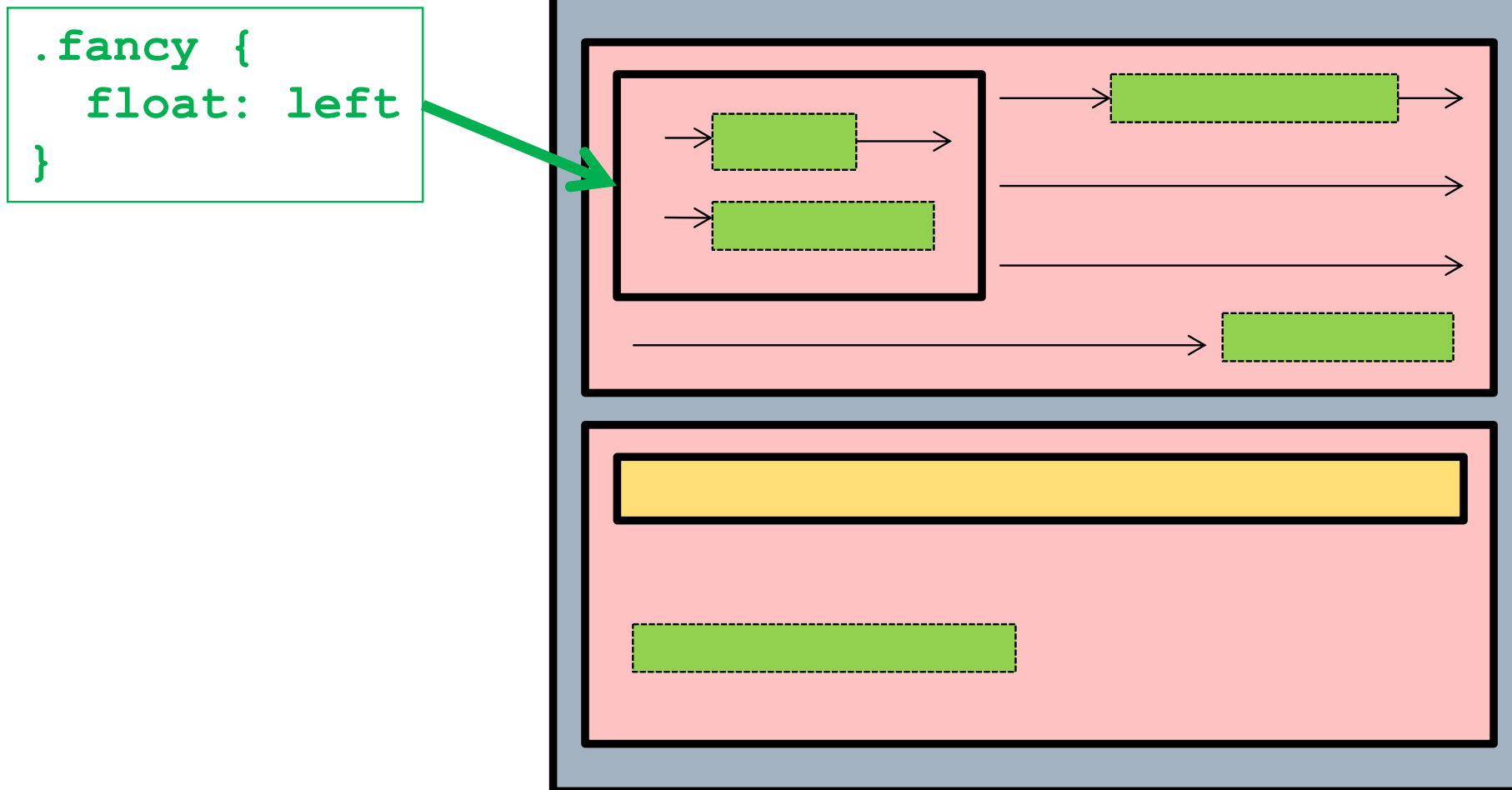
Recall: Blocks, Inline, and Flow



Floating: Remove From Flow

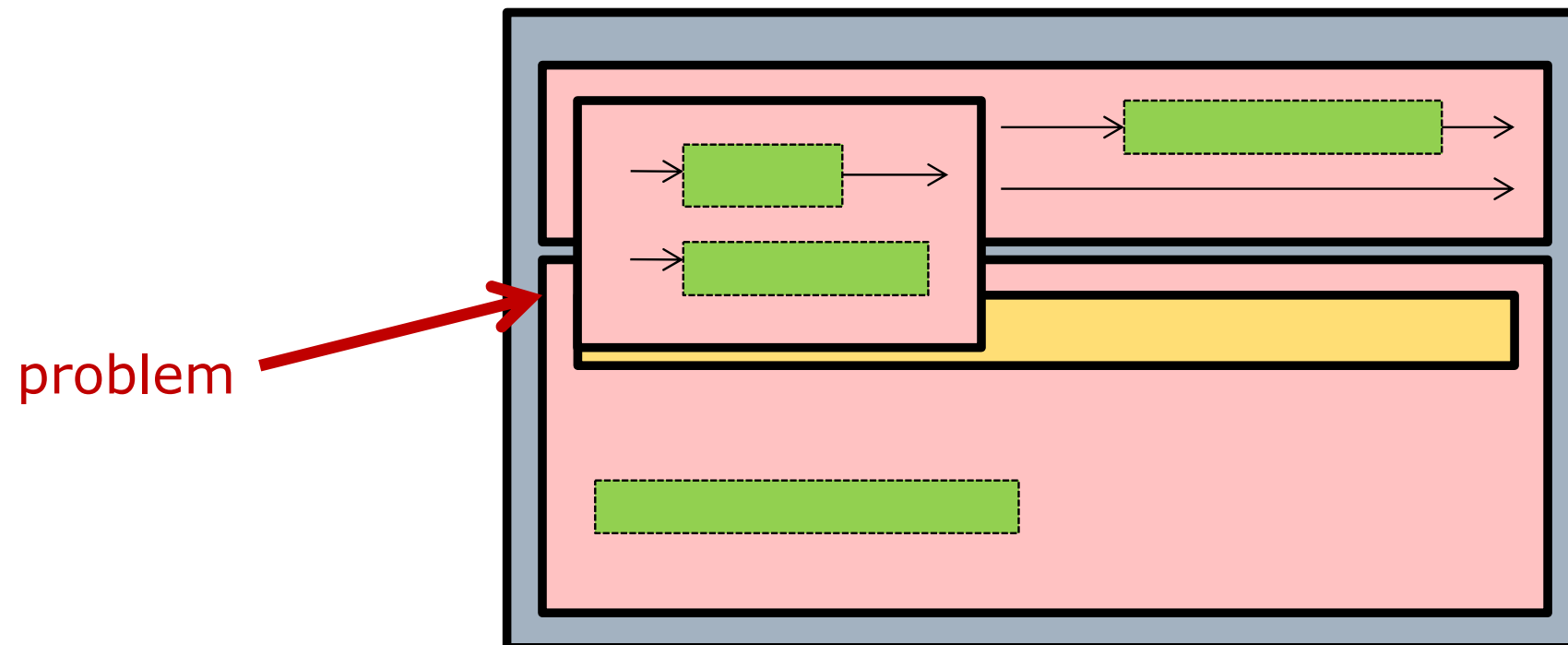


Floating: Overlays Block



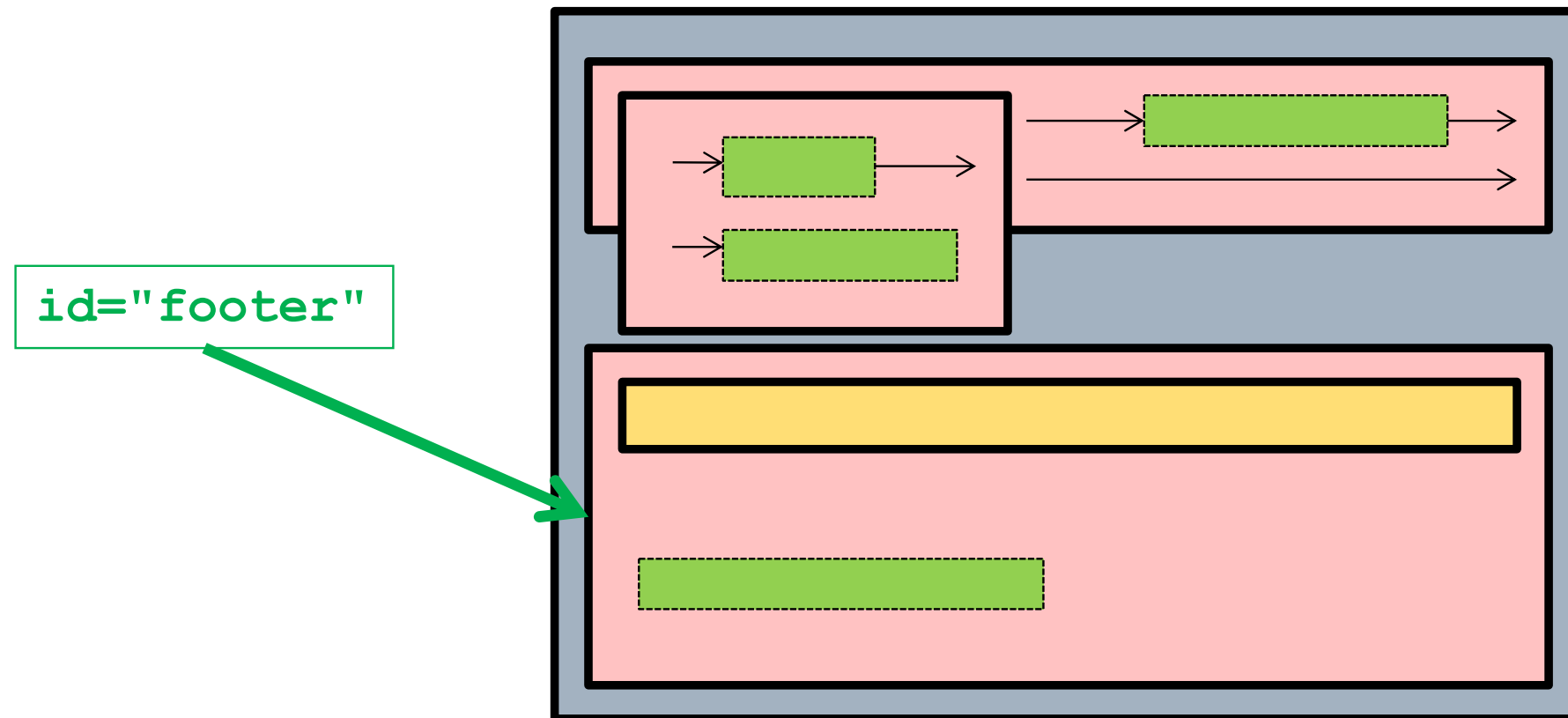
Problem: Blocks Below

- ❑ Floating element may be taller than containing element
- ❑ May be undesirable, eg for footer that should be below everything *including floats*



Solution: clear

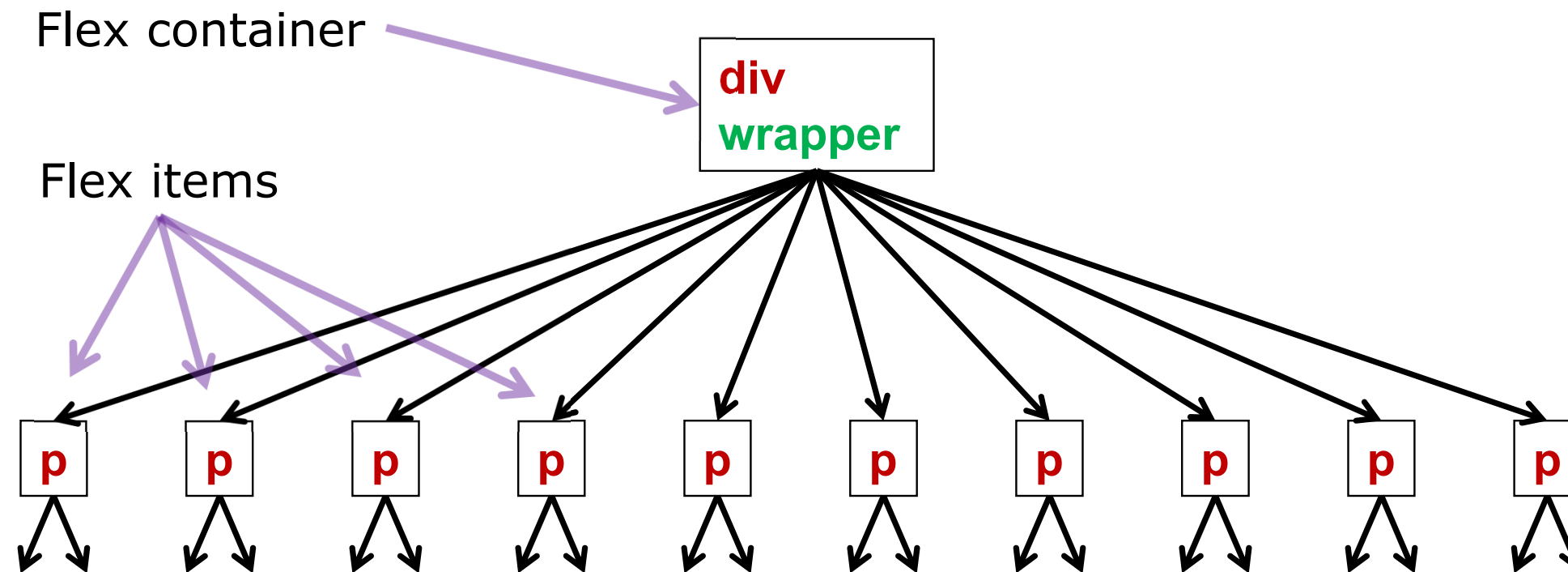
- ❑ Styling for block element *after* float
 - `#footer { clear: left; }`
- ❑ Requires *that* side to be clear of floats



CSS: Flexbox

- ❑ Display property for controlling whether elements are block or inline
- ❑ Parent element is the *flex container*
 - Style with CSS property (`display: flex`)
 - Set direction/wrap of children
 - Set justification/alignment of children
- ❑ Direct children are the *flex items*
 - Set order of appearance in container
 - Set (relative) size of each item

Flex Content as a Tree



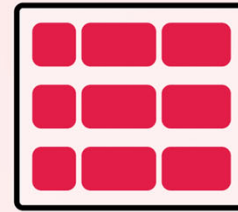
FlexBox Layout: Example

```
.wrapper {  
  display: flex;  
  flex-direction: row; /* default */  
  justify-content: space-evenly;  
  align-items: flex-start;  
}
```

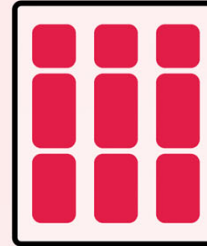
```
<div class="wrapper">  
  <p>1</p>  
  <p>2</p> ...  
</div>
```

CSS Flexbox

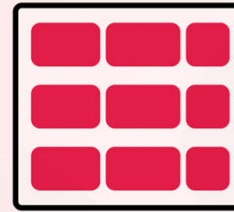
flex-direction



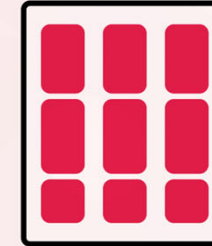
row



column

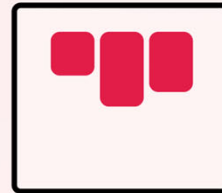


row-reverse

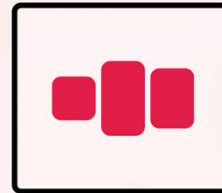


column-reverse

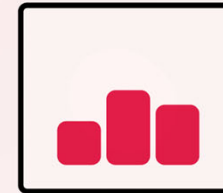
align-items



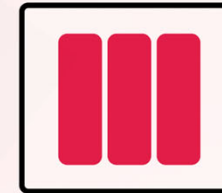
flex-start



center



flex-end

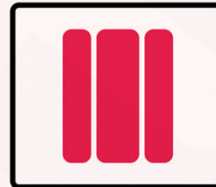


stretch

justify-content



flex-start



center



flex-end



space-between

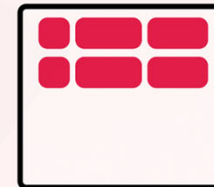


space-around

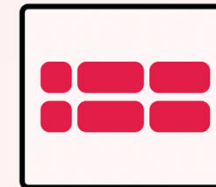


space-evenly

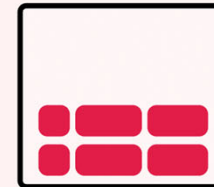
align-content



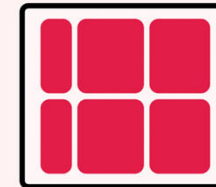
flex-start



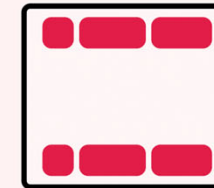
center



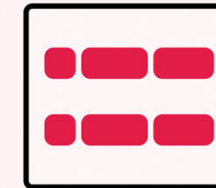
flex-end



stretch



space-between

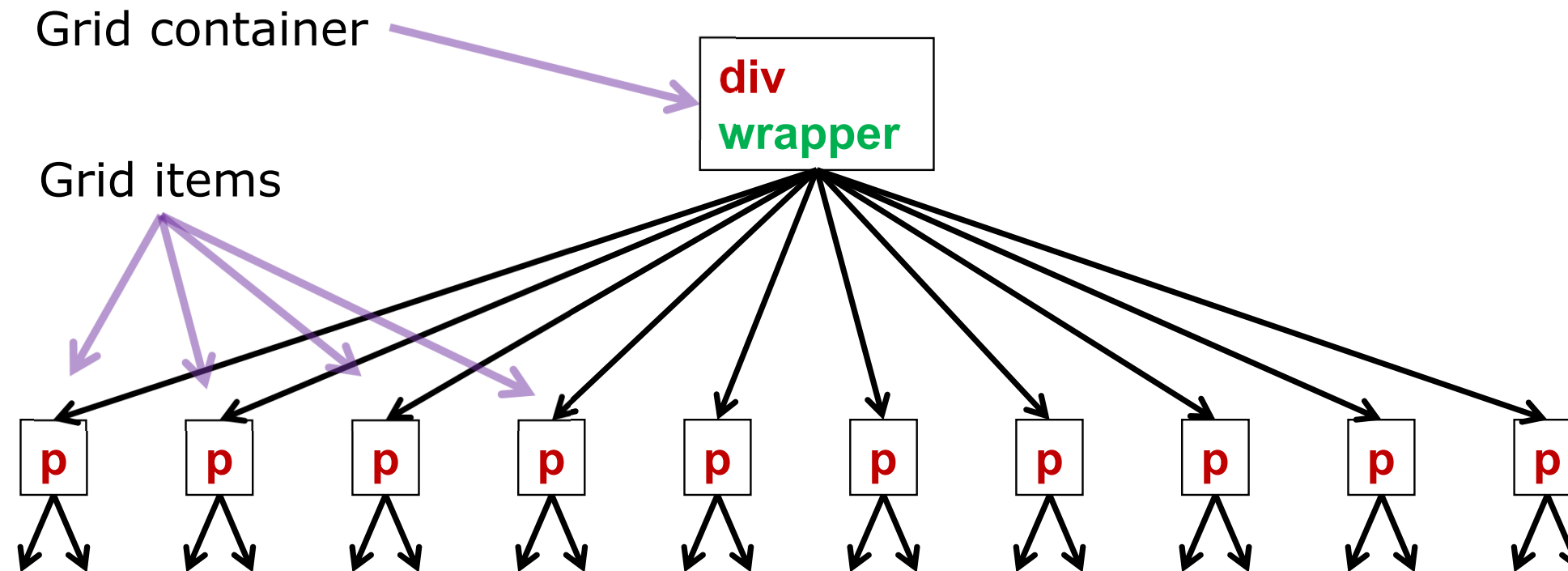


space-around

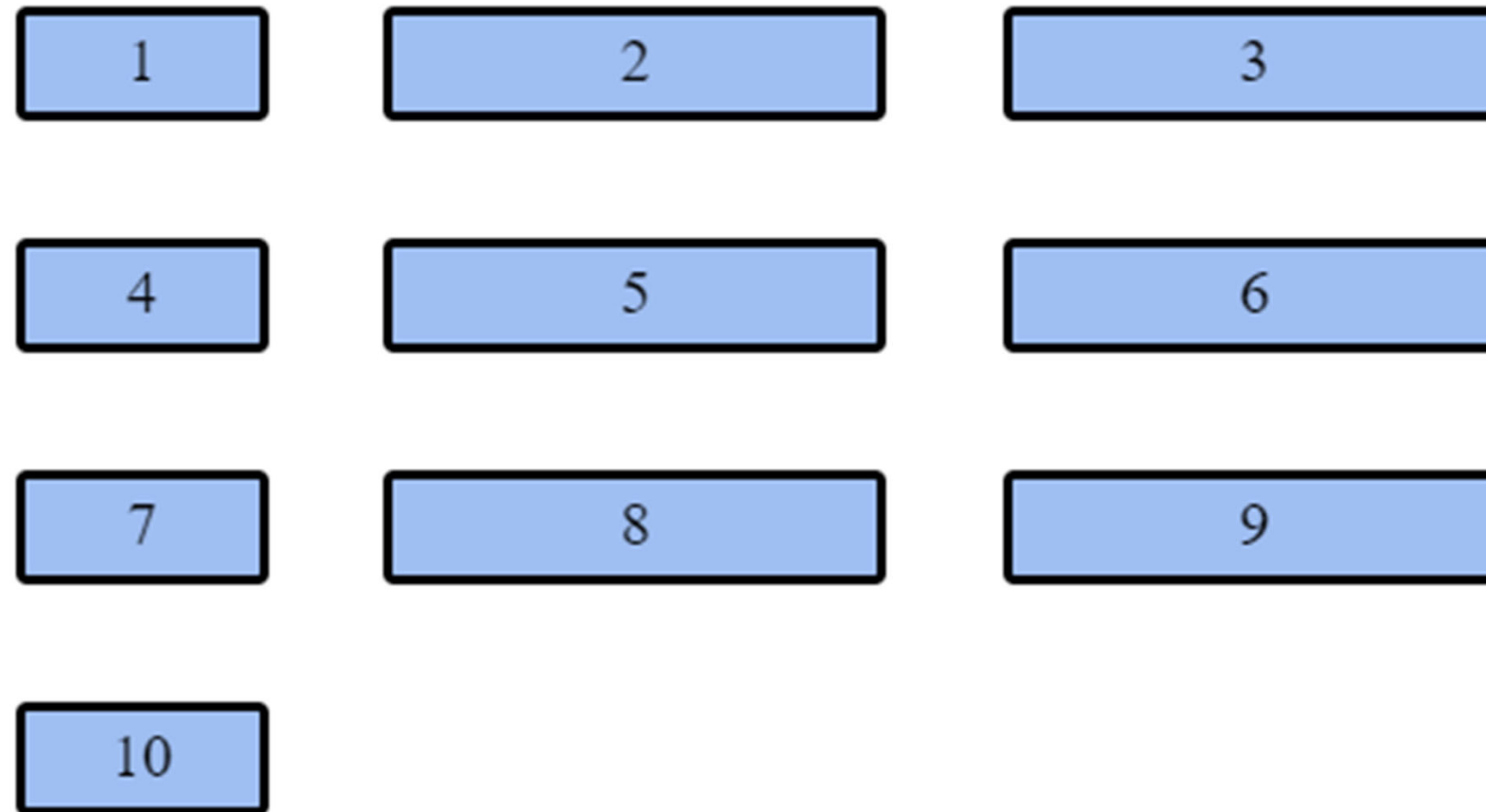
CSS: Grid Layout

- ❑ Display property for arranging elements in a 2D grid
- ❑ Parent element is the *grid container*
 - Style with CSS property (`display: grid`)
 - Set number/size of rows/columns
 - Set gap between rows/columns
- ❑ Direct children are the *grid items*
 - Set alignment, justification, placement
 - One item can be sized/placed to a *grid area* (ie a rectangular subgrid)

Grid Content as a Tree



Grid Layout: Example (Result)

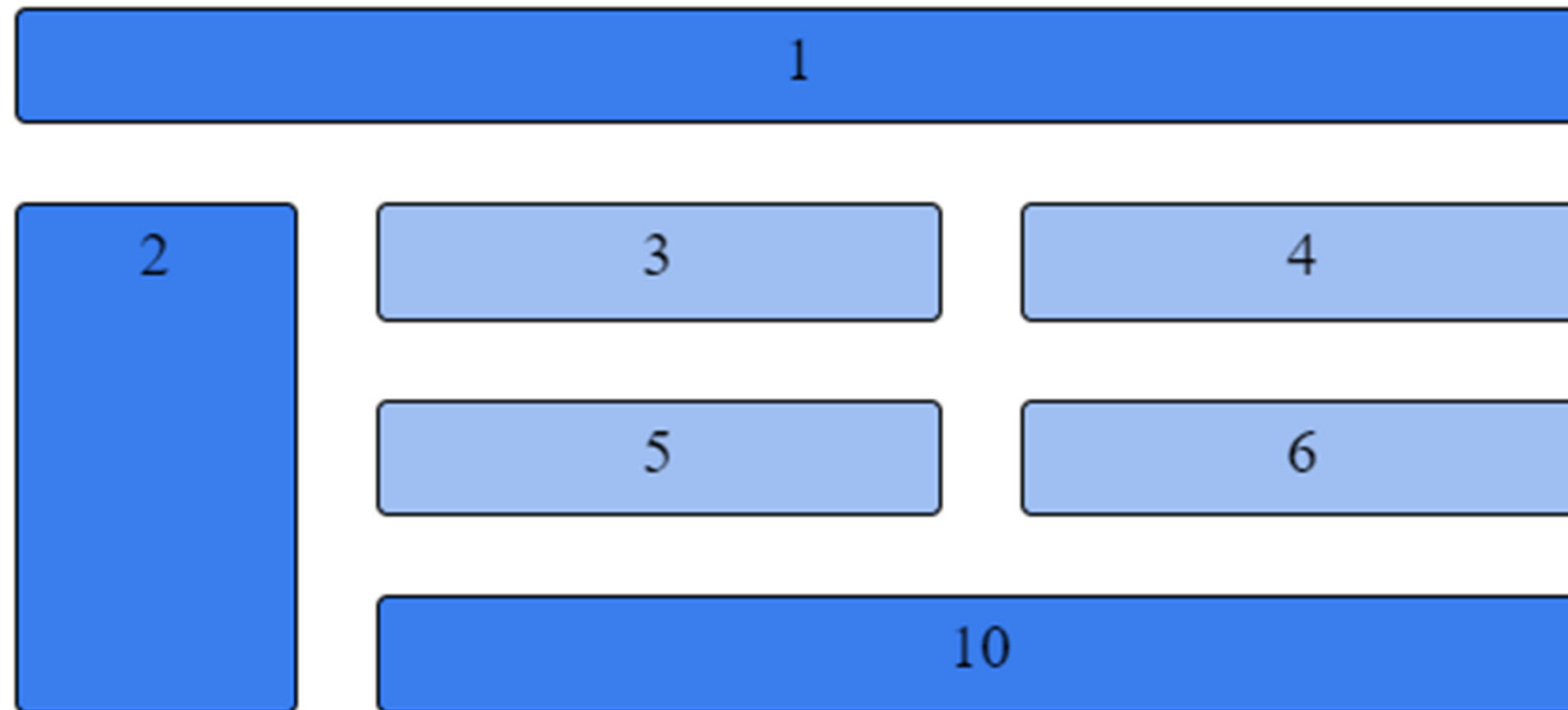


Grid Layout: Example (Source)

```
.wrapper {  
  display: grid;  
  grid-template-columns: 1fr 2fr 2fr;  
  grid-template-rows: repeat(4, 20px);  
  grid-gap: 20px;  
}
```

```
<div class="wrapper">  
  <div>1</div>  
  <div>2</div> ...  
</div>
```

Grid Areas: Example (Result)



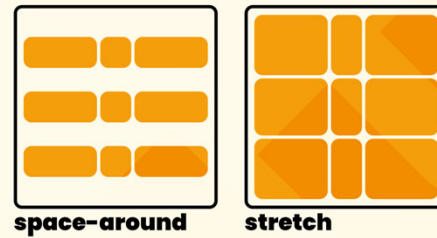
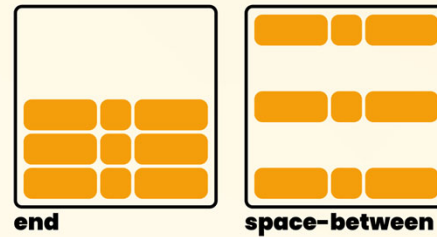
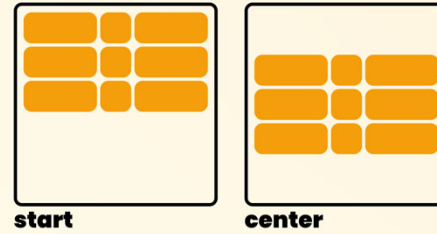
Grid Areas: Example (Source)

```
.top { grid-area: tp; }
.sidebar { grid-area: sd; }
#footer { grid-area: ft; }

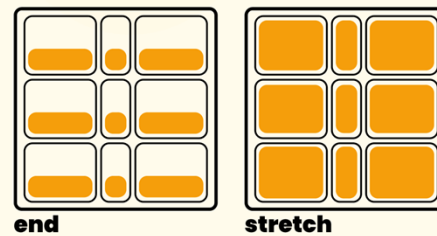
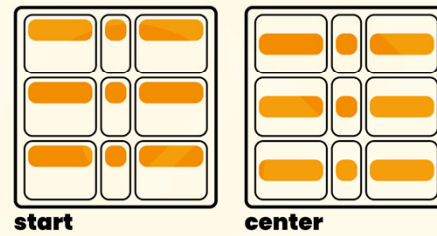
.wrapper {
  display: grid;
  grid-template-columns: 1fr 2fr 2fr;
  grid-template-areas:
    "tp tp tp"
    "sd . ."
    "sd . ."
    "sd ft ft";
}
```

CSS Grid Layout

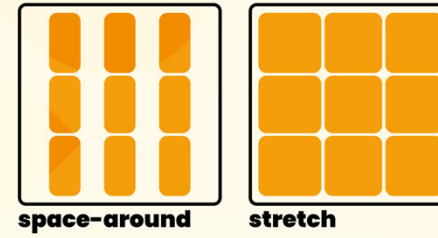
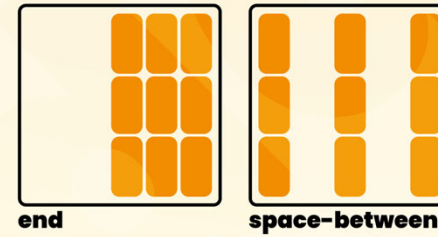
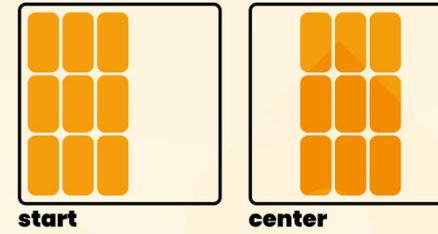
align-content



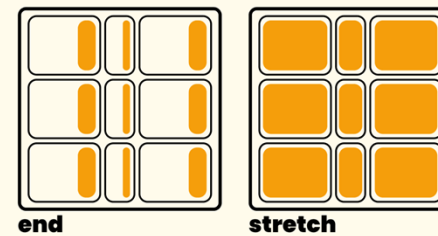
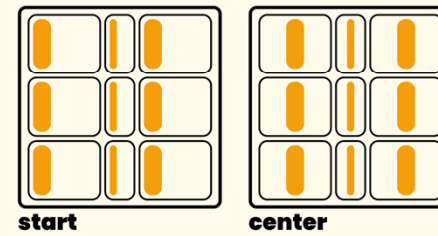
align-items



justify-content



justify-items

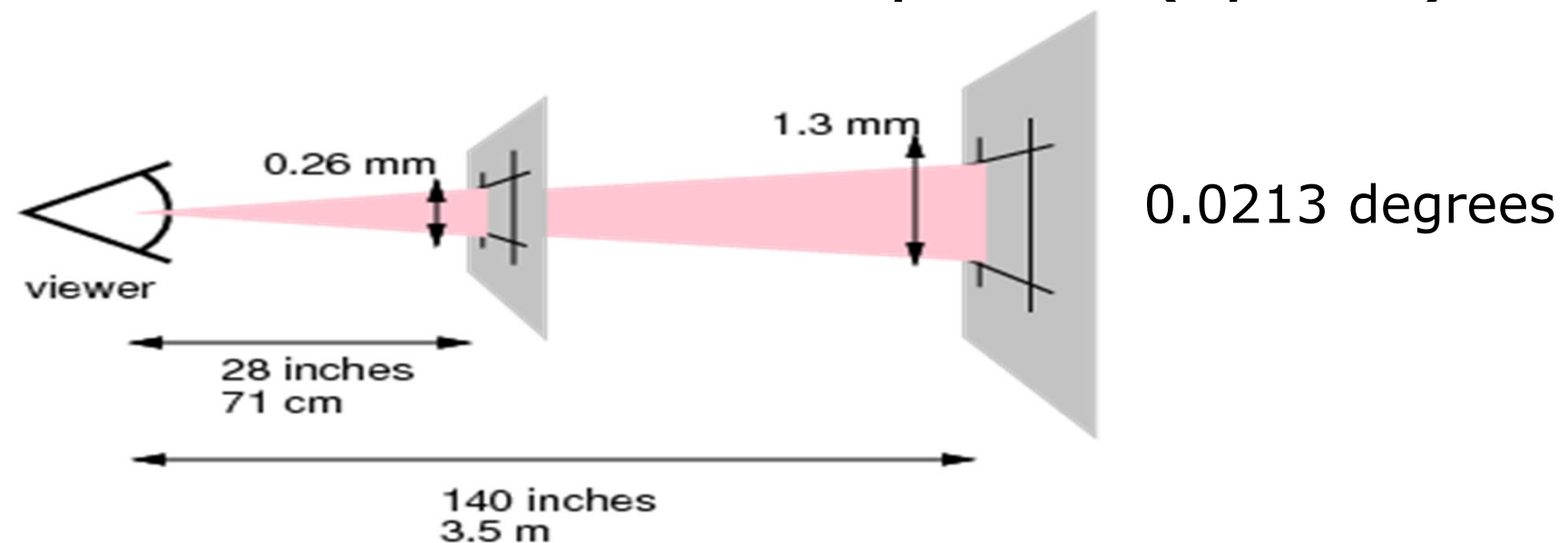


CSS Units for Size

- “Absolute” units (but browsers cheat)
 - `in`, `cm`, `mm`
 - `pt` (point) = 1/72 inch, `pc` (pica) = 12 pts
- Absolute (for a given resolution)
 - `px` (pixels)
- Relative to current element’s font
 - `em` = width of 'm' in element’s font
 - `ex` = height of 'x' in element’s font
- Relative to parent (or ancestor) size
 - `%`, `rem` (like `em`, but with root’s font)
- Standard advice for fonts:
 - Prefer relative units

Aside: The Problem with Pixels

- Historically, pixel size was determined by hardware (screen resolution)
 - ppi: pixels per inch
- Problems using **px** unit:
 - Different resolutions = different size of **px**
 - Different devices = different view distances
- Solution: W3C's "reference pixel" (*optics*)

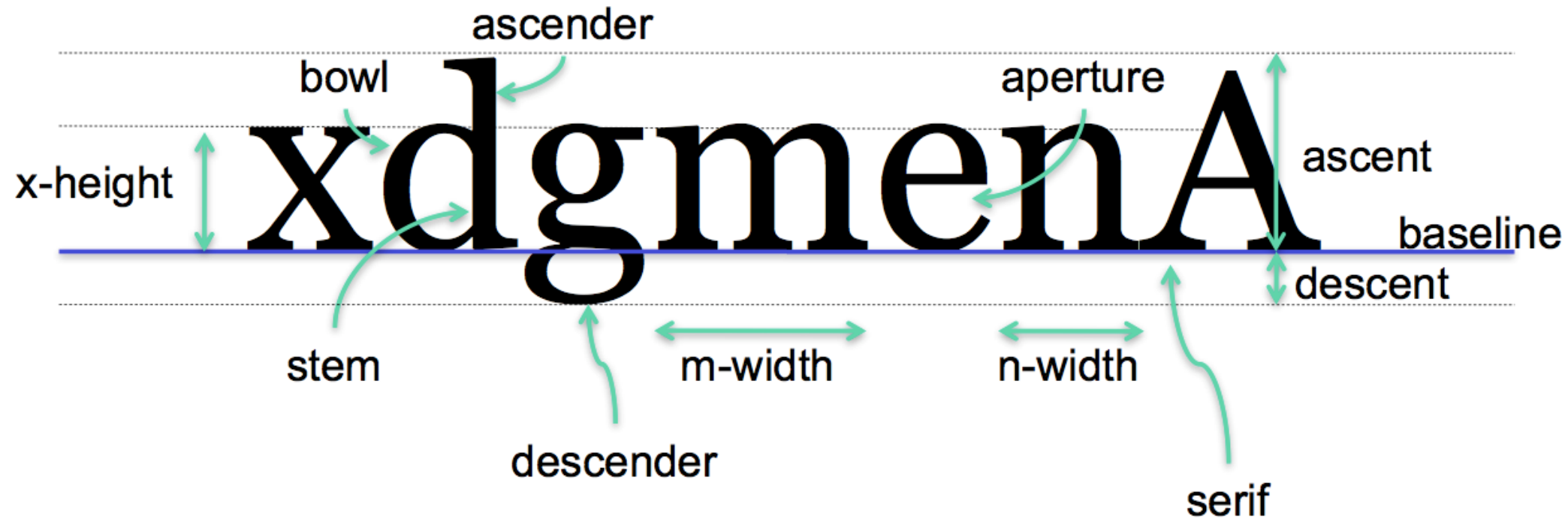


Fonts: Concepts

- Fonts are a key part of visual design
 - Serious, technical, whimsical, friendly...
- Font = family, weight, slant, etc
 - Family: **font-family**
 - Arial, Helvetica, Times, Courier, Palatino, Garamond, Verdana, Tahoma, Lucida,...
 - Weight: **font-weight**
 - thin, light, normal, bold, ...
 - 100, 200, 300, ..., 900
 - Slant: **font-style**
 - Normal, oblique, italic
- Font family should be “typeface”

Properties and Metrics

- Serif vs sans-serif
- Kerning: proportional vs monospace
- Size = ascent + descent (usually)
- m-width, x-height



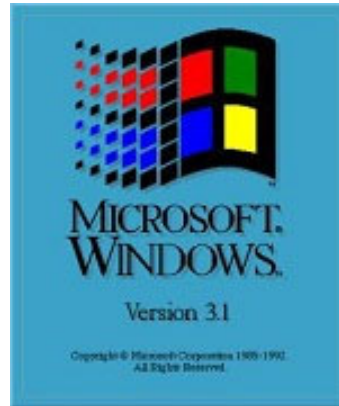
Whitespace

- ❑ Critical for aesthetics, readability
- ❑ Margins around body text, headings
- ❑ Leading
 - Space from baseline to baseline
 - CSS property: `line-height`
- ❑ Larger x-height = easier to read
 - But larger x-height also requires more line spacing
- ❑ “Music is the silence between the notes”

Font Families

- *De gustibus non est disputandum*
- Nevertheless, some common opinions
- Less is more: Use fewer fonts/sizes
 - Cohesive appearance
- Helvetica/Arial: clean but ubiquitous
 - They are identical / completely different
- Times is hard to read (on a monitor)
 - Better for print
- Comic Sans is for 12-year-olds and owners of NBA basketball teams

Identical & Completely Different



Fallback Fonts

- ❑ Not sure what fonts host OS will have
- ❑ CSS font-family: List alternatives in decreasing order of preference

```
font-family: Helvetica, Arial,  
            "Liberation Sans", sans-serif;
```

- ❑ Always end with one of 5 *generic* fonts:

- | | |
|----------------------------|----------------|
| ■ sans-serif (Arial?) | example |
| ■ serif (Times New Roman?) | example |
| ■ monospace (Courier New?) | example |
| ■ cursive (Comic Sans?) | example |
| ■ fantasy (Impact?) | example |

- ❑ OS (and browser) determine which font family each generic actually maps to

CSS3: Web Fonts @font-face

- Looks like a selector, but actually a “directive”

```
@font-face {  
    font-family: HandWriting;  
    src: url('PAGSCapture.woff2');  
}
```

- Font family is then available in rest of CSS

```
body { font-family: HandWriting; ... }
```

- User agent dynamically downloads font
- Font files come in a variety of formats
 - For web, prefer .woff2 (optimized for faster load times)
 - Other formats: .ttf, .otf, .eot, .svg, ...
- Beware: copyright issues!
 - See fontsquirrel.com, fonts.google.com

Summary: Floats, Flex, Grids, Fonts

- Controlling the flow
 - Floating elements: Removed from flow, layered on top
 - CSS flexbox: 1D layout (with wrap)
 - CSS grid: 2D layout
- Fonts
 - Fallback fonts to account for uncertainty
 - Web fonts for dynamic loading