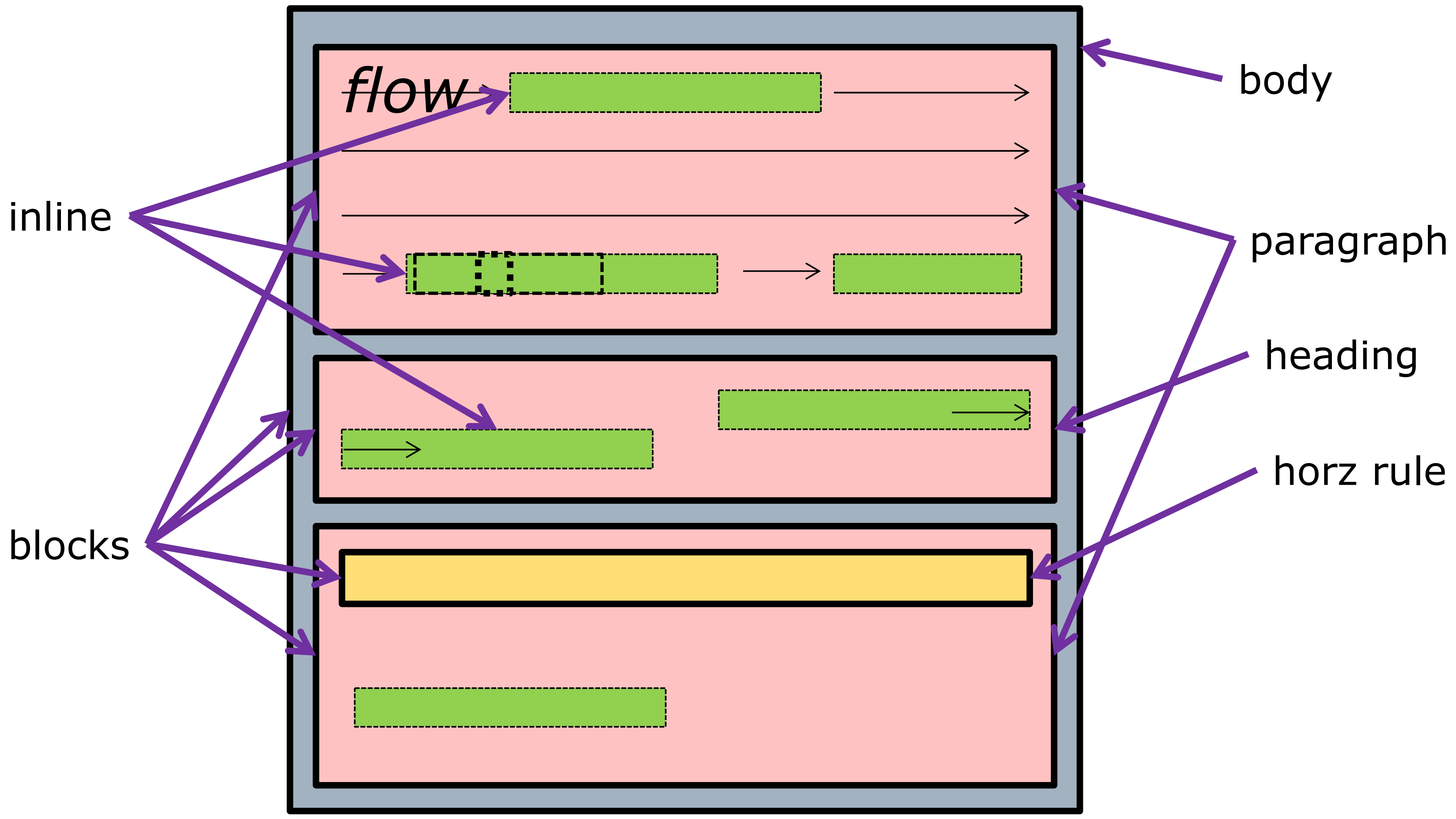
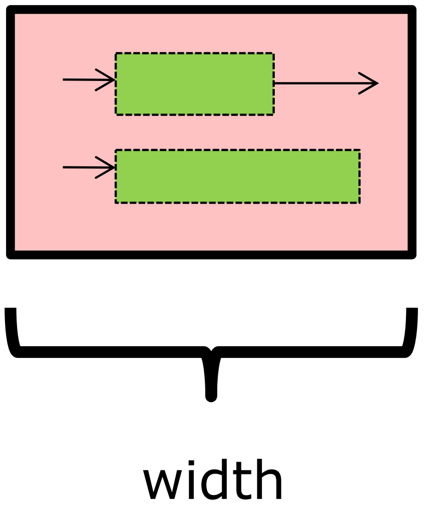
Style: Flow, Fonts, Images

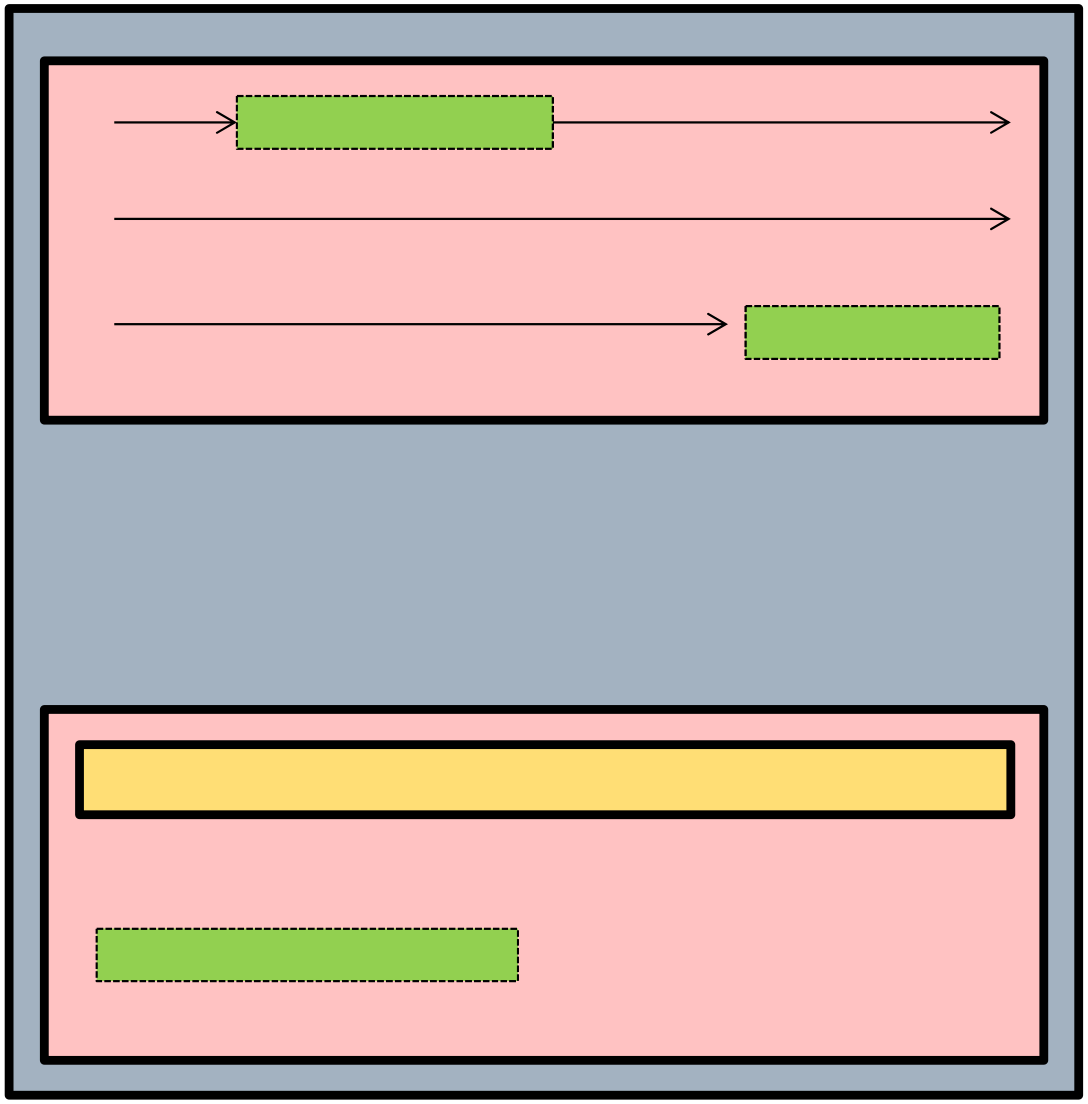
Lecture 20

Recall: Blocks, Inline, and Flow



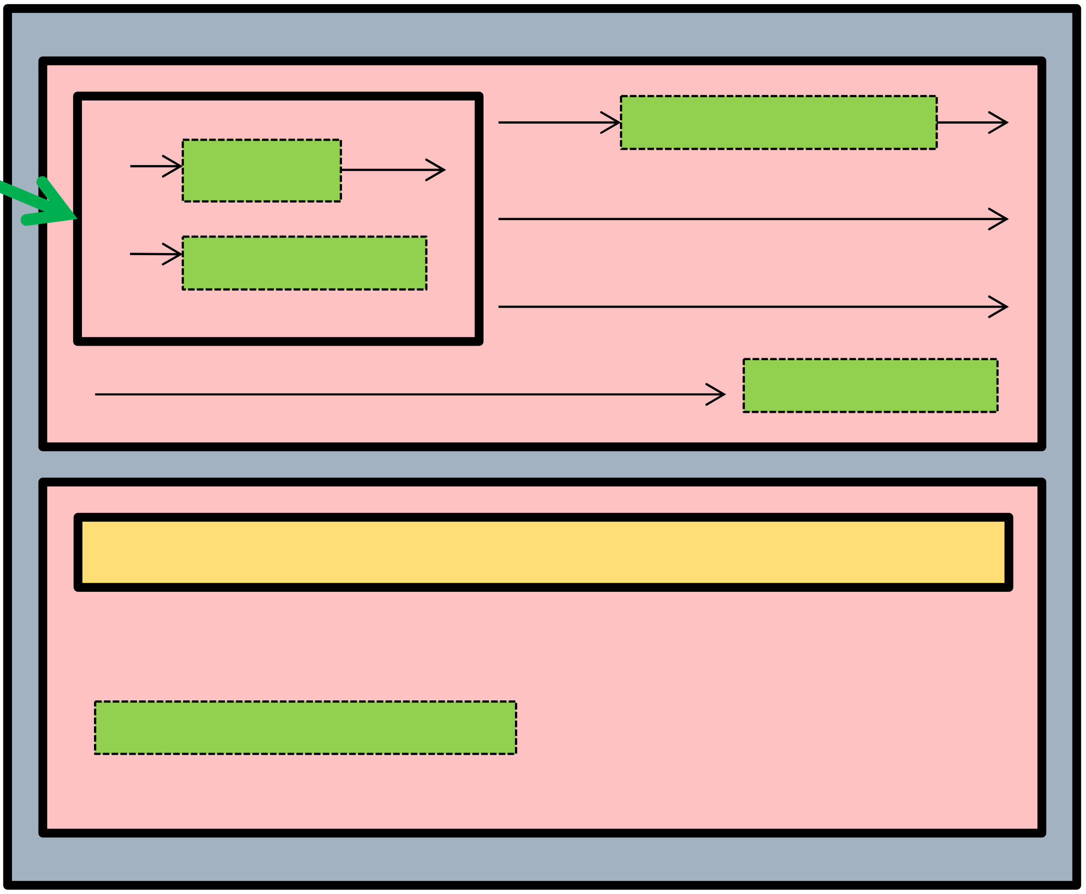
Floating: Remove From Flow





Floating: Overlays Block

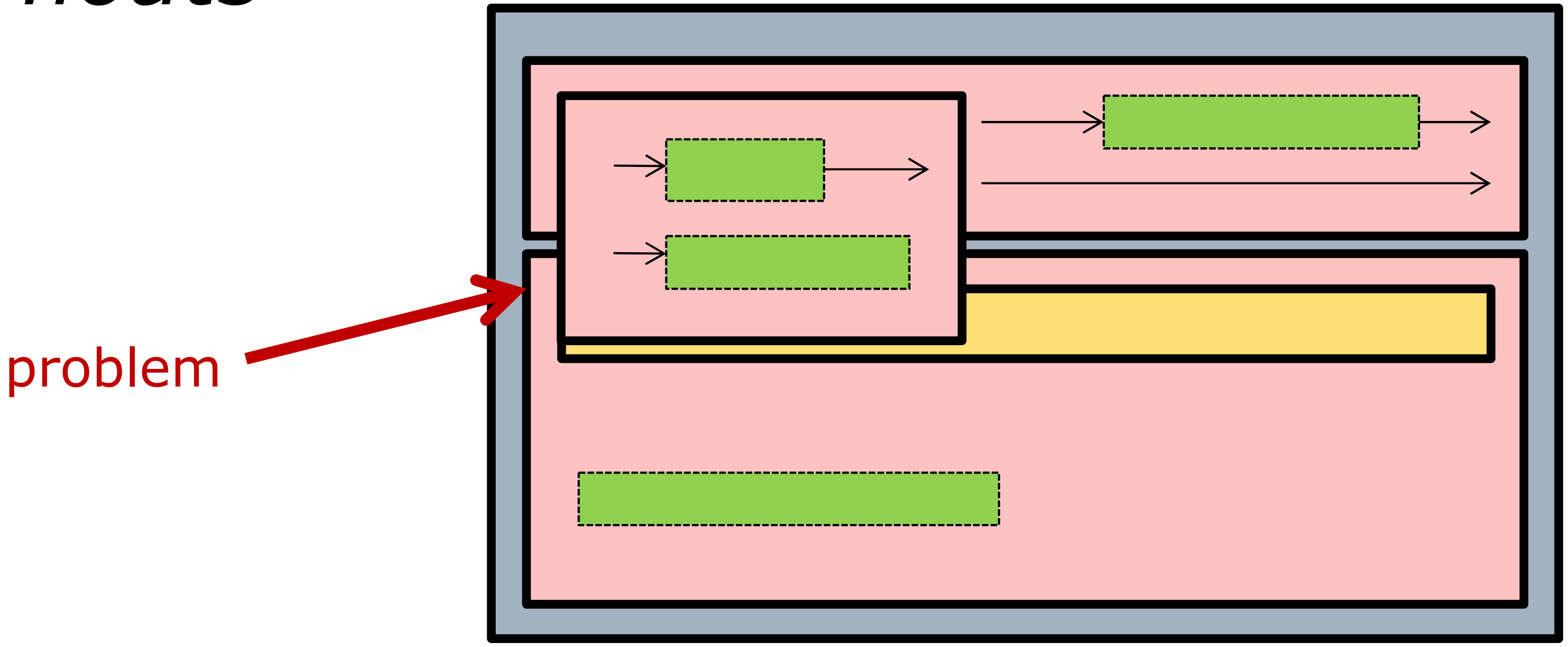
```
.fancy {
  float: left
}
```



codepen.io/cse3901/pen/bLYdLz

Problem: Blocks Below

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

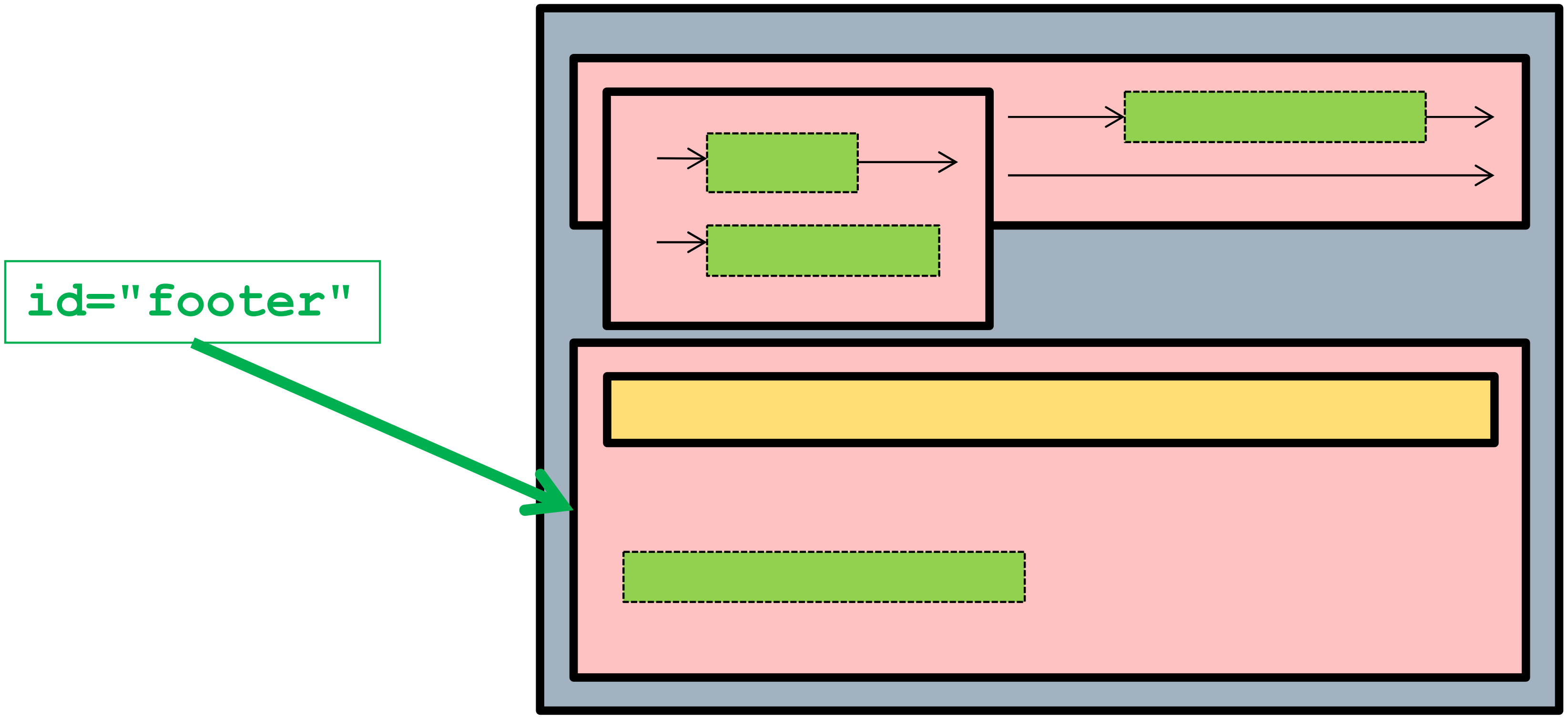


Solution: clear

- Styling for block element *after* float

```
#footer { clear: left; }
```

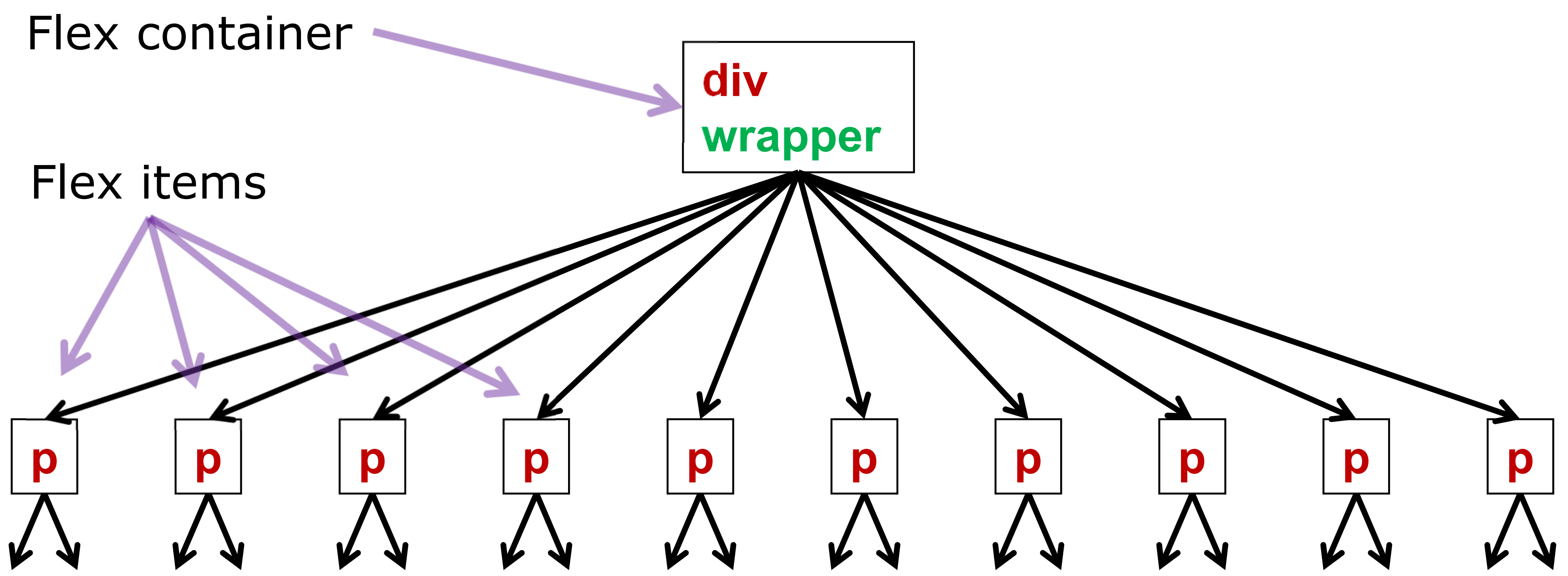
- Requires *that* side to be clear of floats



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



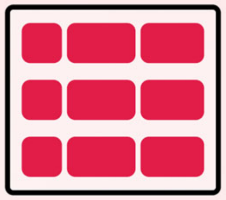
FlexBox Layout: Example

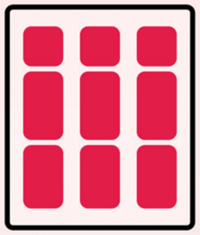
```
.wrapper {
  display: flex;
  flex-direction: row; /* default */
  justify-content: space-evenly;
  align-item: flex-start;
}

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

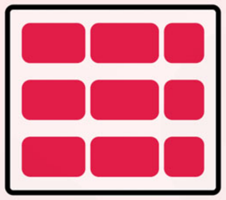
codepen.io/cse3901/pen/poMyppJ

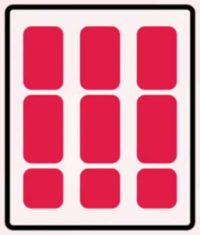
CSS Flexbox

flex-direction

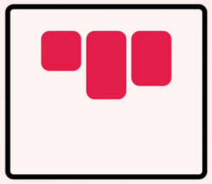

row

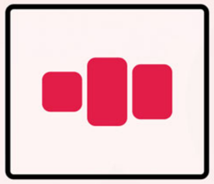

column

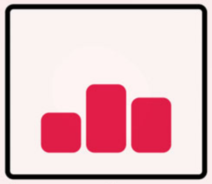

row-reverse

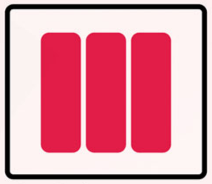

column-reverse

align-items

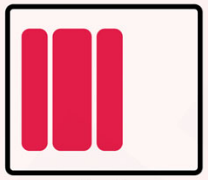

flex-start

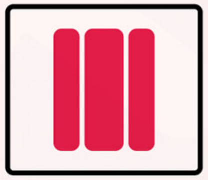

center

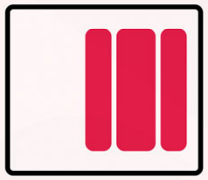

flex-end

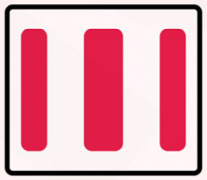

stretch

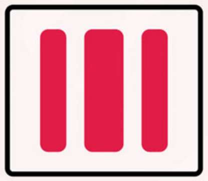
justify-content

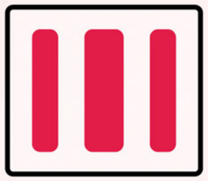

flex-start


center

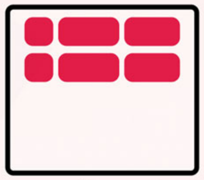

flex-end

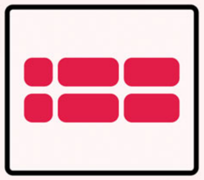

space-between

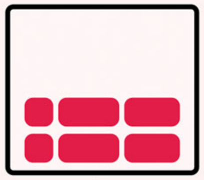

space-around

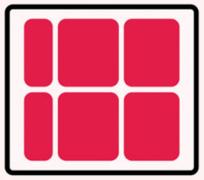

space-evenly

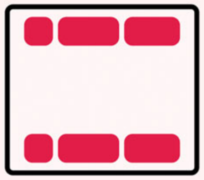
align-content

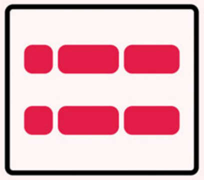

flex-start


center


flex-end


stretch


space-between

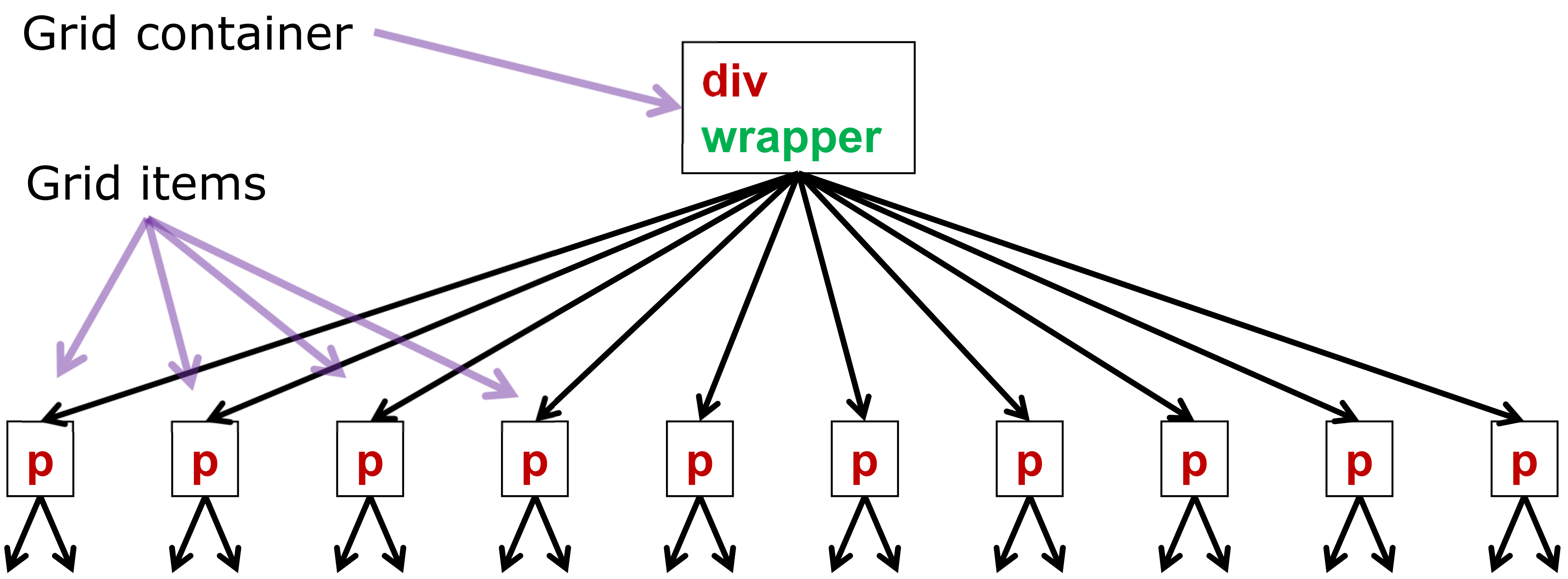

space-around

@eludadev

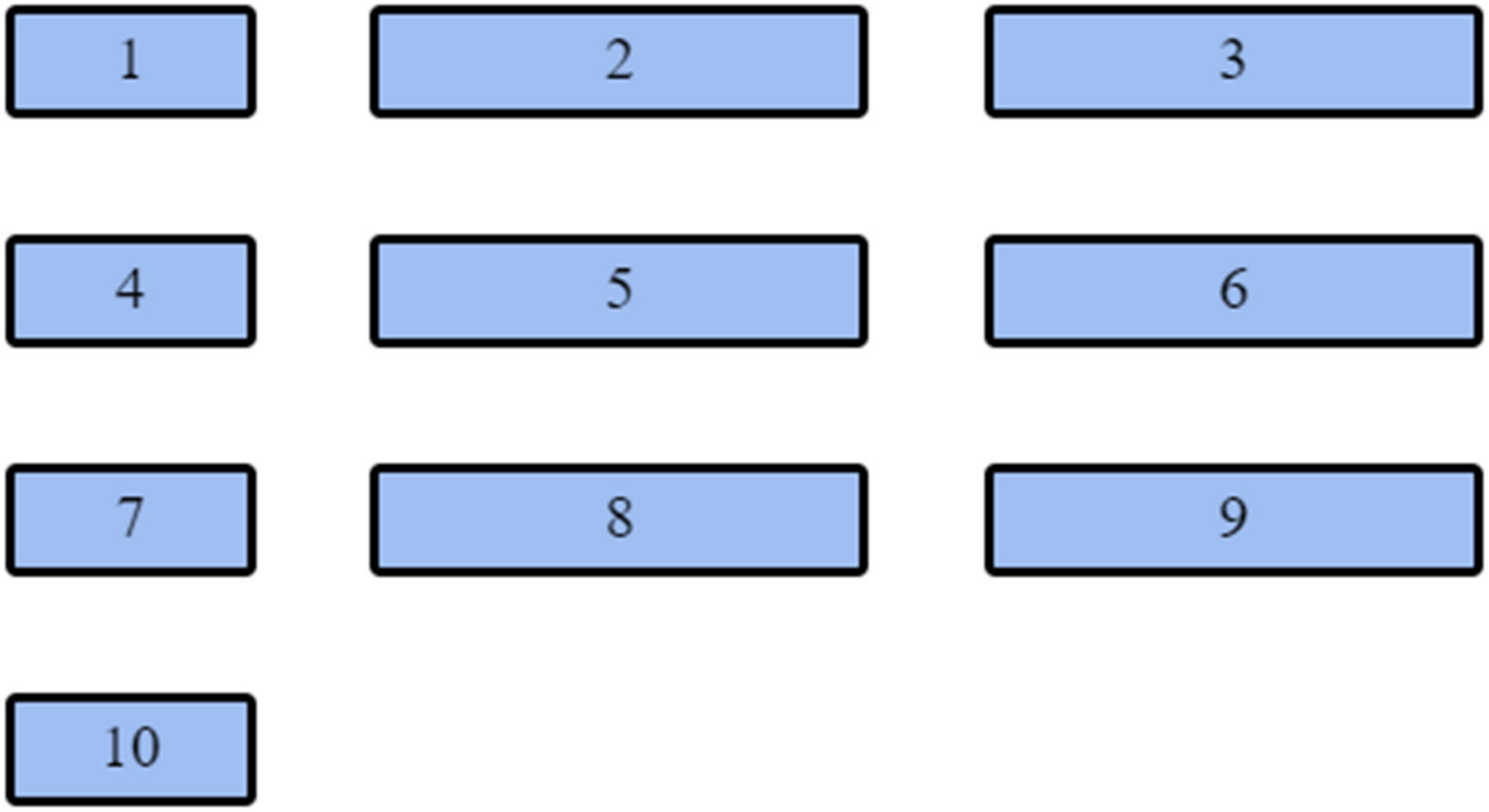
CSS: Grid Layout

- 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

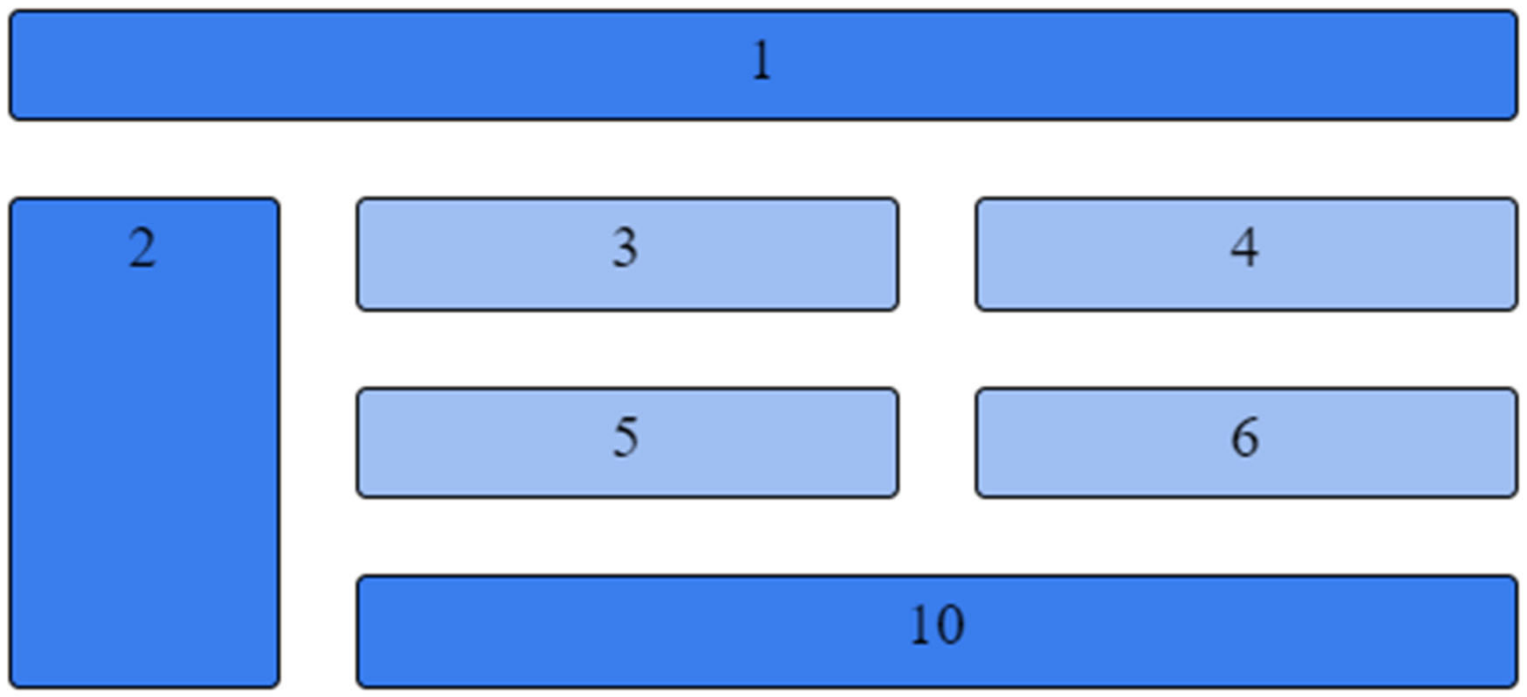


Grid Layout: Example

```
.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>
```

codepen.io/cse3901/pen/aqVNJN

Grid Areas: Example



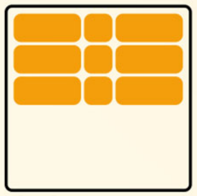
Grid Areas

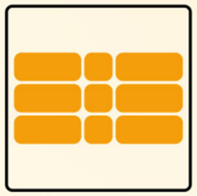
```
.top { grid-area: tp; }
.sidebar { grid-area: sd; }
#footer { grid-area: ft; }

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

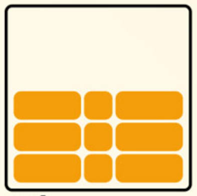
codepen.io/cse3901/pen/oEoKXV

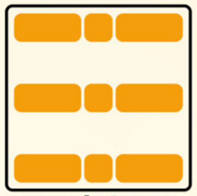
CSS Grid Layout

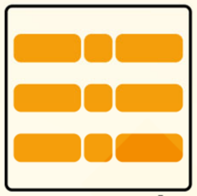
align-content

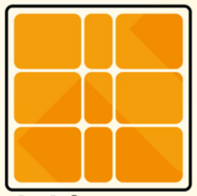

start


center

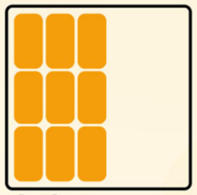

end

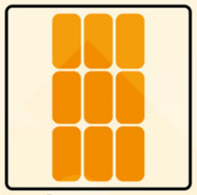

space-between

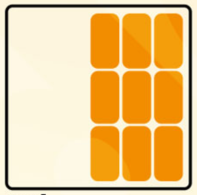

space-around

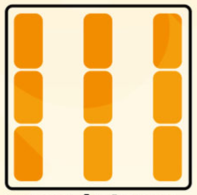

stretch

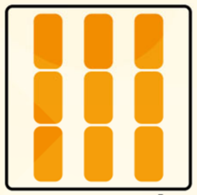
justify-content

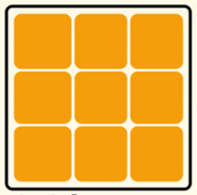

start


center

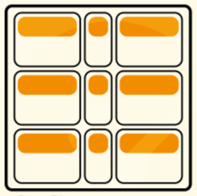

end

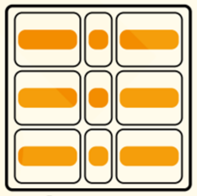

space-between

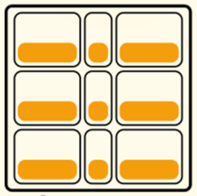

space-around

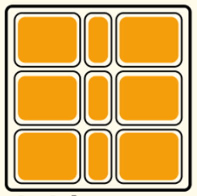

stretch

align-items

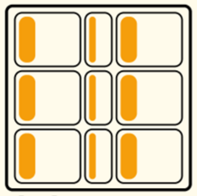

start

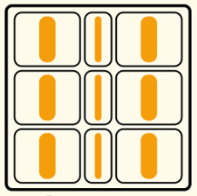

center

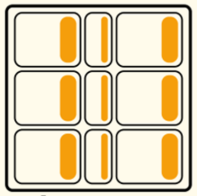

end

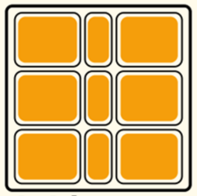

stretch

justify-items


start


center


end


stretch

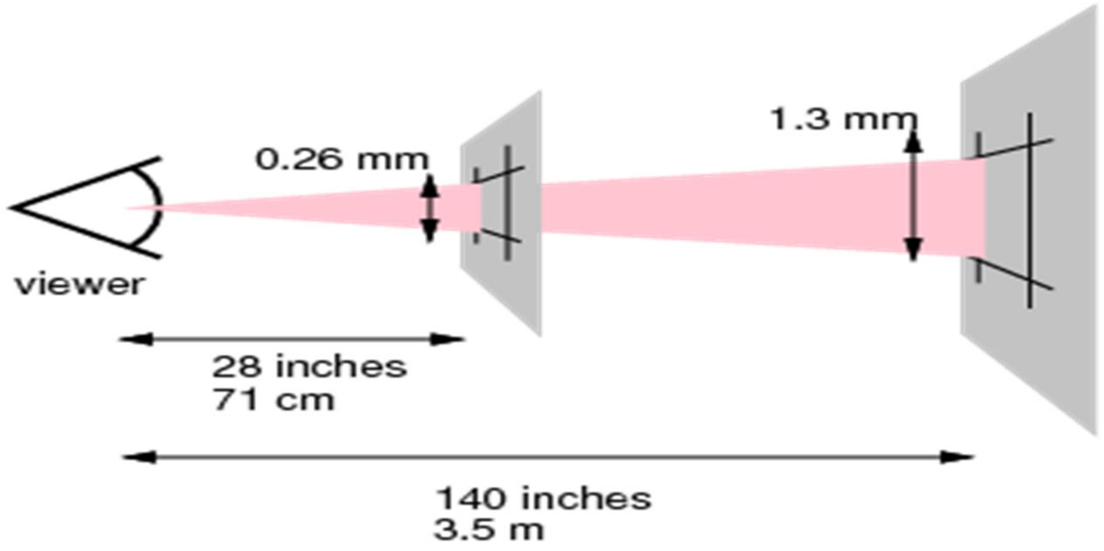
github.com/eludadev/css-docs

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*)



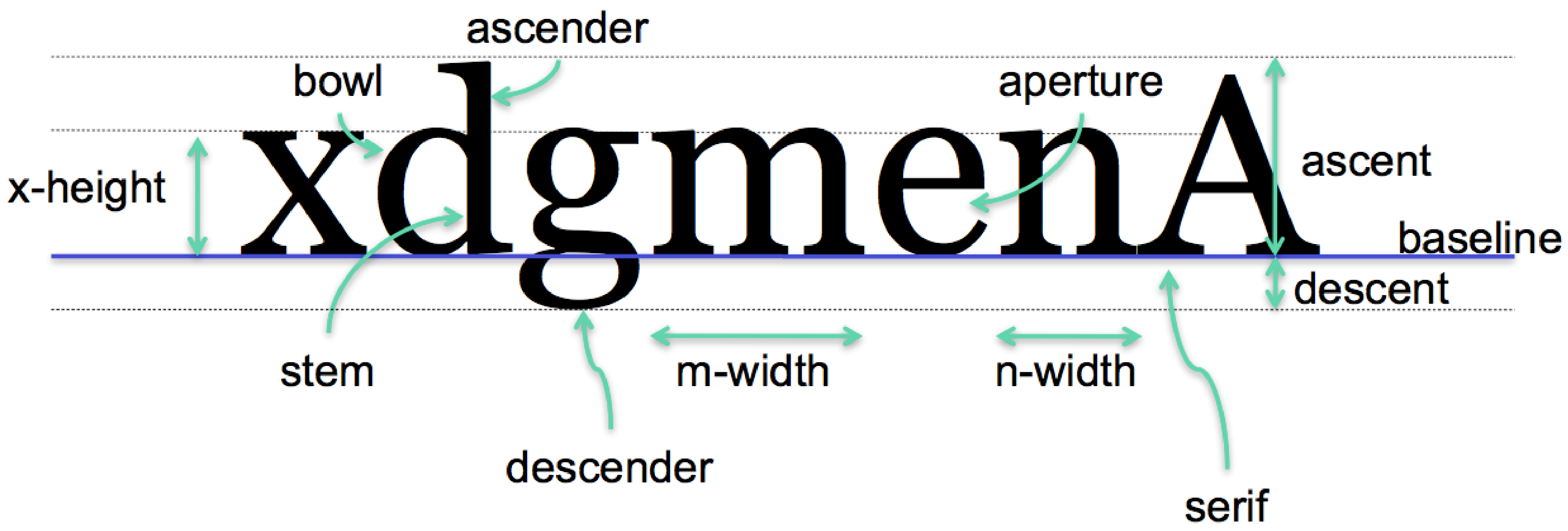
0.0213 degrees

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



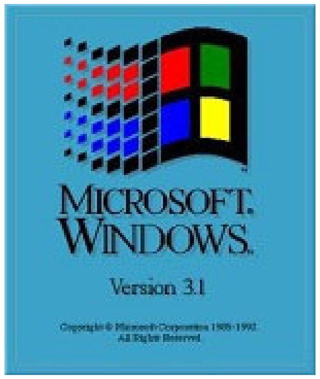
Whitespace

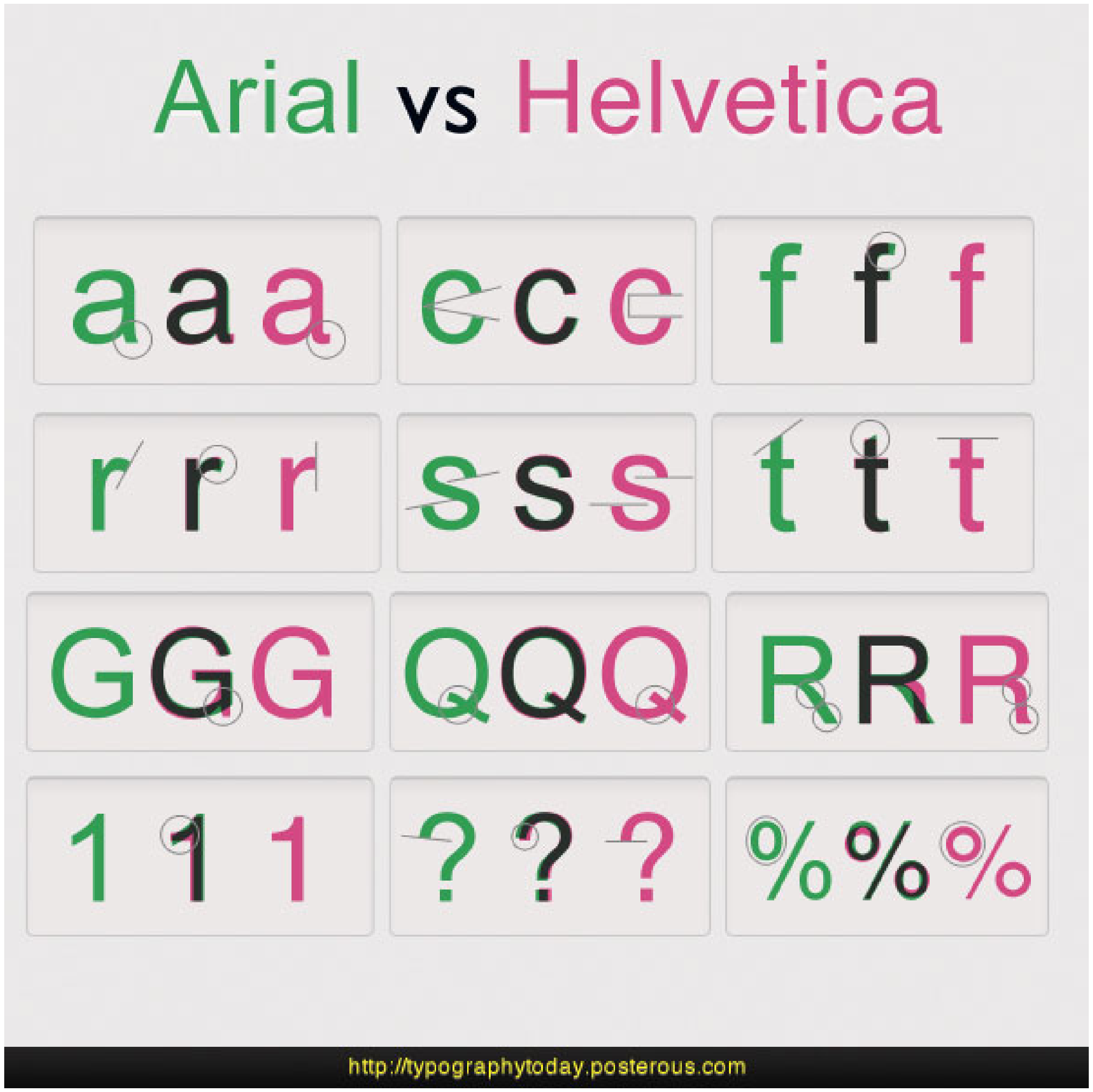
- Critical for aesthetics, readability
- Margins around body text, headings
- Leading
  - Space from baseline to baseline
  - CSS property: **`line-height`**
- Larger x-height = easier to read
  - But larger x-height also requires more line spacing
- "Music is the silence between the notes"

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


MICROSOFT.
WINDOWS.
Version 3.1
Copyright © Microsoft Corporation 1985-1992. All Rights Reserved.



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 is a "directive"

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

- Font family then available in rest of CSS

```
p { font-family: HandWriting; … }
```

- User agent dynamically downloads font
- Different syntaxes for font files
  - .ttf, .otf, .eot, .woff, .svg, …
- Beware: copyright issues!
  - See fonts.google.com

CSS Color Values

- Keywords: case-insensitive identifiers

```
red, navy, firebrick, chocolate
```

- RGB as decimal (0-255), percentage, or hex

```
rgb (255, 0, 0)  /* pure red */
rgb (100%, 0%, 0%)
#ff0000
#f00 /* expand by doubling each digit */
```

- HSL (Hue, Saturation, Light)
  - Hue (0-360) is angle on color wheel: 0 is red, 120 green, 240 blue
  - Saturation & light are both %'s

```
hsl (0, 100%, 50%) /* full bright red */
```

- Alpha channel (transparency): 1 is opaque!

```
rgba (255, 0, 0, 0.5)
hsla (0, 100%, 50%, 1)
```

Color Keywords: 147 (141 dist.)

aliceblue	antiquewhite	aqua	aquamarine	azure	beige
bisque	[illegible]	blanchedalmond	blue	blueviolet	brown
burlywood	cadetblue	chartreuse	chocolate	coral	cornflowerblue
cornsilk	crimson	cyan	darkblue	darkcyan	darkgoldenrod
darkgray	darkgreen	darkkhaki	darkmagenta	darkolivegreen	darkorange
darkorchid	darkred	darksalmon	darkseagreen	darkslateblue	darkslategray
darkturquoise	darkviolet	deeppink	deepskyblue	dimgray	dodgerblue
firebrick	floralwhite	forestgreen	fuchsia	gainsboro	ghostwhite
gold	goldenrod	gray	green	greenyellow	honeydew
hotpink	indianred	indigo	ivory	khaki	lavender
lavenderblush	lawngreen	lemonchiffon	lightblue	lightcoral	lightcyan
lightgoldenrodyellow	lightgray	lightgreen	lightpink	lightsalmon	lightseagreen
lightskyblue	lightslategray	lightsteelblue	lightyellow	lime	limegreen
linen	magenta	maroon	mediumaquamarine	mediumblue	mediumorchid
mediumpurple	mediumseagreen	mediumslateblue	mediumspringgreen	mediumturquoise	mediumvioletred
midnightblue	mintcream	mistyrose	moccasin	navajowhite	navy
oldlace	olive	olivedrab	orange	orangered	orchid
palegoldenrod	palegreen	paleturquoise	palevioletred	papayawhip	peachpuff
peru	pink	plum	powderblue	purple	rebeccapurple
red	rosybrown	royalblue	saddlebrown	salmon	sandybrown
seagreen	seashell	sienna	silver	skyblue	slateblue
slategray	snow	springgreen	steelblue	tan	teal
thistle	tomato	turquoise	violet	wheat	white
	whitesmoke	yellow	yellowgreen		

HTML **`<img>`** Tag Attributes

- **`src`**: location (URL) of image file
- **`width`**, **`height`**:
  - Area in *window* to reserve for image
  - Image is *scaled* to those dimensions
  - These attributes affect browser flow, regardless of when/if image is displayed
- **`alt`**: text to show if graphic can not be displayed or seen (ie alternative)
- **`title`**: text to *augment* displayed graphic (eg tooltip)

Image Representation

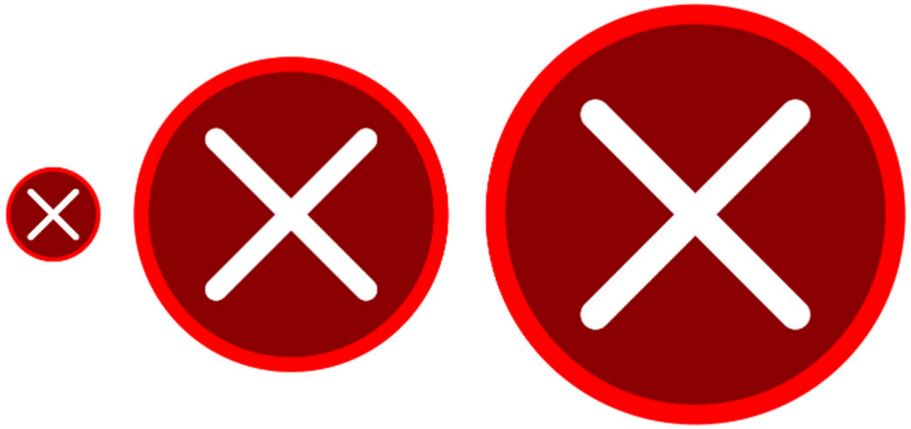
- Raster vs vector graphics
  - Raster: stored pixel-by-pixel
  - Vector: mathematical description
- Compression of raster images
  - Lossy: better compression, lower quality image
  - Lossless: largest file size, best quality

Major Formats

- GIF
  - Raster graphics, lossy compression (oldest)
  - 8 bit, basic transparency (on/off)
  - Frame-based animation (groan)
  - Good for small file size, crisp lines, logos
- JPEG
  - Raster, lossy compression
  - 24 bit, no transparency
  - Good for photos, gradual gradients
- PNG
  - Raster, lossless (but still often good) compression
  - Variable depth, full alpha transparency
  - Good replacement for GIF (but no animation)
- SVG
  - vector graphics
  - Good for crisp lines, simple logos, graphs

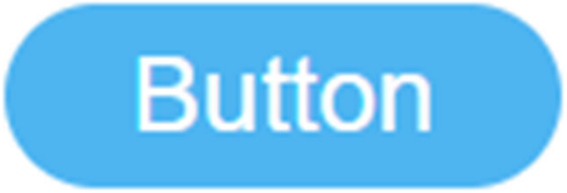
Scaling Images

- Vector graphics scale perfectly



- Raster images should be *pre-scaled*
  - Width (height) attributes of image tag should match actual width (height) of image
  - Why?
  - Cloud services can help (eg cloudinary.com)

Alternative: CSS



```
.button {
  display: inline-block;
  padding: 0.3em 1.2em;
  margin: 0 0.3em 0.3em 0;
  border-radius: 2em;
  box-sizing: border-box;
  text-decoration: none;
  font-weight: 300;
  color: #FFFFFF;
  background-color: #4eb5f1;
  text-align: center;
  transition: all 0.2s;
}
```

Summary

- 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
- Images
  - Formats jpeg, png, gif, svg
  - Tradeoffs of size, quality, features