Plan for today

**Flexbox stuff**
- Direction, align, and justify
- Growing, shrinking, wrapping
- Aside: a couple more selectors

**Absolute and relative units**
- Percent, viewport, em, and rem

**position**
- Breaking out of page flow

**CSS strategies and best practices**
**display: flex**

Completely changes how element is laid out

The element becomes a "flex container"

Its (direct) children become "flex items"

**Lays out flex items in a row or column**

Default: row. Use `flex-direction: column;` to change
Flexbox properties

**justify-content**: layout along the "main axis"
- main axis = flex-direction
- flex-start, flex-end, center
- space-between: equal space between flex items
- space-around: also leave space on the ends

**align-items**: layout along the cross axis
- cross axis = opposite of flex-direction
- flex-start, flex-end, center
More flexbox properties

Growing and shrinking
Applied to flex item

*flex-grow* (default 0): fill remaining space

*flex-shrink* (default 1): give up space to fit in box

**flex-wrap** *(default nowrap)*
Applied to flex container
Wrap to next row/column if necessary
Aside: more CSS selectors

> (direct child)
  s1 > s2: select s2 if it's a direct child of s1
  E.g. useful for flex items inside container

* (universal selector)
  select all elements
  E.g. .box > *: all direct children of .box

MDN list of selectors and combinators
**Units**

**font-size keywords**

- xx-small, ..., medium, ..., xxx-large
- Scale with browser font
- Absolute--won't scale with container font size

**em: relative unit**

- 1em = font-size
- Useful for margin/padding that needs to scale

**rem (root em)**

- Like em, but uses root font size
- Scale with browser text size

MDN `<length> units`
Bigger units

Percentage
Always relative to container (parent) element
100% isn't special (>100% = overflow container)

Viewport
Definition: the area of the browser window that shows page content
vw and vh: 1/100th of viewport width/height
Aside: calc

calc() is a CSS function

- Use it in place of a value
- Argument is a math expression
- Lets you combine units

E.g. width: calc(100vh - 200px);

(But Flexbox may be easier than writing a complex expression)
**position**

**position:** another way to move elements

Most useful when removing elements from page flow

Takes a keyword

**Default:** static

Normal flow, cannot move

**relative**

Start where it would be normally

Use top, bottom, left, right to move

E.g. `.elem { position: relative; left: 100px; }

`.elem will be 100px right of where it normally would be
**position**

**absolute**
- Relative to most recent positioned element
- Defaults to root element (top-left of viewport)
- Use `position: relative` on ancestor to control reference point

**fixed**
- Relative to root element (top-left of viewport)
- Always same position regardless of scrolling

**These two remove element from flow**
- No space reserved for it
CSS strategies

Many ways to do things
   Generally, pick the simplest one

Keep selectors simple
   Clear class names that describe semantics
   Count on inheritance
   Avoid complex dependency on cascade

Watch out for outdated/less useful CSS
   E.g. float, vendor prefixes (-moz, -webkit)
   Don't just copy/paste CSS
       Fall back on core concepts to understand properties
       Look up the properties for compat and interactions
Summary

That's it for CSS for now

  We'll come back to a few more things throughout
  But you can already build some really cool stuff!

Before next time

  assign2.1

Next time: APIs

  Back to JavaScript
  Working with data, interacting with servers