Plan for today

A few more things about HTML
Styling elements in JavaScript
Intro to styling and CSS
  Useful style attributes
  Separate CSS files
  Block vs. inline
Semantic elements

Elements serve two purposes: semantics and presentation

Semantics: how elements behave
  - Show images
  - Click and tab between links and buttons

Presentation: how elements look
  - Headings bold and large
  - Links blue and underlined

Can override presentation easily
  - Harder (and more confusing) to override semantics
Page regions

<header>, <main>, <footer>
<nav>: navigation (top bar, menus)
<article>: content that can stand alone (blog/forum post, news article)
<section>: a section of the page, often with a heading

These don't generally have default presentation
But still important for overall page structure
HTML quirks

Whitespace collapses
Any number of spaces shown as a single space

Some elements take up their own line
We'll get back to this

Escaping symbols with &name; (entities)
& ; (&), &lt; (&lt), &gt; (&gt)
A bunch more, but often can just put the char

Handling of "custom" (bogus) elements and attributes
No warning, just displays them
Don’t count on this
Styling elements

Element's style prop is an object

Can set properties on the element

These are called CSS properties

Most values are strings

```javascript
elem.style.backgroundColor = "#8c1515"; /* Cardinal */
/* Reset to "default" value (before we changed it) */
 elem.style.display = "";
 elem.style.border = "1px solid blue";
```

We'll learn more about all the different properties next time
Grouping styles

**HTML class attribute**

Space-separated list of CSS classes
Completely different from JS classes!
Can be used to find/select elems in JS, or to group styles (next time)

`classList` is a `DOMTokenList`

Exact type doesn't matter, but see link for methods

```javascript
if (elem.classList.contains("foo"))
    elem.classList.add("bar");
else
    elem.classList.remove("baz");
```
Styling with CSS

Cascading Style Sheets
Also not a programming language

Per-element style attribute
<p style="color: red">This text is red</p>

Not ideal
Mixes semantics (HTML) and presentation
Can't reuse styles
Can't change styles separately
<link> separate file with styles

<link rel="stylesheet" href="styles.css">
(in <head>)

.css Syntax:

selector {
    property: value;
    another-property: another-value;
}

Useful CSS properties

font-family: type face
   Specific ("Comic Sans") or generic name ("sans-serif")
font-style: for italics
font-weight: for bold
text-decoration: for underline/overline/strikethrough
color: text color
background-color: background color
text-align: alignment (left, center, right)
border: [length] [style] [color]
A bunch of **color names**

You should use them ...except IMO some names are strange so maybe only when they're obvious

**rgb(red, green, blue)**

From 0 to 255

**rgba(red, green, blue, alpha)**

Alpha is transparency (from 0 to 1)

**Hex color: #rrggbb**

You'll probably see this most often

Don't know hex? 00 = none, 80 = half, FF = full

Aside: **Stanford colors** (who knew)
### CSS selectors

**type: all type element on page**

```html
p { ... }
<p>...</p>
```

**class: elements with specified class attr**

```html
<p class="announcement">...</p>
```

**#id: element with specified id attr**

```html
<h2 id="quarter">...</h2>
```

### class vs. id

- ids must be unique across the page
- Use classes to describe the type of element
- **Best practice: describe type, not style**
  
  E.g. class="highlight", not class="yellow-bg"
Combining CSS selectors

type.class: type elements with class class)

p.announcement { ... }

(type#id works, but redundant

.class1.class2 works too

s1 s2: select s2 if descendant of s1

Need not be direct child

header a { ... }

All links inside the header section

s1, s2: select s1 and s2

h1, h2, h3 { ... }

Apply to all level 1, 2, and 3 headings
Cascade: when multiple selectors apply

Rule of thumb: most specific rule wins
ID > class > type
Longer selector > shorter selector
If tie, last definition > prior ones

Inheritance: many properties apply to descendants already
E.g. font/text properties
No good rule of thumb here
Selector tips

Keep your selectors simple
- Often, a class is enough
- Don't reuse class name for different meanings
- Shorter selectors mean fewer surprises with cascade

Count on inheritance and cascade
- Wrap similar element in a container
- Don't duplicate classes on sibling elements
Page flow

Recall: some elements take up full width
  <p>, <h1>

Others lay out left to right
  <a>, <strong>

Block vs. inline
  block: has width and height, defaults to full width
  inline: can't set width or height, can't have block children

Exceptions
  <img>: can be sized, but it's inline
display CSS property

Override default flow

Values: block, inline,

  none: hide the element completely
  inline-block: inline, but has width/height (like <img>)

<div> and <span>

  <div>: generic block element
  <span>: generic inline element

No semantics or default presentation
Useful for custom styling, layouts
...But don't use when more precise element exists
Summary

Today
  CSS, styling elements

Before next time
  assign1
  assign2 will go out this weekend, due Tue May 2

Next time
  Page layout, box model, flexbox