HTML and CSS

Michael Chang

Spring 2020

Announcements

Enrollment: please reply ASAP if enrollment delay

Avoid private chat questions in lecture

- Not trying to ignore you, I literally can't read them

Assign0 posted, due next Monday

- Updated instructions for Mac (thanks Kashif for troubleshooting)

References for HTML and CSS

- Will make a page for them

Plan for today

Last time: intro to web, HTML

- Assorted elements and topics

HTML structure

- Parts of a page, boilerplate, some useful elements
- Asides: standards, debugging

Intro to styling and CSS

- Selecting things to style
- Useful style attributes
- Separate CSS files
- Block vs. inline

HTML

Hypertext Markup Language

Not a programming language

Overall structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>...</title>
  </head>
  <body>
    ...
  </body>
</html>
```

head

Information about the page, not displayed

`<meta>`: page metadata

Author, search engine keywords, character set

`<title>`: title shown in tab/taskbar

`<script>` and `<link>`: refer to other files

Discussed later

Some elements

<h1>, <h2>, ..., <h6>: headings

<p>: paragraph

: line break (leaf)

<a>: link (different from <link>!)

<img>: image

: content is important (bold)

: content should be emphasized (italics)

: bulleted (unordered) list

: numbered (ordered) list

: list items (only inside <ul> and <ol>)

Semantic elements

Elements serve two purposes: semantics and presentation

Semantics: how elements behave

- Show images
- Click and tab between links and buttons
- Other cool things with certain tools

Presentation: how elements look

- Headings bold and large
- Links blue and underlined

Can override presentation easily

- Harder (and more confusing) to override semantics

Aside: (mostly) obsolete elements

- <b> for bold, <i> for italics, <u> for underline, to center text

Don't use these; they were designed for presentation only

- Some have been repurposed, but generally not used anymore anyway

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;

`&` (&), `<` (<), `>` (>)

A bunch more, but often can just put the char

Handling of "custom" (bogus) elements and attributes

No warning, just displays them

Graceful degradation

Browsers don't complain when they encounter issues

- Unrecognized elements, attributes

Design principle: best-effort rendering

- Newer pages won't break on older browsers
- They might look bad or behave incorrectly)

Is this good or bad?

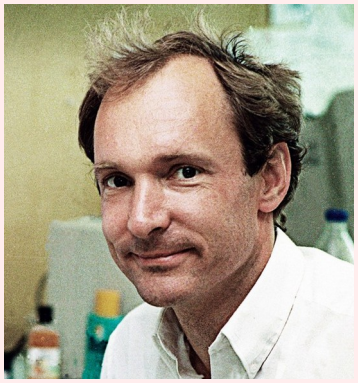
- Hard to say...

The nerdy, mostly* accurate backstory for HTML today

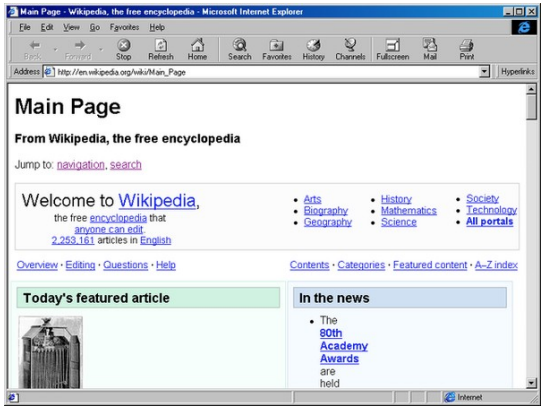
Victoria Kirst, Spring 2017

*I would be more accurate, but it's hard to get valid sources online... so I'm going off of what I can + the lore I've heard while working on a browser.

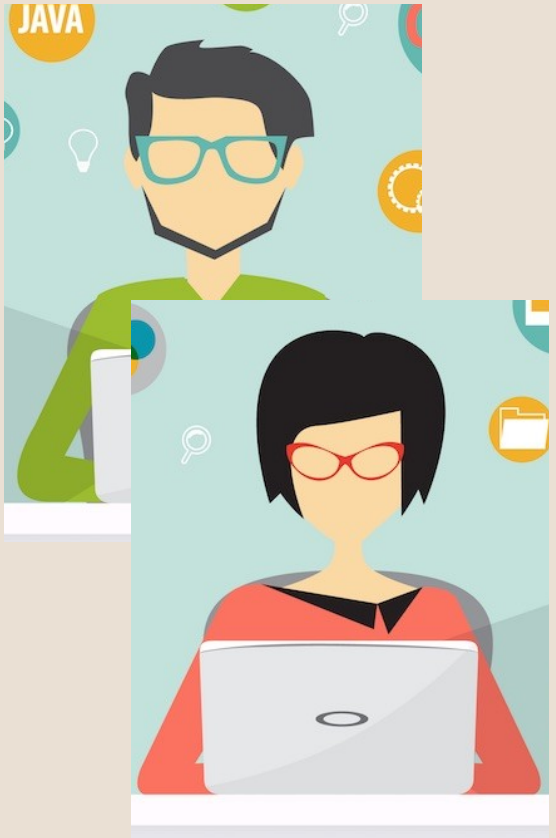
State of the world, 1997:



Standards say one thing,

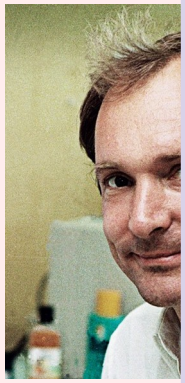


Browsers do another thing,



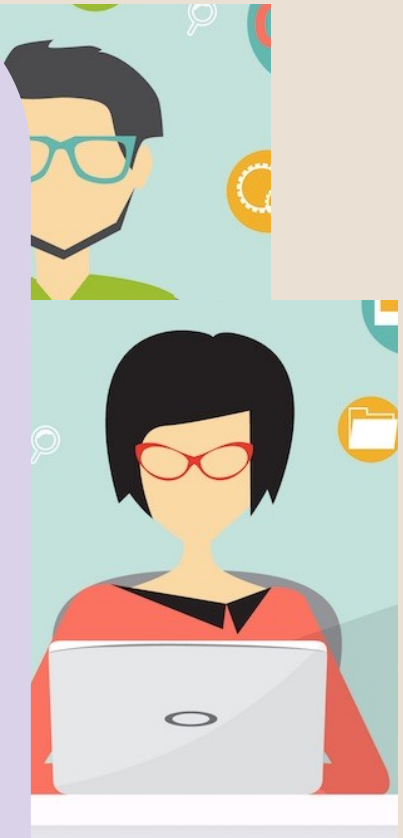
Developers write weird, non-standard code.

State of the world, 1997:



Standard
one thing,

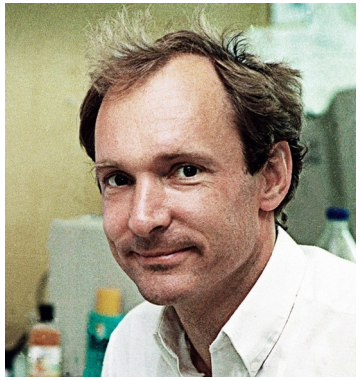
another thing,



lopers write
weird, non-
standard code.

In 1997, things are kind of a mess!

2000ish:



W3C

Oops, that "degrading gracefully" thing was a mistake.

Browsers, stop rendering pages that have invalid HTML.

(This was the proposal of XHTML 1.1)

2000ish: (not totally accurate)

W3C

What?!?!

Sure whatever

That would break the internet!

We can't do that!

2004: WHATWG formed

Let's burn everything and start from scratch with XHTML 1.1
(break approx. 64 million websites)

WHATWG

Let's work on HTML5
(an imperfect but realistic standard)

Fast forward 2017?!

- W3C gave up XHTML 1.1 in 2007
- W3C and WHATWG are mostly friends (I think), though they are still separate entities

So far

Last time: intro to web, html

- Assorted elements and topics

HTML structure

- Parts of a page, boilerplate, some useful elements
- Asides: standards, debugging

Intro to styling and CSS

- Selecting things to style
- Useful style attributes
- Separate CSS files
- Block vs. inline
- Box model

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

Separate .css files

<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]`

CSS colors

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

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

.class: elements with specified class attr

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

#id: element with specified id attr

```
<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 and inheritance

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

- Wrap similar element in a container
- Don't duplicate classes on sibling elements

So far

Last time: intro to web, HTML

- Assorted elements and topics

HTML structure

- Parts of a page, boilerplate, some useful elements
- Asides: standards, debugging

Intro to styling and CSS

- Selecting things to style
- Useful style attributes
- Separate CSS files
- Block vs. inline

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>`)

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: HTML and CSS

- A bunch of elements and properties
- For more, check out the references
- ...But don't stress about committing them to memory

Before next time

- (Hopefully) finalize enrollment
- assign0

Next time

- Box model
- Flexbox: a new era of `display`