Intro Web Development

Spring 2024

Web Dev in a sentence

Software development using the web as a deployment and execution platform

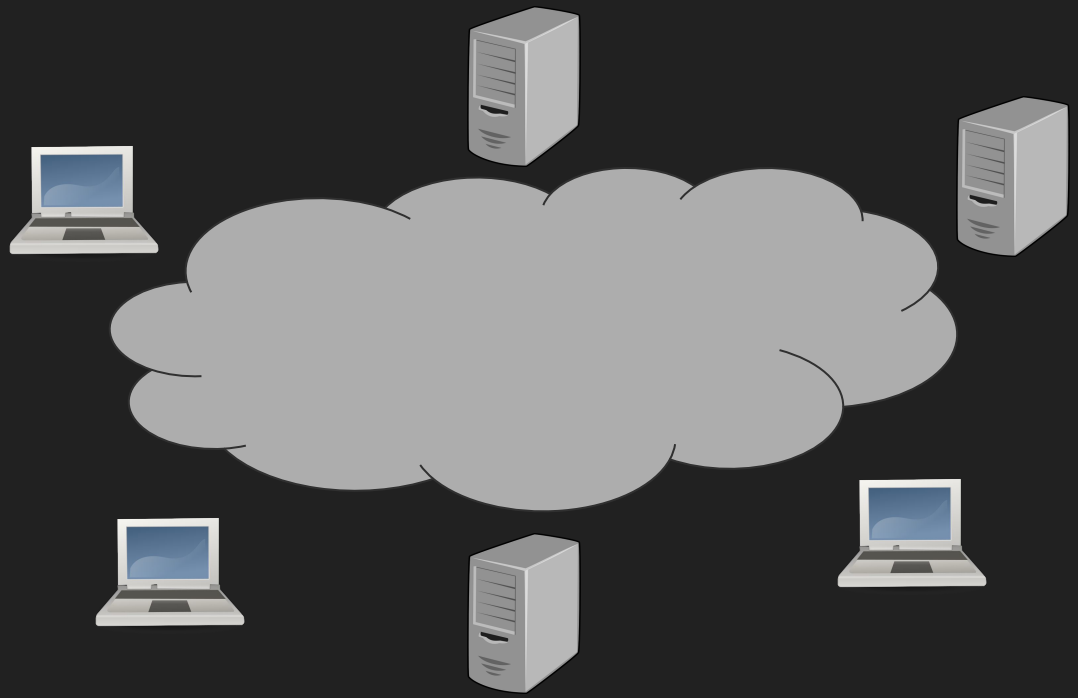
- Key: you use a web browser to interact with the created artifact

Web dev doesn't dictate the WHAT, it dictates the HOW

To understand the HOW, we need some history

- Helps explain why web dev is the way it is today

The Internet

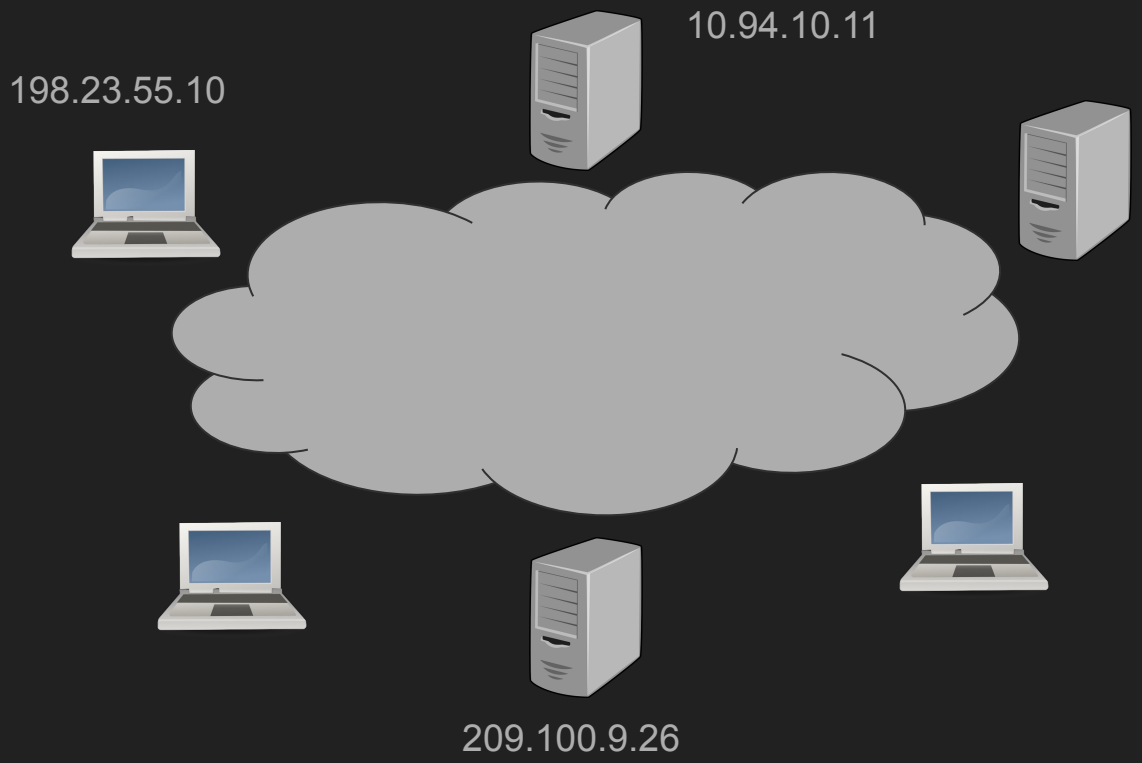


Set of computers connected via a network

Network lets computers send messages

Each computer has an address so that you can direct messages to it

The Internet

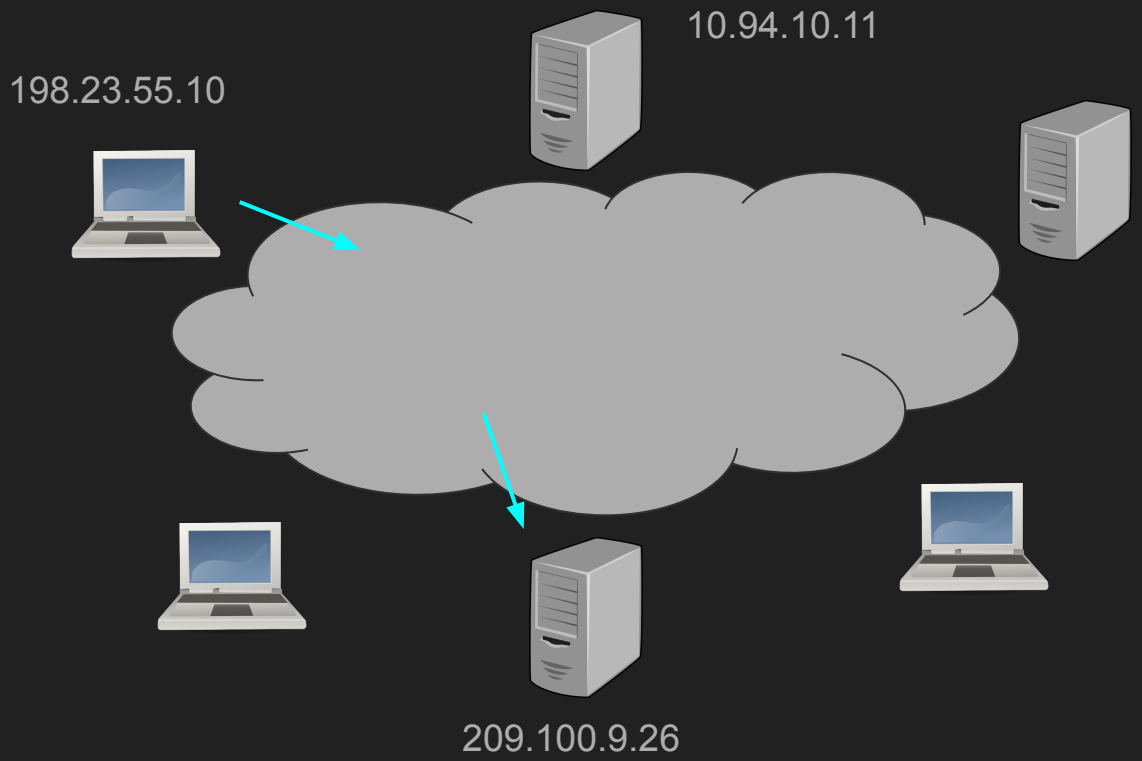


Set of computers connected via a network

Network lets computers send messages

Each computer has an address so that you can direct messages to it

The Internet

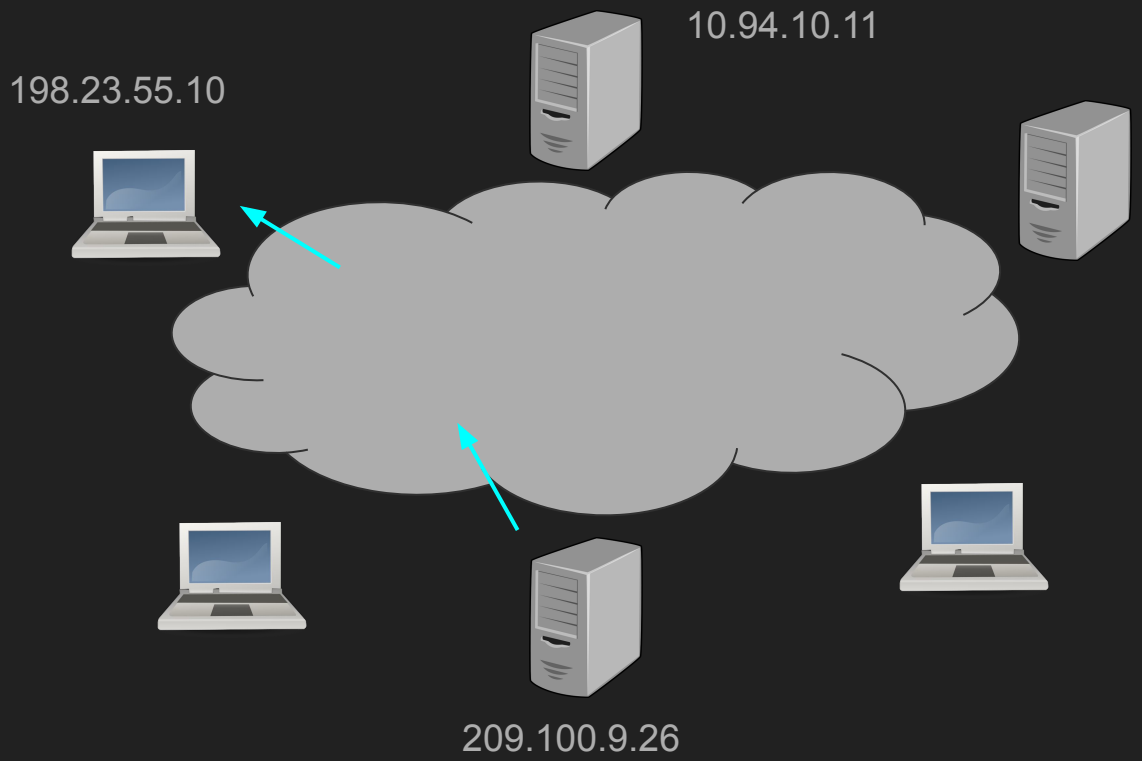


Set of computers connected via a network

Network lets computers send messages

Each computer has an address so that you can direct messages to it

The Internet

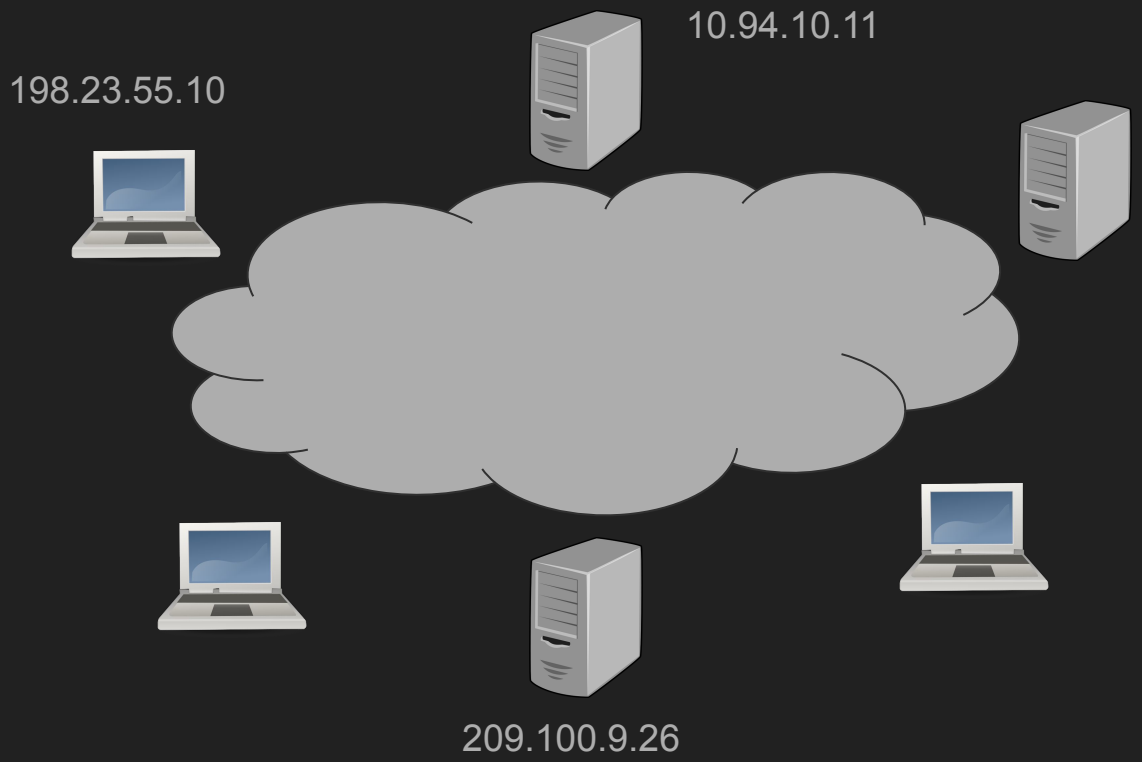


Set of computers connected via a network

Network lets computers send messages

Each computer has an address so that you can direct messages to it

The Internet



Many different kind of messages can be exchanged by various applications

Each defined by a protocol (message format, etc)

- ping
- FTP
- SSH
- VNC
- Web (= HTTP)

The World Wide Web

A grandiose name — the initial idea (1990s):

- A computer could make a set of files containing text + pictures available
- Those files could contain links to other files possibly on other computers
    - Hypertext - think Wikipedia

A *web server* makes those files available to everybody

A user would use a special application (*web browser*) to access those files using the address of the computer and the location of the file on that computer

- browser understand links and when you click on one, you could fetch the linked file and display it
- web browsers request files from web servers using HTTP

The World Wide Web

A grandiose name — the initial idea (1990s):

- A computer could make a set of files containing text + pictures available
- Those files could contain links to other files possibly on other computers
  - Hypertext - think Wikipedia

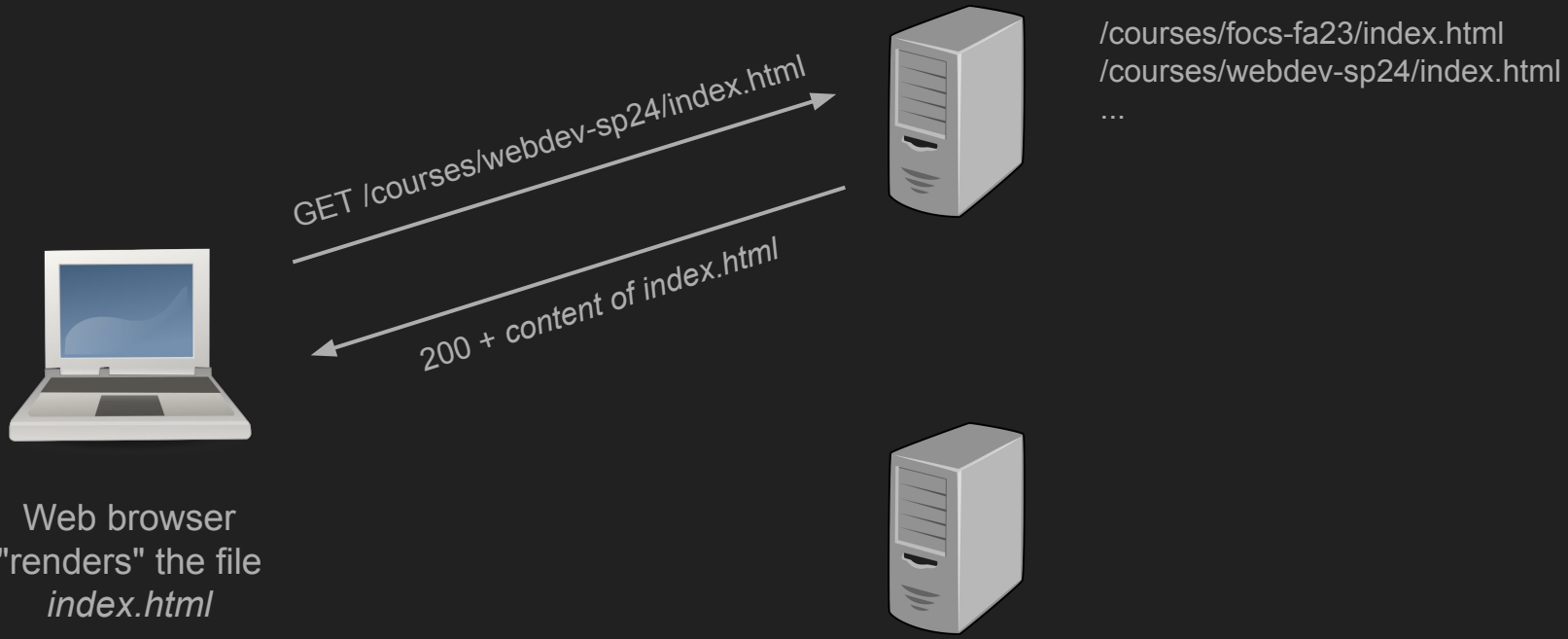
A *web server* makes those files available to everybody

A user would use a special application (*web browser*) to access those files using the address of the computer and the location of the file on

- browser understand links and when you click on one, fetch the linked file and display it
- web browsers request files from web servers using HTTP

HyperText Transfer Protocol

The World Wide Web



Hypertext

The web was initially designed to exchange hypertext

Web browsers are applications that can display hypertext files smartly

- format of hypertext = HTML
- basically, structured text
- got a lot more complicated since initial v1.0
- structured text broken down into a tree of elements
- browser "renders" the tree using a fairly complicated algorithm that only a few people at Mozilla and Google genuinely understand

Hypertext

The web wa

Web brows

- format
- basical
- got a lo
- structu
- browse
  people

```
<html>
  <body>
    <p> This is a paragraph with a <b>bold</b> word </p>
    <ul>
      <li> Item 1 </li>
      <li> Item 2 </li>
    </ul>
    <p> This is a <a href="http://google.com">link</a> </p>
  </body>
</html>
```

Great — then what?

That was good enough for a short while

- web browsers get and render HTML files from web servers = everybody happy

People wanted more "dynamic" behavior — that's where things get fun

Two distinct ways to add more dynamic behavior:

(1) make the HTML file sent to the browser more dynamic
(2) make the generation of the HTML page on the server more dynamic

Dynamicity on the browser side

A web browser gets an HTML file

- transforms it into a tree
- renders the tree

Idea: create a scripting language for the browser

- add code to the web page to, e.g., react to events (clicks, hovers, keypresses) and do something to the tree in response

That scripting language evolved into the JavaScript of today

- An HTML file creates a tree that becomes an "active" artifact
- A web browser is an HTML renderer + a JavaScript interpreter

Dynamicity on the browser side

A web browser gets an HTML fi

- transforms it into a tree
- renders the tree

Idea: create a scripting languag

- add code to the web page keypresses) and do somet

That scripting language evolve

- An HTML file creates a tre
- A web browser is an HTML

Extreme picture:

Send an "empty" HTML file, with code to create the tree programmatically on the browser!

Dynamicity on the server side

Initial picture:

- Web server delivers existing HTML files from the file system
- Good for static information, not so much when information changes
- E.g., student records
  - one HTML file per student?
  - make a change manually or pragmatically to a file when something changes? Urgh…

Better solution:

- Store student records in a database
- Generate the HTML file in response to a browser request

Dynamicity on the server side

Classical Web Servers (Apache, NGINX)

- use an external program to generate HTML files dynamically (CGI)
  - this includes PHP

Web Application Servers (templates, or generate in code):

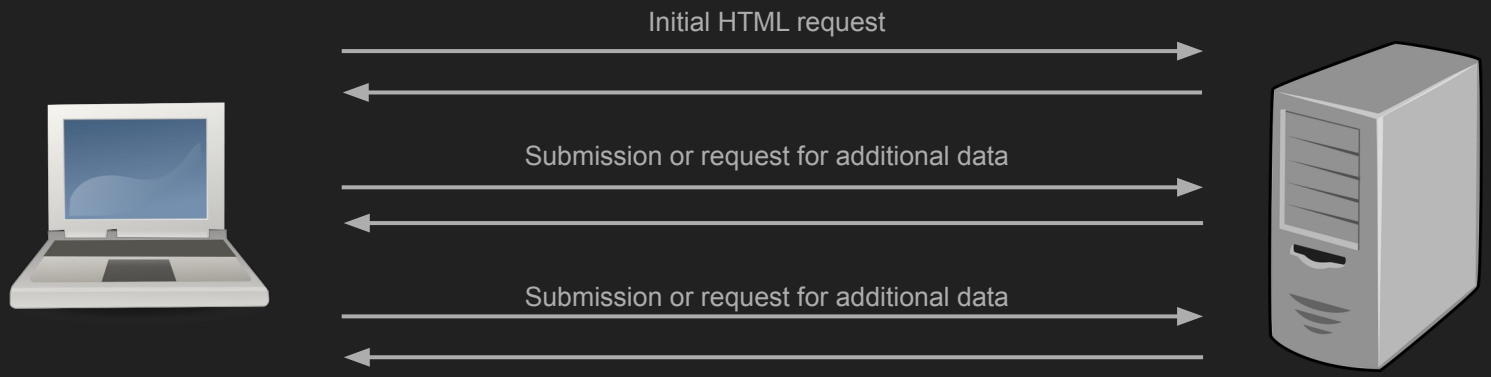
- Ruby / Rails
- Python / Django, Python / flask
- Java / Spring
- .NET / ASP
- JavaScript / Express

Dynamicity über Alles

Web Application servers let you send more general data than just HTML files

Javascript can make calls to a Web Application server after an HTML file has rendered

⇒ Modern web app (including SPAs)



Learning path

We're going to look at all the models above, and gets our hands dirty with pretty much all of them

- HTML / CSS
- JavaScript
- Frameworks for coding in JavaScript more reasonably (React, ...)
- Web application servers (NodeJS, Python / Flask, ...)
- Data persistence (databases)

Course work

Regular homeworks

- mostly getting our hands dirty with what we've seen
- idea for an ongoing web app that we'll be tinkering with as the course progresses

Final project (2nd half of semester)

- implement something you find interesting

Teams of 2

HTML Overview

HTML

A notation for "structured" text documents with links to other documents

- The basis of web content — or the skeleton

The small print:

- HTML = HyperText Markup Language
- An instance of SGML (Standard Generalized Markup Language) like DocBook and others
- SGML is an ancestor (uncle?) of XML, yet another markup language
- You still encounter XML in Java and .NET — rest of the world has moved to JSON
- We're on version 5 of HTML called HTML 5.
- The history of HTML is interesting and frustrating
- Compatibility after 30 years — the first web pages mostly still render

The basics

```
<html>
  <head>
    <title> Best Web Page Evah! </title>
  </head>

  <body>
    <!-- this is the part that actually gets rendered by browsers -->
    <h1> Sample HTML </h1>
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
    <p> Phasellus ipsum tellus, <em>malesuada</em> efficitur venenatis ut, aliquam at ipsum. </p>
    <ol>
      <li> Fusce consequat tellus nec diam aliquet, sit amet viverra dolor feugiat. </li>
      <li> Etiam nec euismod mauris, et elementum purus. </li>
    </ol>
    <img src="cat.jpg">
  </body>

</html>
```

The basics

```
<html>
  <head>
    <title> Best Web Page Evah! </title>
  </head>

  <body>
    <!-- this is the part that actually gets rendered by browsers -->
    <h1> Sample HTML </h1>
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
    <p> Phasellus ipsum tellus, <em>malesuada</em> efficitur venenatis ut, aliquam at ipsum. </p>
    <ol>
      <li> Fusce consequat tellus nec diam aliquet, sit amet viverra dolor feugiat. </li>
      <li> Etiam nec euismod mauris, et elementum purus. </li>
    </ol>
    <img src="cat.jpg">
  </body>

</html>
```

Tags describe pieces of the document
(via an opening tag and closing tag)

The basics

```
<html>
  <head>
    <title> Best Web Page Evah! </title>
  </head>

  <body>
    <!-- this is the part that actually gets rendered by browsers -->
    <h1> Sample HTML </h1>
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
    <p> Phasellus ipsum tellus, <em>malesuada</em> efficitur venenatis ut, aliquam at ipsum. </p>
    <ol>
      <li> Fusce consequat tellus nec diam aliquet, sit amet viverra dolor feugiat. </li>
      <li> Etiam nec euismod mauris, et elementum purus. </li>
    </ol>
    <img src="cat.jpg">
  </body>

</html>
```

Some tags have attributes
(think of them as tag parameters)

The basics

```
<html>
  <head>
    <title> Best Web Page Evah! </title>
  </head>

  <body>
    <!-- this is the part that actually gets rendered by browsers -->
    <h1> Sample HTML </h1>
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
    <p> Phasellus ipsum tellus, <em>malesuada</em> efficitur venenatis ut, aliquam at ipsum. </p>
    <ol>
      <li> Fusce consequat tellus nec diam aliquet, sit amet viverra dolor feugiat. </li>
      <li> Etiam nec euismod mauris, et elementum purus. </li>
    </ol>
    <img src="cat.jpg">
  </body>

</html>
```

Everything between an opening and closing tag is the content of the tag

The basics

```
<html>
  <head>
    <title> Best Web Page Evah! </title>
  </head>

  <body>
    <!-- this is the part that actually gets rendered by browsers -->
    <h1> Sample HTML </h1>
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
    <p> Phasellus ipsum tellus, <em>malesuada</em> efficitur venenatis ut, aliquam at ipsum. </p>
    <ol>
      <li> Fusce consequat tellus nec diam aliquet, sit amet viverra dolor feugiat. </li>
      <li> Etiam nec euismod mauris, et elementum purus. </li>
    </ol>
    <img src="cat.jpg">
  </body>

</html>
```

Everything between an opening and closing tag is the content of the tag

The basics

```
<html>
  <head>
    <title> Best Web Page Evah! </title>
  </head>

  <body>
    <!-- this is the part that actually gets rendered by browsers -->
    <h1> Sample HTML </h1>
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
    <p> Phasellus ipsum tellus, <em>malesuada</em> efficitur venenatis ut, aliquam at ipsum. </p>
    <ol>
      <li> Fusce consequat tellus nec diam aliquet, sit amet viverra dolor feugiat. </li>
      <li> Etiam nec euismod mauris, et elementum purus. </li>
    </ol>
    <img src="cat.jpg">
  </body>

</html>
```

That content may itself be made up of others tags with their own content

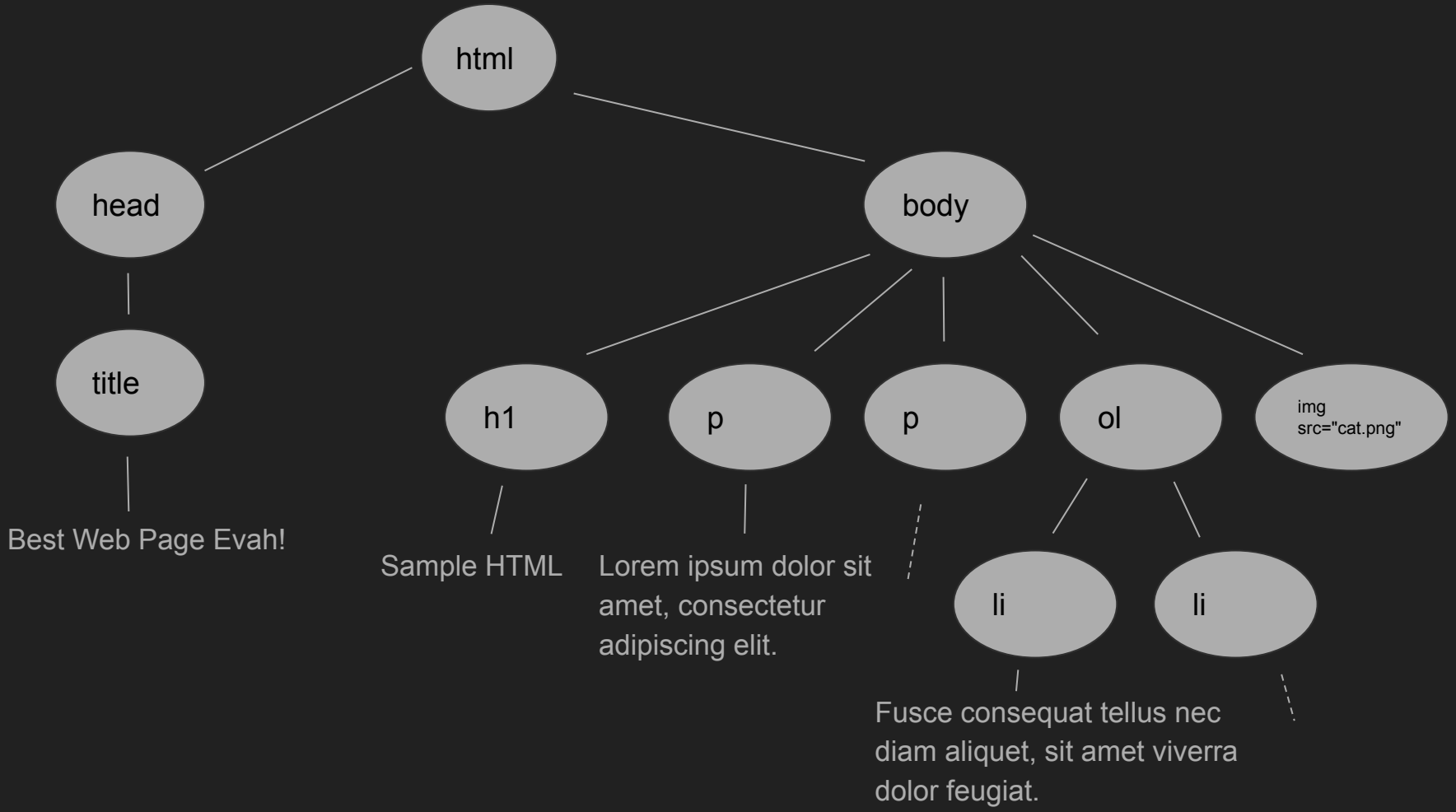
The basics

```
<html>
  <head>
    <title> Best Web Page Evah! </title>
  </head>

  <body>
    <!-- this is the part that actually gets rendered by browsers -->
    <h1> Sample HTML </h1>
    <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
    <p> Phasellus ipsum tellus, <em>malesuada</em> efficitur venenatis ut, aliquam at ipsum. </p>
    <ol>
      <li> Fusce consequat tellus nec diam aliquet, sit amet viverra dolor feugiat. </li>
      <li> Etiam nec euismod mauris, et elementum purus. </li>
    </ol>
    <img src="cat.jpg">
  </body>

</html>
```

Tags naturally define a tree structure for the document


html
head
body
title
h1
p
p
ol
img
src="cat.png"
Best Web Page Evah!
Sample HTML
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
li
li
Fusce consequat tellus nec diam aliquet, sit amet viverra dolor feugiat.

Categories of tags

Structural	<head>, <body>, <h1>, … , <h6>, <p>, ... ,
Formatting	, , <code>, <b>, <i>, <u>, <pre>, <sup>, <sub>, ...
List	, , , ...
Table	<table>, <thead>, <tbody>, <tr>, <th>, <td>
Metadata	<title>, <style>, <meta>, …
Embedded Content	<img>, <audio>, <video>, <canvas>, <svg>, <iframe>, <object>, ...
Form (controls)	<button>, <input>, <select>, ...
Scripting	<script>, ...

Entities

How do we handle "special" characters?

 This is an inequation : x < y

Entities let you escape characters that have meaning in HTML: <, >, &

 This is an inequation : x < y

Lots of entities defined, including mathematical symbols: ∞

Can use an arbitrary unicode characters: ✂

HTML → DOM tree representation

Web browsers transform HTML into a tree defined by the Document Object Model

Types of nodes in the tree:

- Element: mostly corresponding to tags
- Attributes: attributes in tags are treated as special children of elements
- Text: text inside tags

Content of a tag are children of the corresponding Element node

Browsers implement a specific rendering algorithm

Every node is either a block node or an inline node (depending on the tag — text is inline)

- block nodes are rendered vertically, one above the other
- children of a block node are rendered into sub-blocks (one per child), except that:
- adjacent inline nodes get merged into a single block

Intuition: paragraphs (blocks) of text (inline)

Layout follows the tree structure — e.g., A B <h2>C</h2> D

- layout algorithm defines a block for , within it there are three subblocks:
  - A B (the two inline nodes A and B form one block)
  - <h2>C</h2>
  - D (the one inline node forms a block by itself)

Things get more complicated when you add layout-affecting styling

Styling

Tags and the tree structure describe what the "content" of the page is

- what it means

In the olden days, the browser was free to decide how to render this content

It wasn't too long before people decided they wanted to control how things looked

- styling: how elements look
- that's the most intricate part of HTML — we'll look at it next time

With the ability to change how arbitrary elements look and how they're laid out, the thrust to respect the meaning of tags dropped

- we'll see examples of that later in React

Your first homework

Send me an individual email with:

- your preferred name
- your pronouns
- your background in webdev if any:
  - do you know HTML/CSS?
  - any javascript experience?
  - any server-side experience (and if so, what frameworks)?
  - example of something you've done (if applicable)

There are no wrong answers — I just want to know roughly where everybody is