Introduction: The Web, HTML and CSS

CPEN 322 - Building Modern Web Applications - Winter 2021-1

Karthik Pattabiraman

The University of British Columbia
Department of Electrical and Computer Engineering
Vancouver, Canada

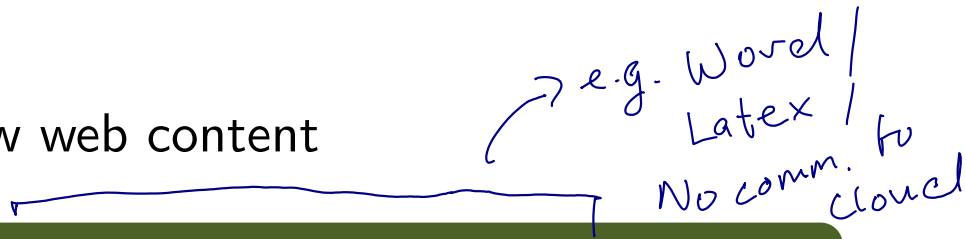
Electrical and Computer Engineering

September 9, 2021

Web Applications: what are they ?

Web Application: Definition

- A client-server software application in which the client (or UI) runs inside a web browser
- What's a client-server application ?
  - Distributed between 2 machines, client and server
- What's a web browser ?
  - Software application to view web content



Web Application: Differences from Desktop Applications

- Web Applications need a browser connected to the web all the time (for the most part)
- Web applications do their processing and storage at both the client and the server
- Web applications do not require cumbersome software installation (on the client)

Why Web Applications ?

- Ease of installation and use → No installation if browser intuitive → No platform specific UI
- Standard UI and interface across platforms
- Service provider retains control of code distribution. No need to patch, upgrade etc. → economic
- Allow compute and data-storage intensive functionality to be offloaded to the server → e.g. google search on whole web
  - Cannot install web
  - Computation reqd enormous.
  - Computation by google servers instead

History of the Internet and the Web

1960's:-1980's The early days

Army research lab

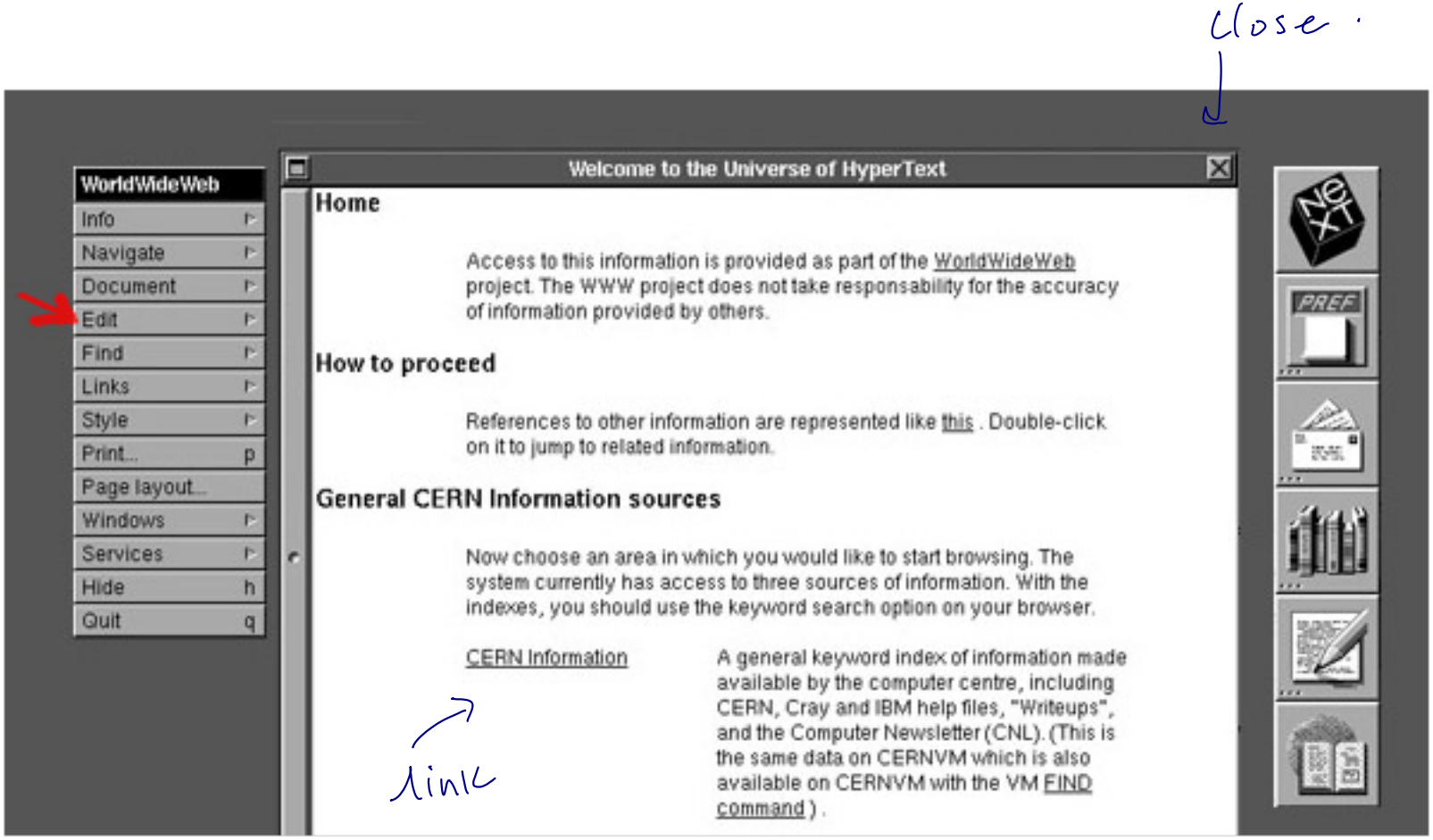
No fixed line, w any 2 path

- 1960s: Rise of the ARPAnet and packet switched networks at UCLA and DARPA
- 1970s: Development of TCP/IP protocol stack at Stanford, Univ. College London, BBN Tech.
- Internet in the 1980's:
  - TCP/IP becomes standard protocol on ARPANET
  - Tim Berners Lee invents Hyper Text Markup Language (HTML)

1985-1990: HTML and Web Browsers

- HTML: Way to embed hyperlinks within Documents to navigate to new documents
  - Invented by Tim Berners Lee at CERN for physicists to share documents in the late 80's
  - Extension of the SGML (Standard Generalized Markup Language) in the mid 1980's
  - Formalized as the HTTP protocol by Lee (1992)
  - Early web browsers were text-based e.g., Lynx
  - Competition: Gopher protocol, Archie client

Tim Berners Lee's Nexus Browser



- Source: Digital-archeology.org

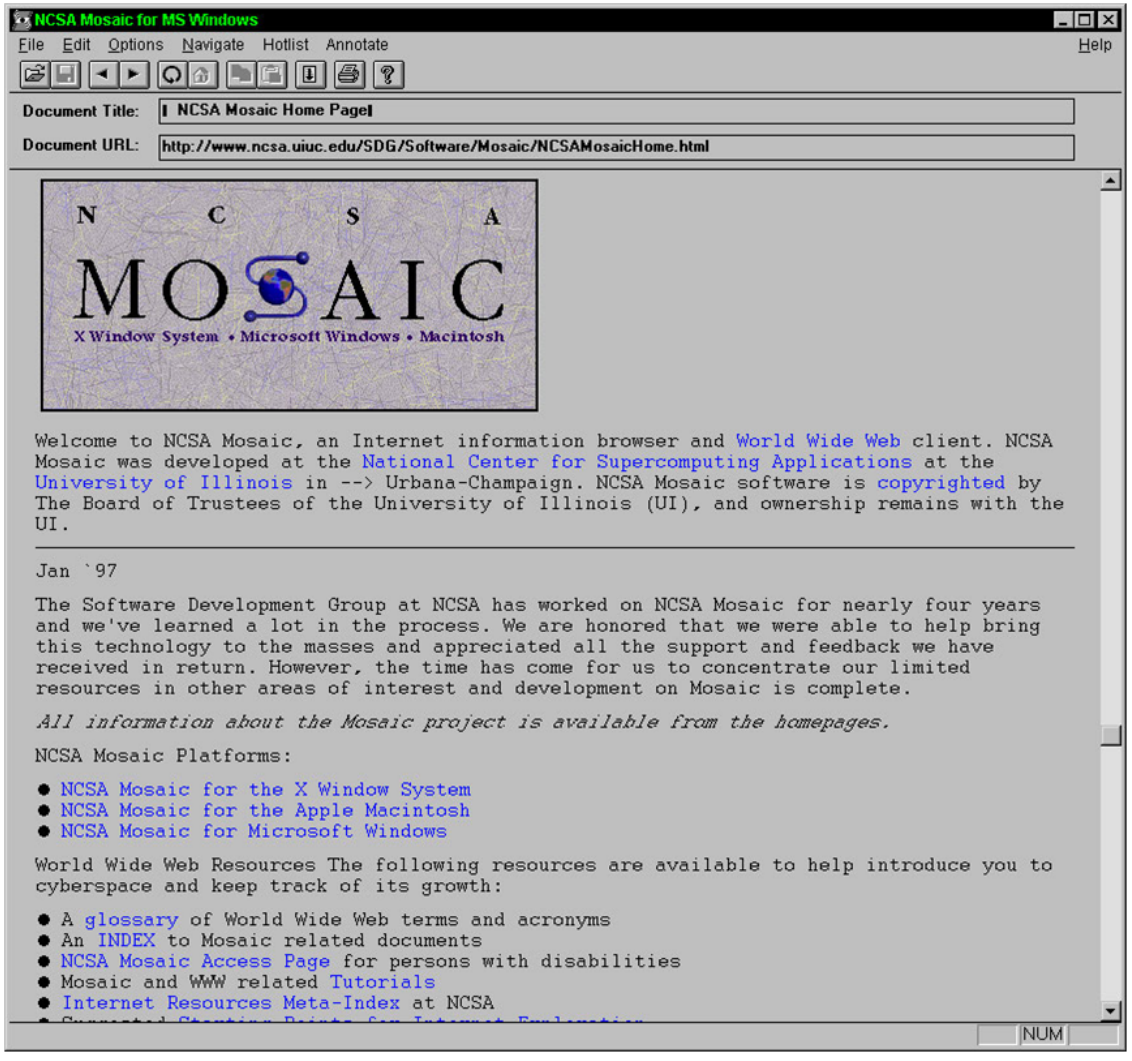
1990-95: Web Browsers

Influential development in web app

1/2

- Mosaic: One of the first graphical browsers developed by the Univ. of Illinois/NCSA → National Center for supercomputing applications
  - Led by Marc Anderson and Eric Bina in 1993 → UIUC did not allow
  - Marc and Eric left UIUC to start Netscape
  - Netscape Navigator released in 1994 → lots of "borrowed" code
  - UIUC licenses Mosaic code to SpyGlass Inc. which released Spyglass Mosaic
  - Microsoft acquires Spyglass. Releases it as IE 1.0 in 1995. Browser wars start !

Mosaic Web Browser: Screenshot



X Mosaic for free distr. and not commercialized (Netscape)

- Source: www.nsf.org

1995-2000: Browser Wars

- Web browser market was hotting up
  - Netscape Navigator Versus Internet Explorer → superior.
  - Netscape had more features and was more robust
  - IE had the marketing power of Microsoft and was bundled with Windows as a free download ! → More market share.
  - JavaScript 1.0 was developed and integrated with Netscape by Brendan Eich (more on this later)
  - In 1997, Microsoft released IE4 which had feature parity with Netscape and integrated with Windows
    - Netscape was never the same again. They were sold to AOL and essentially lost the browser wars by the early 2000s

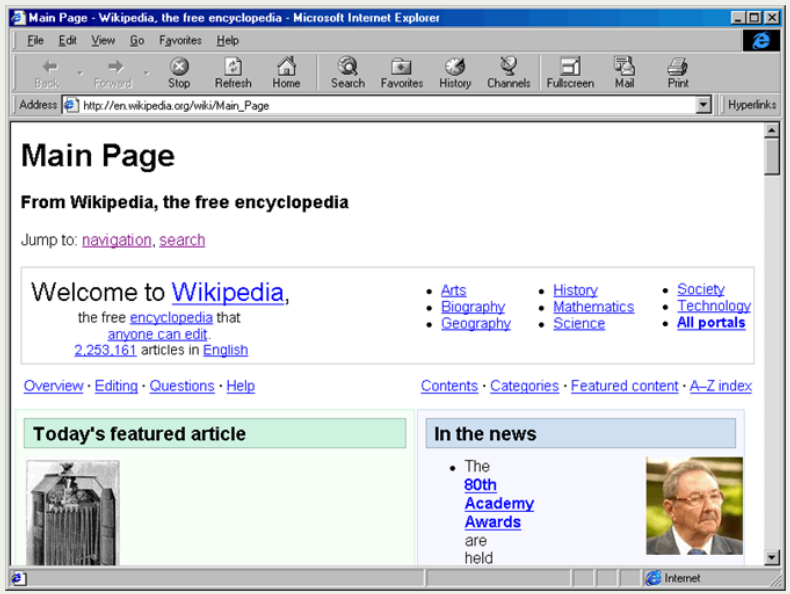
→ America online.

→ ISP.

Netscape Navigator 4 vs IE 4

Netscape Navigator 4

Internet Explorer 4



2000-2005: Browser Wars Part 2

- But, while Netscape was finished, they released their code to the Mozilla foundation
  - Team of dedicated volunteers that rebuilt Navigator from scratch and ironed out its quirks
  - First release in 2004. Rapid releases in 2005,2006
  - First browser that was standards compatible
  - Microsoft became complacent. Removed most core staff from the IE team and didn't develop it.
  - By 2006, Mozilla (Firefox) was back in the game

horrible

Won the browser war

2005-2009: The Rise of AJAX

→ Defines modern web apps

- AJAX (asynchronous JavaScript and HTML)
  - Feature introduced by Microsoft IE in their Outlook Web Access (OWA) client in the early 2000s
  - Google used it for Google maps and Gmail (2004), and suddenly it was all the rage (2005)
  - JavaScript became the new popular kid on the block, and JavaScript performance started to become increasingly important with Mozilla taking the lead
  - Google introduced Chrome in 2008 which was primarily about faster JavaScript execution and support – based on Webkit development engine

Corporate email client.

→ Repeatedly Poll thru AJAX req. → Dynamic update page.

"magic!"

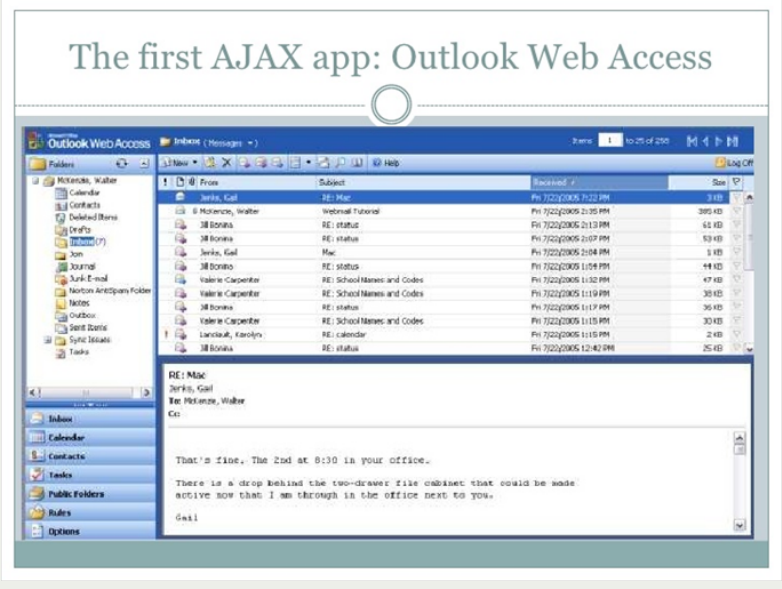
Completely in browser, apps w/ smooth experience - Done w/ AJAX

+ AJAX → Do cool things

↳ Shows other ppl methods for fastness to increase adoption

AJAX Applications

Outlook Web Access (OWA)



Google Search (AutoComplete)

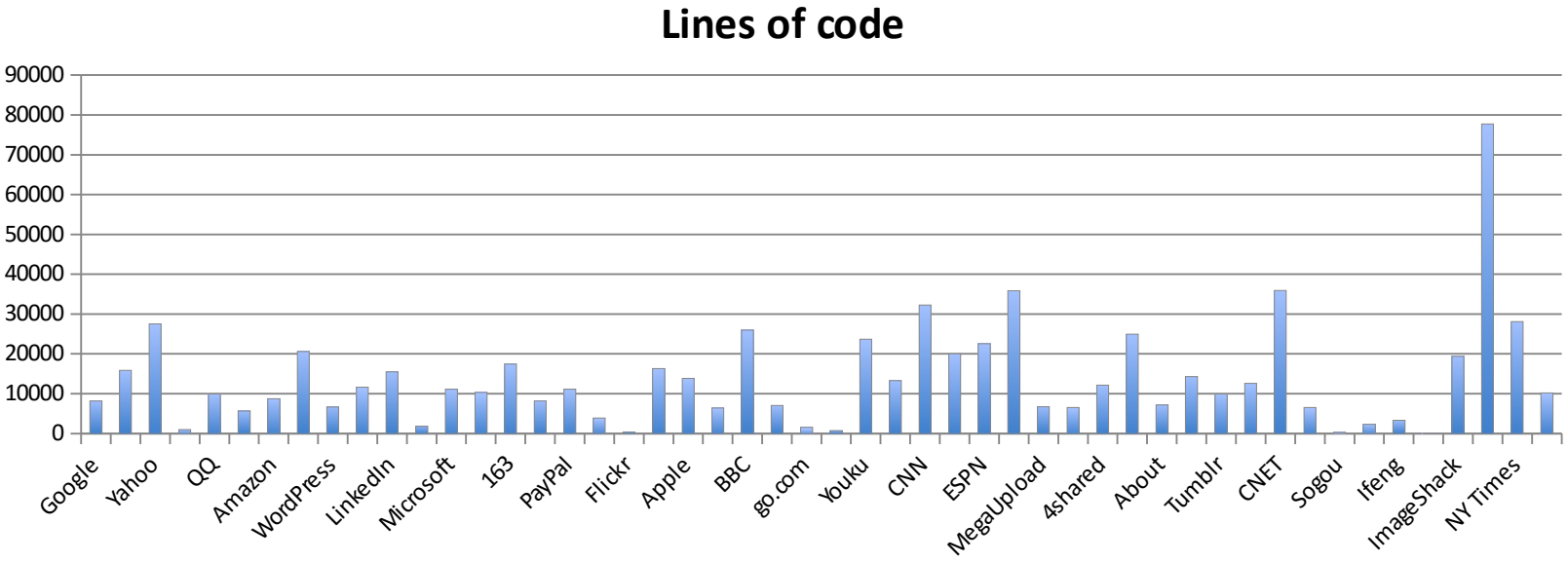
w/ autocomplete using AJAX.

2010-2015: JavaScript Everywhere

- JavaScript becomes mainstream
  - More and more webpages start heavily using JS, often with thousands of lines of minified code
  - New applications (e.g., Google docs, Office live etc.) and frameworks (e.g., Jquery)
  - Academic papers are written about JavaScript in terms of its performance, reliability, security → Dispel myths
  - JavaScript is the most popular language on Github, Stackoverflow and is in the top 5 on the Tiobe Index } Stronghold
  - JavaScript is also used to teach introductory CS by Khan Academy – often taught as the first language
  - Many variants of JavaScript: TypeScript (MS), DART (Google) and Flow (Facebook). Also, HTML5
  - Language for programming IoT devices (Samsung)

Prevalence of JavaScript (around 2012)

- 97 of Alexa top 100 websites use JavaScript
- Many of them have thousands of lines of code



Pure javascript, excl HTML/CSS

x10 nowadays

Anatomy of a Web Application

Web Application Components

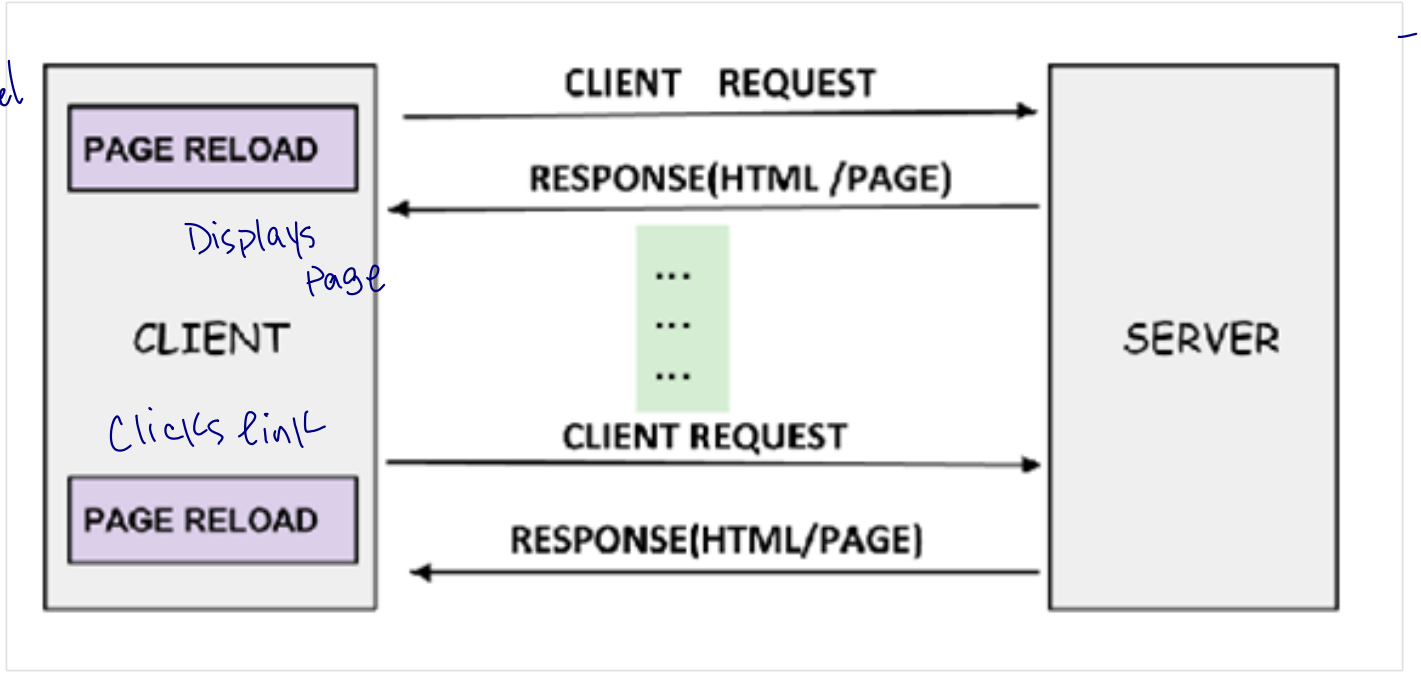
↳ traditional or modern.

- Three pre-requisites for a web application
  - Server: To "serve" the web-page and to send content to the client
  - Client: To receive content from the server and display them on the web browser window
  - HTTP connection for client-server interactions
- Everything else is optional.

↳ fancy JavaScript, Mongo DB, Cookies] barebones ver. not req'd

Traditional Web Applications

- Do not have (much) interactivity on the client
- Getting new content involves navigating to a new webpage or reloading the webpage / entire frame → See entire transition
  - Typically through a hyper-link
  - Limited information passing from the client (typically in the form of a URL with '?') → or forms (not as interactive)
- E.g., http://www.google.com?search="CPEN400A"

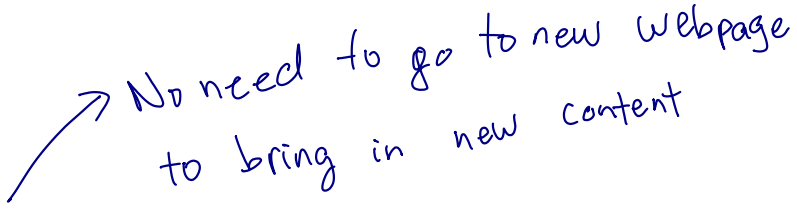


- Many pages still use model

- Modern
  - expand/shrink page w/o redraw.

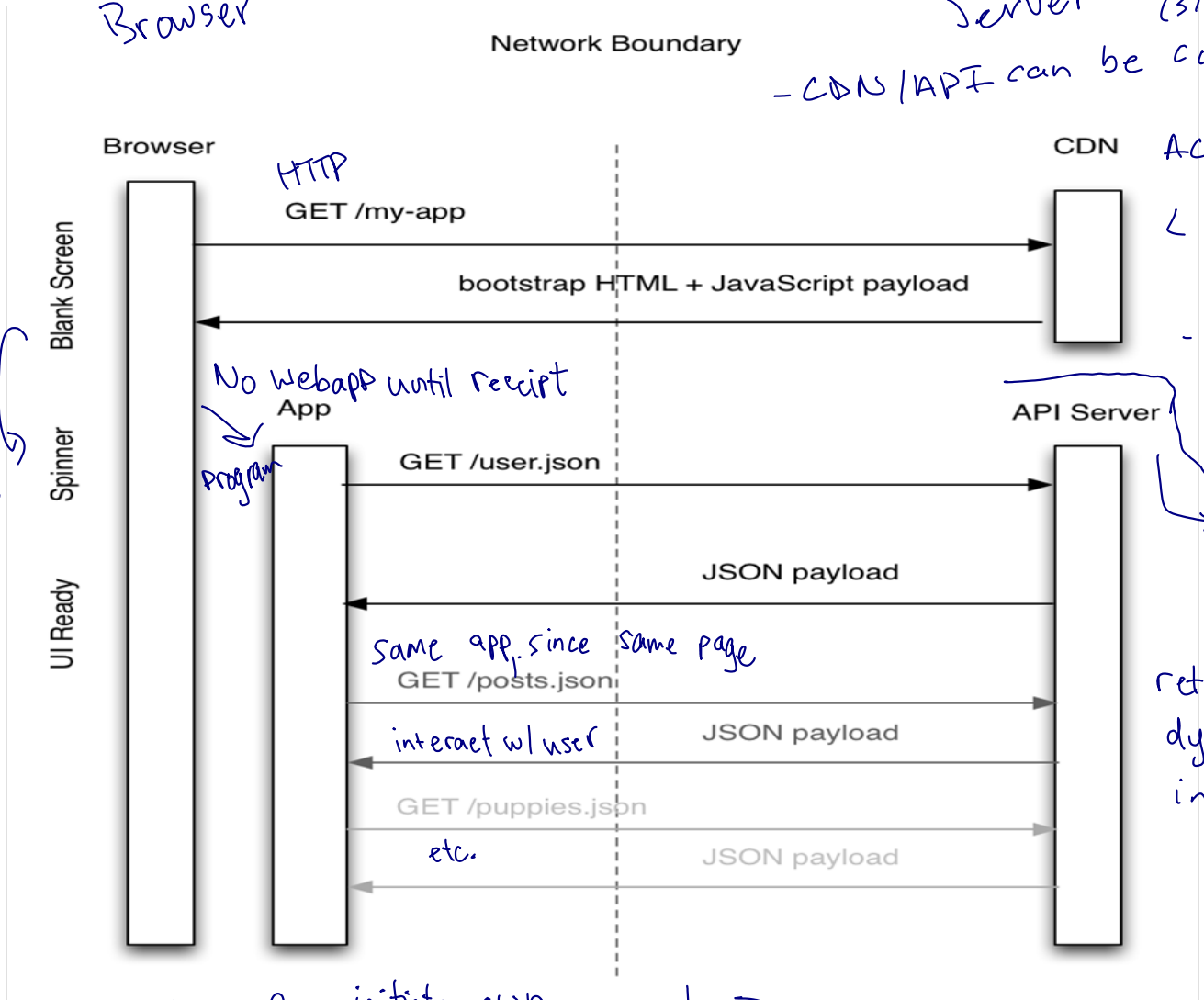
"clunky"

Modern Web Applications



- Client is a lot more interactive and actively renders content within the browser
- Application logic is split between client and server – client can execute (JavaScript) code
- No need to reload the web page for updating the state of the page being displayed (DOM)
- Rich message passing interface with the server through AJAX messages

Modern Web Applications (2)



client / Browser

Server (simple)
- CDN / API can be combined, typically sep. for large

ACAMI
(Cache static parts of page)
- content same for all in geographic area

Respond to AJAX req. returns any content dynamically brought in.

No webapp until receipt

Setup on client side

Waiting to setup

execute JS code

- 2 phase exec. model.

same app, since same page

App Can initiate own request.

Components of a Modern Web App

- Client-side components
  - HTML/DOM
  - CSS
  - JavaScript
- AJAX messages (client-server interactions)
- Server code (node.js or any other platform)

Client: HTML/DOM

- Hyper-text markup language to describe the structure and contents of the initial page
  - Also has pointers to the JavaScript code (e.g., <script>)
- Is retrieved by the browser and parsed into a tree called the Document Object Model (DOM)
  - Common way for elements to interact with the page
  - Can be read and modified by the JavaScript code
  - Modifications to the DOM are rendered by browser

-> uniform abstraction,

No HTML
Passing (trouble)
just DOM

- Accessible/modifiable by JS

- Instant
- JS <-Read / Writeback-> DOM

Client: CSS

- CSS (Cascading style sheets) separate the content of the page from its presentation
- Written in the form of declarative rules with a element on LHS and action to apply on RHS
- Ensure uniformity by applying the rule to all elements of the webpage in the DOM

* No need to individually style elements – write once and done

Client: JavaScript

- Unique to modern web applications and provides active functionality
- Executed when <script> tag is encountered or when events are triggered on DOM elements
- Is a full-fledged programming language with many advanced features (and some bad ones)

AJAX Messages

overlap I/O & Computation ← careful w/ parallel ops?

- Way for client and server to interact with each other without navigating to a new webpage
- Initiated by the client by sending a message asynchronously (activities can happen in parallel after send on the client) → Adv. & Disadv.
- Request is processed by server, and a response is returned. Response triggers a registered callback at the client.
- Call-back can do whatever it wants with the response, though typically it uses it to update the DOM

↑ Depend on JS.

Server

- Server can be written in whatever language one wants, as long as it serves the content and responds to AJAX requests from the client
- Typical languages used: Java, RoR, PHP etc.
- Recently, JavaScript is also being used to write server-side applications for uniformity and portability. Example of this is Node.js

→ Ruby on Raze

- can be non traditional → C based, C, C++
  ↳ CGI.

Other State Elements

X Covered

- Cookies ✓ some aspects
- Persistent local storage (HTML5 and variants)
- Server Database (e.g., MongoDB) ✓ some aspects
- Canvas (HTML5)

HTML: The Base Substrate of a Web App

HTML: What is it?

Primarily in this course

- Hierarchical way to organize documents and display them (typically in a web browser) + other ways too.
- Combines semantics (document structure) with presentation (document layout) e.g. Head, body, etc.
- Allows tags to be interspersed with document content e.g., <head> - these are not displayed, but are directives to the layout engine

↳ e.g. Word / Acrobat. -> traditionally one.
- structure external to content
- imposed in arbitrary way by program interpreting in
- Not portable. -> require program or file used to intersperse content
- HTML does not require ext. software to render.
  - popular — web browser; portability largest constraint

HTML: Example

head

```
<!DOCTYPE html>
<html>
  <head>
    <title>Photo Gallery</title>
  </head>

  <body>
    <div class="photo">
      <h3>My first photo</h3>
      <img src="picture1.jpg"/>
    </div>
  ...
  </body>
</html>
```

body

HTML: Head

- Is typically NOT displayed by web browser
- Contains metadata to describe the page
  - Title of the webpage: <title> TITLE </title>
  - Style of the webpage: <style> style rules </style>
  - Link to CSS stylesheets: <link rel="stylesheet" type="text/css" href="">
  - For search engines: <meta name="" content="">
  - Embed JavaScript: <script > Javascript code </script> OR <script src="Javascript file"></script>

HTML: Body

- Contains the actual contents of the page with HTML tag descriptors for the structure
- Common tags used in HTML

	group elements spanning multiple lines line break before and after
	group elements within a single line
<p>	new paragraph
 	line break

HTML: Body (2)

`<h1>,…,<h6>`	headings
`<img src=''>`	images
`<a href=''>`	hyperlink

`<table><tr><td>`	tables
`<ul><li>`	unordered list
`<ol><li>`	ordered list
`<form><input>`	forms that take in user input

The Curious "div" element

< Semantic implication >

- Div's are a way to separate different sections of a page and have no meaning by themselves
  - Used to group together semantically related elements in the same portion of the document
  - Allows semantic attributes such as CSS styling or JavaScript code to be applied to div elements
  - div elements can be nested within each other → hierarchial
- Use of div's allows easier rendering of pages, and adds semantic meaning to webpages (good)

- Performance penalty w/ overuse

Div element: Example

- Div element is used to group menu items
  - id can be used within JavaScript (JS) code to access it – must be unique to the element
  - class is used for indicating type of element – need not be unique, and is used for multiple elements in JS
  - background is a style to apply to all elements in div

not unique to divs though using div assoc. w/ group

Use CSS where possible

Sep.

```
1 <div id="menu1" class="Menu" style="background: #d3e7dd;">
2     <a href="index.html">Home</a> |
3     <a href="about.html">About Us</a> |
4     <a href="faq.html">FAQ</a> |
5     <a href="contact.html">Contact Us</a>
6 </div>
```

4 links.

Semantic commonality

The span element: Inline div

- span is an inline version of 'div', for separating small chunks of the document without a line break. Cannot contain other div elements in it
- Mostly for applying styling rules to small segments of the webpage without line-breaks

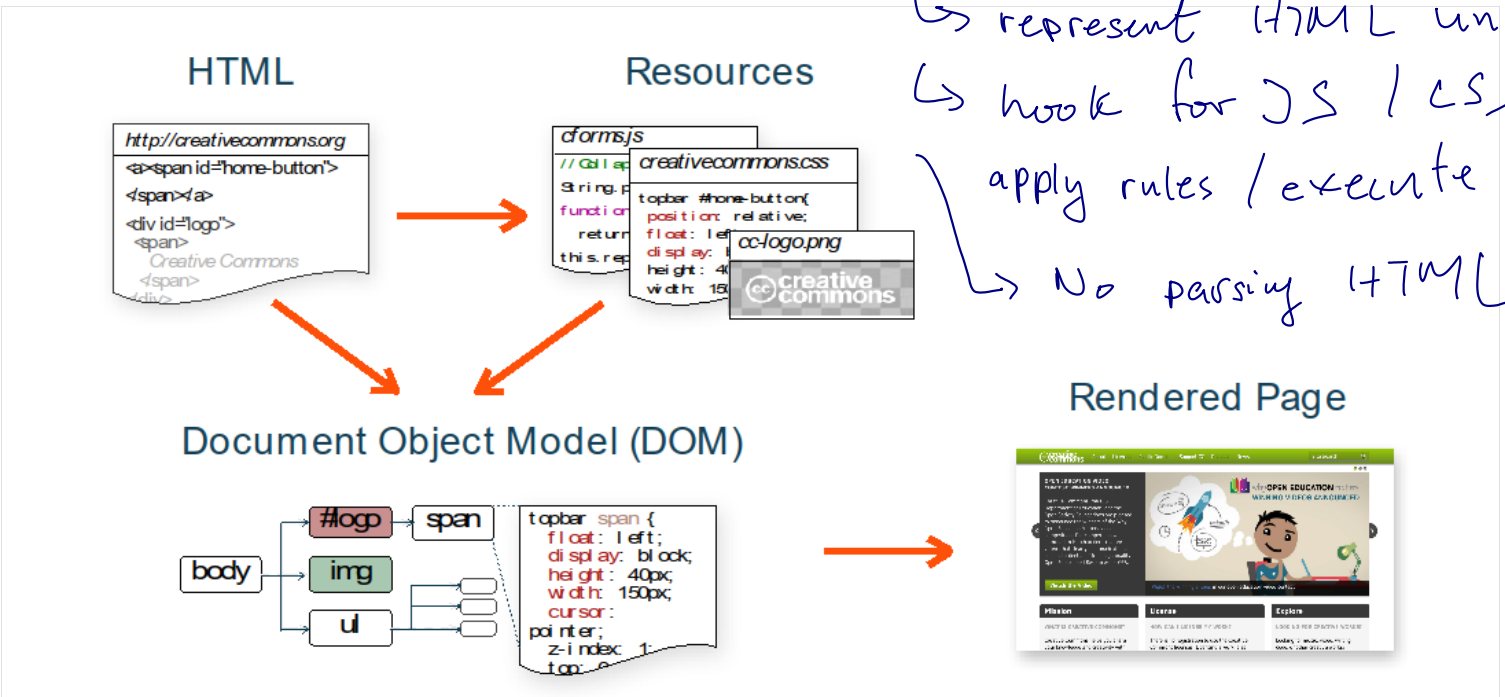
```
1 <div id="menu1" class="Menu" style="background: #d3e7dd;">
2     <a href="index.html">Home</a> |
3     <a href="about.html">About Us</a> |
4     <a href="faq.html">FAQ</a> |
5     <span class="italics"><a href="contact.html">Contact Us</
          a></span>
6 </div>
```

Why use Divs and Spans?

- Divs and spans are very useful to break a page into semantically related elements
  - Search engines like Google rely on these to find related information
  - Provide hooks to your webpage from CSS, and especially JavaScript code (more on this later)
  - Makes it easier to render across platforms
- However, overuse of these makes webpages hard to read, and also slower (Especially on mobile)

Browser's View of HTML: DOM

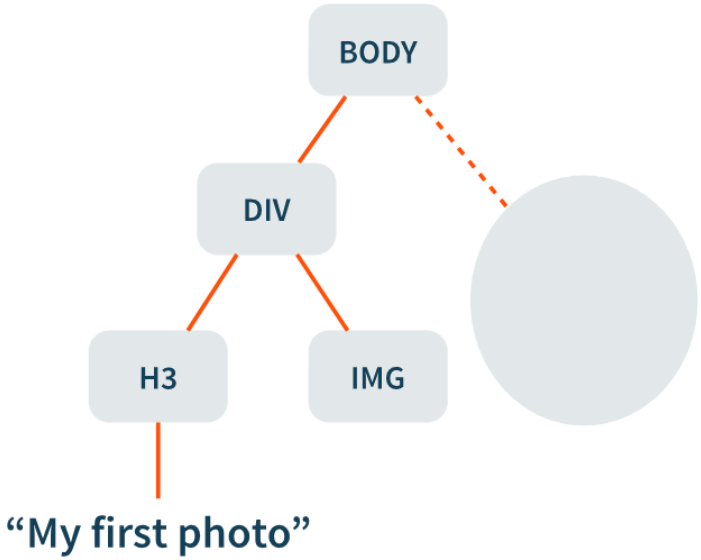
- HTML is parsed by the browser into a tree structure - Document Object Model (DOM) → Abstract syntax tree.
  - ↳ represent HTML uniformly
  - ↳ hook for JS / CSS to apply rules / execute code
  - ↳ No parsing HTML



DOM: Example

- Often one-to-one correspondence between HTML and the DOM rendered by browser

```
<body>
  <div class="photo">
    <h3>My first photo</h3>
    <img src="picture1.jpg"/>
  </div>
...
</body>
```



DOM: Why is it Important?

- Common data-structure for holding elements of a web-page (HTML, CSS, JavaScript etc.)
  - No need to worry about parsing HTML, CSS etc.
- Corresponds almost exactly to the browser's rendered view of the document
  - Changes to the DOM are made (almost) immediately to the rendered version of the webpage
  - Heavily used by JavaScript code to make changes to the webpage, and also by CSS to style the page

→ Periodic DOM refresh interval (by browser)

- Virtual DOM
  - basically the same

Virtual DOM → DOM → HTML

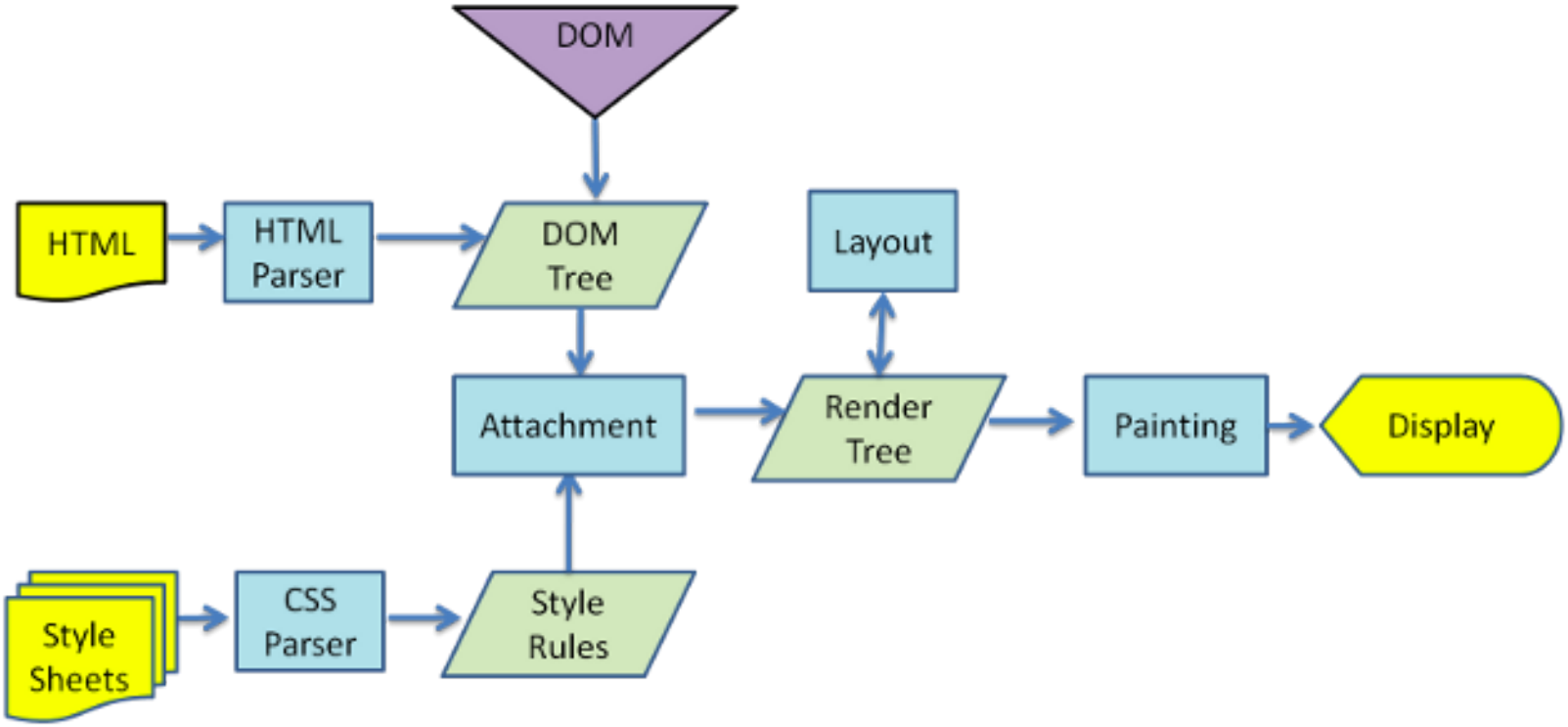
- if use frameworks

DOM Structure: Disadvantages

DOM state depends on initial + changes (by JS)
- reason about DOM state when you alter.

- No isolation between different parts of the DOM tree for a script as long as its from the same origin
  - All scripts from same origin (i.e., domain) can access the entire DOM tree from that origin
  - Scripts can clobber the DOM and leave it that way
  - Highly dynamic - difficult to reason about DOM state
- DOM is also very browser-specific (not standard)
- Can be a significant bottleneck in rendering webpages in parallel as it is a single global structure -> shared

What do Web Browsers do?



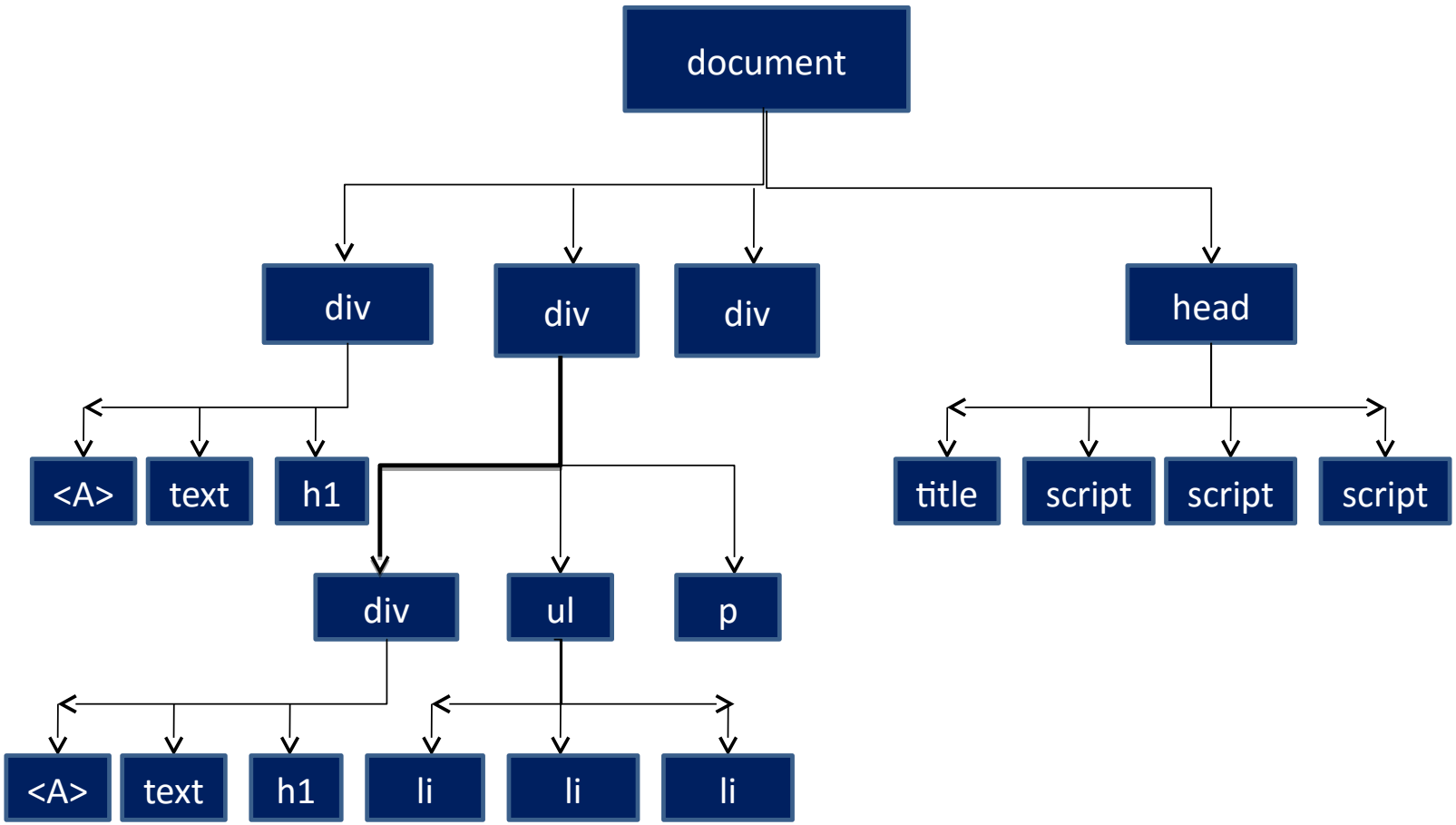
- Example from Webkit: Used by Chrome and Safari
- (source: http://www.html5rocks.com/en/tutorials/internals/howbrowsersworl

Class Activity

Draw the DOM tree corresponding to the following HTML code:

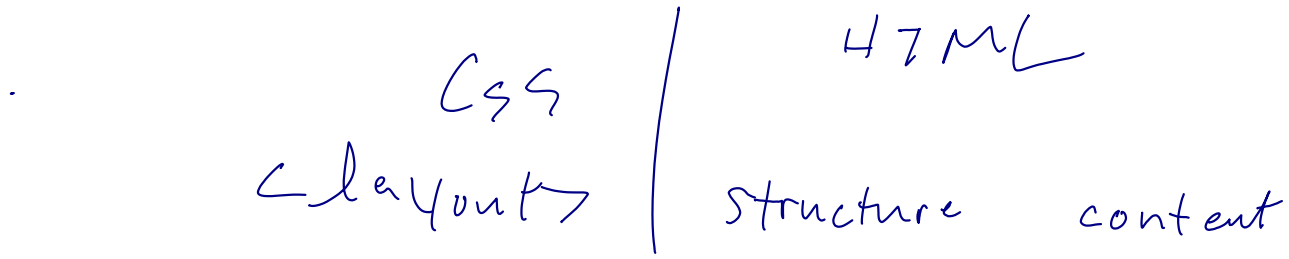
```
<html>
<head>
    <title> ... </title>
    <script> ... </script>
    <script> ... </script>
    <script> ... </script>
</head>
<body>
    <div> <A> ... </A> <text> ... </text> <h1> ... </h1> </
        div>
    <div>
        <div> <A> ... </A> <text> ... </text> <h1> ... </h1> <
            /div>
        <ul> <li> ... </li> <li> ... </li><li> ... </li> </ul>
        <p> ... </p>
    </div>
    <div> ... </div>
</body>
</html>
```

Class Activity - Solution



CSS: Syntax and Semantics

CSS: Philosophy and Motivation



- Language for specifying how (HTML) documents are presented to users (Separate from content)
- Declarative – set of rules and their actions
  - Makes it easy to modify and maintain the website
- Allows different rules to be specified for different display formats (e.g., printing versus display)

Including CSS in HTML: Example


```
1  <html>
2      <head>
3          <title>Sample document</title>
4          <link rel="stylesheet" href="style1.css">
5      </head>
6      <body>
7          <p>
8              <strong>C</strong>ascading
9              <strong>S</strong>tyle
10             <strong>S</strong>heets
11         </p>
12     </body>
13 </html>
```

link.

- (Anyone including their CSS code directly in the HTML?
  - I won't judge, but...)

~ Bad practice.

Example of CSS stylesheet (style1.cc)


```
1  strong {color: red;}
```

- strong: tag to match
- color: red;: attribute: value

```
1  <p>
2      <strong>C</strong>ascading
3      <strong>S</strong>tyle
4      <strong>S</strong>heets
5  </p>
```

⇒ Cascading Style Sheets

How does CSS work?

- Apply styles to the DOM tree of the web page
- CSS rule applies to DOM nodes matching tag, and their descendants (unless overridden)

```
P
├─STRONG
│ └─"C"
├─"ascading"
├─STRONG
│ └─"S"
├─"tyle"
├─STRONG
│ └─"S"
└─"heets"
```

Here, all STRONG tags will be matched ⇒ all descendants of STRONG tags will be *styled*.

CSS Inheritance

- All descendants of a DOM node inherit the CSS styles ascribed to it unless there is a "more-specific" CSS rule that applies to them
- Always apply style rules in top down order from the root of the DOM tree and overriding the rules as and when appropriate
  - Can be implemented with an in-order traversal

```
1 p {color:blue; text-decoration:underline}
2 strong {color:red}
```

```
1 <p>
2     <strong>C</strong>ascading
3     <strong>S</strong>tyle
4     <strong>S</strong>heets
5 </p>
```

⇒ Cascading Style Sheets

CSS Class and IDs

- CSS rules can also apply to elements of a certain class or an element with a specific ID

For a given class:

```
1 .key {
2   color: green;
3 }
```

For a given id:

```
1 #principal {
2   font-weight: bolder;
3 }
```

Example

```
1 <p class="key" id="principal">
```

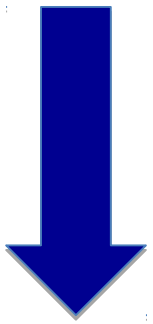
CSS Rules and Priority

- What to do when rules conflict with each other ?
  - Always apply the "most specific selector"
- "Most-specific" ('>' represents specificity):
  - Selectors with IDs > Classes > Tags
  - Direct rules get higher precedence over inherited rules (as before)

CSS Class and IDs: Example


```html
<!doctype html>
<html>
  <head>
  <meta charset="UTF-8">
  <title>Sample document</title>
  <link rel="stylesheet" href="style1.css">
  </head>
  <body>
    <p id="first">
      <strong class="carrot">C</strong>ascading
      <strong class="spinach">S</strong>tyle
      <strong class="spinach">S</strong>heets
    </p>
    <p id="second">
      <strong>C</strong>ascading
      <strong>S</strong>tyle
      <strong>S</strong>heets
    </p>
  </body>
</html>
```

```css
strong { color: red; }
.carrot { color: orange; }
.spinach { color: green; }
#first { font-style: italic; }
```



Cascading Style Sheets

Cascading Style Sheets

CSS Selectors based on Relationships

- Selectors can also be based on relationships between elements in the DOM tree
  - A E : Any element E that is a **descendant** of A
  - A > E: Any element E that is a **child** of A
  - E : first-child: Any element E that is the first child of its parents
  - B + E : Any element E that is the next sibling of B element (i.e., B and E have the same parent)

CSS Pseudo-Class Selectors

- CSS Selectors can also involve actions external to the DOM called pseudo-classes
- Visited: Whether a page was visited in the history
- Hover: Whether the user hovered over a link
- Checked: Whether a check box was checked

```
1 Selector : pseudo-class {
2     property: value
3 }
```

CSS Selectors Relationships: Example


```html
<div class="menu-bar">
  <ul>
    <li>
      <a href="example.html">Menu</a>
      <ul>
        <li>
          <a href="example.html">Link</a>
        </li>
        <li>
          <a class="menu-nav" href="example.html">Submenu</a>
          <ul>
            <li>
              <a class="menu-nav" href="example.html">Submenu</a>
              <ul>
                <li><a href="example.html">Link</a></li>
                <li><a href="example.html">Link</a></li>
                <li><a href="example.html">Link</a></li>
                <li><a href="example.html">Link</a></li>
              </ul>
            </li>
            <li><a href="example.html">Link</a></li>
          </ul>
        </li>
      </ul>
    </li>
  </ul>
</div>
```

```css
div.menu-bar ul ul {
  display: none;
}

div.menu-bar li:hover > ul {
  display: block;
}
```

The first rule says that for all ‘div’ elements of class ‘menu-bar’, in which an ul element is a descendant of another ul, do not display the second element

The second rule says that for all ‘div’ elements of class ‘menu-bar’, in which an ul element is a child of an li element, display it and the entire block, if the mouse hovers over the second element

Class Activity: CSS Rules

- What's the effect of the following CSS spec. on the HTML code in the next two slides ?

```
#news { background-color: silver; font: italic; color: black; }

.sports { color: blue; text-decoration: underline; }

H3, H4 { font-family: sans-serif; }

.latest { color: green;}

#news span { color: red; }

P.select { font-size: medium; }
```

Source: Ali Mesbah and Shabnam Mirshokraie. 2012. Automated analysis of CSS rules to support style maintenance. In Proceedings of the 34th International Conference on Software Engineering (ICSE '12). IEEE Press, Piscataway, NJ, USA, 408-418.

Class Activity: HTML -1

- Use a rich text editor (i.e., Word, LibreOffice Writer, etc.) to draft a preview of the output of this example.

```
<HTML>
 <HEAD>
  <LINK href="example.css" rel="stylesheet" type="↩
       text/css"/>
 </HEAD>
 <BODY>
   <P id='news' style='font:normal'>World
     <SPAN class='sports'>Sports news</SPAN>
   </P>
   <DIV class='sport'>Football</DIV>
 </BODY>
</HTML>
```

Class Activity: HTML -2

- Use a rich text editor (i.e., Word, LibreOffice Writer, etc.) to draft a preview of the output of this example.

```
<HTML>
 <HEAD>
 <LINK href="example.css" rel="stylesheet" type="↩
      text/css"/>
 </HEAD>
 <BODY>
   <P id='news' style='font:normal'>World
     <SPAN class='latest'>latest news</SPAN>
   </P>
 </BODY>
</HTML>
```