Hackers of UMass

The Web

Web Applications

Hypertext Markup Language

Cascading Stylesheets

Overview

- **The Web**
- **HyperText Markup Language (HTML)**
- **Cascading Stylesheets (CSS)**

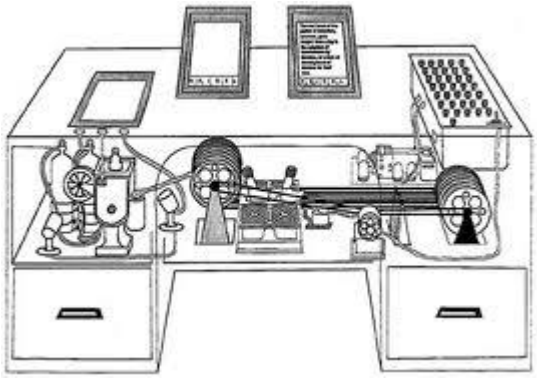
- **Happy Hacking!**

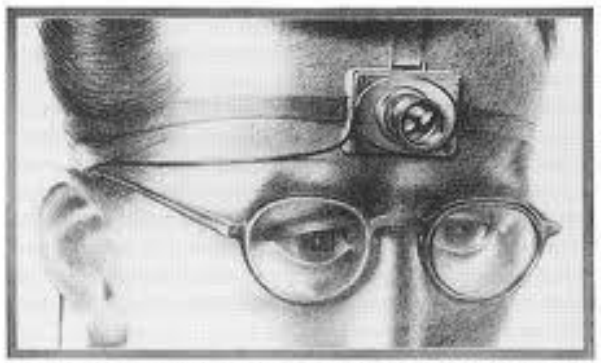
A Brief History of the Web

1945 Vennevar Bush "As We May Think"

a conceptual machine that could store vast amounts of information, in which a user had the ability to create information "trails": links of related text and illustrations. This trail could then be stored and used for future reference. Bush believed that using this associative method of information gathering was not only practical in its own right, but was closer to the way the mind ordered information.

Memex on Wikipedia & As We May Think
Vennevar's original article in the Atlantic Monthly.



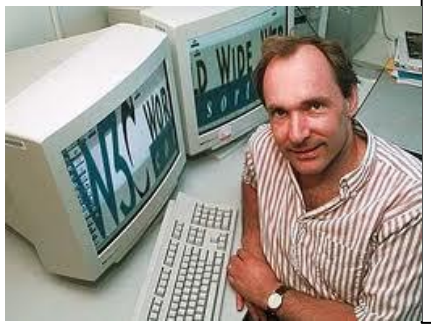


A scientist of the future records experiments with a tiny camera fitted with universal-focus lens. The small square in the eyeglass at the left sights the object (*LIFE* 19(11), p. 112).

A Brief History of the Web

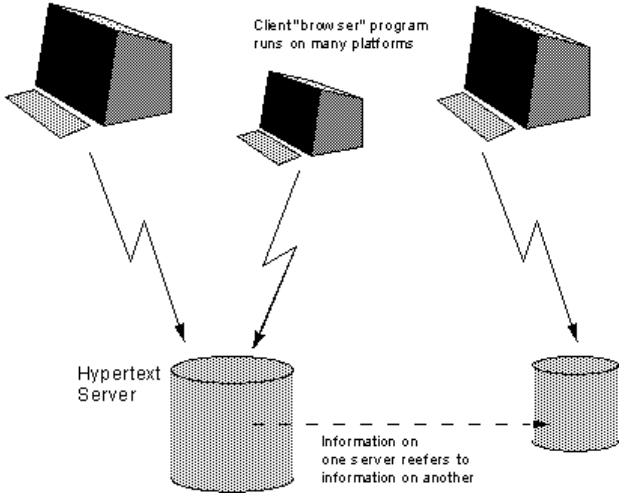
Tim Berners-Lee (1990) CERN

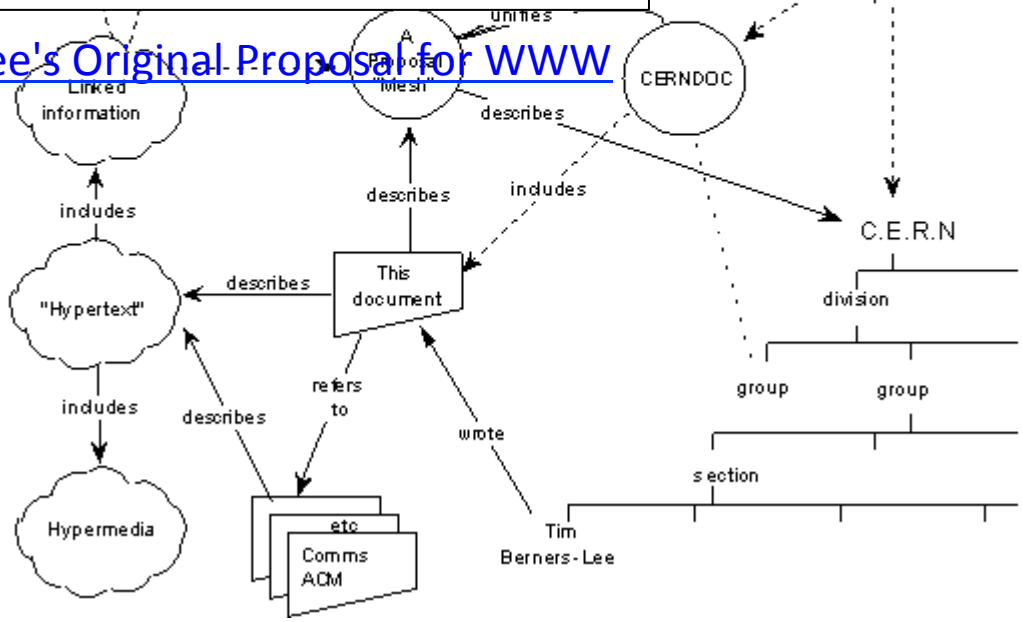
European Organization for Nuclear Research



In providing a system for manipulating this sort of information, the hope would be to allow a pool of information to develop which could grow and evolve with the organization and the projects it describes. For this to be possible, the method of storage must not place its own restraints on the information. This is why a "web" of notes with links (like references) between them is far more useful than a fixed hierarchical system....The system we need is like a diagram of circles and arrows, where circles and arrows can stand for anything.

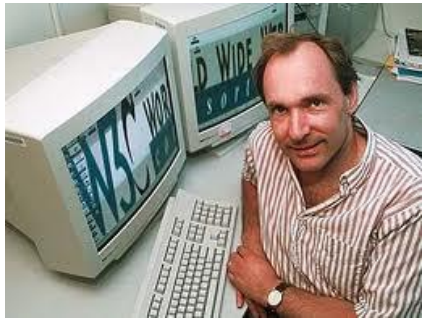
Tim Berners-Lee's Original Proposal for WWW





A Brief History of the Web

Tim Berners-Lee (1990) CERN



By Christmas 1990 Berners-Lee had built all the necessary tools for the *early web*:

HyperText Transfer Protocol (HTTP)

HyperText Markup Language (HTML)

HTTP Web Browser

HTTP Web Server

A Brief History of the Web

Growth of the Web (1992 - 1995)

- Mostly Universities

Mosaic web browser, X Windows (1993)

Development team led by Marc Andreessen

Al Gore's High Performance Computing and Communication Act of 1991

- Cello web browser, MS Windows (1993)
  Cornell Law School
- Netscape is born (1994)
  Andreessen & former CEO of Silicon Graphics James Clark

A Brief History of the Web

Growth of the Web (1992 - 1995)

- Berners-Lee + MIT (1994)

  Founded World Wide Web Consortium (W3C)

  Companies willing to create standards
  and recommendations to improve
  the quality of the web.

- A Free Web

  Berners-Lee made the web free
  with **no patent and no royalties due**

  W3C decided that their standards must be based on royalty-free technology to be easily adopted by anyone.

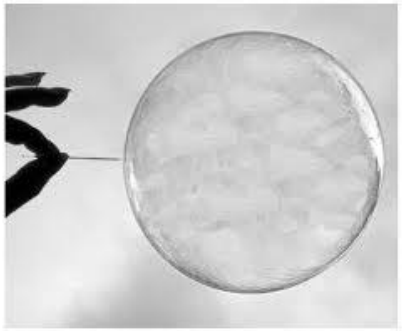
A Brief History of the Web

Commercialization of the Web (1996 - 1998)

- Companies realize that a web presence is no longer optional.
- The rise of the dotcoms.

Boom and Bust (early 2000's)

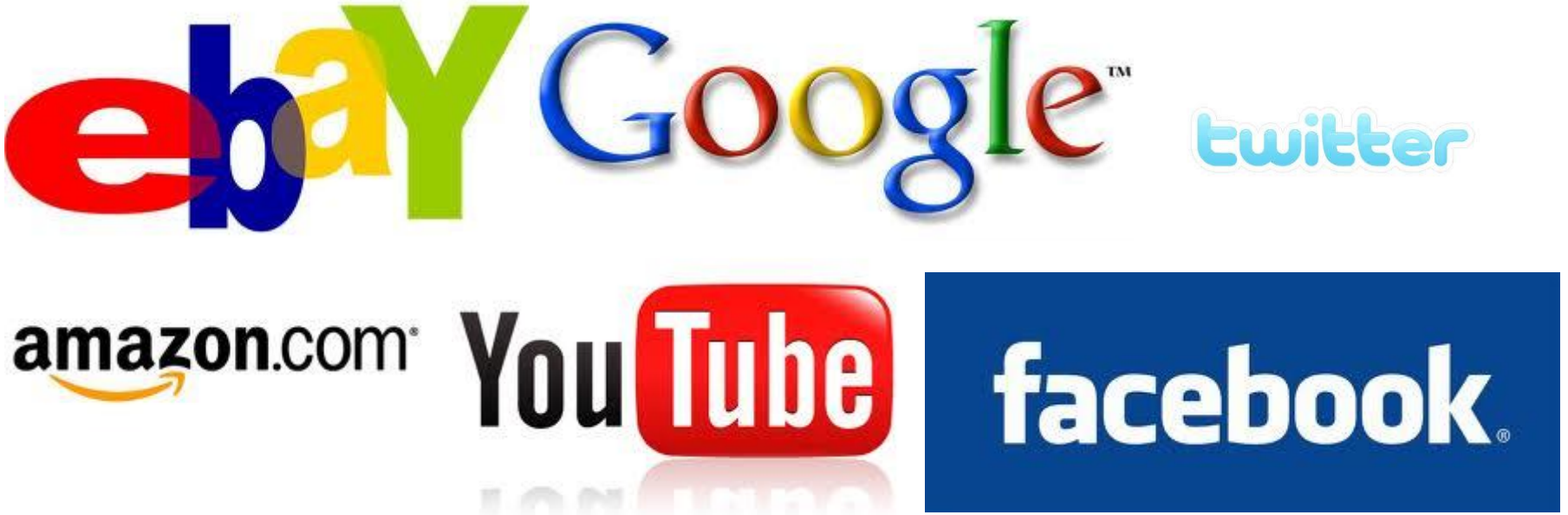
- Good ideas, bad business plans
- Burned through venture capital and failed to make profits!



A Brief History of the Web

The Ubiquitous Web (2002 - 2009)

- Better business models & technology
- The web is **everywhere**.



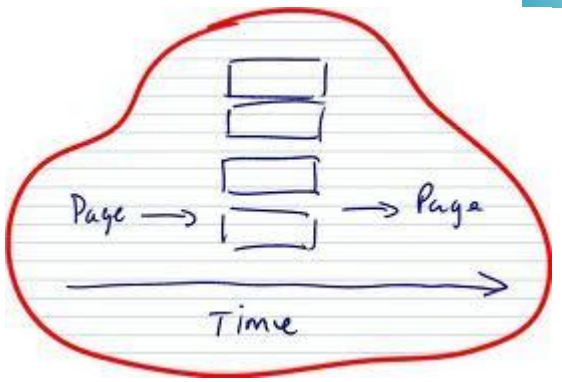
A Brief History of the Web

Web 2.0 (2002 - 2009)

- User Generated Content (Blogs, RSS)
- Sophisticated "web sites"
- The beginning of the *Web Application*

Asynchronous Page Updates

Behaves more like a native application



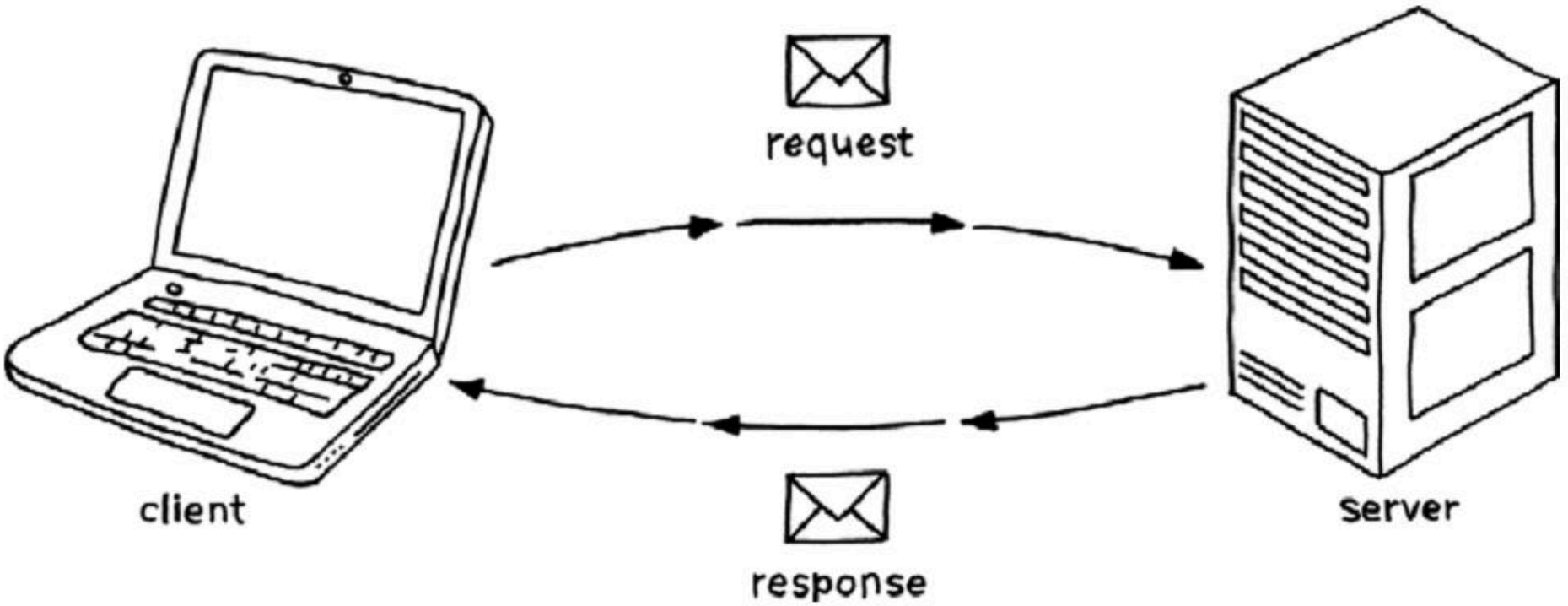
A Brief History of the Web

Next Generation Web (2010 - future)

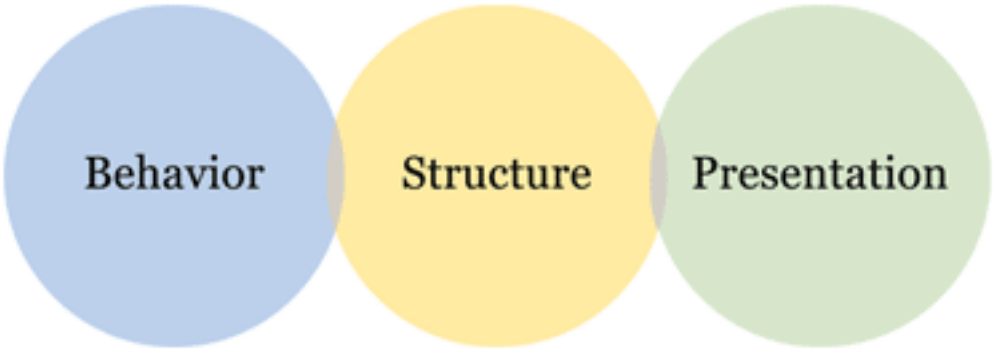
- **More Sophisticated Web Applications**
- **JavaScript is ubiquitous**
  - client-side
  - server-side
- **HTML5 + CSS3 + JavaScript**
  - Online & offline capabilities
  - Speed comparable to native performance
  - Graphics capabilities, Efficient communication
- **"Cloud Computing"**

Web Communication

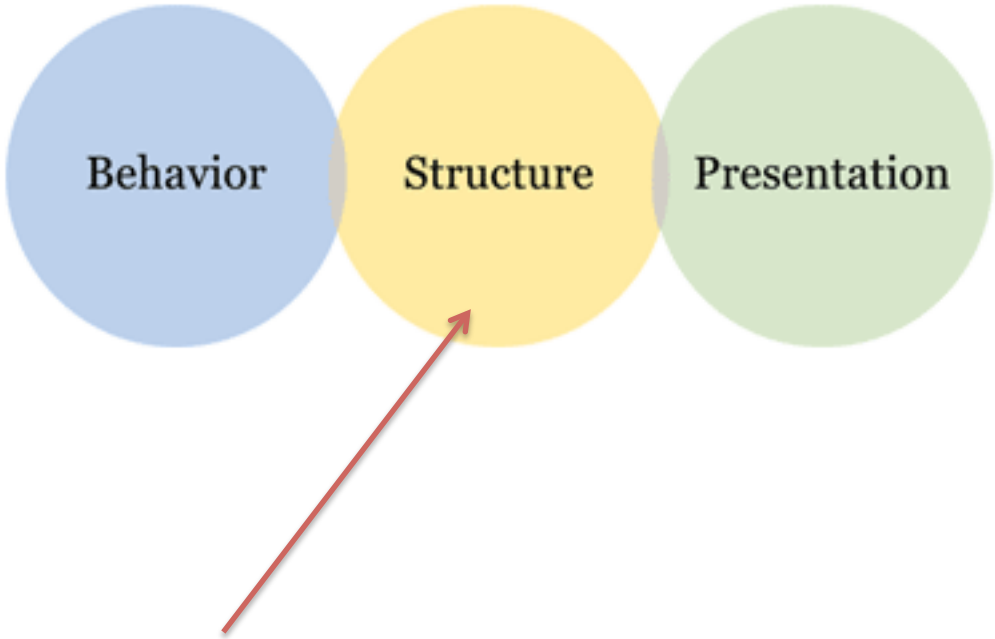
HyperText Transfer Protocol (HTTP)



Web Application Parts

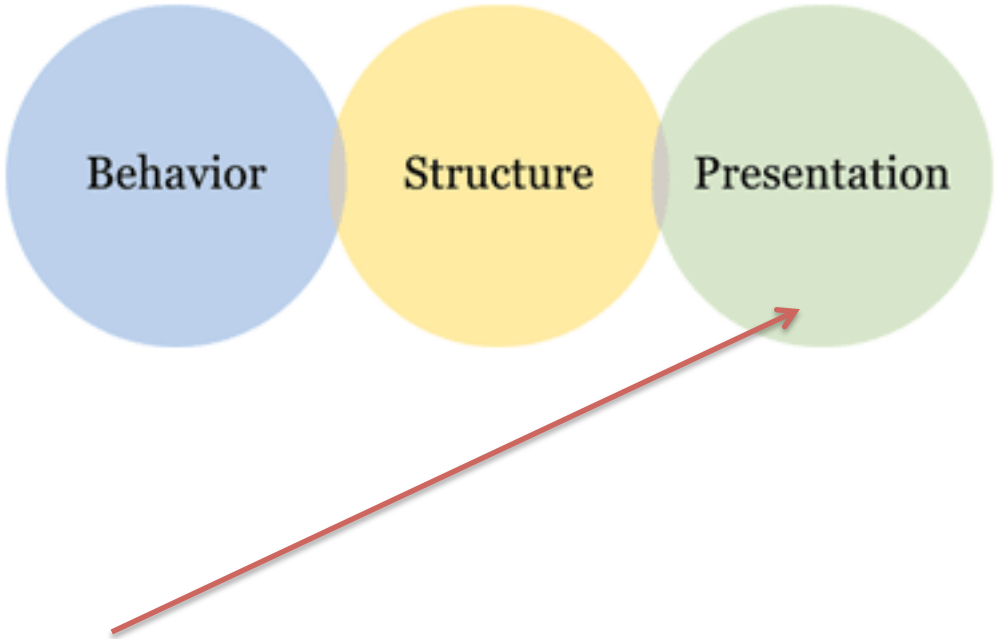


Web Application Parts



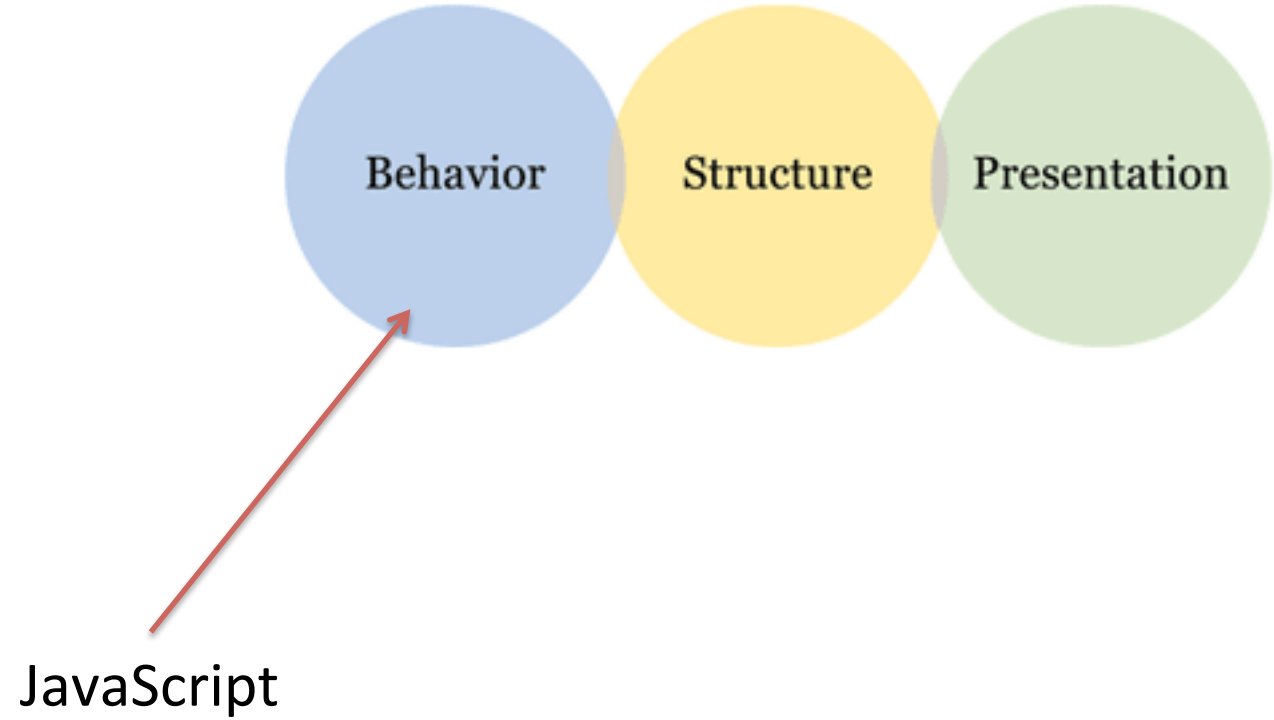
HyperText Markup Language (HTML)

Web Application Parts

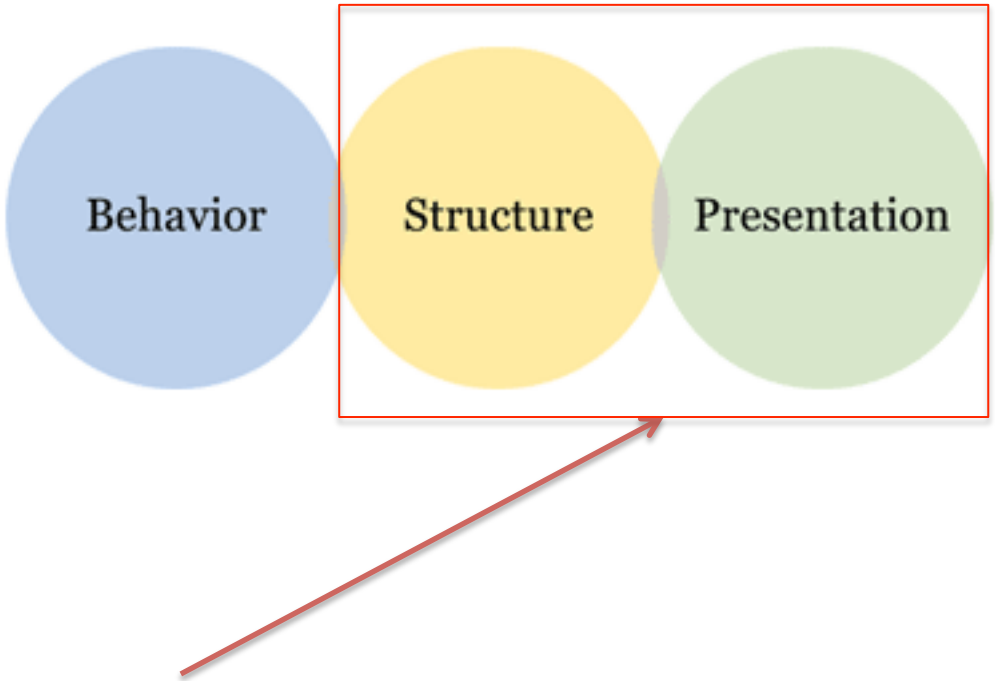


Cascading Style Sheets (CSS)

Web Application Parts



Web Application Parts



We are focused on these tonight!

Structure: HTML

HTML is a text-based language used to describe the structure and content of a document.

HTML uses “markup” (a.k.a tags) to provide structural information about the content contained within it.

HTML is understood and manipulated by the browser, CSS, and JavaScript.

Markup Languages

- **What is a markup Language?**
  - Not a programming language
  - A language used to annotate text
    - Syntactically distinguishable
    - Instructions contained within to describe content
- **HyperText Markup Language (HTML)**
  - Is a markup language!

HTML Versions

HTML Versions

- HTML 2.0 (1995)
- HTML 3.2 (1997)
- HTML 4.0 (1997)
- HTML 4.01 (1998, 3 different versions)
- ISO HTML (2000)
- HTML5

XHTML Versions

- XHTML 1.0 (2000)
- XHTML 1.1 (2001)
- XHTML 2.0
- XHTML5

HTML Tags

- **Tags**
  - **Start Tag:** <*tagname*>
  - **End Tag:** </*tagname*>
  - **Singleton Tag:** <*tagname*> or <*tagname*/>
- **Content**
  - A start and end tag surround *content*:
    - <T> *content* </T>
    - Where *content* can be character data or tags

HTML Attributes

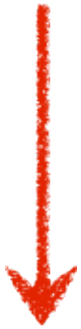
- **Tag Attributes**
  - Start tags may have additional properties
  - These properties can relate to anything
    - URL of image
    - URL of another resource (hyperlink)
    - Styling information
    - Behavior
    - Identification, equivalence class
    - User defined

```
<a href=http://www.google.com>A link to google</a>
```

HTML Example

Regular Text

```
This course studies a variety of web technologies
including HTML5, CSS, and JavaScript
```



```
<p>This course studies a variety of web technologies
including <b>HTML5</b>, <i>CSS</i>, and <u>JavaScript</u></p>
```

HTML Text

HTML5 Main Structure

- **First, the DOCTYPE**
  - Tells the browser which version of html used
  - For HTML5 it is:

```
<!doctype html>
```

  - For HTML 4.01, XHTML it looks something like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

HTML5 Main Structure

- **Character Encoding**

```
<!doctype html>
<meta charset=utf-8>
```

Provides additional information to web server, web client, or both

Indicates the character set used in the document you are writing

```
Content-Type: text/html; charset=utf-8
```

The web server might use this to set the Content-Type

HTML5 Main Structure

- **Character Encoding**

It also does not force you
to use an '/' for a singleton tag

```
<!doctype html>
<meta charset=utf-8>
```

HTML5 does not require you
to quote the value of an attribute

HTML5 Main Structure

- The basic structure of an HTML5 document:

```
<!doctype html>
<meta charset=utf-8>
<html>
<head>
<title>Message</title>
</head>
<body>
<p>If you don't know where you are going,
   any road will take you there.</p>
</body>
</html>
```

The html tag defines the start of an html document

HTML5 Main Structure

- The basic structure of an HTML5 document:

```
<!doctype html>
<meta charset=utf-8>
<html>
<head>
<title>Message</title>
</head>
<body>
<p>If you don't know where you are going,
   any road will take you there.</p>
</body>
</html>
```

The head tag defines meta information *about* the html document.

HTML5 Main Structure

- The basic structure of an HTML5 document:

```
<!doctype html>
<meta charset=utf-8>
<html>
<head>
<title>Message</title>
</head>
<body>
<p>If you don't know where you are going,
   any road will take you there.</p>
</body>
</html>
```

The title tag defines the *title* of the document.

This is the text that appears in the browser tab.

HTML5 Main Structure

- The basic structure of an HTML5 document:

```
<!doctype html>
<meta charset=utf-8>
<html>
<head>
<title>Message</title>
</head>
<body>
<p>If you don't know where you are going,
   any road will take you there.</p>
</body>
</html>
```

The body tag defines the main content area of the document.

HTML5 Main Structure

- The basic structure of an HTML5 document:

```
<!doctype html>
<meta charset=utf-8>
<html>
<head>
<title>Message</title>
</head>
<body>
<p>If you don't know where you are going,
    any road will take you there.</p>
</body>
</html>
```

The paragraph tag is an example of a structural tag that allows you organize content.

HTML5 Main Structure

- The basic structure of an HTML5 document:

```
<!doctype html>
<meta charset=utf-8>
<html>
<head>
<title>Message</title>
</head>
<body>
<p>If you don't know where you are going,
   any road will take you there.</p>
</body>
</html>
```

Then we have character data.

HTML5 Main Structure

- The basic structure of an HTML5 document:

```
<!doctype html>
<meta charset=utf-8>
<html>
<head>
<title>Message</title>
</head>
<body>
<p>If you don't know where you are going,
   any road will take you there.</p>
</body>
</html>
```

Lowercase tag names are preferred in html5.

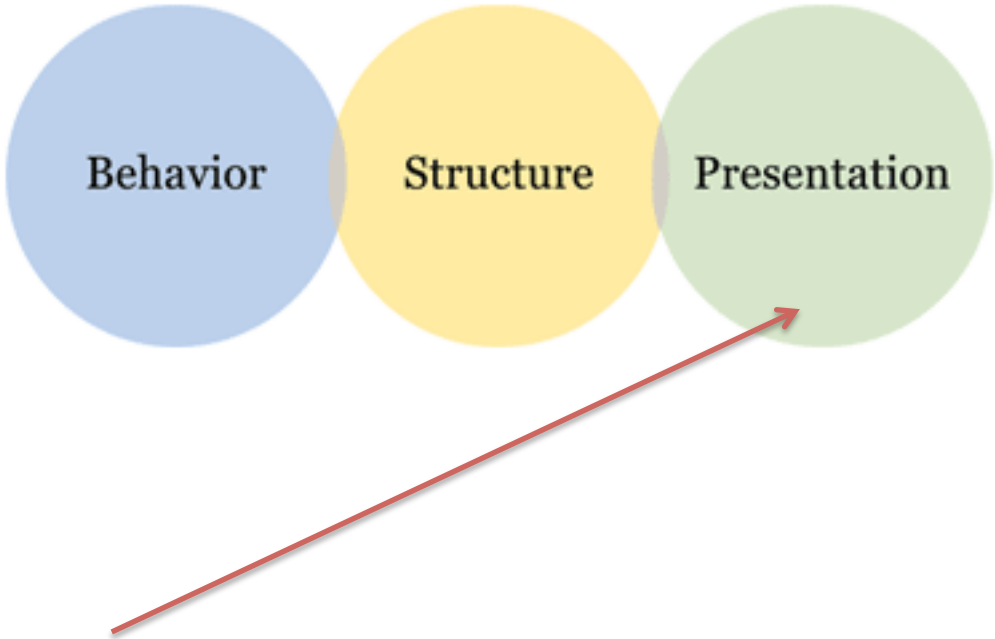
Examples

Very nice! Great Success! Examples!

Cascading Style Sheets

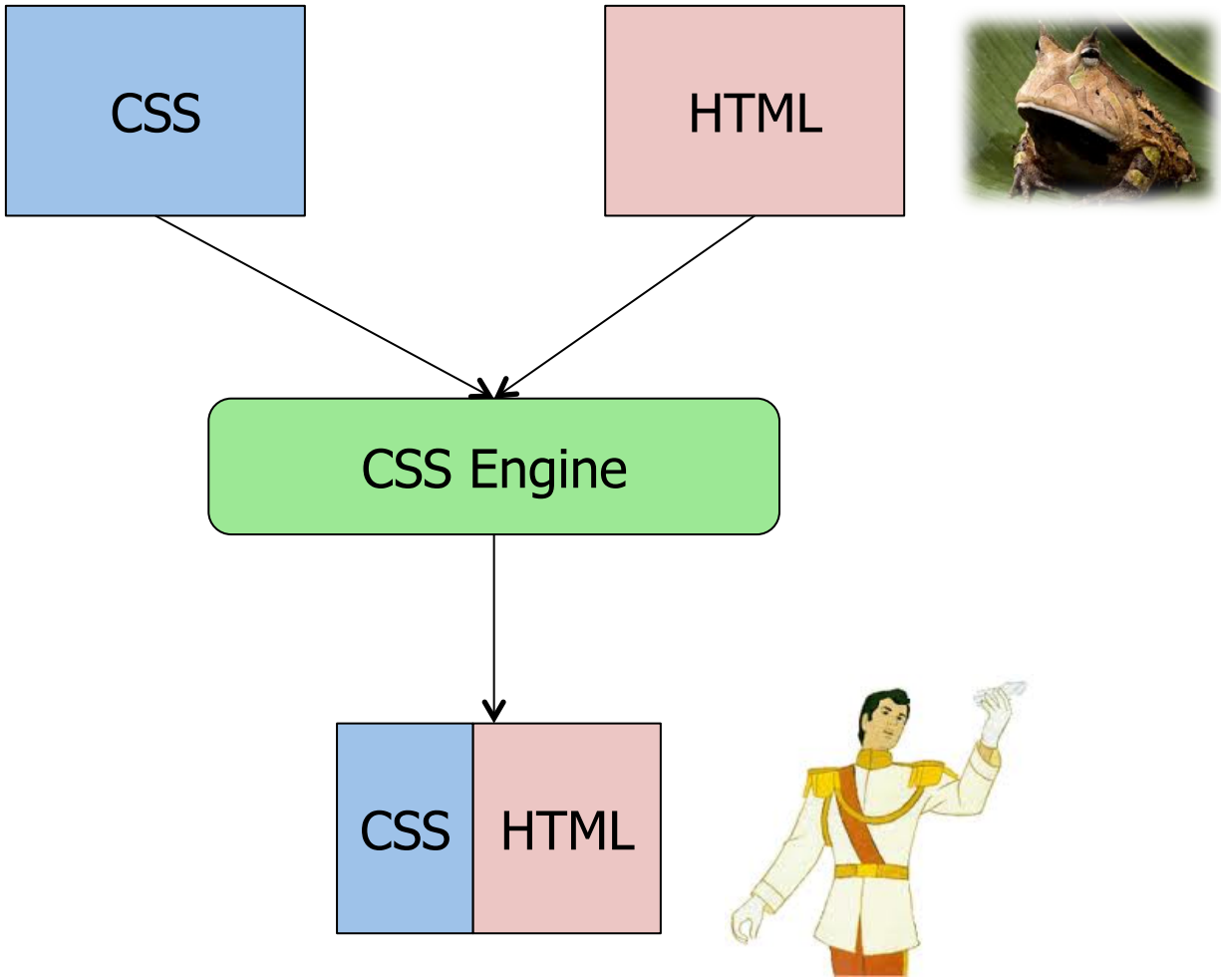
- **What is CSS?**
  - Style & Presentation
  - A Language to Manipulate HTML elements
- **Why is it important?**
  - Usability of HTML documents/user interfaces
  - Separation of concerns
- **How is it used?**
  - Internal: within tags or <*style*> element
  - External: imported with <*link*> element

Web Application Parts



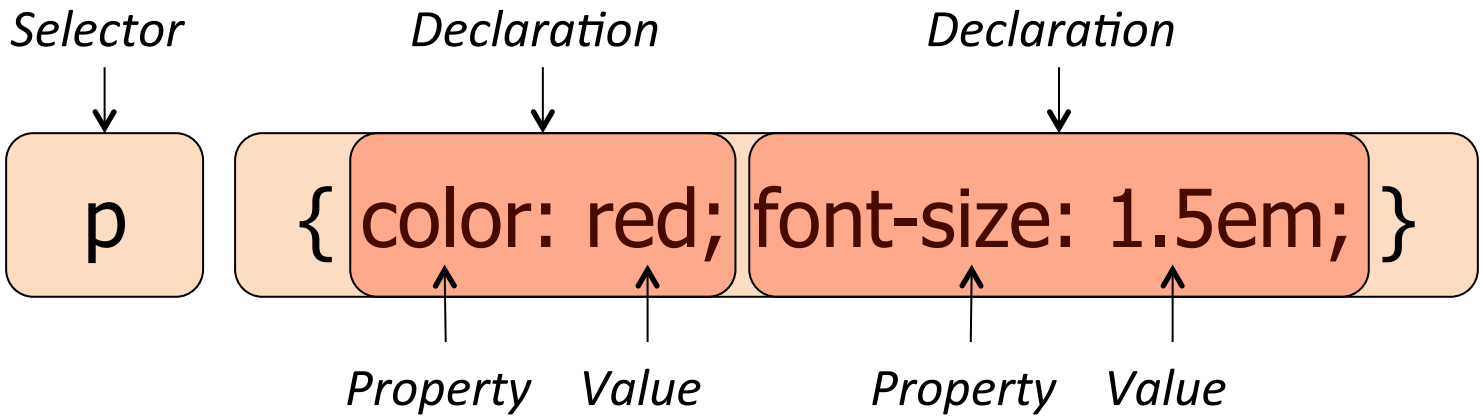
Cascading Style Sheets (CSS)

CSS Usage



Anatomy of a CSS Style Rule

```
p {
    color: red;
    font-size: 1.5em;
}
```



Make all paragraphs have a font color of red and font size of 1.5em

Creating Styles & Style Sheets

- **Where do we "put" style rules?**
  - Internal
    - We can do this by embedding CSS in a *style attribute* string value or as character data within a *<style> element*.
  - External
    - We can "include" CSS rules using the *src* attribute of a *<link> element*.
    - **This is the better way.**

Interal Style Sheets

- **HTML *style* Attribute**
  - Syntax: <E style="CSS declarations"> … </E>

Interal Style Sheets

- **HTML *style* Attribute**
  - Syntax: <E style="CSS declarations"> … </E>
- **Example:**

```
…
<body>
  <p style="color: red; font-size: 1.5em;">
    The rule is, jam tomorrow and jam
    yesterday – but never jam today.
  </p>
</body>
…
```

Interal Style Sheets

- **HTML *<style>* Element**
  - Syntax: <style type="text/css"> … </style>
  - Goes inside the *<head>* element

Interal Style Sheets

- **HTML *<style>* Element**
  - Syntax: <style type="text/css"> … </style>
  - Goes inside the *<head>* element
- **Example:**

```
<head>
  <style type="text/css">
    p { color: red; font-size: 1.5em; }
  </style>
</head>
<body>
  <p>The rule is, jam tomorrow and jam
     yesterday – but never jam today.</p>
</body>
…
```

External Style Sheets

- **HTML *<link>* element**
  - Syntax:
    <link rel="stylesheet" src="filename.css" type="text/css" />
  - Goes inside the *<head>* element

External Style Sheets

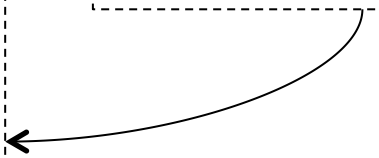
- **HTML *<link>* element**
  - Syntax:
    <link rel="stylesheet" src="filename.css" type="text/css" />
  - Goes inside the *<head>* element
- **Example:**

index.html

```
<head>
  <link rel="stylesheet"
        src="main.css"
        type="text/css">
</head>
<body>
  <p>The rule is, jam tomorrow and jam
     yesterday - but never jam today.</p>
</body>
```

main.css

```
p {
  color: red;
  font-size: 1.5em;
}
```



Selectors

- **Scope of a selector**
  - Indicates the *element* or *elements* of a page to style
- **They can be broad**
  - Apply to all elements of a particular kind
- **Or they can be specific**
  - Apply to an element with a particular name

Identifying What to Style

- **Page Wide Styling**
  - CSS rules that apply to every occurance of an HTML element in the document

```
p {
    color: red;
    font-size: 1.5em;
}
```

Identifying What to Style

- **Page Wide Styling**
  - CSS rules that apply to every occurance of an HTML element in the document

```
p {
    color: red;
    font-size: 1.5em;
}
```

Perhaps we want to be a little more specific…

Styling Classes of Tags

- **Class Selectors**
  - Allow you to style elements that belong to a group or serve some special purpose.

CSS

```
.important {
    color: red;
    font-size: 75px;
}
```

HTML

```
<h2 class="important">
Headlines
</h2>
…
<h2 class="important">
Birthdays
</h2>
…
```

Styling Named Elements

- **ID Selectors**
  - Allow you to style elements with a specific name or identifier.
  - Applies to ***only*** a single element.

CSS

```
#headlines {
    color: red;
    font-size: 75px;
}

#birthdays {
    color: green;
}
```

HTML

```
<h2 id="headlines">
Headlines
</h2>
…
<h2 id="birthdays">
Birthdays
</h2>
…
```

Styling Groups of Tags

- **Style across element types**
  - Same style information that you want to apply across many different tags

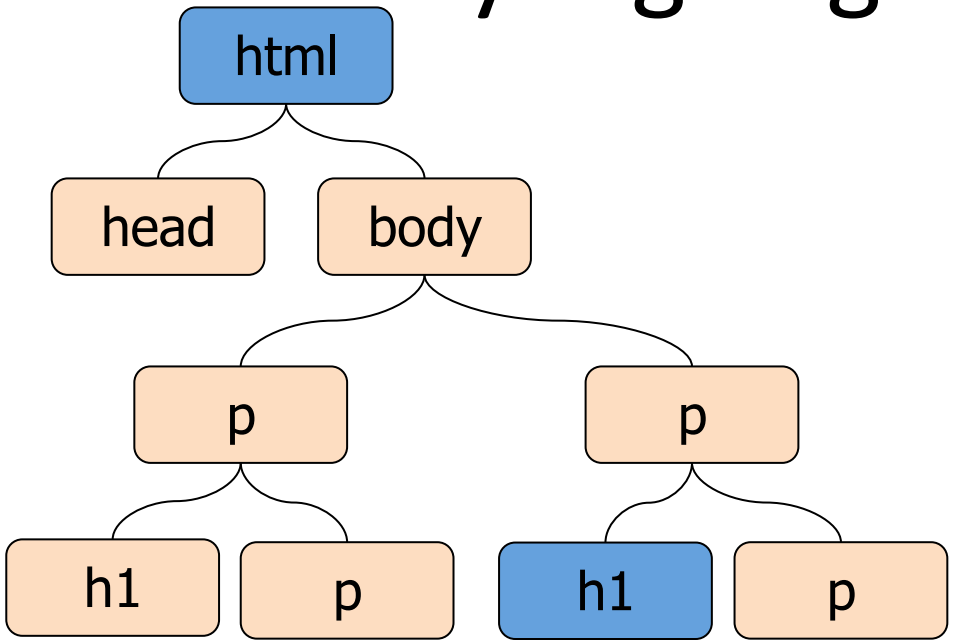
```
h1, h2, p, .hitem, #todo {
    color: #F134AC;
}

* {
    font-weight: bold;
}
```

Styling Tags within Tags

- **What if you want to style elements that are *relative* to another element?**
- **You can use *descendant selectors*.**
  - Ancestor: a tag that wraps another tag
  - Descendant: a tag inside one or more tags
  - Parent: the closest ancestor to another tag
  - Child: tag directly enclosed by another tag
  - Sibling: children of the same tag are siblings

Styling Tags within Tags

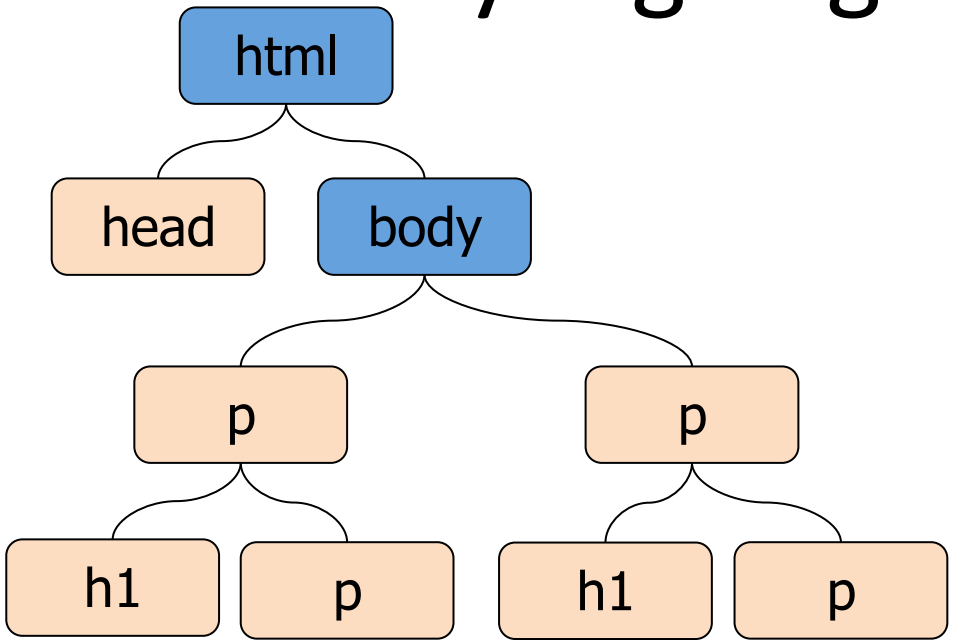


Ancestor Relationship

<html> is an ancestor of <h1>

In fact, <html> is an ancestor of all tags.

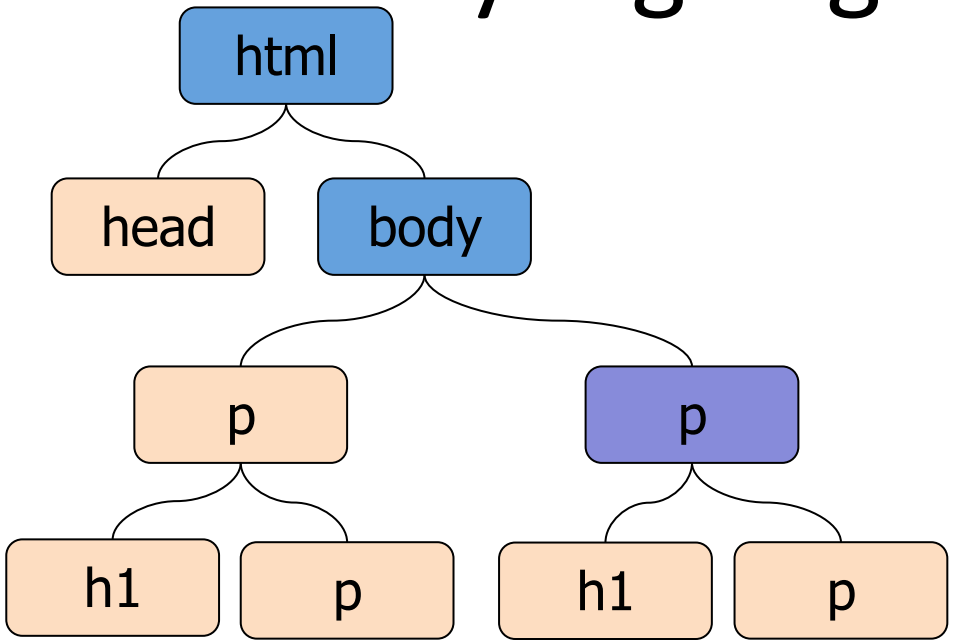
Styling Tags within Tags



Descendant Relationship

The <body> tag is a descendant of the <html> tag.

Styling Tags within Tags

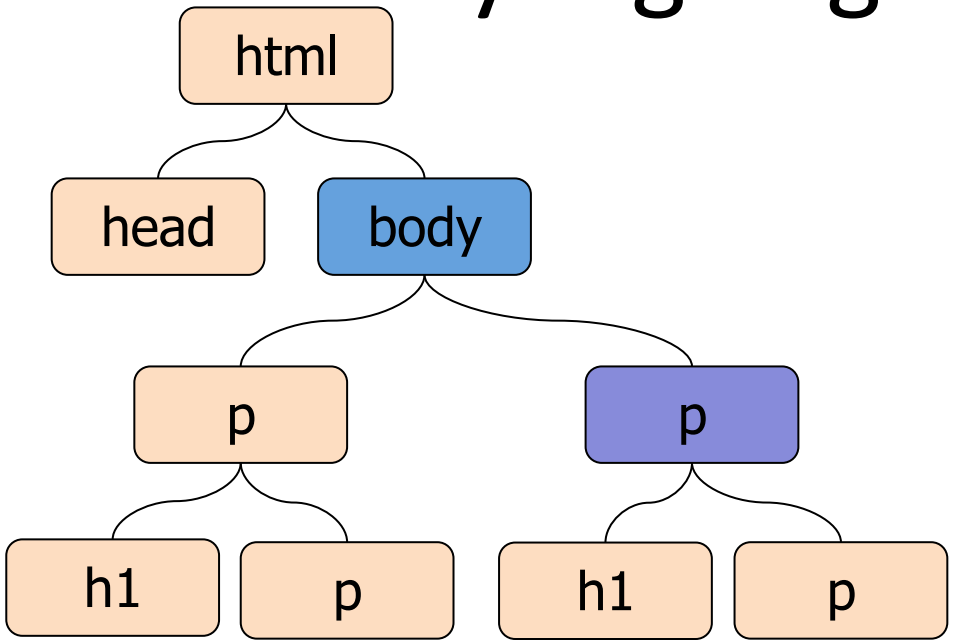


Descendant Relationship

The <body> tag is a descendant of the <html> tag.

The <p> tag is a descendant of both the <body> and the <html> tags.

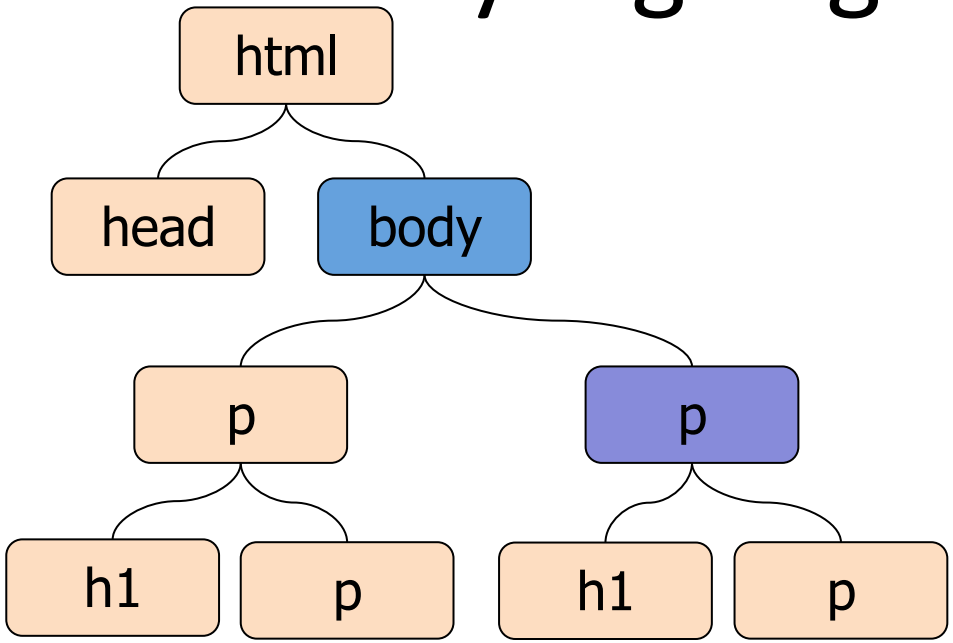
Styling Tags within Tags



Parent Relationship

The <body> tag is the parent of this <p> tag.

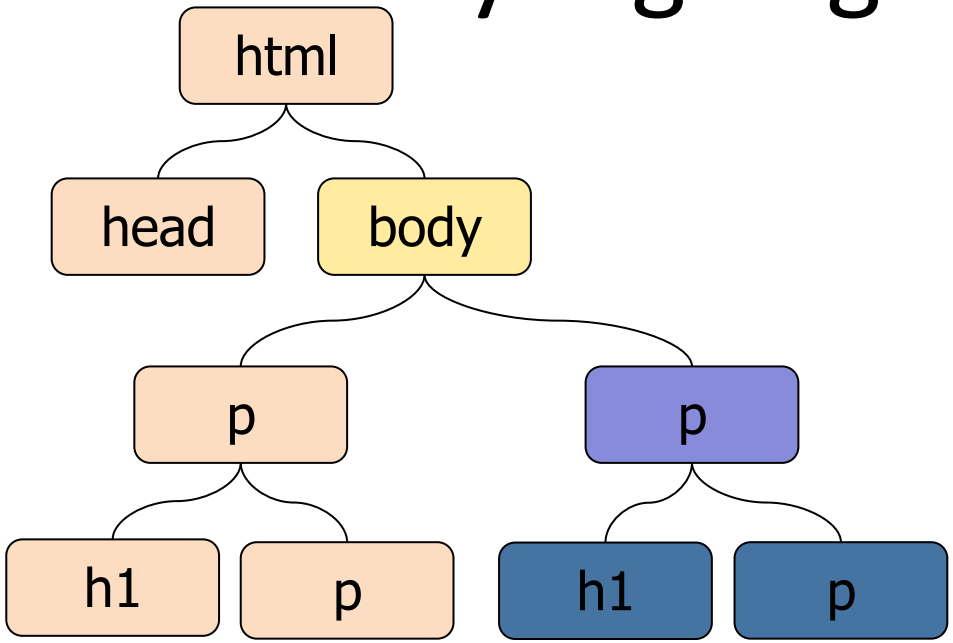
Styling Tags within Tags



Child Relationship

This <p> tag is the child of the <body> tag.

Styling Tags within Tags

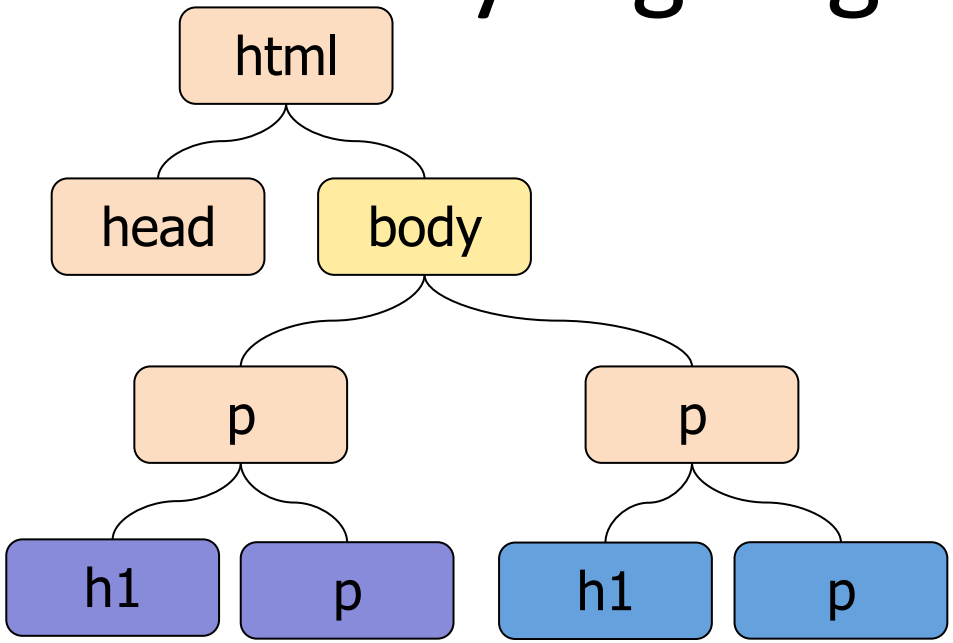


Sibling Relationship

These <h1> and <p> tags are siblings of each other.

They are both *children* of the containing <p> tag.

Styling Tags within Tags



Building Descendant Selectors

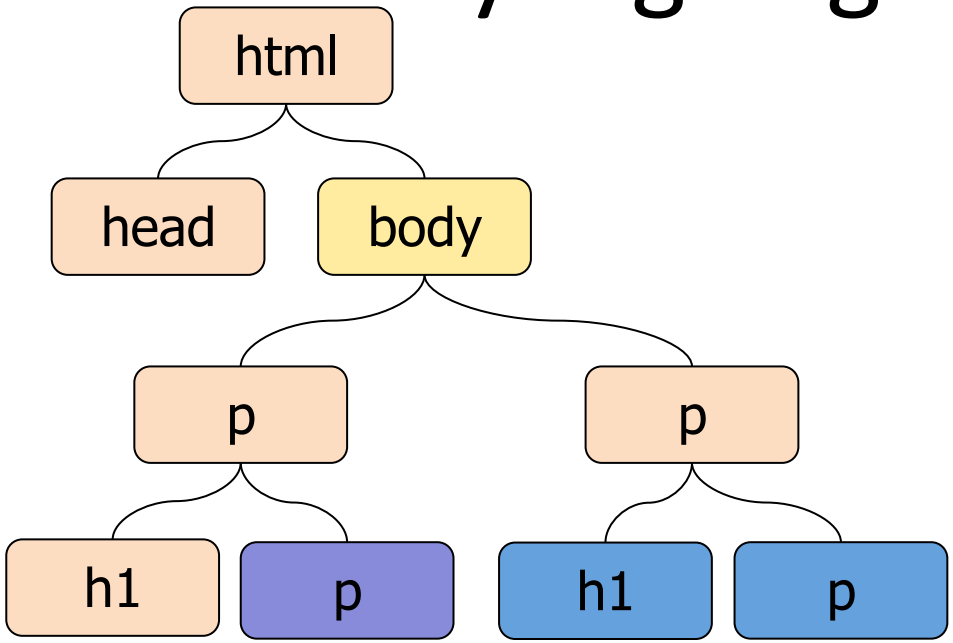
Make all paragraphs that are descendants of a <p> tag red.

Make all <h1> tags inside a <p> tag green.

```
p p {
    color: red;
}
```

```
p h1 {
    color: green;
}
```

Styling Tags within Tags



Building Descendant Selectors

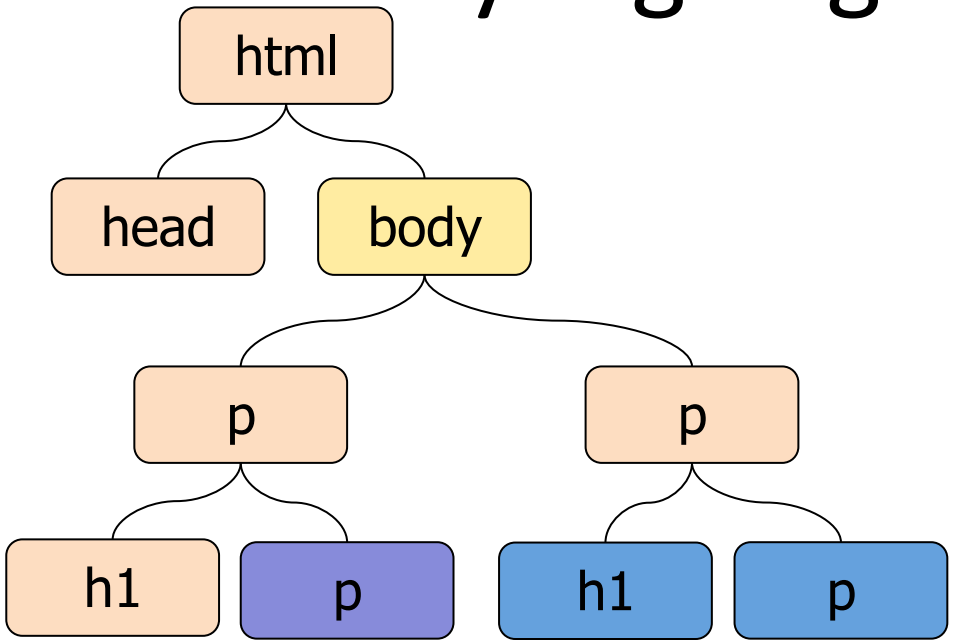
Make all paragraphs that are descendants of a <p> tag red.

Make all <h1> tags inside a <p> tag green. Only <h1> tags that are of the class "emphasize".

```
p p {
    color: red;
}
```

```
p h1.emphasize {
    color: green;
}
```

Styling Tags within Tags



Building Descendant Selectors

Make all paragraphs that are descendants of a <p> tag red.

Make all <h1> tags inside a <p> tag green. Only <h1> tags that are of the class "emphasize".

```
p p {
    color: red;
}
```

```
p h1.emphasize {
    color: green;
}
```

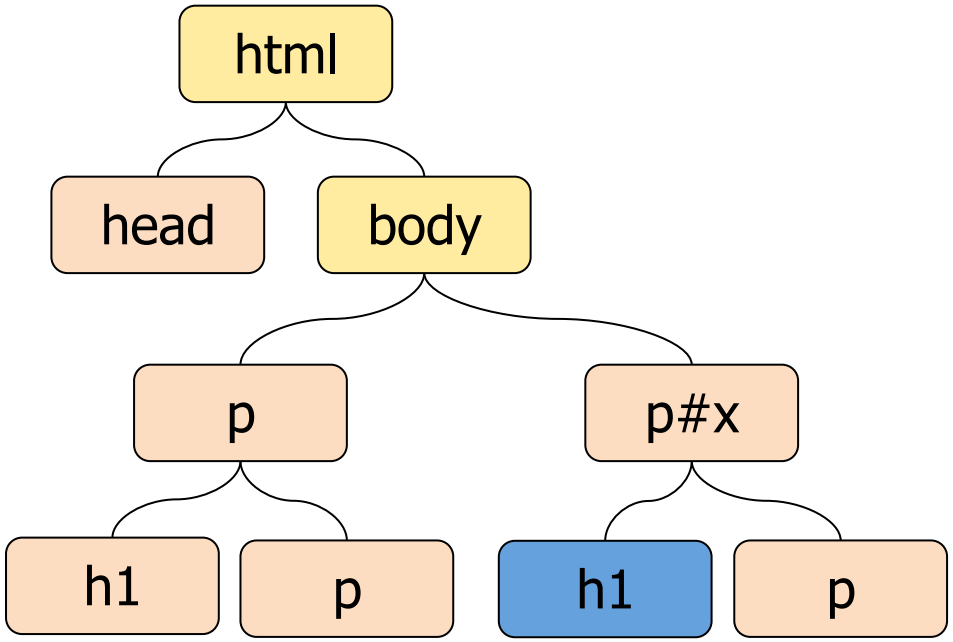
```
p h1 .emphasize {
    color: green;
}
```

What would this do?

Child Selectors

- **What if we want to style a *direct* descendant of an element?**
  - Use child selectors

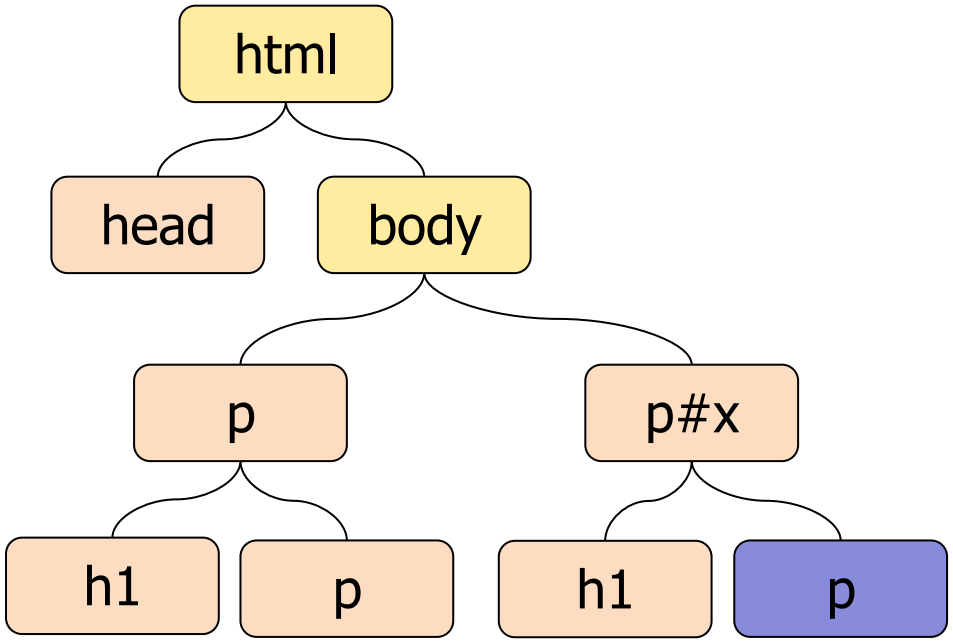
```
body > p#x > h1 {
   color: red;
}
```



Sibling Selectors

- **What if we want to style a *sibling* of an element?**
  - Use sibling selectors

```
body > p#x > h1 + p {
   color: red;
}
```



Attribute Selectors

- **Perhaps we care about elements with particular attributes.**
  - img[title]
    - images with a title attribute
  - a[href="http://google.com"]
    - links to google
  - input[type="text"]
    - text input boxes

Attribute Selectors

- **Perhaps we care about elements with particular attributes that match the beginning of some text value:**
  - a[href^="http://"]
    - Links to external sites
  - a[href^="http://"], a[href^="https://"]
    - Regular and secure links to external sites

Attribute Selectors

- **Perhaps we care about elements with particular attributes that match the end of some text value:**
  - a[href$=“.pptx”]
    - Links to external sites

```
a[href$=“.docx”] {
  background-image: url(docx.png) no-repeat;
  padding-left: 15px;
}
```

Attribute Selectors

- **Perhaps we care about elements with particular attributes that match any part of some text value:**
  - img[src*="face"]
    - Any image containing "face" in its src value

Examples

Very nice! Great Success! Examples!

HTML/CSS Hacking Necessities

- **A computer ☺**
- **A good text editor**
  - Atom, Sublime, Emacs, VIM, Notepad++
- **A good browser (or two or three)**
  - Chrome, Firefox, …
- **An Internet connection**
- **Github/Git (Optional)**
- **Template to start off with!**
  https://github.com/hackers-of-umass/photo-gallery-template

Happy Hacking!

https://github.com/hackers-of-umass/photo-gallery-template