HTML Basics

CMPS 369 Lecture 3

Today's Topic

- HTML - what is it for?
- Markup Syntax
- Attributes
- Core HTML Elements
- Semantic Formatting
- Links & Navigation

But first...

Lets enhance our Node.js server from last time

We want to write HTML in external files, so Node.js should extract the **path** from the URL requested and read the requested file contents.

- This is basically just like the very first web servers...
- I could have also used the **connect** package for node, and done this in 3 lines of code!

Note about Node

I don’t expect you to understand JavaScript or Node.js at this point.

We will cover JavaScript in depth, and iteratively you will become more familiar with it and Node and how it relates to Web Development

Be patient for now - and focus on HTML

/ Requests

A request for / indicates that the browser wants the "default" document. We've programmed our server to return a simple HTML file - **index.html**

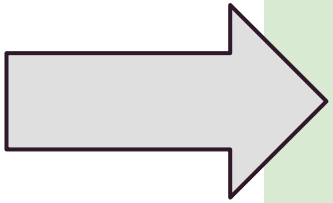
A nice program for looking at raw responses is **curl**.

```
curl -v http://localhost:3000/
```

```
HTTP/1.1 200 OK
Content-Type: text/plain
Date: Mon, 06 Jan 2014 18:45:15 GMT
Connection: keep-alive
Transfer-Encoding: chunked

<!doctype html>
<html>
    <head>
        <title>index.html</title>
    </head>
    <body>
        <p>Hello World</p>
    </body>
</html>
```

Response body is the HTML we put in index.html



doctype

There have been five versions of HTML.

There are only 3 that are primarily "active"

- HTML 4.01
- XHTML
- HTML 5

```
<!doctype html>
<html>
    <head>
        <title>index.html</title>
    </head>
    <body>
        <p>Hello World</p>
    </body>
</html>
```

- The doctype tells the browser what type of HTML you are writing.
- It also provides instructions to the browser on how strictly to interpret and use the HTML standard.
- This is a **big** deal. Lets discuss for a bit...

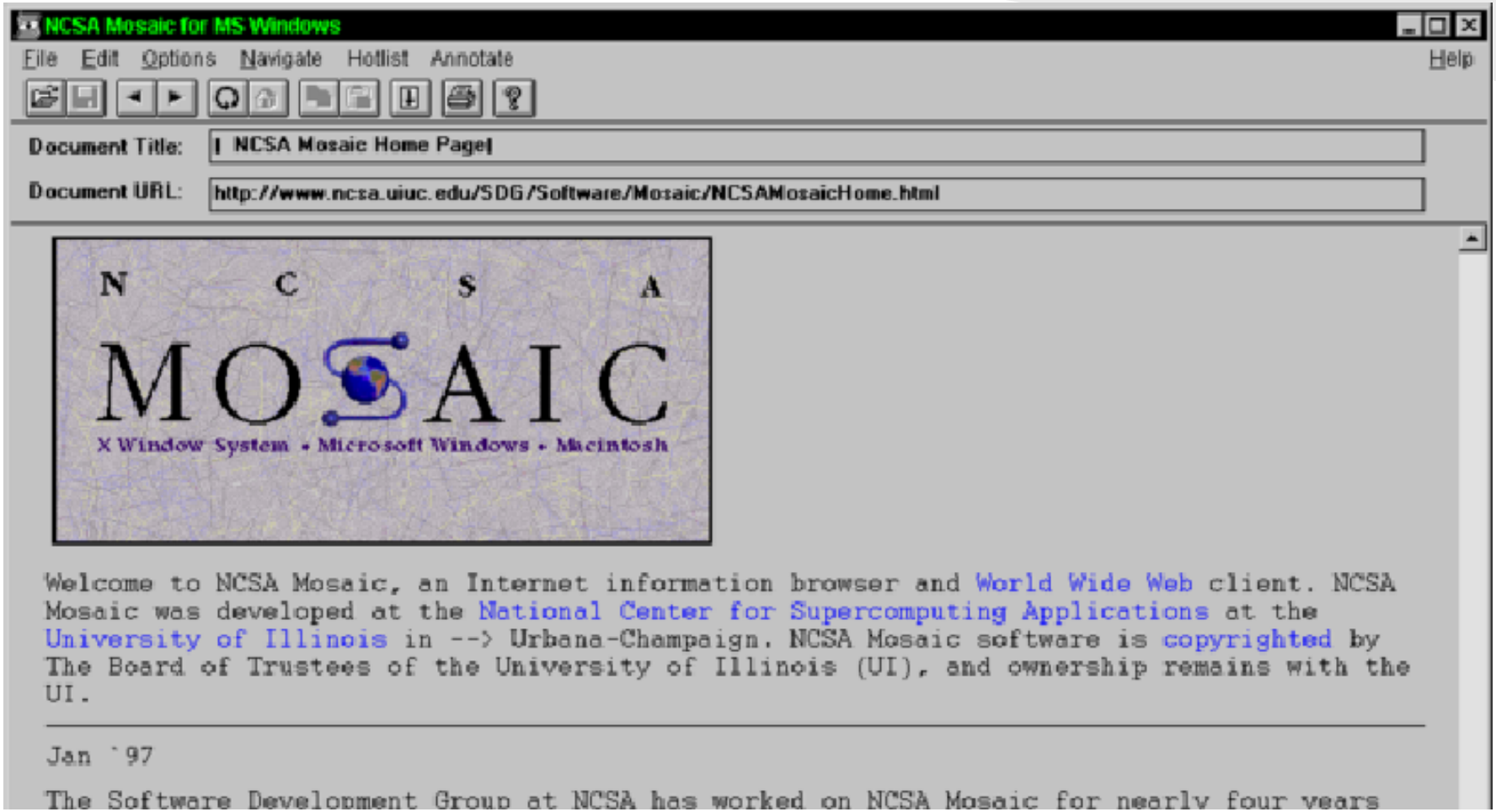
A bit of history

HTML is old!

- 1990: Tim Berners-Lee develops HTML for scientific/engineering applications
- 1992 Revisions - HTML contains

title	paragraphs
hyperlinks	headings
simple list	glossary
example	address

A bit of history

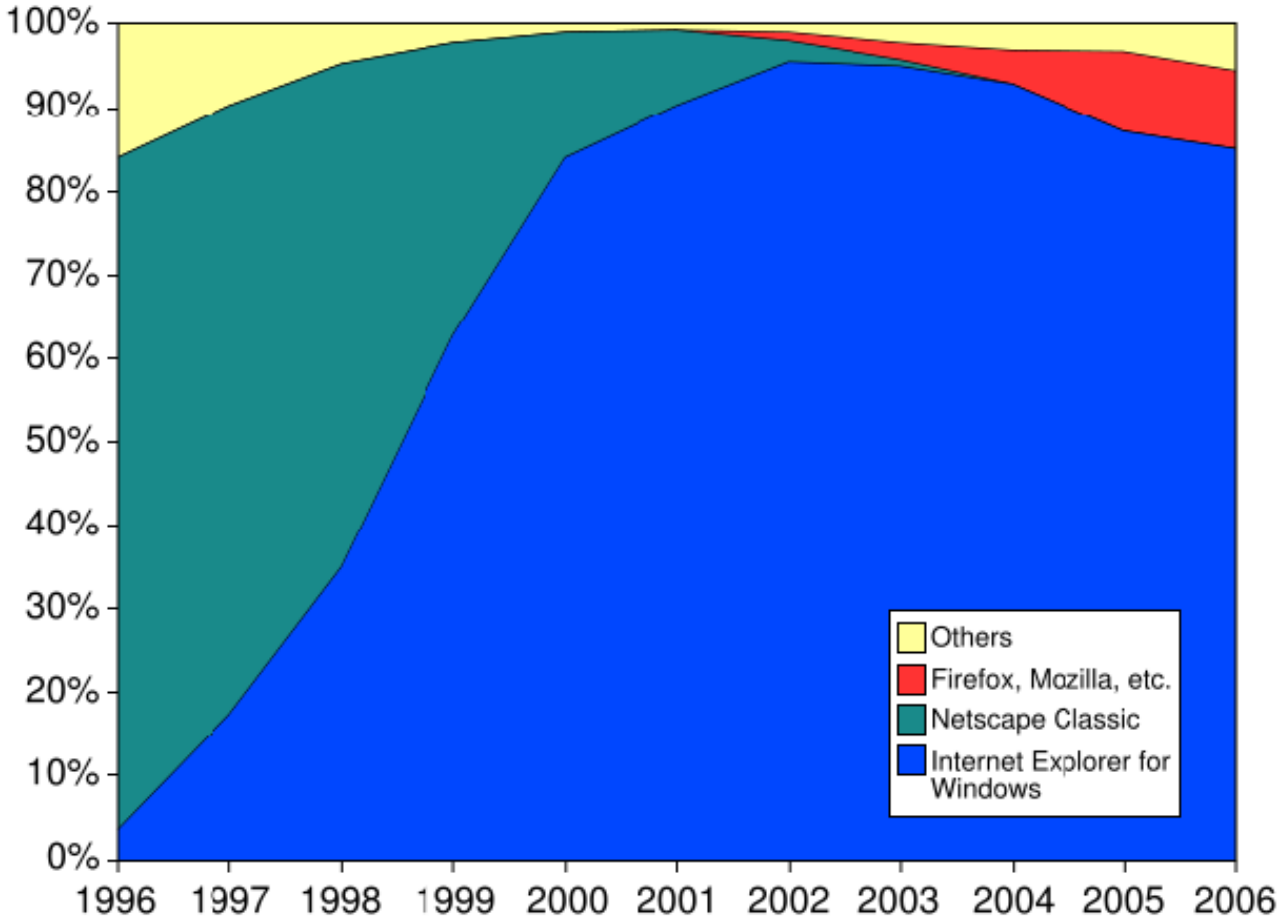


A bit of history

- 1993: Berners-Lee creates W3C
  - Goal is to publish HTML standard and to approve subsequent modifications
  - Ignored: Netscape & IE were moving too fast and had no interest in sharing ideas
- ~1997: IE 4.0 wins war for MS
  - development slows considerably
- HTML 4.01: First effective recommendation
  - Subset of IE implementation
  - Tries to separate display from content

Internet Explorer only “loosely” followed standards.. because it was the standard...

Browser Wars



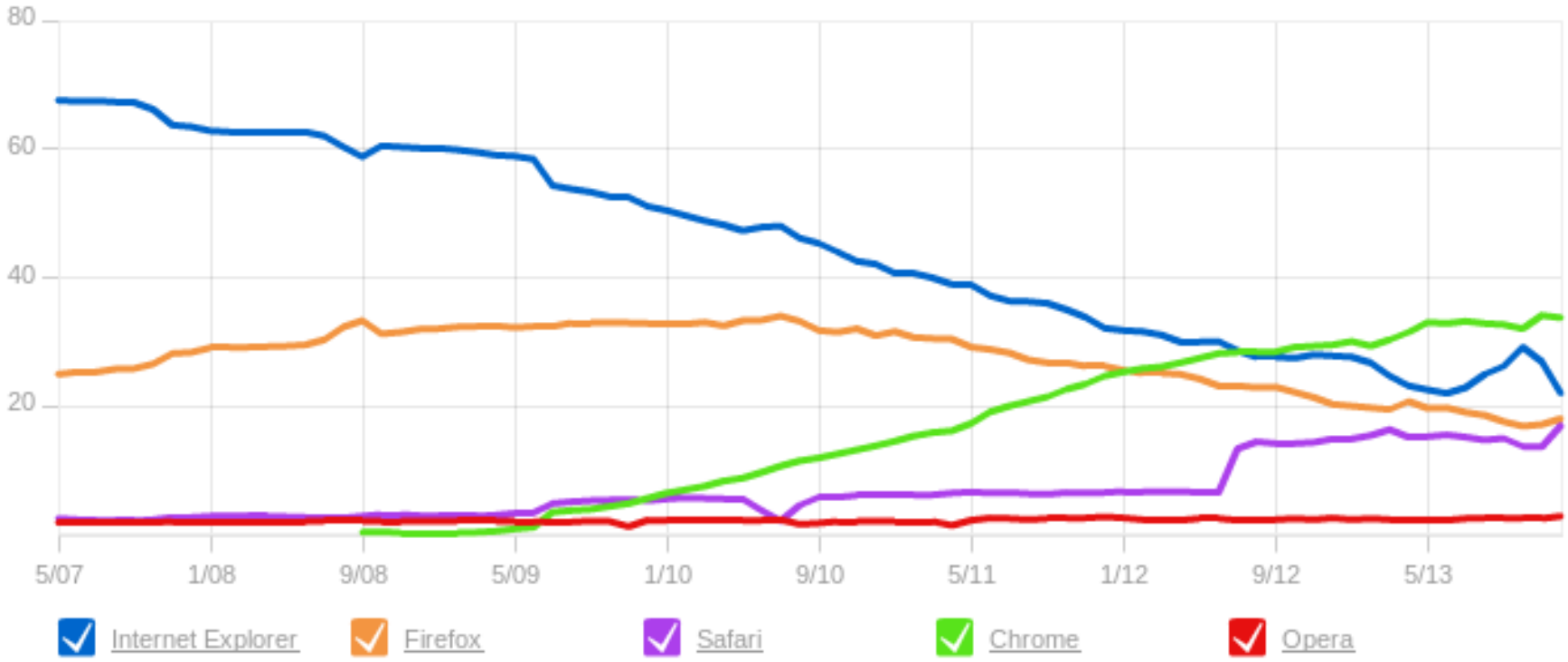
Browser Wars - Fallout

The state of web development in early 2000's was drastically different than it is today

Any website of sufficient complexity needed to be written largely twice -

- Once for Internet Explorer
  - And perhaps more, for IE4, IE6, IE7, etc.!!!
- Another time for Firefox/Chrome/Safari

Current trends



http://www.w3counter.com/trends

Parity among browsers

Now that no single browser dominates… they all need to play by the rules (standards)

End of the story

If you [view source] on the web, you will see some very, very scary stuff…

- Old HTML where elements aren't properly closed
- Old HTML element names that shouldn't be used
- Mixed upper/lower case element names
- HTML that only renders properly in "quirks" mode
- And lots of embedded CSS and terrible JavaScript

If you want a job as a web developer… don't be one of these people…

HTML5

If you write standard's based HTML5 code you can be quite confident your pages will work nicely (and consistently) on:

- Google Chrome (any OS, including Android)
- Apple Safari (Mac OS X, IOS)
- Firefox (any OS)
- Internet Explorer 9+, Edge
  - And there are bandaids to make IE8 and below work properly too.

If you leave out the doctype element, you transport yourself back to the internet dark ages… where you can never be too sure how your page will look on a different device/browser/platform!

HTML - What is it for?

HTML describes the **content** and **structure** of a document

- *Note, it does not describe how a browser will render the document.*
- CSS will handle the rendering instructions

If you keep this subtle point clear in your mind, you will be able to do remarkable things with HTML.

Markup

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Popular Websites: Google</title>
  </head>
  <body>
    <h1>About Google</h1>
    <p>Google is best known for its search engine, although Google now offers a number of  other services.</p>
    <p>Google's mission is to organize the world's information and make it universally accessible and useful.</p>
    <p>Its founders Larry Page and Sergey Brin started Google at Stanford University.</p>
  </body>

</html>
```

We need to know the difference between **elements**, **tags**, and **content**.

Core Elements

All HTML pages have an `<html>` root element, with *exactly* two children: `<head>` and `<body>`

- The `<head>` element contains information *about* the page.

- The `<body>` element contains the content itself.

<head> Elements

The <head> element is just a container for head elements. These element include:

- **<title> -** Specifies the title of the page, which typically is shown in the browser window's title bar
- **<link> -** Allows you to link external resources (usually CSS files that contain rendering rules)
- **<style> -** Allows you to specify CSS rendering rules directly in the page
- **<script> -** Allows you to add JavaScript to the page
- **<meta> -** Can be used to define additional information about the page, such as author information

Content Elements

Inside the `<body>` element can include all the rest of the HTML content markup

Common ones include:

- Headings: `<h1>, <h2>, <h3>, <h4>, <h5>, <h6>`
- Paragraphs: `<p>`
- Grouping elements: `<div>, <span>, <header>, <nav>`
- Phrase elements: `<em>, <strong>, <code>`
- Lists and Tables: `<ul>, <ol>, <li>, <table>`

We'll learn many more today and over the next few lectures

Text and Whitespace

HTML uses “white space collapsing”

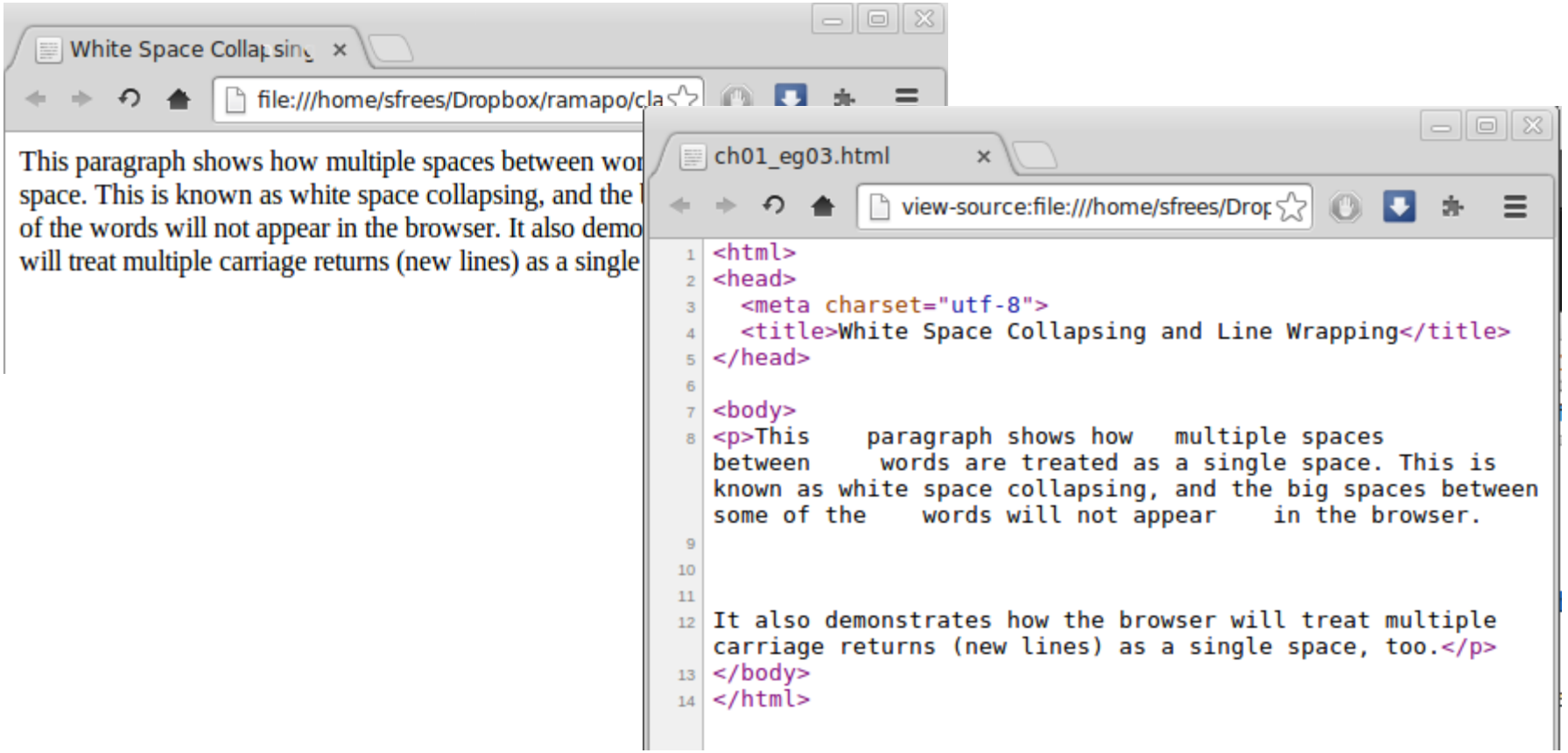
- White space characters (in your HTML content) include the space bar, tab key, return key, etc.
- Consecutive whitespace characters are always collapsed to ONE “space”
- New lines are converted to ONE space

This is actually great news, it means

1. When we write HTML in a text editor, tabs and lines used to keep our file organized will never affect the **rendered** HTML
2. The browser is tasked with handling how word-wrapping is handled.

Text and Whitespace

http://localhost:3000/c01/ch01_eg03.html



Headings

- Headings always appear on a separate line.
- There are 6 levels
  - the browser typically renders headings 1-3 larger than the normal text,
  - h4 is normally the same size
  - h5 and h6 are by default smaller
- Heading elements are **block** elements

http://localhost:3000/c01/ch01_eg04.html

http://localhost:3000/c01/ch01_eg05.html

Text Markup

- The `<p>` element wraps paragraphs. Paragraphs are also **block** elements
- To cause a new line to be rendered *within* a paragraph, you can use the `<br/>` element
  - Note, br is an *empty* element.
- To ask the browser to render your text *exactly* as written, you can use the `<pre>` element

http://localhost:3000/c01/ch01_eg07.html

http://localhost:3000/c01/ch01_eg08.html

`<br/>` and `<pre>` are **inline** elements

Attributes

- Attributes can be attached to elements
  - Attributes are name=”value” pairs
  - Any number (distinct) can appear within the start tag of an element
- Attributes don’t render, they convey additional meaning or options to the element
- Lots of element support specific attributes

  `<a href=“http://google.com“>Google</a>`
- However there are several “universal” attributes

Universal Attributes

id: Allows you to specify a unique "identifier" for the element.

- The value must start with [a-z] or [A-Z] and can be followed by any number of letters, digits, hyphens, underscores, colons, or periods.
- The value must be unique for the entire page
- Often useful when using CSS and JavaScript

Universal Attributes

class: Allows you to specify a "classification" for the element.

- The value adheres to the same rules as **id**
- You can assign multiple classes, separated by spaces

```
<p class="a b c">The paragraph is part of a, b, and c</p>
```

- Again, this is quite useful for CSS and JavaScript

Universal Attributes

class: Allows you attach CSS to the element directly.

- The value is full CSS code, which will be applied by the browser.

```
<p style="color:red">Red text.</p>
```

- Generally this isn't a good way to apply CSS, it creates HTML that is very difficult to maintain.

Universal Attributes

title: Provides a *suggested* title for the element

- The browser often uses this text to display a tooltip when the mouse is hovering over the element
- Note however - its not required to do this!
- It may be used differently with touch-screens
- Can also be used with screen readers

CSS and JavaScript

We won't cover these for a bit - however, we should recognize them...

- `<link rel="stylesheet" href="css/main.css">`
  - Links an external file called main.css found in the css directory (relative to this page).
  - The file should contain CSS instructions

- `<script src="js.main.js"></script>`
  - Links (and executes) a JavaScript file.
  - Note that script is never an empty element!

Block vs. Inline

Block elements: Occupy their own vertical space

Inline elements: Do not disrupt the vertical layout of the block they are contained within

- Block elements can contain other block elements
- Block elements can contain inline elements
- Inline elements can contain other inline elements
- Inline elements **cannot** contain block elements

Grouping

It is important to convey to the browser the organization of the document

Before HTML5, there were only a few ways of doing this.

- and grouped together elements, but did not alter the presentation at all
- They were used instead to work with CSS and JavaScript to “group” elements visually

Grouping

- HTML5 introduces additional grouping elements, which actually have more *meaning* attached to them
  - **<header>** - *vertical grouping of heading elements*
  - **<nav>** - a block element containing navigational elements
  - **<section>** - a block element that should be considered part of the document's outline
  - **<article>** - a block element grouping an "independent content group"

Grouping

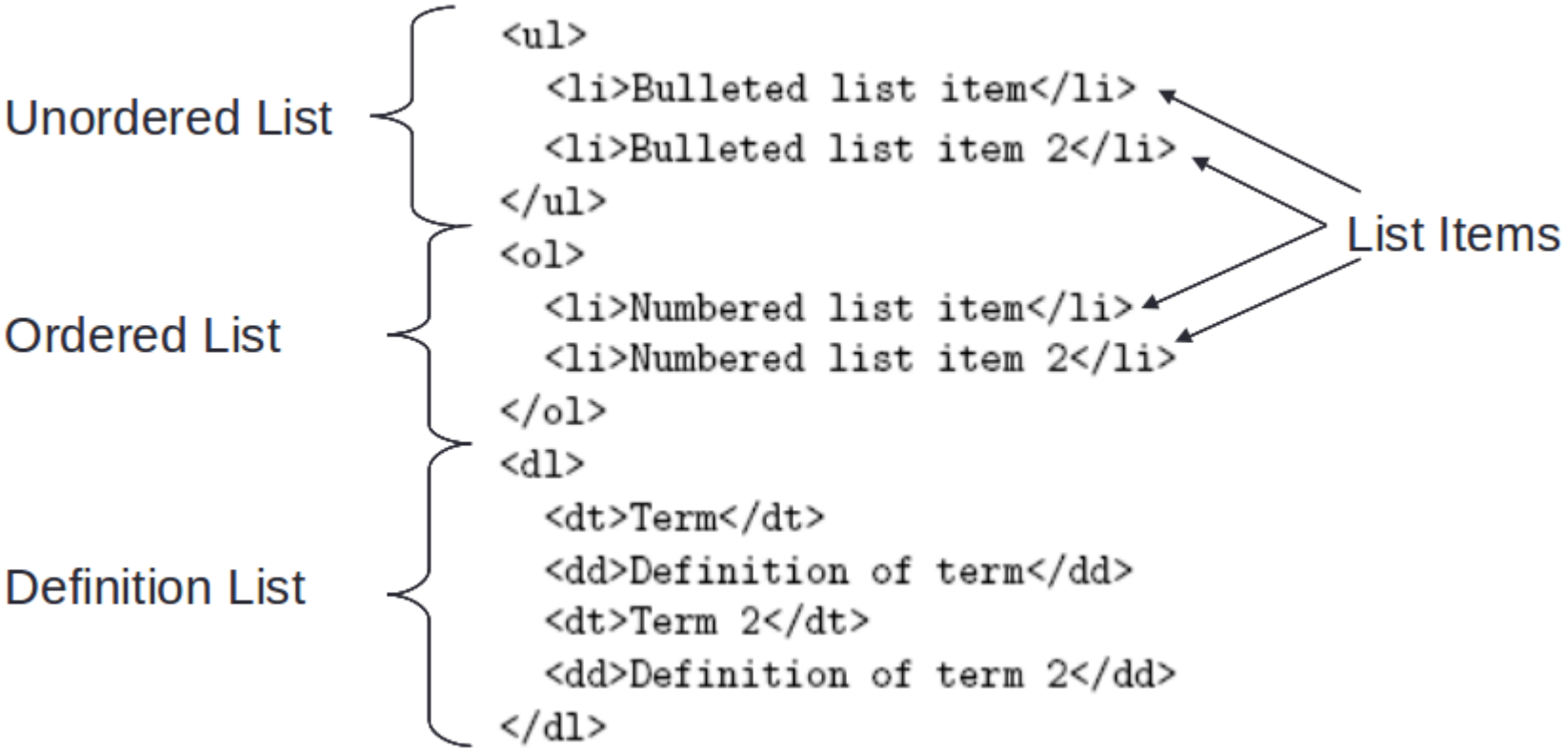
<hr/> - this has been around forever - creates a horizontal line across the page

<blockquote> - a block element which contains text that should be set apart from surrounding content (probably with increased margins)

<aside> - block element that provides supporting information for the text that contains it. Often will be moved aside from the content (although browsers don't do it automatically)

<footer> - block element containing content that would be at the bottom of the page

Lists

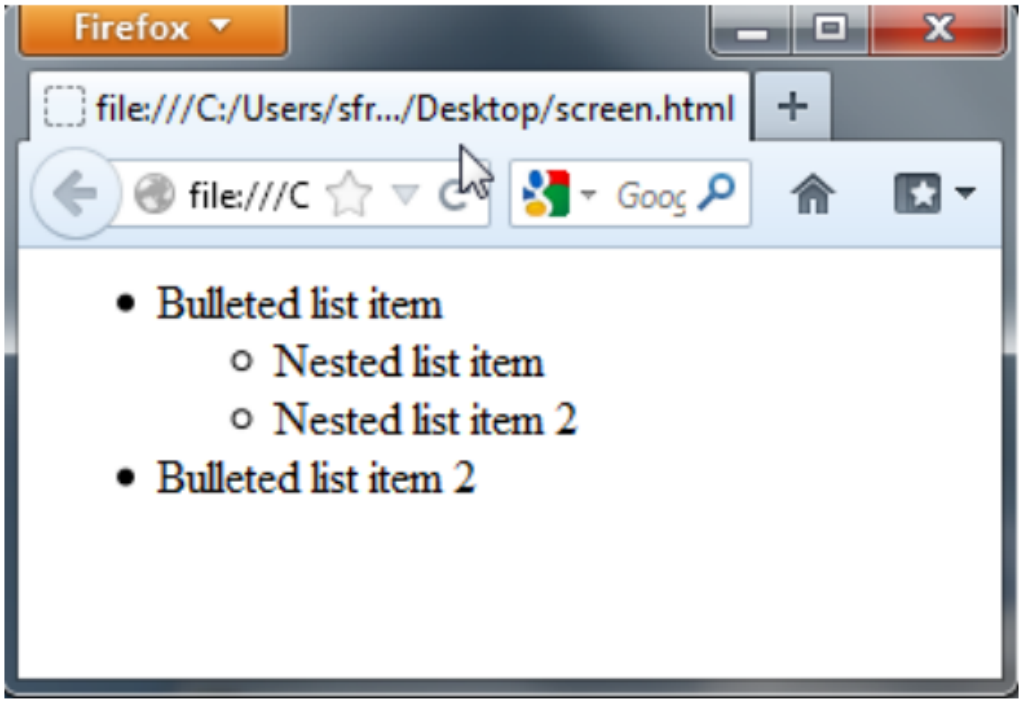

Unordered List
Ordered List
Definition List

Bulleted list item
Bulleted list item 2

Numbered list item
Numbered list item 2

<dl>
<dt>Term</dt>
<dd>Definition of term</dd>
<dt>Term 2</dt>
<dd>Definition of term 2</dd>
</dl>
List Items

Lists




```
<ul>
  <li>Bulleted list item
  <ul>
    <li>Nested list item</li>
    <li>Nested list item 2</li>
  </ul>
 </li>
 <li>Bulleted list item 2</li>
</ul>
```

HTML Text - Chapter 2

Fine-tuning text

Inline elements

There are several elements that do not alter the vertical layout of text, but do affect rendering

- `<em>` - places emphasis on the contained text
  - Browser will typically make this *italics*
- `<strong>` - places strong emphasis on text
  - Browser will typically make this **bold**.

Note, there are also `<i>` and `<b>` inline elements, which do similar things (but not quite as good)

Inline Elements

<small> - used for fine-print

<cite> - contains some sort of citations - normally it is contained within a hyperlink

<q> - an inline quote (will surround with “)

<abbr> - tells the browser you are using an abbreviation

```
<abbr title="HyperText Markup Language>HTML</abbr>
```

<time> - associate a computer-readable time with text

```
<time datetime="2014-1-1">New Years</time>
```

* *see the text book for allowable time formats*

Inline Elements

<figure> and **<figcaption>** - a figure can contain a figcaption and some other block element (code?) or an image

<sup> and **<sub>** wrap superscripts and subscripts

<mark> highlights the text (usually yellow)

Inline Element - just for us...

`<code>` - wrap some code in this and the browser will typically make it pre-formated

`<var>`, `<samp>`, `<kbd>` also help formatting "code"

Why all the elements?

In the good old days (~2008) we had:

```
h1-h6
p
lists
tables
div, span
br and hr
```

HTML5 Elements

- Before HTML 4 standardization, there were a lot of "rendering" tags (i.e <font>)
  - This was generally understood as *bad*.
- However, we want the web to be understandable
- We want machines to be able to understand out documents
- Notice that the current HTML elements mainly convey meaning, relationships, and structure
  - Only indirectly convey rendering preference

Special Characters

There are a number of characters that cannot be in your HTML **content** because HTML uses them in its syntax...

We can replace the characters with numeric or named entities in the text - and they will render correctly

Character	Numeric Entity	Named Entity
“	&#034;	&quot;
&	&#038;	&amp;
<	&#060;	&lt;
>	&#062;	&gt;

Comments

Just like any code you write - its often helpful to add comments

HTML comments must be explicitly closed

```
<p> hello <!-- world --> </p>
```

HTML Text - Chapter 3

Links and Navigation

Hyperlinks

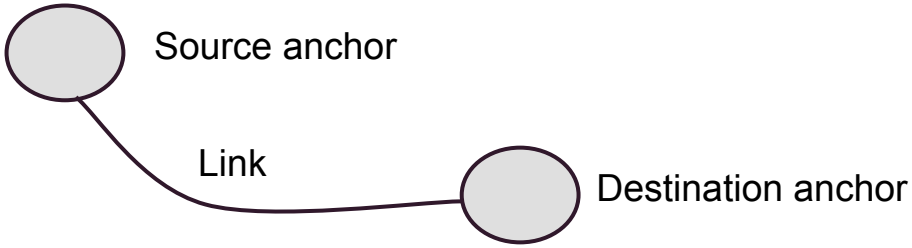
There were actually some alternative "document" languages and strategies when the internet first began… **internet != html**

The reason HTTP and HTML became **the** standard was because of HyperLinks (HyperText)

The idea was around since 1945, but there was never a way to implement it. Its actually ***revolutionary***.

Anchors and Links

You don't put *links* in your HTML - you define **anchor** points



- Both source and destination anchors are declared with the **<a>** element
- A URL (absolute, or relative) can also be an anchor **destination**

Source anchors

- The simplest source anchor is when you want a hyperlink from your page to either an external page or another (internal) page on your site

- The href attribute specifies the destination

```
<a href="details.html">details are here </a>
or go <a href="http://google.com"> find them
yourself</a>
```

- Note - you can have **absolute** or **relative** destinations.

Absolute vs. relative

An absolute href value must have the full scheme, domain, path (a full URL)

A relative URL is prepended with the current page's **base** URL

- This defaults to the page's URL
- Can be overridden by the **<base>** element in the **<head>** of your page.

Always use relative href values for pages on your own site!

Relative links

Your site will often be a directory structure

/ (root)

...html pages at top level...

- (folders with sub-content for html)
- css/
- js/
- images/

You should always use relative links, since you never know when you will deploy or run at a different location (or port!)

Destination anchors

- It is also possible to link to **specific parts** of a destination page - even the current page
- However, this requires an explicit anchor to be placed at the destination
- An destination is specified with the <a> element, but instead of href, use and id.

```
<a id="destination">Destination</a>
```

- Then you can set a link *to* that location with another anchor

```
<a href="#destination">Go to destination</a>
```

http://localhost:3000/c03/ch03_eg06.html

You can combine external and id anchors:

```
<a href="thepage.html#thelocation">The page and location </a>
```

Anchor options

There are several attributes you can place on a source anchor to alter the browser behavior

target attribute controls where the new page should open

_blank	Opens the linked page in a new window or tab
_self	Opens the linked page in the same window/tab (default)
_parent	Opens the linked page in the full body of the window (suitable for links in pop-up pages…!)

See page 68 in text for more attributes

HTML Text - Chapter 4

Images, Audio, and Video

Images

- We need to discuss how images work:
  - How do you put them on your page?
  - How do you make them into links?
  - What image format should you use?
  - What are some best-practices?

<img> element

The <img> element is an *inline* element

- It has 4 critical attributes
  - **src** - specifies the URL where the image can be found
  - **alt** - text content describing the image
    - Will be shown if image fails to load
    - Can be used by search engines
  - **height** and **width** - lets you specify dimensions (in pixels)

```
<img src="photo.png" alt="A cool picture" height="200" width="500"/>
```

http://localhost:3000/c04/ch04_eg02.html

HTTP and the image element

```
<!DOCTYPE html>
<html>
    <head>
        <title>Pictures</title>
    </head>
    <body>
        <img src="images/apple.jpg" width="130" height="130" >
        <img src="images/apple.jpg" width="160" height="160" >
        <img src="images/orange.jpg" width="80" height="150" >
    </body>
</html>
```

- **There are 3 HTTP round trips here**.
  - One to get the HTML, one to get apple.jpg, and the other for orange.jpg.
    - They are distinct HTTP request/responses - the browser initiates new requests for any linked (src attribute) resources.
    - apple.jpg won't be requested twice - the browser just draws it twice (at different sizes)

Height and Width

When we cover CSS, we'll see you could also specify height and width there - however

- For images - you should always use the primary element attributes. Why?

Be careful about sizing your images!

- Use image editing software to resize your image before linking to your HTML
- Don't use your "customer's" browser... its not good at resizing, and it costs time and money!

If you upload an 800x800 image, and show it on your HTML at 200x200, the browser still downloads 800x800!

Images as Links

A <a> element typically has text as its content - but it can have an <img> as its child as well.

```
<a href="http://www.apple.com">
  <img src="images/apple.jpg" width="130" height="130" >
</a>
```

As with any link, the image itself will take on the "pointer" cursor when the mouse hovers over it.

Image formats

Image formats matter, because they affect the quality and amount of bytes an image uses

- **GIF** - The file stores a color map of up to 256 colors. Very efficient, not great if you have lots of colors/gradients.
  - Supports transparent backgrounds
- **JPEG** - Lots of compression variability, best for high resolution, detailed photographs
- **PNG** - Better than GIF, use it for anything other than large, detailed photographs

Avoid formats like TIFF - some browsers will not be able to render them.

Other media

There was a time where Adobe Flash was essentially **the** media provider on the web.

- With the advent of HTML5, this is no longer the case.
- We're going to cover strategies that only work on:
  - Chrome, Safari, Firefox
  - Internet Explorer 9 and above
  - Lets pretend IE 6, 7, and 8 don't exist for a while...

Note - Flash is still "easy", but IOS doesn't support it... which effectively means you shouldn't use it!

Easy way out.... YouTube

- Video files are big, and need to be uploaded to your web server, just like linked images.
- A very common, and free, approach is to upload your videos to YouTube instead
- YouTube gives you a snippet of HTML

```
<iframe
    width="560"
    height="315"
    src="//www.youtube.com/embed/9gTw2EDkaDQ">
</iframe>
```

Use YouTube?

- Advantages:
  - You aren't using your own bandwidth
  - High Speed
  - Near universal browser support

- Problems:
  - Your content (?)… Google's server
  - There are significant privacy and intellectual property concerns with using YouTube in a commercial application

The <video> element

```
<video width="720" height="480" src="central.mp4"/>
```

Note - its essentially the same as the <img> element...

Additional attributes

- poster - specify image to show until video is loaded/started
- preload, autoplay, loop, muted, controls - all boolean attributes

http://localhost:3000/c04/ch04_eg10.html

Audio

Audio (HTML5) is also quite easy

```
<audio src="audio/white_noise.mp3"/>
```

Codec Nightmare

This looks like a very happy situation - however it is not.

- For both video and audio, the file formats involve highly complex compression of binary data
- A **codec** is a program that encodes/decodes these binary formats.
- Unfortunately, there is no **single** format for video and audio that everyone has agreed to build into their browsers!

Codecs

Proprietary Formats

Video: h.264 and MPEG4

Audio: MP3

- Protected by patents, and charge royalties to the implementing software (browser)
- Can support DRM

"Open Formats"

Video: WebM and OGG/Theora

Audio: OGG/Vorbis

Codecs

Microsoft and Apple support h.264, MPEG4, and MP3. They do not provide support for open standard codecs in their browsers.

Google, Opera, and Mozilla (Firefox) do not support proprietary formats

- *Chrome does have support on Windows and Mac, but not on Linux.*
- *The situation is evolving with Firefox...*

Current situation (w3schools.com)

Browser	MP4	WebM	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox Firefox v26 released as of Jan 2014	NO **Update:** Firefox 21 running on Windows 7, Windows 8, Windows Vista, and Android now supports MP4	YES	YES
Safari	YES	NO	NO
Opera	NO	YES	YES

Browser	MP3	Wav	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	NO **Update:** Firefox 21 running on Windows 7, Windows 8, Windows Vista, and Android now supports MP3	YES	YES
Safari	YES	YES	NO
Opera	NO	YES	YES

...oh, wait...

There are a lot of people still using Internet Explorer 6, 7 and 8!

IE < 9 simple does not support HTML5 (it barely "supports" 4!)

For these people, you are going to need to use Flash or some other format

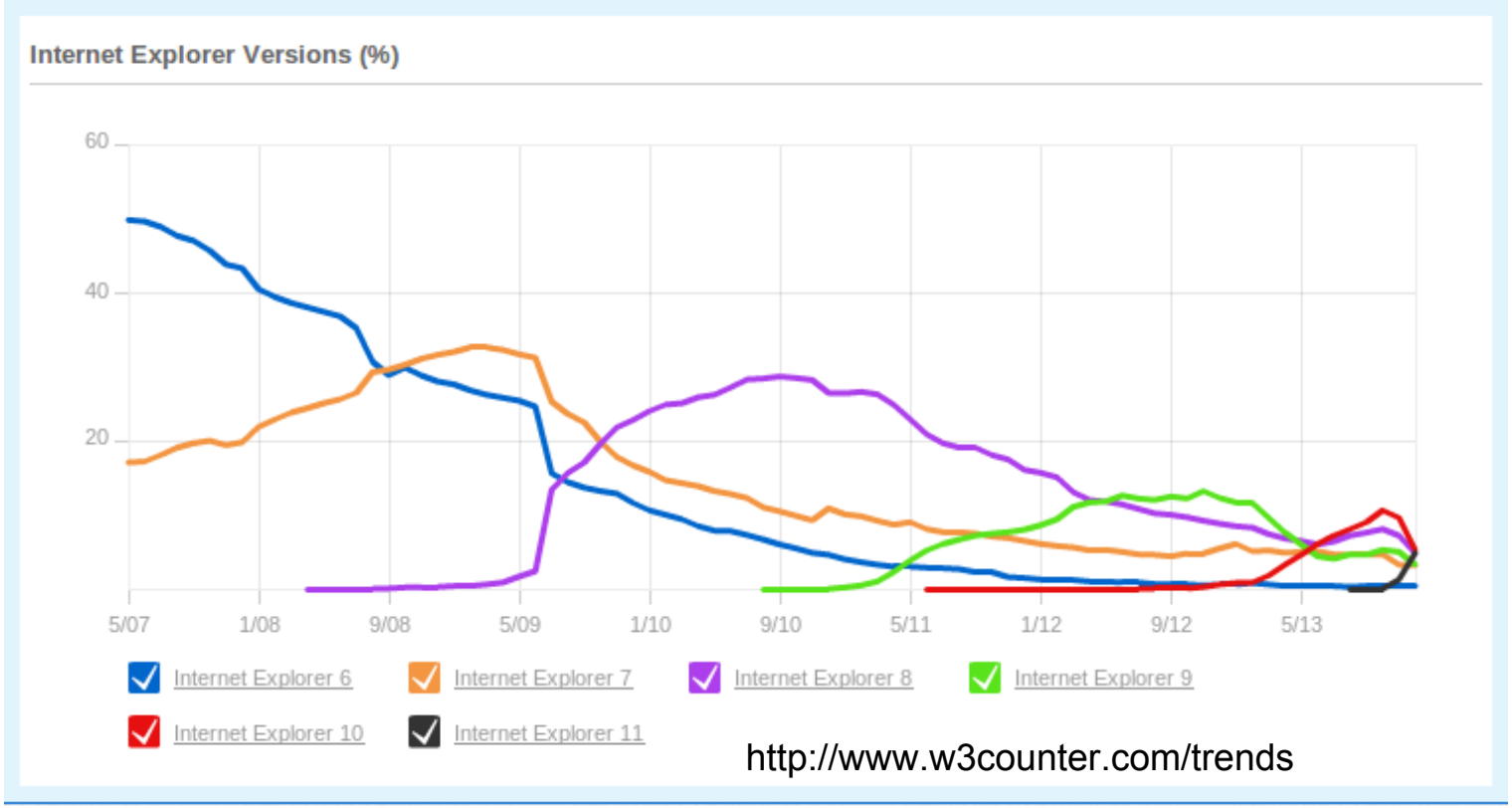
See pages 100-103 in the HTML text for details

Count your blessings… things used to be a lot worse...

Internet Explorer

Most web developers equate Internet Explorer with the antichrist.

IE 10+ works well though, as long as you use your <!doctype>!



HTML Text - Chapter 5

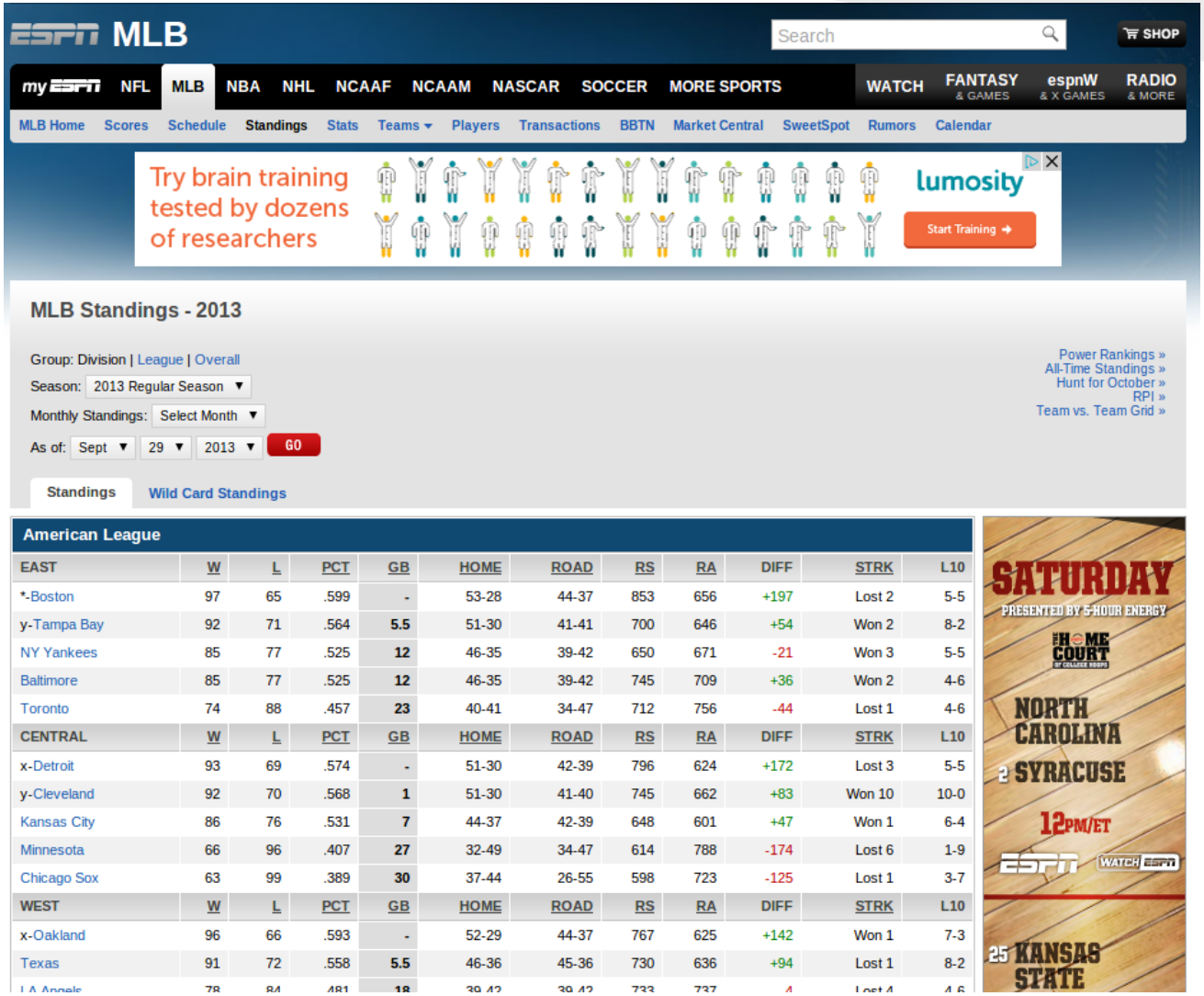
Tables

Tables...

Some people shriek in horror when you say you are going to use a table...

- There was a time where **many** people controlled page layout with tables.
- Using tables to control page layout is a *horrible* solution - CSS is for layout
- However - lots of HTML documents truly contain tabular data - and that's when you want to use a table!

To table, or not to table…



ESPN MLB

Search SHOP

my ESPN | NFL | MLB | NBA | NHL | NCAAF | NCAAM | NASCAR | SOCCER | MORE SPORTS | WATCH | FANTASY & GAMES | espnW & X GAMES | RADIO & MORE

MLB Home | Scores | Schedule | Standings | Stats | Teams | Players | Transactions | BBTN | Market Central | SweetSpot | Rumors | Calendar

MLB Standings - 2013

Group: Division | League | Overall
Season: 2013 Regular Season
Monthly Standings: Select Month
As of: Sept 29 2013 GO

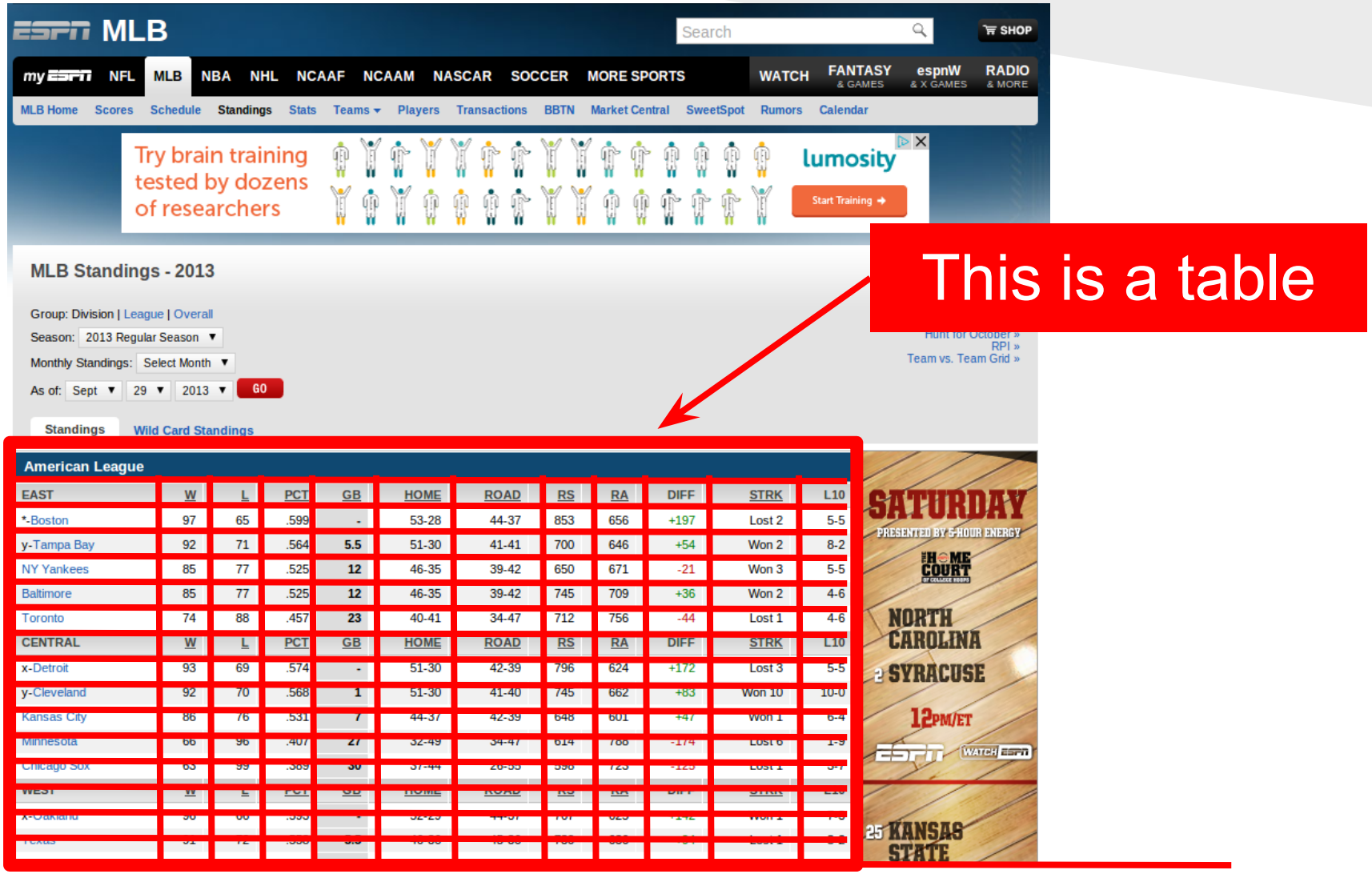
Power Rankings »
All-Time Standings »
Hunt for October »
RPI »
Team vs. Team Grid »

Standings | Wild Card Standings

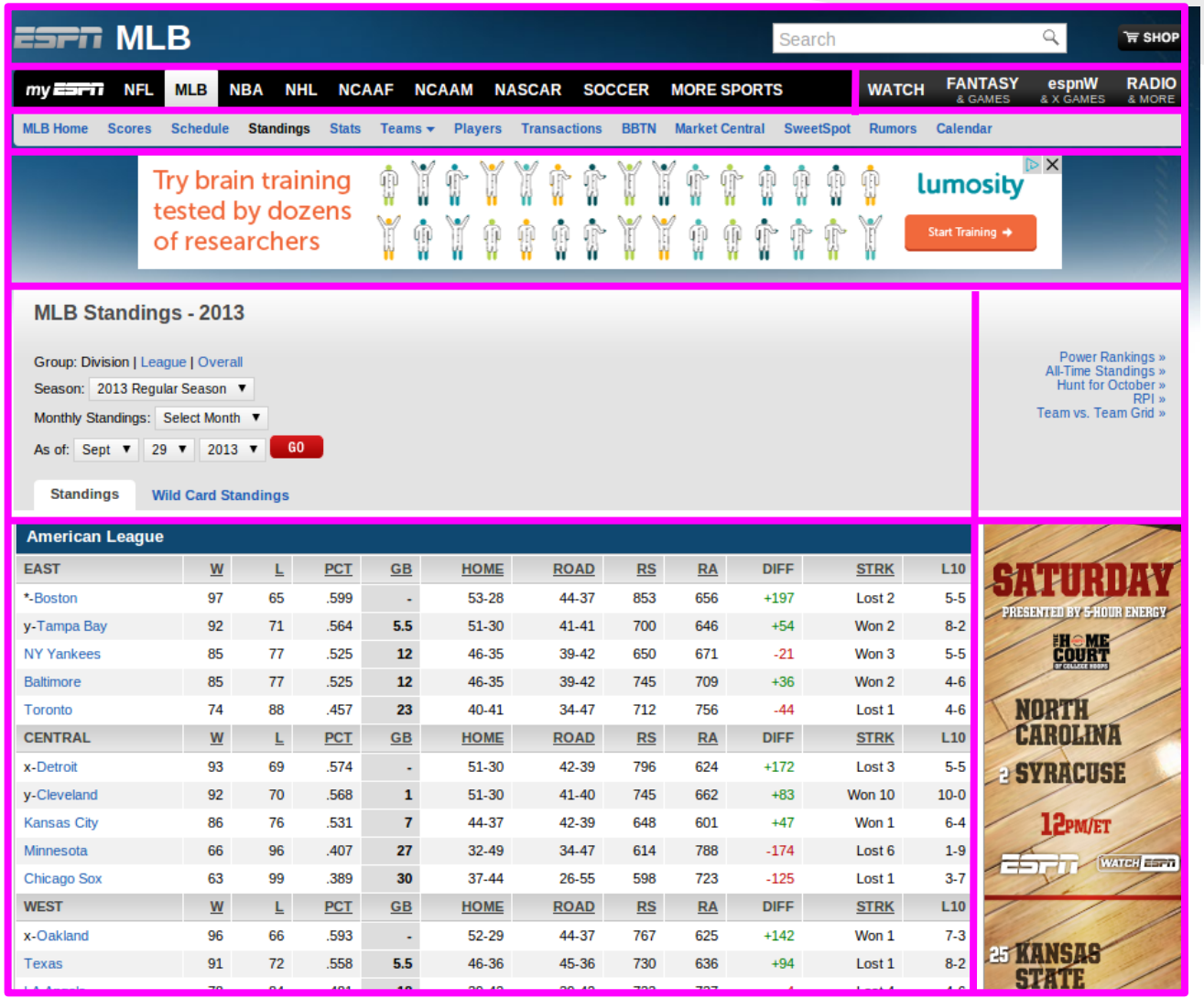
American League

EAST	W	L	PCT	GB	HOME	ROAD	RS	RA	DIFF	STRK	L10
*-Boston	97	65	.599	-	53-28	44-37	853	656	+197	Lost 2	5-5
y-Tampa Bay	92	71	.564	5.5	51-30	41-41	700	646	+54	Won 2	8-2
NY Yankees	85	77	.525	12	46-35	39-42	650	671	-21	Won 3	5-5
Baltimore	85	77	.525	12	46-35	39-42	745	709	+36	Won 2	4-6
Toronto	74	88	.457	23	40-41	34-47	712	756	-44	Lost 1	4-6
CENTRAL	W	L	PCT	GB	HOME	ROAD	RS	RA	DIFF	STRK	L10
x-Detroit	93	69	.574	-	51-30	42-39	796	624	+172	Lost 3	5-5
y-Cleveland	92	70	.568	1	51-30	41-40	745	662	+83	Won 10	10-0
Kansas City	86	76	.531	7	44-37	42-39	648	601	+47	Won 1	6-4
Minnesota	66	96	.407	27	32-49	34-47	614	788	-174	Lost 6	1-9
Chicago Sox	63	99	.389	30	37-44	26-55	598	723	-125	Lost 1	3-7
WEST	W	L	PCT	GB	HOME	ROAD	RS	RA	DIFF	STRK	L10
x-Oakland	96	66	.593	-	52-29	44-37	767	625	+142	Won 1	7-3
Texas	91	72	.558	5.5	46-36	45-36	730	636	+94	Lost 1	8-2
LA Angels	78	84	.481	18	39-42	39-42	733	737	-4	Lost 4	4-6

To table, or not to table…



To table, or not to table...



EAST	W	L	PCT	GB	HOME	ROAD	RS	RA	DIFF	STRK	L10
*-Boston	97	65	.599	-	53-28	44-37	853	656	+197	Lost 2	5-5
y-Tampa Bay	92	71	.564	5.5	51-30	41-41	700	646	+54	Won 2	8-2
NY Yankees	85	77	.525	12	46-35	39-42	650	671	-21	Won 3	5-5
Baltimore	85	77	.525	12	46-35	39-42	745	709	+36	Won 2	4-6
Toronto	74	88	.457	23	40-41	34-47	712	756	-44	Lost 1	4-6
CENTRAL	**W**	**L**	**PCT**	**GB**	**HOME**	**ROAD**	**RS**	**RA**	**DIFF**	**STRK**	**L10**
x-Detroit	93	69	.574	-	51-30	42-39	796	624	+172	Lost 3	5-5
y-Cleveland	92	70	.568	1	51-30	41-40	745	662	+83	Won 10	10-0
Kansas City	86	76	.531	7	44-37	42-39	648	601	+47	Won 1	6-4
Minnesota	66	96	.407	27	32-49	34-47	614	788	-174	Lost 6	1-9
Chicago Sox	63	99	.389	30	37-44	26-55	598	723	-125	Lost 1	3-7
WEST	**W**	**L**	**PCT**	**GB**	**HOME**	**ROAD**	**RS**	**RA**	**DIFF**	**STRK**	**L10**
x-Oakland	96	66	.593	-	52-29	44-37	767	625	+142	Won 1	7-3
Texas	91	72	.558	5.5	46-36	45-36	730	636	+94	Lost 1	8-2

This is NOT a table!!!

<table> should be used to show a table of information, that logically takes on rows and columns.

CSS should be used to set up the *layout* of components on your page.

<table> element

All tables have a **body** - <tbody>

- Table bodies are divided first into rows - <tr>
- Each row is divided into columns (cells) - <td>

Optionally (and recommended) a table can have a heading and footer

- <thead> - can use <th> instead of <td>
- <tfoot> - goes before <tbody>, but drawn at the bottom

<caption> is used for a title on the table

Table Example

Transaction date	Payment type and details	Paid out	Paid in	Balance
12 Jun 12	Amazon.com	$49.99		$8411.16
13 Jun 12	Total	$60.00		$8351.16
14 Jun 12	Whole Foods	$75.28		$8275.88
14 Jun 12	Visa Payment	$350.00		$7925.88
15 Jun 12	Cheque 122501		$1450.00	$9375.88
17 Jun 12	Murco	$60.00		$9315.88
18 Jun 12	Wrox Press		$1000.00	$10315.88
18 Jun 12	McLellans Bakery	$25.00		$10290.88
18 Jun 12	Apple Store	$1350.00		$8940.88
		$1970.27	$2450.00	$8940.88

http://localhost:3000/c05/ch05_eg05.html

Tables - addition options

You can develop quite complex tables

- You can group columns together with `<colgroup>`
  - Specified outside of `tbody`
  - Useful for applying CSS
- You can make cells (`td`) span multiple columns or rows using `colspan` and `rowspan` attributes
- You can even nest tables (be careful, things start to get messy!)

Next

We've spent a lot of time talking about presentation of pages with HTML...

HTML can be interactive, through **forms**, which send data from the user to the server for processing...

However, forms aren't that exciting if your server doesn't work with the data..

Next

Before covering forms, we're going to take a detour into JavaScript

We'll learn

- The fundamentals of JavaScript
- How to capture data from a form in Node.js
- How to display some dynamic data on our HTML page using EJS rendering templates

You need to read Chapter 1-8 in the JavaScript text book over the next 2 weeks!