Document Architecture:
HTML & CSS

Noah Mendelsohn
Tufts University
Email: noah@cs.tufts.edu
Web: http://www.cs.tufts.edu/~noah
What you should get from today’s session

- Quick look at design goals for HTML
- Learn how to write a simple Web page using HTML
- Learn the basics of using CSS for styling
- Learn how to validate your HTML and CSS
- Learn about higher level design issues relating to document formats
Quick Review of Web Architecture
Architecting a universal Web

- Identification: URIs
- Interaction: HTTP
- Data formats: HTML, JPEG, GIF, etc.
Three pillars of Web Architecture

URI is http://webarch.noahdemo.com/demo1/test.html

HTTP GET

Identification with URIs

HTTP RESPONSE

demo1/test.html
Host: webarch.noahdemo.com
Three pillars of Web Architecture

URI is http://webarch.noahdemo.com/demo1/test.html

Interaction Using HTTP

HTTP GET

demo1/test.html
Host: webarch.noahdemo.com

HTTP RESPONSE
Three pillars of Web Architecture

Representations using media types like text/html, image/jpeg, etc
Architecting a universal Web

- Identification: URIs
- Interaction: HTTP
- Data formats: HTML, JPEG, GIF, etc.
Introduction to HTML
HTML Goals

- Ubiquity – Metcalfe’s law (previous lecture)
- Usable for a very wide range of *document* content – new and legacy
- Easy to learn
- To be the preferred format for Web publishing
- Usable with many languages (French, Chinese, English, etc.)
- Same HTML renders on wide range of devices – can look different
- Extensible…
- …and forwards compatible (old browsers still work!)
- A framework for Web *applications*

We will dive deeper into these characteristics later in the term.
A simple HTML document

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #1</title>
</head>
<body>
<h1>A very simple Web page</h1>
</body>
</html>
```
A simple HTML document

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #1</title>
</head>
<body>
<h1>A very simple Web page</h1>
</body>
</html>
```

Identifies this as an HTML document
(DOCTYPE optional, but for COMP 150-IDS, please provide it)
Matched *tags*

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #1</title>
</head>
<body>
<h1>A very simple Web page</h1>
</body>
</html>
```
Nested matched tags

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #1</title>
</head>
<body>
<h1>A very simple Web page</h1>
</body>
</html>
```
A simple HTML document

The whole matched grouping is called an *element*

```html
<!DOCTYPE html>
<html>
<head>
  <title>Demo #1</title>
</head>
<body>
  <h1>A very simple Web page</h1>
</body>
</html>
```
A simple HTML document

The <head> element has general information about the document

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #1</title>
</head>
<body>
<h1>A very simple Web page</h1>
</body>
</html>
```
A simple HTML document

The `<title>` element gives a title – usually shown in browser title bar

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #1</title>
</head>
<body>
<h1>A very simple Web page</h1>
</body>
</html>
```
A simple HTML document

The `<body>` element is the actual content

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #1</title>
</head>
<body>
<h1>A very simple Web page</h1>
</body>
</html>
```
Headings

`<h1>` is a top-level heading

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #3</title>
</head>
<body>
<h1>A very simple Web page</h1>
<p>We've added a paragraph</p>
</body>
</html>
```

<h2>, <h3> etc. for lesser headings
Paragraphs are the main text content

<p>is a paragraph</p>

<!DOCTYPE html>
<html>
<head>
<title>Demo #3</title>
</head>
<body>
<h1>A very simple Web page</h1>
<p>We’ve added a paragraph</p>
</body>
</html>

A very simple Web page
We’ve added a paragraph
Elements can have *attributes*

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #2</title>
</head>
<body>
<h1 id="simple">A very simple Web page</h1>
</body>
</html>
```

Elements *may* have one or more attributes...attribute values (typically) in matched quotes.
The *id* attribute

The *id* attribute gives the element a name…id’s *must* be unique within a document

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #2</title>
</head>
<body>
<h1 id="simple">A very simple Web page</h1>
</body>
</html>
```

http://example.org/demo2.html#simple

...because ID’s can be used in URI fragment identifiers
Self-closing tags

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #4</title>
</head>
<body>
<h1>A very simple Web page</h1>
<img src="noah.jpg" alt="picture of Noah">
<p>We’ve added a paragraph</p>
</body>
</html>
```

Some tags are or may be self-closing (there’s no `</img>` in this document)
<html>
<head>
<title>Demo #4</title>
</head>
<body>
<h1>A very simple Web page</h1>
<img src="noah.jpg">
<p>We’ve added a paragraph</p>
</body>
</html>
Line breaks

<!DOCTYPE html>
<html>
<head>
<title>Demo #4</title>
</head>
<body>
<h1>A very simple Web page</h1>
<img src="noah.jpg">
<p>We’ve added
<br>
a paragraph</p>
</body>
</html>

<br> starts new line, but not new para
Marking up text with nested elements

<!DOCTYPE html>
<html>
<head>
<title>Demo #5</title>
</head>
<body>
<h1>A very simple Web page</h1>
<p>We can have markup <em>within</em> text!</p>
</body>
</html>

<em>is for “emphasis”

mixed content – more on that later.
Some other important HTML features

- Lists: `<ul>` & `<ol>`
- Tables: `<table>`
- Block layout and formatting: `<div>`
- Input forms and submission: `<form>`
- Graphics: `<svg>` & `<canvas>`
- Video: `<video>`
- Tables: `<table>`
- Programmable documents and Web apps: `<script>`
- Etc.
Links
Linking to other documents gives the Web its richness!

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #6</title>
</head>
<body>
<h1>A very simple Web page</h1>
<p>We can link to <a href="http://example.org/another.html">another document</a>.</p>
</body>
</html>
```

<a> is for “anchor”…this is how you put in links
Linking to other documents gives the Web its richness!

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #5</title>
</head>
<body>
<h1>A very simple Web page</h1>
<p>We can link to another document.</p>
</body>
</html>
```

`href =` identifies the document we’re linking.
Linking to other documents gives the Web its richness!

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #5</title>
</head>
<body>
<h1>A very simple Web page</h1>
<p>We can link to <a href="http://example.org/another.html">another document</a>.</p>
</body>
</html>
```
Links don’t have to be text

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #7</title>
</head>
<body>
<h1>A very simple Web page</h1>
<p>Click on picture to follow link:
<a href="http://www.cs.tufts.edu/~noah/">
<img src="noah.jpg" alt="picture of Noah">
</a>.</p>
</body>
</html>
```
HTML is *compositional*

```html
<!DOCTYPE html>
<html>
<head>
<title>Demo #7</title>
</head>
<body>
<h1>A very simple Web page</h1>
<p>Click on picture to follow link:
<a href="http://www.cs.tufts.edu/~noah/">
<img src="noah.jpg" alt="picture of Noah">
</a>
</p>
</body>
</html>
```

<code>&lt;img&gt;</code> can appear in
- Paragraphs
- Links
- Table cells
- &lt;divs&gt;
- Etc.

Recursion
- Tables within tables
- Tables in &lt;divs&gt;
- &lt;divs&gt; in tables
- Images in content in lists in tables
HTML References

- The official HTML5 specification: HTML5: Edition for Web Authors (http://www.w3.org/TR/html5-author/)
  - This is long, detailed and often hard to understand, but it is authoritative

- There are tons of good books on HTML – check Safari or your favorite bookseller for one that works for you

- A Web search will get you good (and bad!) advice on almost anything you might want to do

- A useful, if informal guide to new features in HTML5: Dive into HTML5, by Mark Pilgrim http://diveintohtml5.info/table-of-contents.html
  - Most of these won’t be of interest when you’re starting out, but you might want to take a look

Note: the Web site www.w3schools.com is not affiliated with W3C, and does not convey official advice from the consortium!
Introduction to CSS
HTML and CSS: content and styling

- HTML conveys the logical structure and content of a document
- Each element is given default styling by the browser
  - E.g. `<h1>` is usually in a big font and boldface
- Cascading Stylesheets (CSS) can be used to override the presentation details of any element!
A simple bit of CSS on an element

```
<!DOCTYPE html>
<html>
<head>
<title>Demo #8</title>
</head>
<body>
<h1 style="color:blue">A very simple Web page</h1>
</body>
</html>
```

**color:blue** overrides the color style for this `<h1>`
What you can control using CSS

- color:red
- font-weight:bold
- font-style:italic
- font-size:20px
- background:yellow
- border:solid
- text-align:center
- border:dashed; text-align:center;

Example:
<p style="text-align:center; border:solid; color:red;">CSS is cool!</p>

CSS is cool!
What you can control using CSS

- color:red
- `font-weight:bold`
- `font-style:italic`
- `font-size:20px`
- background:yellow
- border:solid
- text-align:center
- border:dashed; text-align:center;

**How things are positioned:**
- margins: around the outside
- padding: space between content and bounding box
- float: forcing a box right or left, with wraparound

**Advanced**
- Animations
- Opacity
- Different stylesheet for mobile, print, small screen

And lots, lots, lots more!
Where can you put your CSS?

- **On an element**
  - `<p style="color:blue">A blue paragraph</p>`

- **At the top of your HTML file**
  - `<html>
    <head>
      <style type="text/css">
      p {color:blue;} /* all paragraphs blue */
      </style>
    </head>
    <body>
    <p>This will be blue!</p>
    </body>
  </html>

- **In an external stylesheet file**
  - `<html>
    <head>
      <link rel="stylesheet" type="text/css"
      href="http://www.cs.tufts.edu/comp/150IDS/style/150ids.css" />
    </head>
    </html>`

Use sparingly for local changes you won’t want to override

Can be useful to make sure formatting stays w/file, e.g. offline

External stylesheets are usually best practice
CSS Selectors
A simple CSS selector

`p` "selects" all `<p>` elements

```html
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
p {color:red;}
</style>
</head>
<body>
<h1>A very simple Web page</h1>
<p>This will be red</p>
<p>…and this will be red</p>
<p>…and this too</p>
</body>
</html>
```
A simple CSS selector

p “selects” all <p> elements

A very simple Web Page

This will be red
…and this will be red
…and this too

<!DOCTYPE html>
<html>
<head>
<title>Demo #9</title>
<style type="text/css">
p {color:red;}
</style>
</head>
<body>
<h1>A very simple Web page</h1>
<p>This will be red</p>
<p>…and this will be red</p>
<p>…and this too</p>
</body>
</html>
Using element *classes* with CSS

p.hilite “selects” all <p class=“hilite”> elements

A very simple Web Page

This will be red
...this won’t...
...but this will!

You can make up your own class names
Classes can be used with multiple element types

.hilite “selects” all <... class=“hilite”> elements

A very simple Web Page

This will be normal
This will be red and italic
This will be normal

<!DOCTYPE html>
<html>
<head>
<title>Demo #10</title>
<style type="text/css">
.hilite {font-style:italic;
color:red;}
</style>
</head>
<body>
<h1 class="hilite">A very simple Web page</h1>
<p>This will be normal</p>
<p class="hilite">This will be red and italic</p>
<p>This will be normal</p>
</body>
</html>
Using element *identifiers* with CSS

**p#somep** “selects” a single `<p id="somep">` element

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
p#somp {color:red;}
</style>
</head>
<body>
<h1>A very simple Web page</h1>
<p>Normal</p>
<p id="somep">…red…</p>
<p>…normal again</p>
</body>
</html>
```

*A very simple Web Page*

Normal

...red...

...normal again

*Id’s must be unique in each document
By the way: &lt;a href="doc#somep"> links the paragraph*
<span> is useful for styling within text runs

<span> can be styled, but does not introduce line breaks

A very simple Web Page
This is an interesting paragraph.

<!DOCTYPE html>
<html>
<head>
<style type="text/css">
.hilite {font-style:italic; color:red;}
</style>
</head>
<body>
<h1>A very simple Web page</h1>
<p>This is an <span class="hilite">interesting</span> paragraph!</p>
</body>
</html>
CSS Reference Material

- CSS work has been modularized into many separate specifications
  - E.g. selectors are separate from the main language

- W3C Home Page for CSS: [http://www.w3.org/Style/CSS/Overview](http://www.w3.org/Style/CSS/Overview)
  - You can find specifications and other useful material from there

- Many, many tutorials on the Web – search for the feature you want
  - E.g. “CSS Animations”

- View Source
  - If the HTML source has a link to the stylesheet, most browsers let you follow that
Some Things to Note about HTML & CSS
HTML illustrates some key points of document design

- **It’s a text format, not binary**
  - Easy to generate from: editors, programs, scripts, etc.
  - Want to learn how something’s done? Just View Source on a page that does it!

- **Semantic markup**
  - `<H1>` means “This is a top level header”, not “Make this big and bold”
  - Use tags like `<table>` when you have tabular data, not to fudge screen layout
  - Why? Tools, search engines, transcoders etc. can tell more about your document
  - See principle of model/view separation (previous slide)

- **Almost all the formatting of HTML elements is now defined in terms of CSS properties**

- **Model/view separation**
  - Encode content separately from presentation
  - *Use CSS for formatting wherever practical!*
  - An important principle of software design: *supports information extraction & reuse*
  - An example of the broader principle of “separation of concerns”
HTML: Some other things to note

- **Mixed content** is deeply interesting
  - Strings have `<em>` nested structure `</em>` that’s marked up in the string
  - *Many types of documents need this richness*
  - Most important example: hyperlinks
  - *Traditional string types don’t handle this well – hard to import HTML & XML into C, Java, etc.*

- **Linkability**
  - It’s easy to turn most HTML content into a link
  - It’s easy to link not just to an HTML document, but to content within a document
  - *Metcalfe’s law at work again: the power of linking*

- **Extensible**
  - New tags are added over time
  - There’s a rule: older browsers ignore new tags but not the content of those tags
  - Who gets to define new tags is the subject of much controversy
More that may not be obvious about HTML & CSS

- **HTML can do a lot without JavaScript**
  - We’ll explore why that’s important later
  - Note that there is no JavaScript on any of the COMP 150-IDS course pages (so far!)
  - *Please do not use JavaScript in your course submissions for now!*

- **CSS and HTML are Declarative Languages**
  - Not “Do step1 then step2”, but rather “make it so” (e.g. make sure this is centered, make sure it’s blue, etc.)
  - Neither CSS nor HTML is imperative or Turing-complete
  - We’ll discuss later the advantages of declarative languages

- **Constraint-based programming is cool**
  - “Center this” vs. “FirstCharPos = window.left + window.size/2 – string.width/2”
Testing your HTML Files
It’s easy to try out your HTML files

- Create a file on your local machine
- Use your operating system’s convention for file extensions (usually .html)
- If you open the file, your OS will use your default browser
- Which browser?
  - Any up-to-date one should do
  - I recommend Firefox, Safari and/or Chrome
  - For the features we use, compatibility should be quite good

- HTML & CSS Debugging
  - Firefox: right click and “Inspect Element”, or install Firebug addin
  - Safari: Preferences -> Advanced -> Show develop menu
  - Chrome: CTRL-SHIFT-I

- Links
  - Relative links will resolve to your local filesystem
  - Most others will attempt to retrieve from the Web
  - Make sure your stylesheet links will resolve if you’re testing locally
Debugging w/Firefox Inspect Element

A very simple Web page

We can have markup in text!
We can link to another document.
Is your HTML file correct?

1. Go to validator.w3.org
2. Enter the URI for your page
3. Press “Check” to validate
4. You can ignore warning that the HTML5 validator is experimental
Is your CSS file correct?

1. Much easier to check if HTML & CSS are on Web.

2. Go to http://jigsaw.w3.org/css-validator/

3. Enter the URI for your HTML page or your CSS sheet

4. Press “Check” to validate CSS
Validating HTML from your hard drive

1. Go to validator.w3.org
2. Select: Validate by File Upload
3. Browse to find HTML on your local drive
4. Press “Check” to validate

Warning: the validator can find linked CSS, etc. if it’s on the Web and referenced with an absolute URI. It will not go back to your hard drive to find more!
Homework scores may be reduced for any HTML or CSS file you submit that doesn’t validate!
Publishing your HTML Files
Steps for publishing an HTML file at Tufts

- Copy your file to `~yourUTLN/public_html/filename.html` on `linux.eecs.tufts.edu`
- Make sure there are public “execute” permissions on the `public_html` directory (and any subdirectories if you used them):
  ```bash
  chmod o+x ~yourUTLN/public_html
  ```
- Make sure there are public “read” permissions on the file itself:
  ```bash
  chmod o+r ~yourUTLN/public_html/filename.html
  ```
- If you’ve done this right, your file should be accessible from Web browsers all over the world as:
  ```url
  http://www.eecs.tufts.edu/~yourUTLN/filename.html
  ```

*If it doesn’t work, there’s a good chance you didn’t set the permissions right!*
If it’s not working, you’ll want to check permissions

- `ls -ld <dirname>`: lists permissions on named directory itself
- `ls -l <dirname>`: lists permissions on the files in the directory
- **Sample output:**

```bash
$ cd public_html
$ ls -l testfile.html
-rw-r--r--+ 1 noah None 291328 Aug 25 20:55 testfile.html

$ ls -ld public_html
drwx--x--x+ 1 noah None 0 Aug 29 19:56 public_html

Directory is executable by anyone

$ cd public_html
$ ls -l testfile.html
-rw-r--r--+ 1 noah None 291328 Aug 25 20:55 testfile.html

HTML file is readable by anyone

All this is needed so the Apache Web server can get at your file.