How the web works today.

Infrastructure

The Internet isn't much more than what Senator Ted Stevens (R-Alaska) memorably described it in 2006, a "series of tubes."

It's decentralized. Anyone can set up a server; often for non-profits the cheapest thing to do (though not the safest) is to set up a server in-house.

1. How do you publish a web site?

    - Local hosting: plug a computer into a wall and have an ISP (Internet Service Provider) that supports it.
    - Cloud hosting: run a virtual machine or purchase shared time on someone else's virtual machine.

2. How do you get a URL?

- What is the structure of URL?
- What is an IP address?
- How uniquely identifiable is any individual computer?

3. What are protocols for accessing things over the Internet?

- HTTP
- If you're old, maybe you remeber FTP
- The Danger Zone: SSH and the rest.

HTML, nowadays

Back 5 years ago, the web was a mishmash of various different standards; Flash, competing versions of HTML for different browsers, pdfs and text files for certain things.

Nowadays, it's much more possible to build full featured web applications using just the basic standards of the World Wide Web. (Bet you haven't heard that expression in a while).

There are three fundamental pieces of the web today:

1. HTML–Hypertext markup language. This is the markup language for formatting documents.
2. CSS–Cascading Style Sheets. These are
3. Javascript. Javascript is a programming language designed to run inside web browsers. It used to be very slow; due to heavy investment by Google, it's become substantially faster. You can open a javascript *console* in almost any web browser; in Chrome, for example, you can sometimes type `control-shift-i` (or `apple-shift-i`) on a Mac.

- Check that out on something like `gmail.com` or `nytimes.com` to see all the traffic that this javascript runs on your web browser, continuously.

Basic HTML

You should know enough HTML to create a basic site.

The basic structure of HTML is the same as the XML we discussed earlier in this class: in fact, certain well-structured HTML can be considered a subset of XML.

It consists of *text* and *tags*.

Structure A few basic structural elements are the following tags. Remember from XML; a tag both begins and ends.

Some particularly useful ones are

See this cheat sheet for a mercifully short list; much longer lists are available elsewhere. You can ignore the bits on tables; those are not widely used anymore.

- `html`: the HTML document
- `head`: a prefix to the document
- `body`: the part of the page that will actually be displayed.
- `h1`, `h2`, etc.: Header information to display.

```
<html>
<head>
    <title>An example web site</title>
    <author>Me!</author>
</head>
<body>
    <h2>Our not-so-big title</h2>
    This is an example web site.
</body>
</html>
```

Style You can *mark up* text in HTML to display it in a particular way.

- `html`: the HTML document
- `head`: a prefix to the document
- `body`: the part of the page that will actually be displayed.
- `h1`, `h2`, etc.: Header information to display.

```
<html>
<head>
</head>
<body>
    This text will be <em>italic</em>, while <strong>this text will be bold.</strong>.
</body>
</html>
```

Back in the oos, you might just write `<b>` for bold or `<i>` for italic. If you're commenting on a blog, sometimes you can still do this. It's a bad idea, though, because the organizing principle of markup languages is to *separate semantics from appearance*.

What does that mean? Well, `<b>` just means "present in boldface". But `<strong>` means "present in a strong way"; that might mean in red in certain languages, or with a pause for emphasis when read out loud, or something else.

These tags are particularly important when it comes to issues of **accessibility**. Visually impaired persons, for example, may use a screenreader to parse a webpage; depending on how it works now or in the future, there may be special ways of rendering content.

The other reason has to do with elaborate formatting. Suppose you wanted to format a bunch of poems in HTML.

You could specify that each stanza has a two line break after it, that it should be in italics, and the like.

But you can also define specific styles for "stanza" blocks and "lines" and "poems" and "author". At an extreme, this turns into TEI. But in a middle ground, you'll be using CSS–the descriptions of style–to write things like:

```
text.author {
    font-size:larger;
    text-style:italic;
}
```

So that every author shows up in italics, and larger, in the text.

Most of this you won't want to do yourself: instead, you'll either hire a web designer or use a pre-made template. But sometimes–in particular, with Wordpress or with Omeka–you may need to go edit the CSS yourself if you decide you want something to be red instead of green, or whatever.

Non-textual elements One last thing. The two most basic elements of the web as a multimodal medium are images and links.

Both are represented with tags. You can represent an image from anywhere on the Internet in your html, or link to anywhere. The tags are

- `img`: an embedded image.
- `a`: a link to a page.

But how would you do it?

You might think something like this would do the trick: put the url between the opening and closing tags.

```
<a>http://dighist15.benschmidt.org</a>
```

But that doesn't work. It's not quite right conceptually, because anything not in tags will be displayed as text.

Instead you must include *attributes* in the tag. This shows up inside the opening tag, before the angle bracket closes. You can describe any number of attributes: the one that specifies where a link will go to is called, infelicitously, `href`. That looks like this.

```
<a href=http://dighist15.benschmidt.org>Our course web page.</a>
```

The text inside the tag will be the text that gets linked.

Images are much the same, but use an attribute called `src` (for source, obviously).

```
<img src=http://www.neh.gov/files/humanities/articles/2013_0708_images_13a_risemachines.jpg></img>
```

Here's an example.

```
<html>
<head></head>
<body>
<a href=http://dighist15.benschmidt.org>Our course web page.</a>
<br />
<img src=http://www.neh.gov/files/humanities/articles/2013_0708_images_13a_risemachines.jpg></im>
<hline>
</body>
</html>
```

Content Management Systems

We've encountered two content management systems (CMSs) in our class.

The first is Wordpress, the blogging site we use for the class.

The second is Omeka, the tool developed by historians at George Mason University.

Blackboard, the online course management software, is CMS specifically designed for teaching.

CMSs are widely used because they make it easy to separate out formatting from content. If you visit, say, the Atlantic's web site, most of the articles will be submitted without "related links" or ads or the like: instead, it's just the middle column. (Obviously for something like *Snowfall*, things will be different.)

For things like Blackboard and Omeka, this separation means you can borrow someone else's formatting for your own project without a lot of technical mucking about.

Should you use a CMS?

Well, yes. You should. Probably. But they have major issues with sustainability and digital decay. In particular, they all rely on a *database backend*. This is problematic, because it means that you can't just copy them from computer to computer like you would a Word document. Some in our class have encountered this with our course Neatline installations: someone needs to host that web site.

A static website is harder to set up and less powerful, but will last for longer. How to handle this is a Big Problem.