

# Web Programming

Copyright © 2026 by  
Robert M. Dondero, Ph.D.  
Princeton University

# Objectives

- We will cover:
  - The technologies that are at the foundation of web programming...
  - The hypertext transfer protocol (HTTP)
  - The hypertext markup language (HTML)

# HTTP and HTML



Tim  
Berners-Lee

# Agenda

- **HTTP**
- URLs
- HTML
- HTML: Links and Forms

# HTTP

- ***Hypertext Transfer Protocol (HTTP)***
  - A client/server protocol
  - HTTP client requests a file
  - HTTP server provides a file

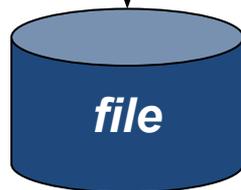
# HTTP



Socket



File system

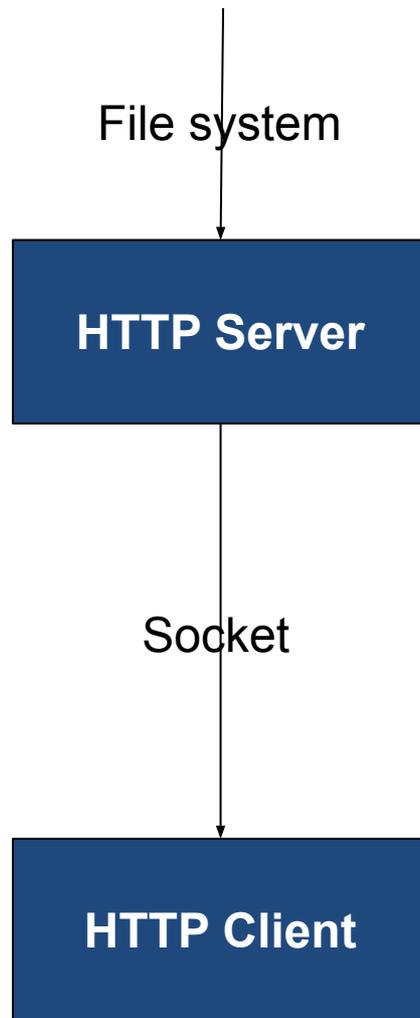


```
GET file HTTP/1.1
Host: host
<Blank line>
```

Or could be POST;  
described soon

Redundant.  
Why? Same IP  
address can have  
more than one  
domain name

# HTTP



```
HTTP/1.1 200 OK
Date: date
Server: server
...
Content-Type: text/plain
<Blank line>
<Contents of file>
```

There are many others...

A "program" interpreted by the HTTP client as per the content type

# HTTP

- **yogi.txt**
  - A simple text file

On **COMPUTER1** (192.168.1.8)

```
$ cat yogi.txt
Baseball is 90% mental and
the other half is physical.
-- Yogi Berra
$
```

# HTTP

- **simplehttpserver.py**
  - A simple HTTP server
  - Enhancement of “standard” Python HTTP server
  - Assume that it has access to **yogi.txt**

# HTTP

- See [httpclient1.py](#)

On **COMPUTER1** (192.168.1.8)

```
$ python simplehttpserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

On **COMPUTER2**

```
$ python httpclient1.py 192.168.1.8 55555 yogi.txt
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.12.3
Date: Sat, 27 Sep 2025 22:34:46 GMT
Content-type: text/plain
Content-Length: 69
Last-Modified: Sat, 12 Feb 2022 23:12:24 GMT

Baseball is 90% mental and
the other half is physical.
-- Yogi Berra
$
```

Note:  
Content-type  
is text/plain

# Agenda

- HTTP
- **URLs**
- HTML
- HTML: Links and Forms

# URLs

- ***Uniform Resource Locator (URL)***
  - **protocol**: //host:port/file
    - **protocol**
      - We'll use `http` now, `https` later
      - Others: `file`, `ftp`, `mailto`, ...
      - See [http://en.wikipedia.org/wiki/URI\\_scheme](http://en.wikipedia.org/wiki/URI_scheme)

# URLs

- Uniform resource locator (cont.)
  - `protocol://host:port/file`
    - **host**
      - Recall *Network Programming* lecture
      - IP address or domain name of HTTP server

# URLs

- Uniform resource locator (cont.)
  - `protocol://host:port/file`
    - **port**
      - Recall *Network Programming* lecture
      - The port at which the HTTP server is listening
      - Default for HTTP: 80
      - Default for HTTPS: 443

# URLs

- Uniform resource locator (cont.)
  - `protocol://host:port/file`
    - **file**
      - The path name of the file that the HTTP server should deliver
      - Default: usually `index.html`

# URLs

- See [httpclient2.py](#)

On **COMPUTER1** (192.168.1.8)

```
$ python simplehttpserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/)
...
```

On **COMPUTER2**

```
$ python httpclient2.py http://192.168.1.8:55555/yogi.txt
Server: SimpleHTTP/0.6 Python/3.12.3
Date: Sun, 28 Sep 2025 01:24:53 GMT
Content-type: text/plain
Content-Length: 69
Last-Modified: Sat, 12 Feb 2022 23:12:24 GMT

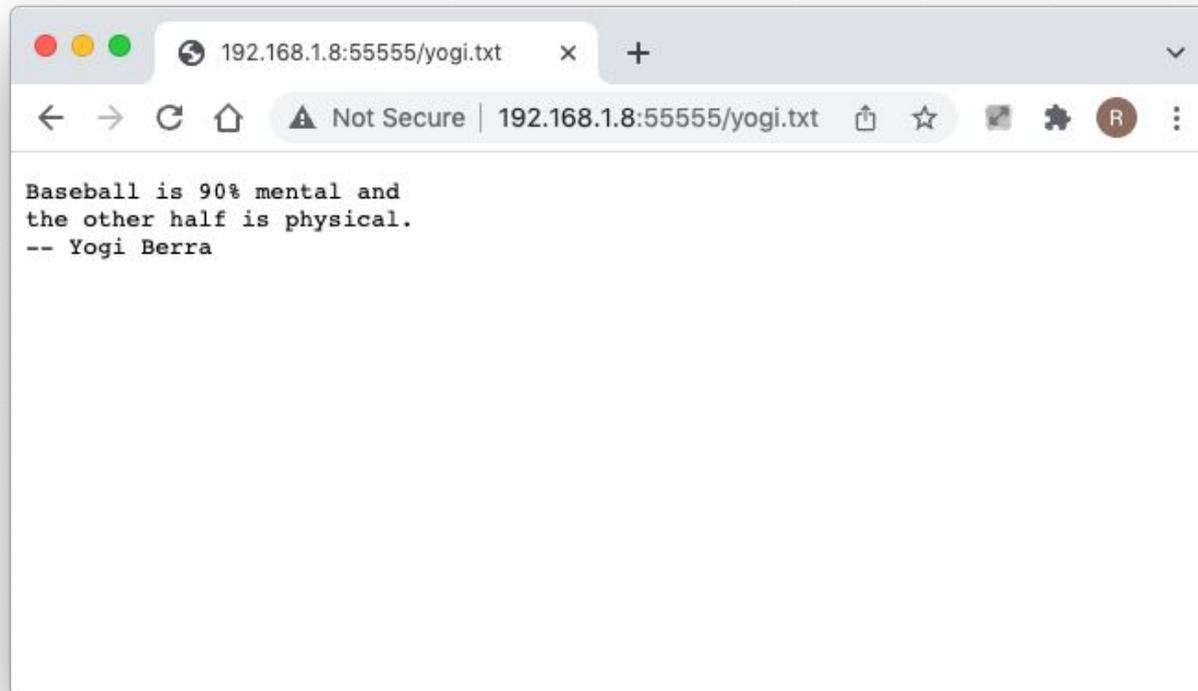
Baseball is 90% mental and
the other half is physical.
-- Yogi Berra
$
```

# URLs

## On **COMPUTER1** (192.168.1.8)

```
$ python simplehttpserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/)
...
```

## On **COMPUTER2**



# URLs

- **Review...**
- **Question:** How to issue HTTP request?
- **Answer 1:** Your own program
- **Answer 2:** Browser
  - Enter appropriate URL

# Agenda

- HTTP
- URLs
- **HTML**
- HTML: Links and Forms

# HTML

- Some HTTP content types:
  - **text/plain**, **text/html**, image/gif, image/jpeg, audio/mp4, **application/xml**, **application/json**, ...
- Complete list of HTTP content types:
  - [http://en.wikipedia.org/wiki/Internet\\_media\\_type](http://en.wikipedia.org/wiki/Internet_media_type)

# HTML

- ***Hypertext Markup Language (HTML)***
  - A language for expressing documents
- HTML document contains...

# HTML

- **Elements**

Example HTML Element	Description
<code>&lt;strong&gt;some text&lt;/strong&gt;</code>	A normal element Delimited by <b>start tag</b> and <b>end tag</b>
<code>&lt;a href="someurl"&gt;some text&lt;/a&gt;</code>	An element with an <b>attribute</b>
<code>&lt;strong&gt;&lt;/strong&gt;</code>	An <b>empty element</b>
<code>&lt;hr&gt;</code>	A <b>void element</b> An element that must be empty and that must consist of a start tag only *
<code>&lt;hr /&gt;</code>	A <b>self-closing tag</b> A void element

\* Not allowed in some “relatives” of HTML

# HTML

- *Processing Instructions*

Example HTML Element	Description
<code>&lt;!DOCTYPE html&gt;</code>	A <i>DOCTYPE</i> processing instruction First line of document Identifies document as HTML 5
<code>&lt;!-- comment --&gt;</code>	A <i>comment</i>

# HTML

- See **fund.html**
  - Artificial example
  - Illustrates fundamentals of HTML

# HTML

```
$ python simplehttpserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/)
...
```

On **COMPUTER1**  
(192.168.1.8)

```
$ python httpclient2.py http://192.168.1.8:55555/fund.html
Server: SimpleHTTP/0.6 Python/3.11.7
Date: Sun, 25 Feb 2024 03:01:48 GMT
Content-type: text/html
Content-Length: 3164
Last-Modified: Sat, 25 Sep 2021 01:58:06 GMT

<!DOCTYPE html>

<!-- ===== -->
<!-- fundamentals.html -->
<!-- Author: Bob Dondero -->
<!-- ===== -->

<html>
  <head>
    ...
</html>
$
```

On  
**COMPUTER2**

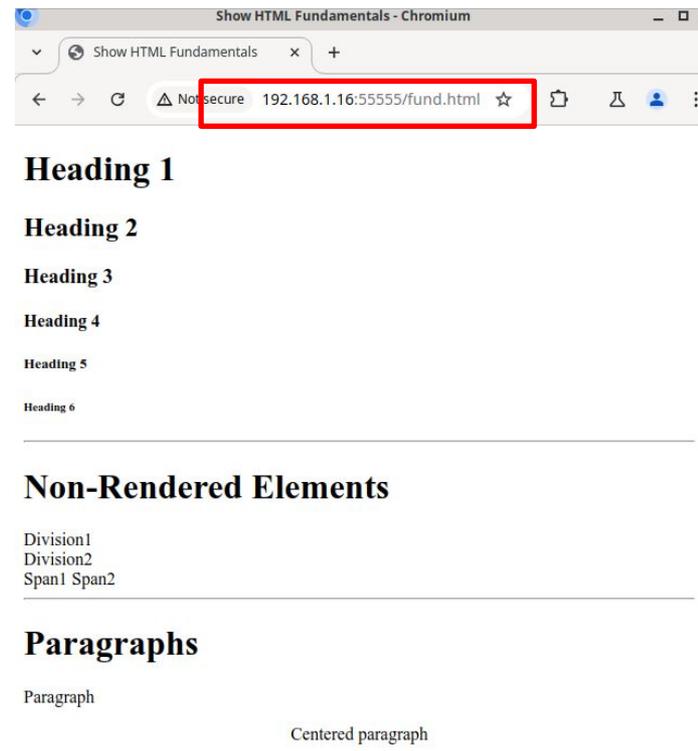
Note:  
Content-type  
is text/html

# HTML

On **COMPUTER1** (192.168.1.16)

```
$ python simplehttpserver.py 55555  
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

On **COMPUTER2**



**Interprets**  
document

Continued on next slide

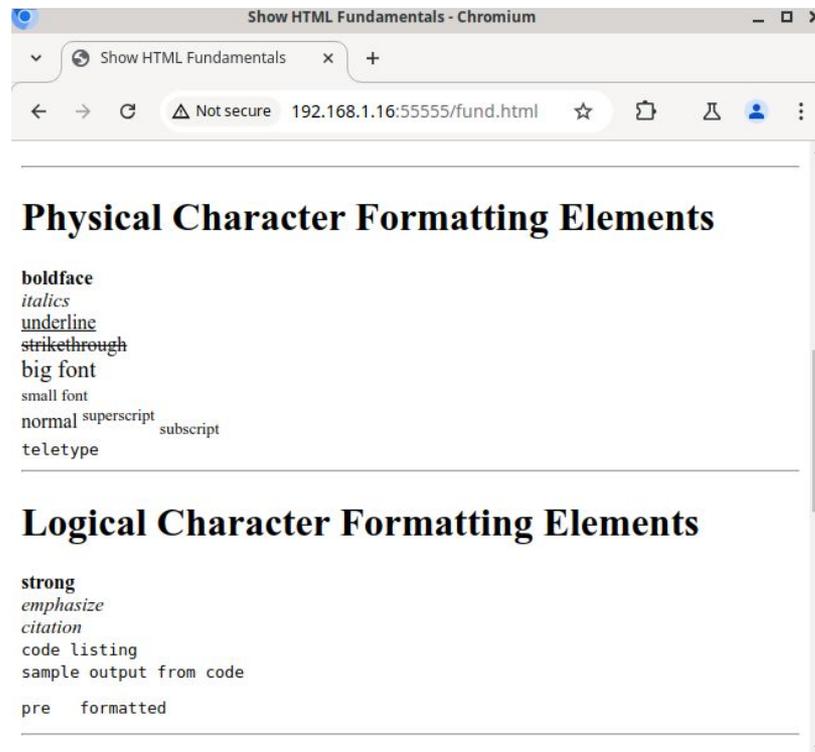
# HTML

On **COMPUTER1** (192.168.1.16)

```
$ python simplehttpserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

On **COMPUTER2**

**Interprets**  
document



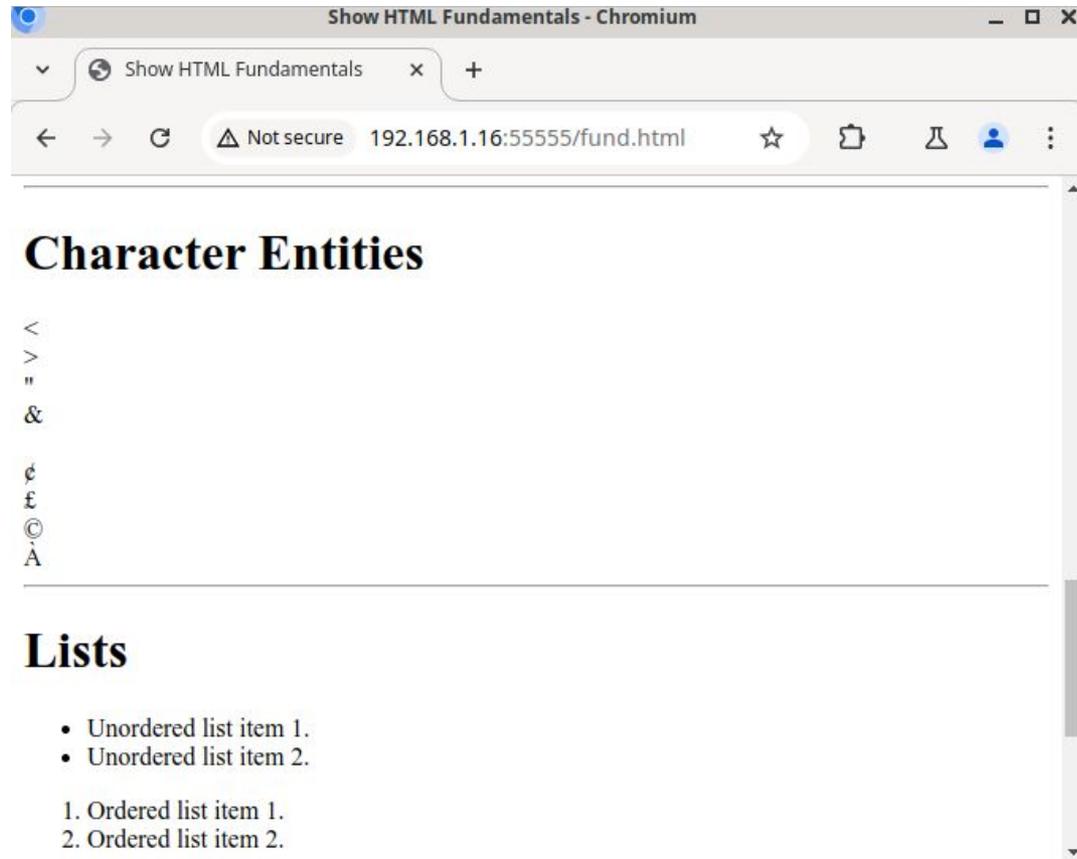
Continued on next slide

# HTML

On **COMPUTER1** (192.168.1.16)

```
$ python simplehttpserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

On **COMPUTER2**



**Interprets**  
document

Continued on next slide

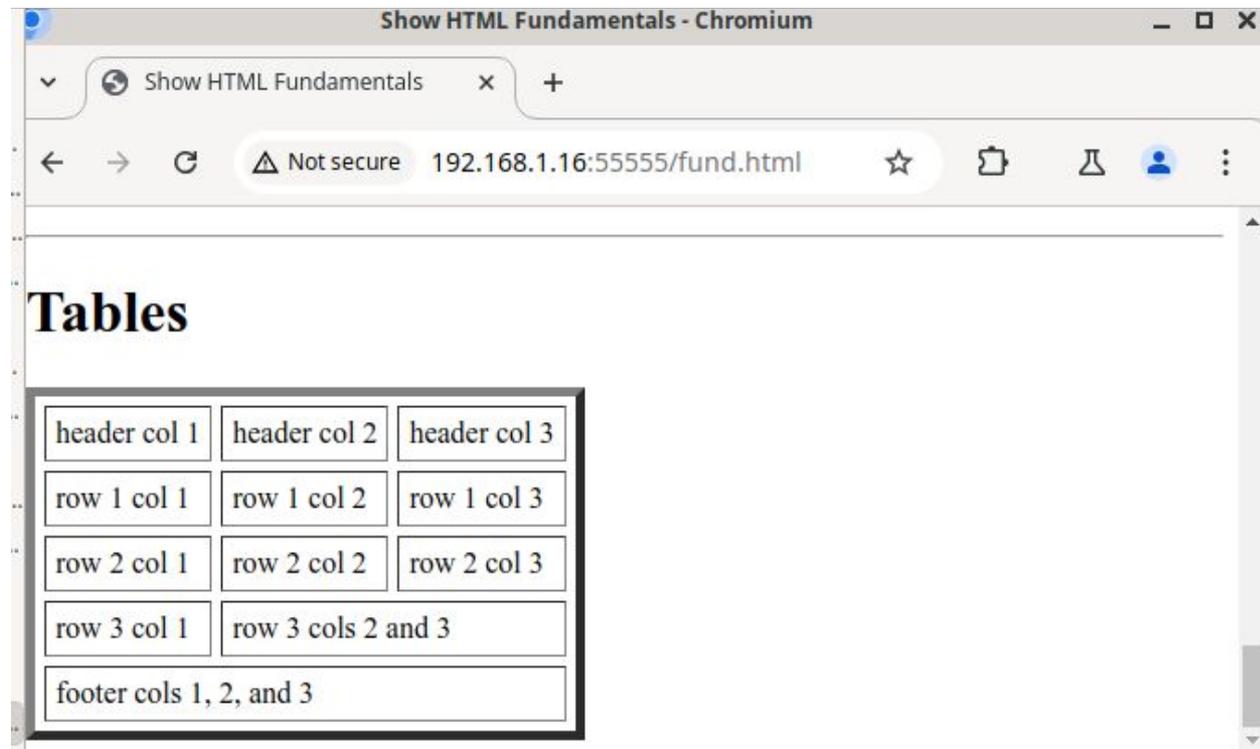
# HTML

On **COMPUTER1** (192.168.1.16)

```
$ python simplehttpserver.py 55555  
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

On **COMPUTER2**

**Interprets**  
document



# Agenda

- HTTP
- URLs
- HTML
- **HTML: Links and Forms**

# HTML: Links and Forms

- See **links.html**

- `<a href="someurl">...</a>`

- Defines a ***page link***

- User clicks on page link => browser sends HTTP request specified by *someurl*

# HTML: Links and Forms

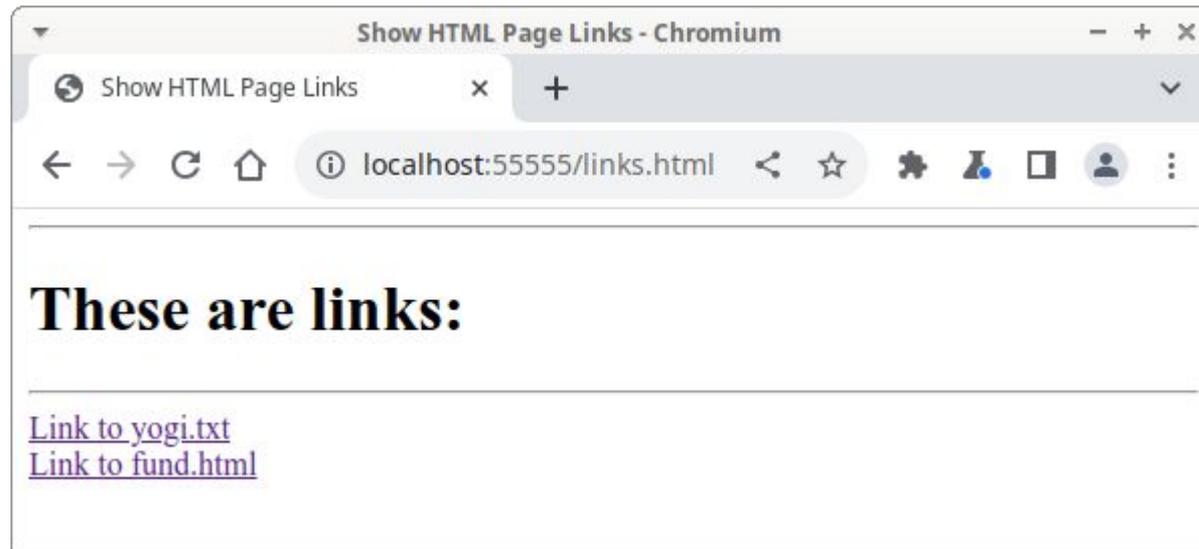
- See **links.html** (cont.)

```
$ python simplehttpserver.py 5555  
Serving HTTP on 0.0.0.0 port 5555 (http://0.0.0.0:5555/) ...
```

# HTML: Links and Forms

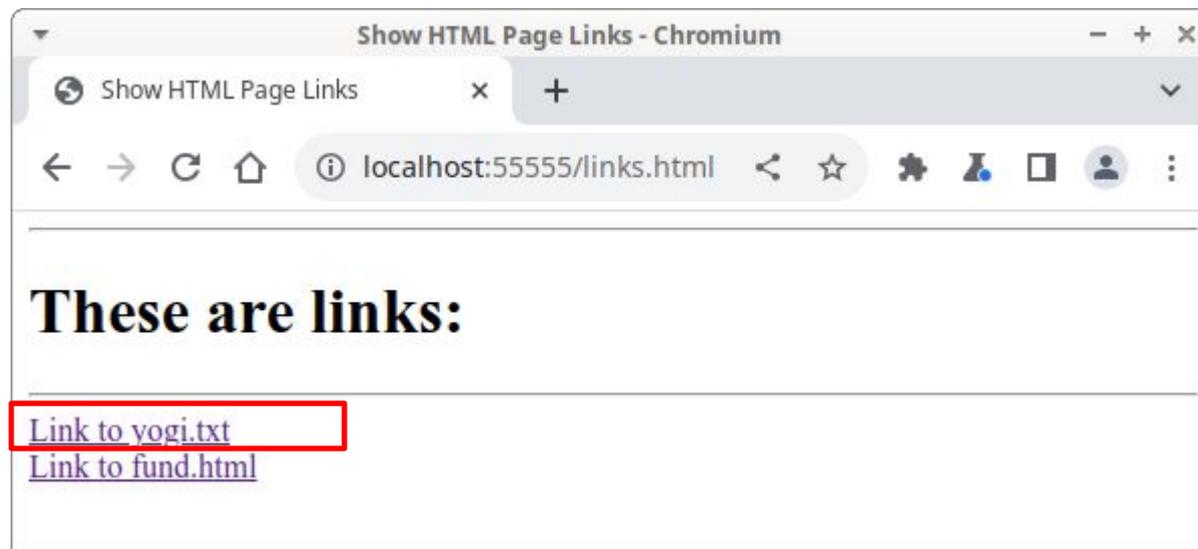
- See **links.html** (cont.)

Use localhost as abbreviation for URL of current computer



# HTML: Links and Forms

- See [links.html](#) (cont.)

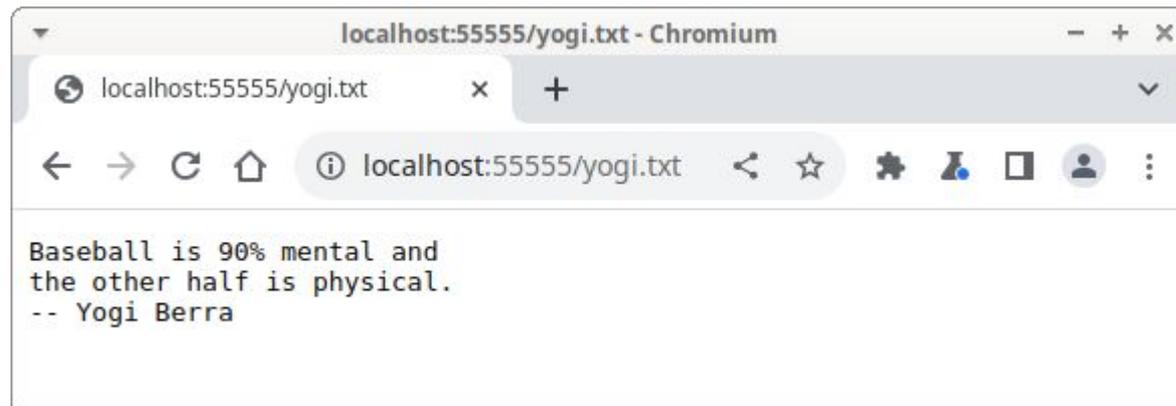


Browser sends HTTP request for  
`http://localhost:55555/yogi.txt`

# HTML: Links and Forms

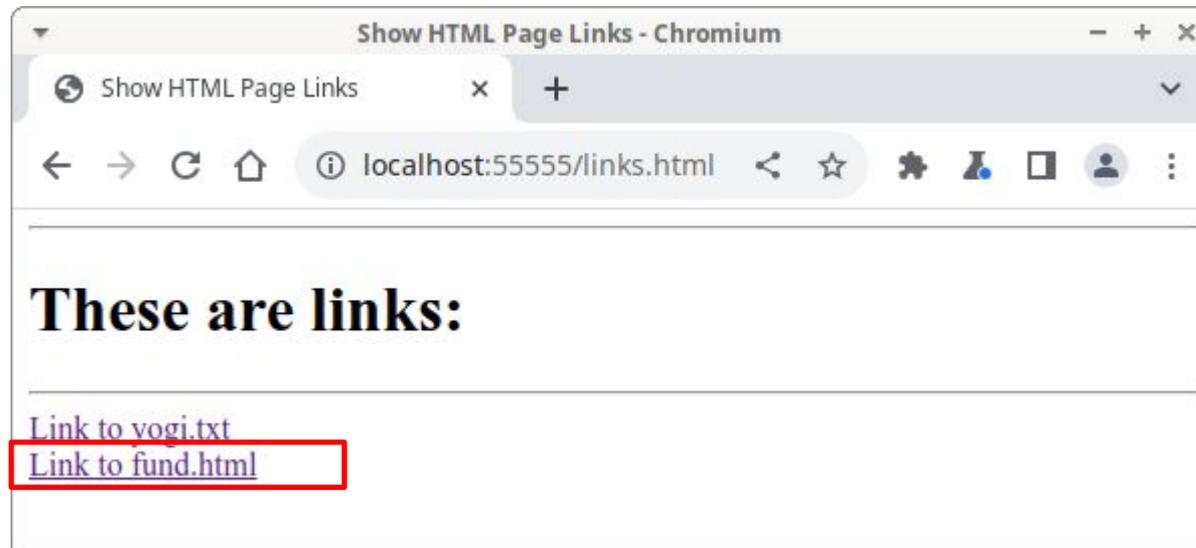
- See **links.html** (cont.)

Protocol, host, port are same as the ones used for previous page



# HTML: Links and Forms

- See [links.html](#) (cont.)

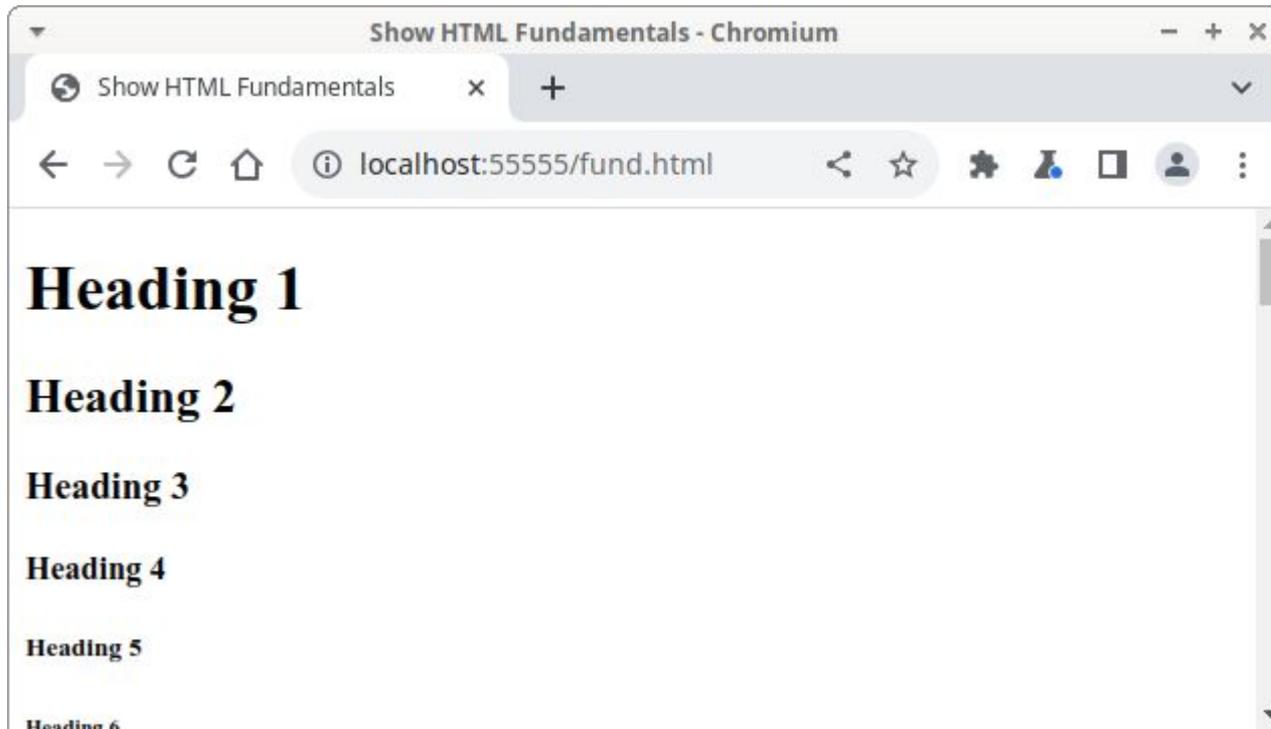


Browser sends HTTP request for  
`http://localhost:55555/fund.html`

# HTML: Links and Forms

- See **links.html** (cont.)

Protocol, host, port are same as the ones used for previous page



# HTML: Links and Forms

- See **forms.html**

- `<form action="someurl">...</form>`
  - Defines a **form**
  - Browser does not render
- `<input type="submit">`
  - Often nested in `form` element
  - Browser renders as a button
  - User clicks on button => browser sends request specified by `someurl` of enclosing form
- `<input type="submit" value="label">`
  - Browser renders as button with label `label`

# HTML: Links and Forms

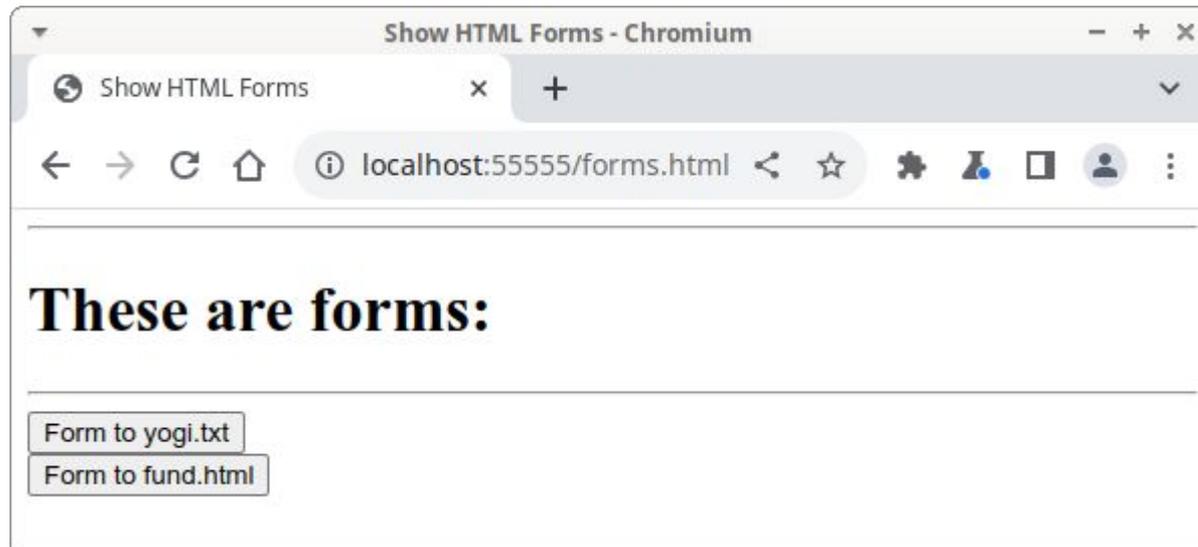
- See **forms.html**

```
$ python simplehttpserver.py 5555  
Serving HTTP on 0.0.0.0 port 5555 (http://0.0.0.0:5555/) ...
```

# HTML: Links and Forms

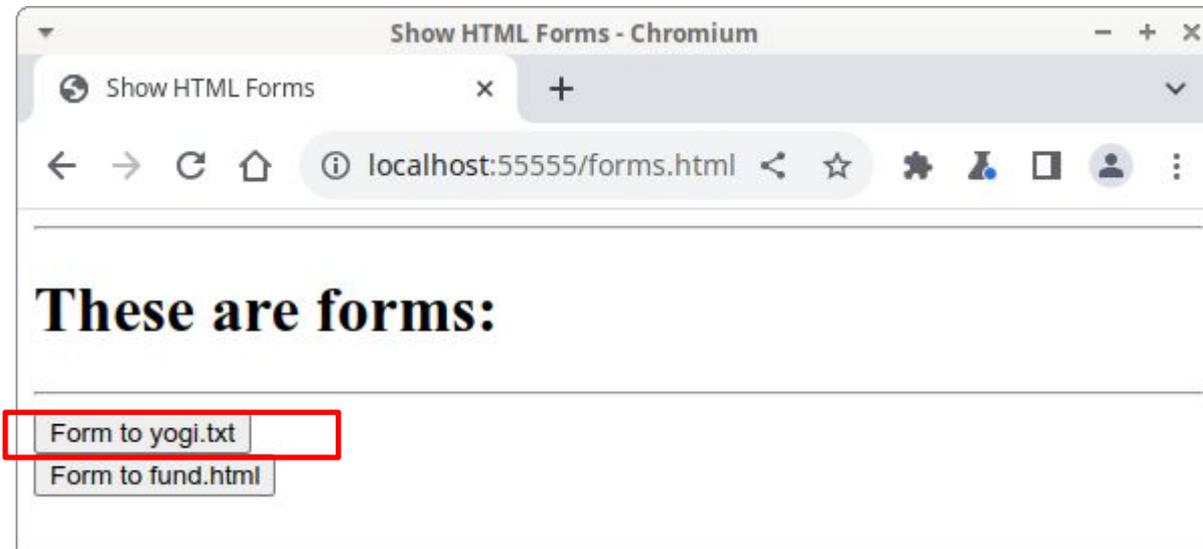
- See **forms.html** (cont.)

Use localhost as abbreviation for URL of current computer



# HTML: Links and Forms

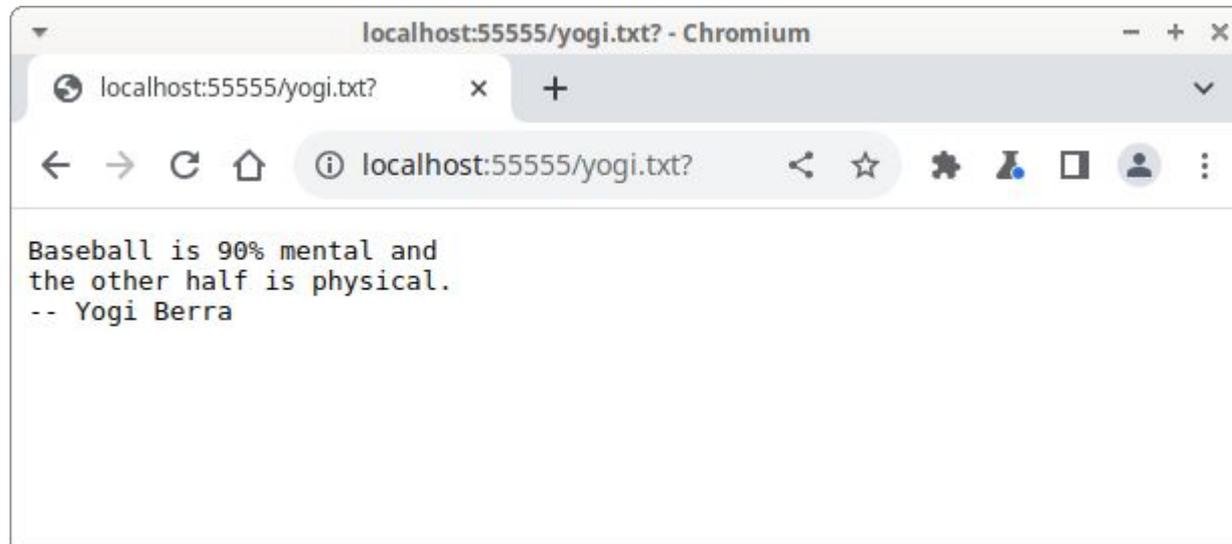
- See **forms.html** (cont.)



Browser sends HTTP request for  
`http://localhost:55555/yogi.txt`

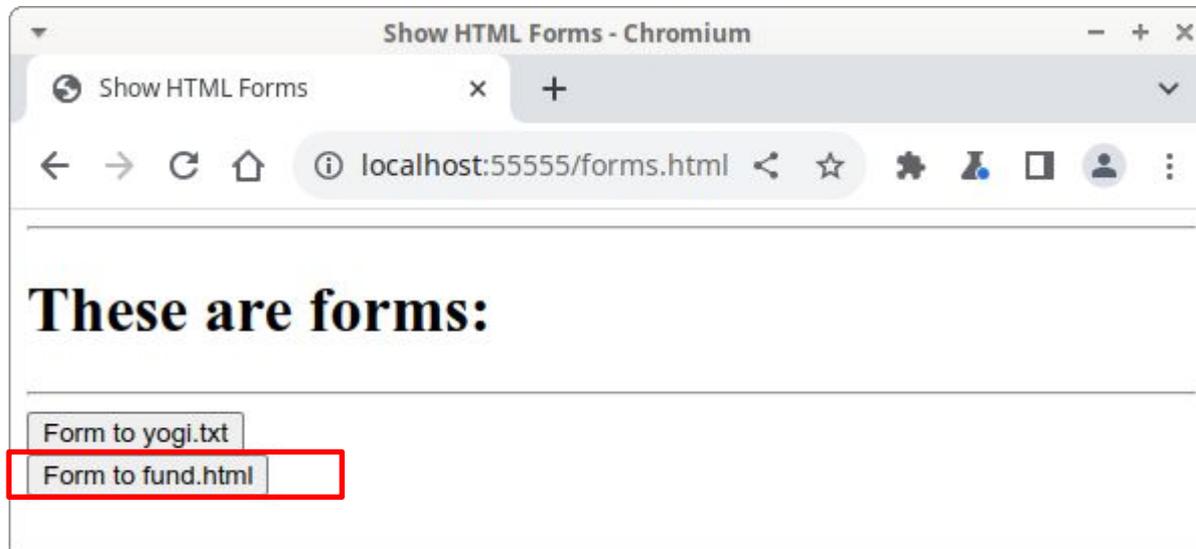
# HTML: Links and Forms

- See **forms.html** (cont.)



# HTML: Links and Forms

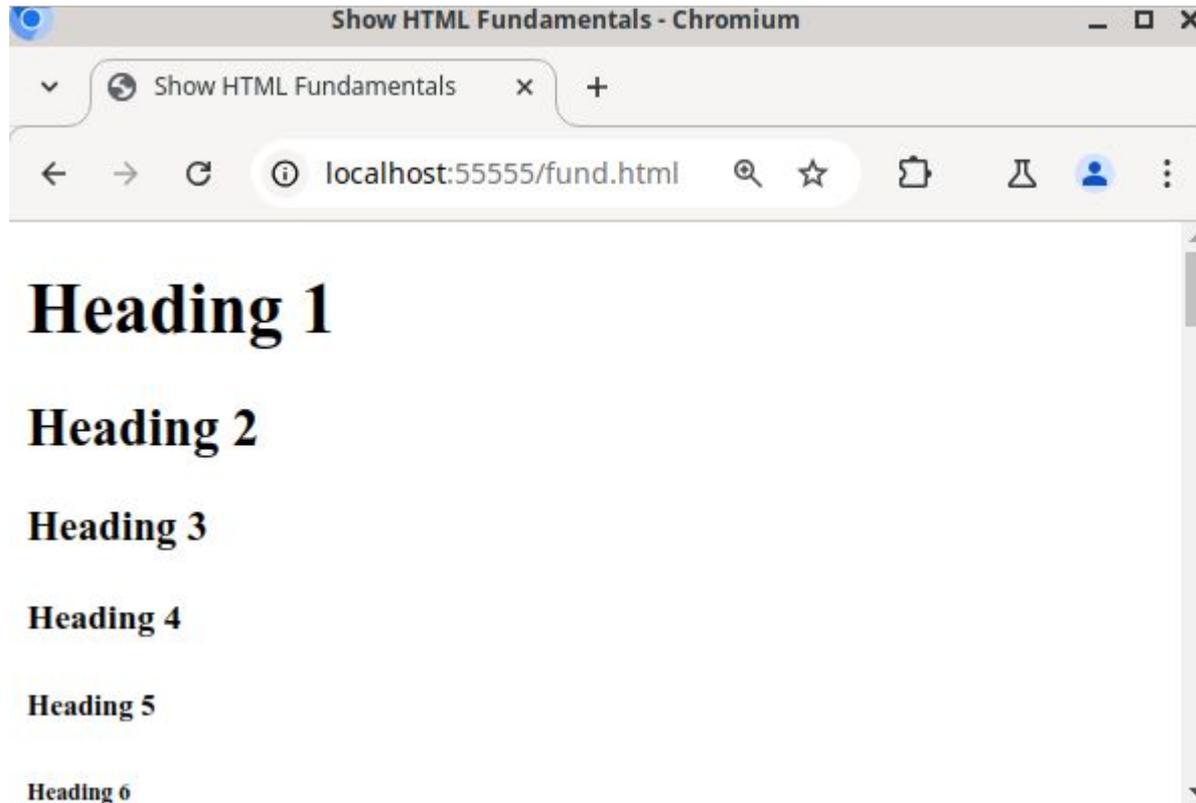
- See **forms.html** (cont.)



Browser sends HTTP request for  
`http://localhost:55555/fund.html`

# HTML: Links and Forms

- See **forms.html** (cont.)



# HTML: Links and Forms

- **Review...**
- **Question:** How to issue HTTP request?
- **Answer 1:** Your own program
- **Answer 2:** Browser
  - Enter URL
  - Click on HTML page link
  - Click on HTML `input` element of type `submit` within a `form` element

# Aside: Examining Raw HTML

- Many browsers allow you to examine raw/uninterpreted HTML doc
  - Chrome & Firefox:
    - Right-click → view page source

# Lecture Summary

- In this lecture we covered:
  - The technologies that are at the foundation of web programming...
  - The hypertext transfer protocol (HTTP)
  - The hypertext markup language (HTML)
- See also:
  - **Appendix 1: Popular HTTP Servers & Browsers**
  - **Appendix 2: HTML History**

# More Information

- The COS 333 *Lectures* web page provides references to supplementary information

# Appendix 1: Popular HTTP Servers & Browsers

# Popular HTTP Servers & Browsers

- As reported by <https://news.netcraft.com/> for May 2024

HTTP Server	Market Share
Nginx	22%
Apache HTTP Server	20%
Cloudflare	11%
OpenResty	10%

# Popular HTTP Servers & Browsers

- As reported by <https://www.w3schools.com/browsers/> for June 2024

Browser	Market Share
Google Chrome	<b>78%</b>
Microsoft Edge	11%
Mozilla Firefox	5%
Apple Safari	4%
Opera	2%

# Popular HTTP Servers & Browsers

- As reported by <https://www.w3schools.com/browsers/> for **Nov 2002**

Browser	Market Share
Microsoft Internet Explorer	<b>83%</b>
Netscape	8%
AOL	5%

# Popular HTTP Servers & Browsers

- Browser notes:
  - Substantial incompatibilities among browsers
    - Lesser problem now
    - But often must design apps for use with all (current and old) browsers!!!

# Appendix 2: HTML History

# HTML History

- ***Structured Generalized Markup Language (SGML)***
  - A language for expressing documents
- SGML document
  - Contains unadorned text and ***markup***

# HTML History

- ***SGML markup***
  - `<tag>...</tag>`
  - `<tag attribute="value" ...>...</tag>`
  - `<tag />`
- Tags and attributes can be anything you want!

# HTML History

- ***Document type definition (DTD)***
  - A specification of allowable tags and attributes (and much more)
- Typically:
  - SGML user group (e.g., pharm industry, drug regulatory agencies) composes DTD
  - SGML users (e.g., pharm companies) compose documents that conform to the DTD
- First line of SGML doc specifies DTD

# HTML History

- HTML
  - 1990
  - Intended to be an application of SGML, but...
  - At the time no clear parsing guidelines were established, so...
  - Many HTML documents are not valid SGML documents

# HTML History

- HTML 2.0
  - 1995
  - First version to be standardized
  - First line of document:
    - `<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">`

# HTML History

- HTML 3.2
  - 1997
  - More of an SGML application, but...
  - Burdened by need for backward compatibility
    - Still had many legacy features that differ from SGML's requirements
  - First line of document:
    - `<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">`

# HTML History

- HTML 4.0
  - 1997
  - Two versions:
    - Strict: deprecated elements are forbidden
    - Transitional: deprecated elements are allowed
  - With strict DTD, an SGML application
    - Conforms to ISO 8879 – SGML

# HTML History

## - First line of document:

- `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
- `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`

# HTML History

- HTML 5
  - 2014
  - Abandons any attempt to define HTML as SGML application
  - Explicitly defines its own syntax rules
  - More closely match existing implementations and documents
  - First line of document:
    - `<!DOCTYPE html>`

# HTML History

- We'll use HTML 5
  - But we'll keep it simple
    - This course is not about HTML