# Web Programming

# Objectives

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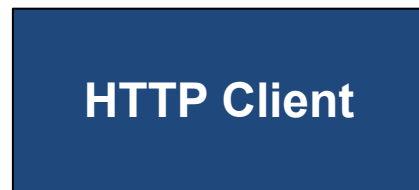  - The hypertext markup language (HTML)

# HTTP and HTML



Tim
Berners-Lee

# Agenda
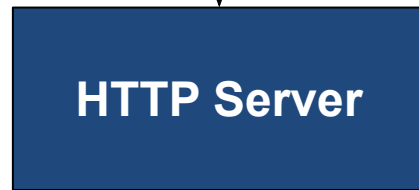
- **HTTP**
- URLs
- HTML
- The World Wide Web

# HTTP

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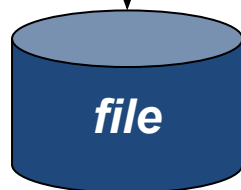
# HTTP

**HTTP Client**
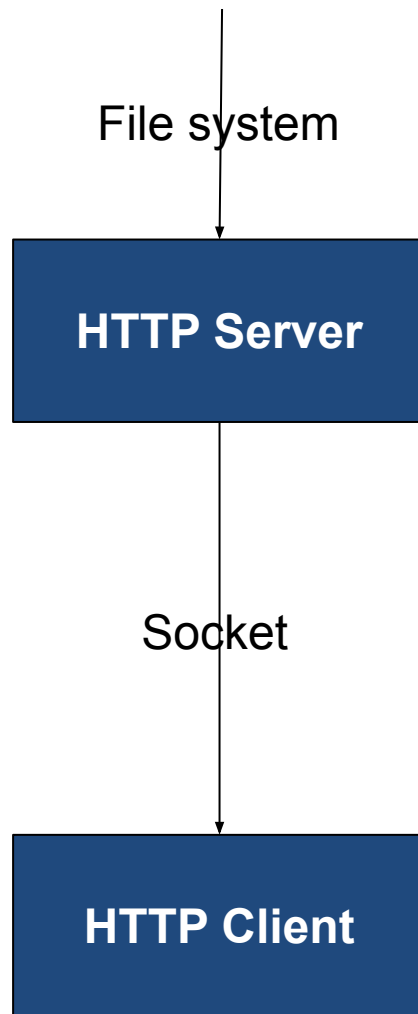
Socket

**HTTP Server**

File system

*file*

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

File system

**HTTP Server**

Socket

**HTTP Client**

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

7

# 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 runs on **COMPUTER1** (192.168.1.8)
  - Assume that it has access to **yogi.txt**

# HTTP

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**


Note: Content-type
is text/plain

```
$ telnet 192.168.1.8 55555
Trying 192.168.1.8...
Connected to 192.168.1.8.
Escape character is '^]'.
GET yogi.txt HTTP/1.1
Host: 192:168:1:8

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.8.10
Date: Sat, 12 Feb 2022 23:14:22 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
Connection closed by foreign host.
$
```

# 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/3.8.10
Date: Sat, 12 Feb 2022 23:16:27 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
- The World Wide Web

# URLs

- *Uniform Resource Locator (URL)*

  - `protocol`://host:port/file
    - **protocol**
      - We'll use `http` now, `https` later
      - Others: `file`, `ftp`, `mailto`, `file`, …
      - See http://en.wikipedia.org/wiki/URI_scheme

# URLs

- Uniform resource locator (cont.)

    - `protocol://`**`host`**`:port/file`
        - **`host`**
            - IP address or domain name of HTTP server
            - Recall *Network Programming* lecture

# URLs

- Uniform resource locator (cont.)

  - `protocol://host:`**`port`**`/file`
    - **port**
      - The port at which the HTTP server is listening
      - Recall *Network Programming* lecture
      - For HTTP, default port is 80
      - For HTTPS, default port is 443

# URLs

- Uniform resource locator (cont.)

  - `protocol://host:port/`**`file`**
    - **`file`**
      - The path name of the file that the HTTP server should deliver
      - Default path name is specified in HTTP server settings
        - » Often index.html or index.php

# 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/3.11.7
Date: Sun, 25 Feb 2024 03:01:19 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
$
```
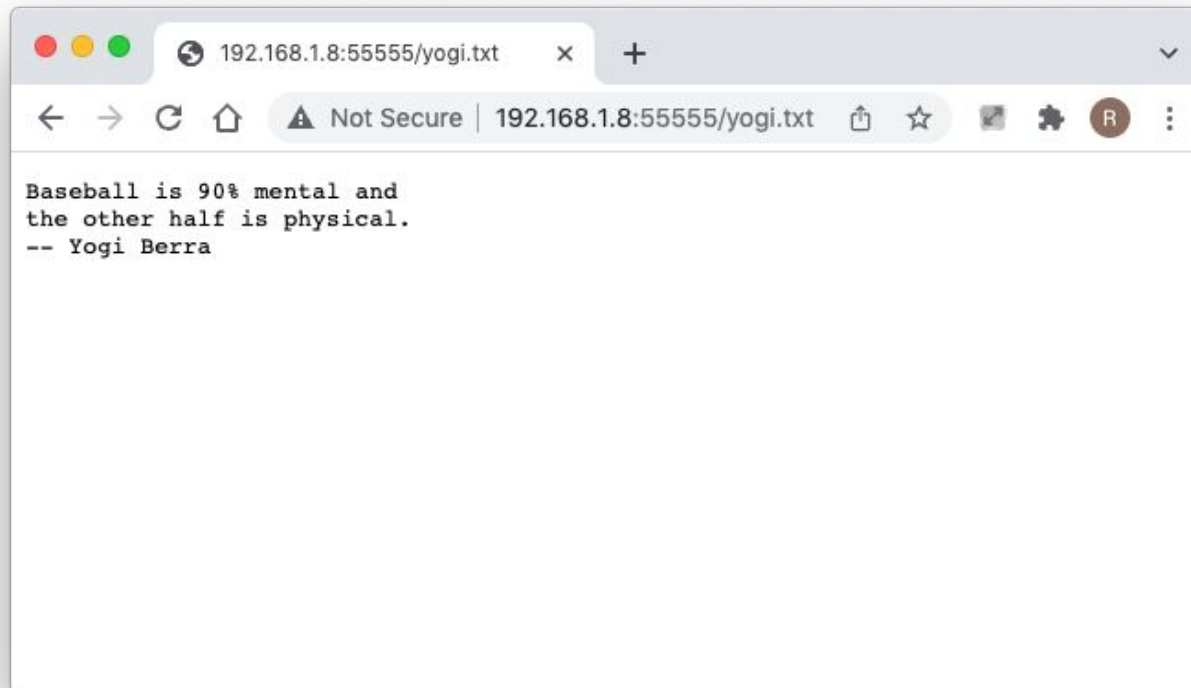
# URLs

## 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**: Telnet
- **Answer 2**: Your own program
- **Answer 3**: Browser
  - Enter appropriate URL

# Agenda

- HTTP
- URLs
- **HTML**
- The World Wide Web

# 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
- The most popular HTTP content type:
  - **text/html**

# HTML

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

# HTML

- *Elements*

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

\* Not allowed in some "relatives" of HTML

# HTML

- *Processing Instructions*

| Example HTML Element | Description |
|---|---|
| `<!DOCTYPE html>` | A *DOCTYPE* processing instruction<br>First line of document<br>Identifies document as HTML 5 |
| `<!-- comment -->` | A *comment* |

# HTML

- See **fund.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>
$
```
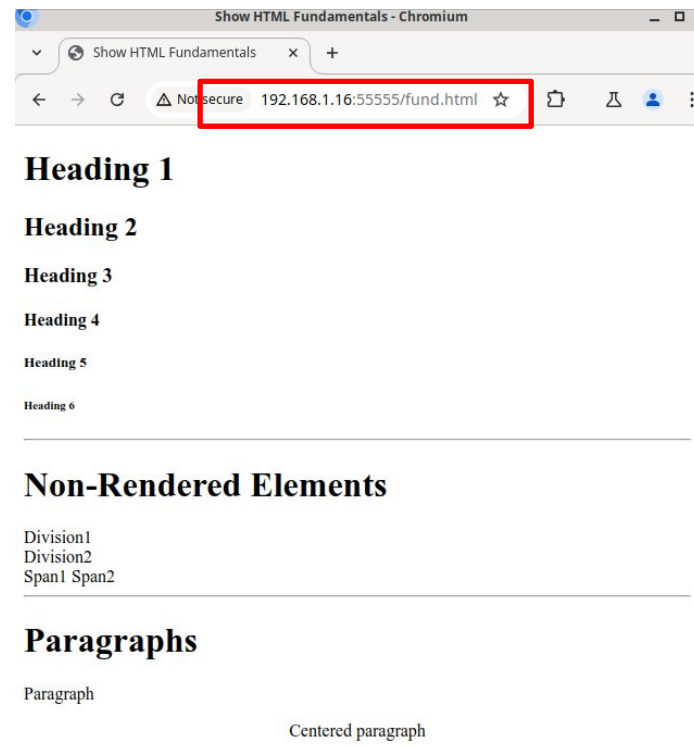
On **COMPUTER2**

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**

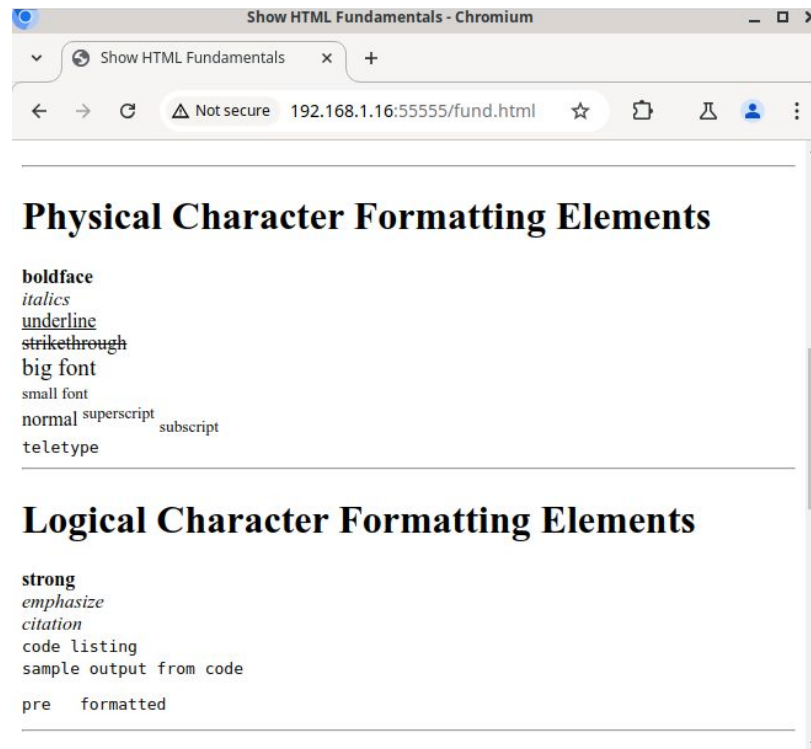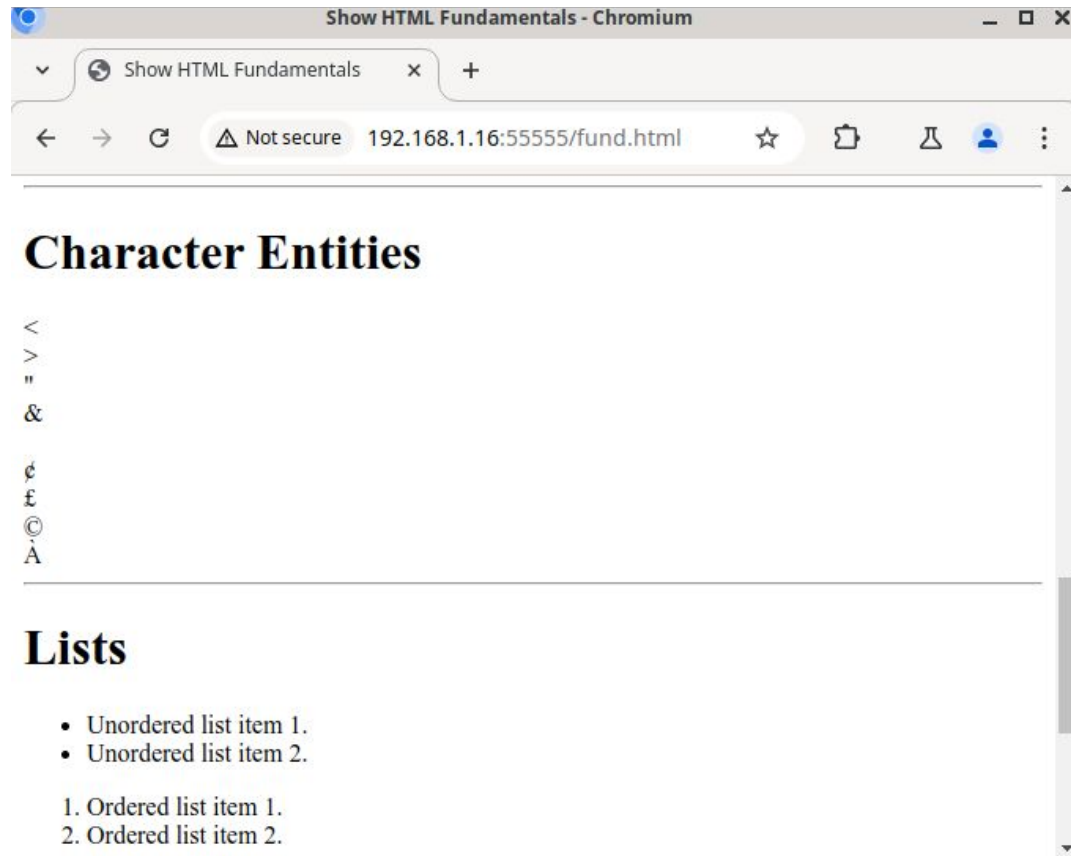**Interprets**
document



Continued on next slide

27

# 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/) ...
```
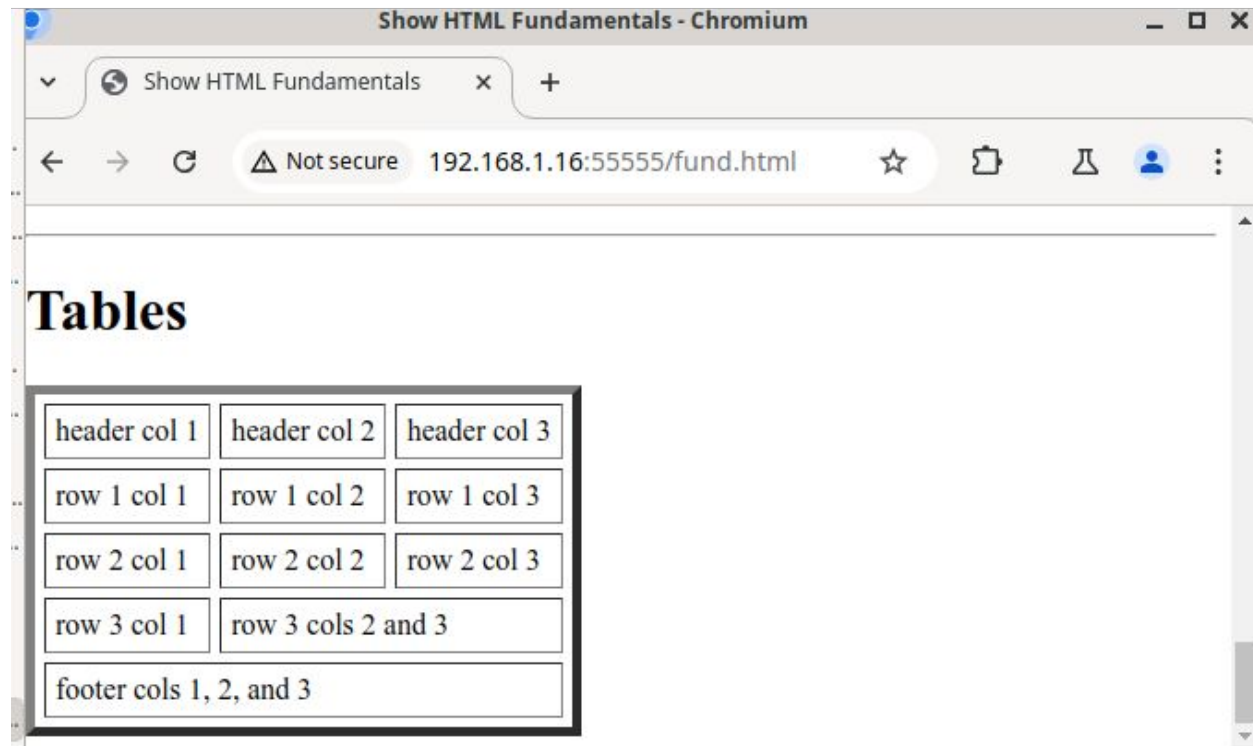
## On **COMPUTER2**

**Interprets**
document

# 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

29

# 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

# HTML

- ## See **links.html**

  - `<a href="someurl">...</a>`
    - Defines a *page link*
    - User clicks on page link => browser sends request specified by `someurl`
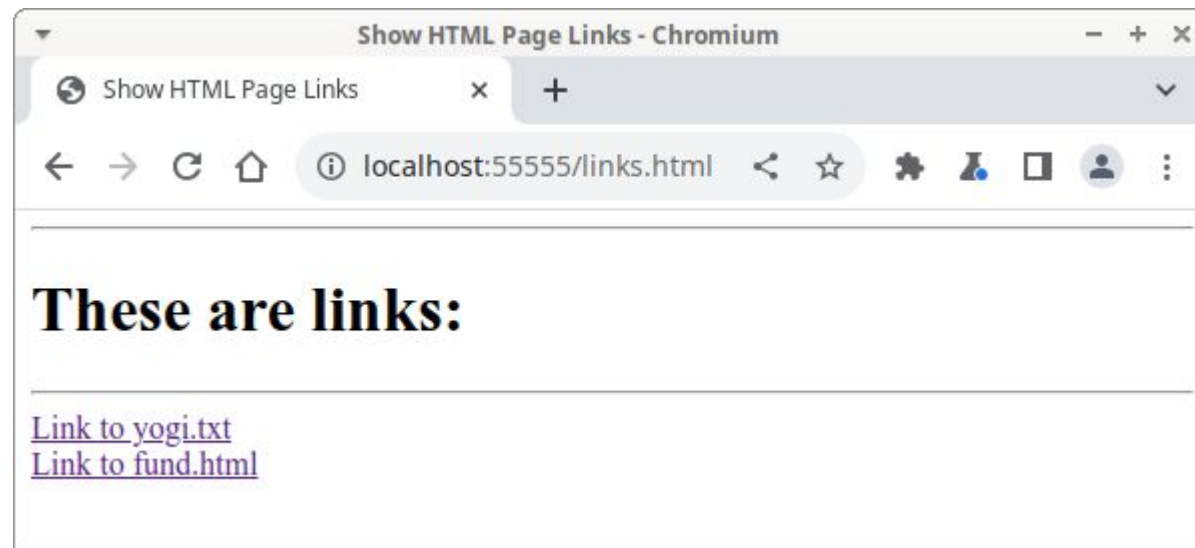
# HTML

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

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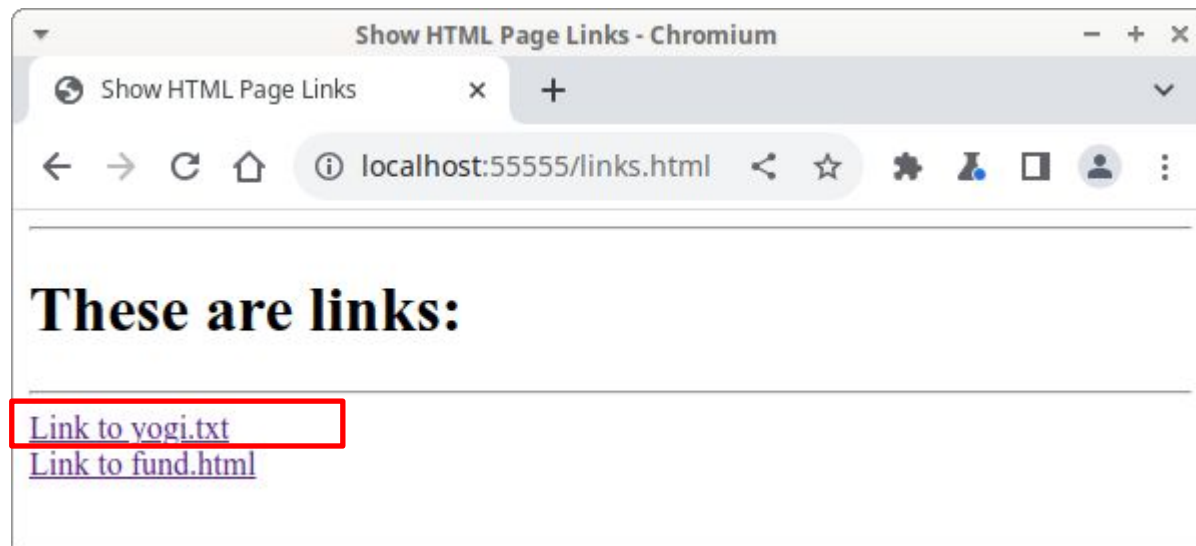
# HTML

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

  Use localhost as abbreviation for
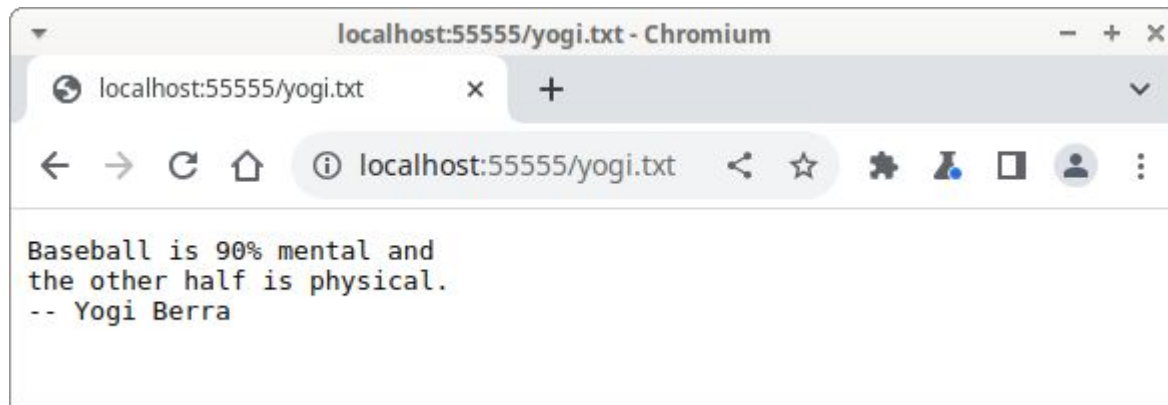  URL of current computer

# HTML

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

# HTML

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

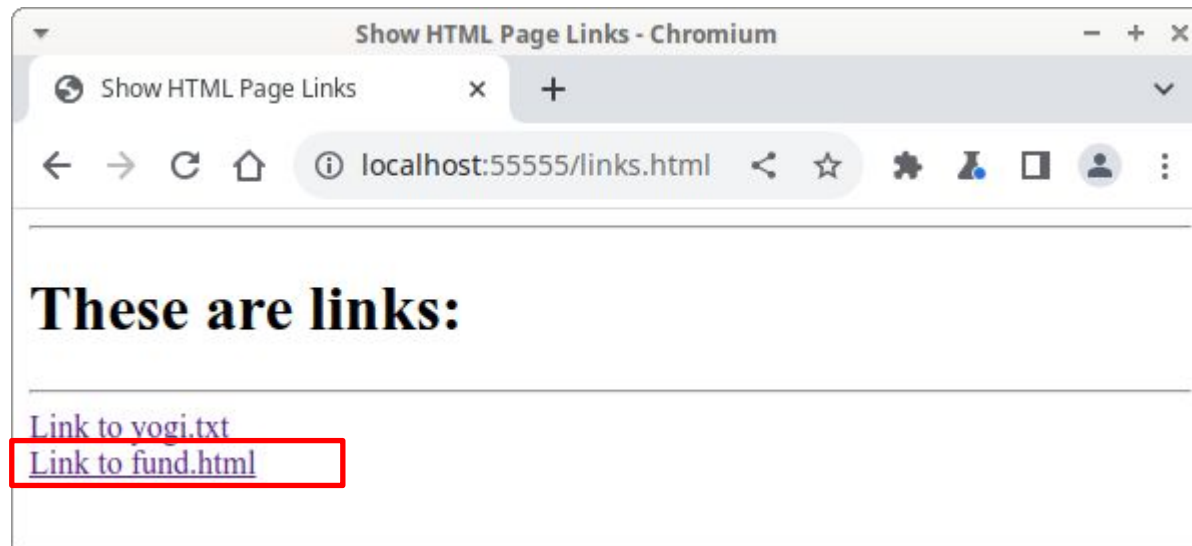Protocol, host, port are same as the
ones used for previous page



Content-type: text/plain
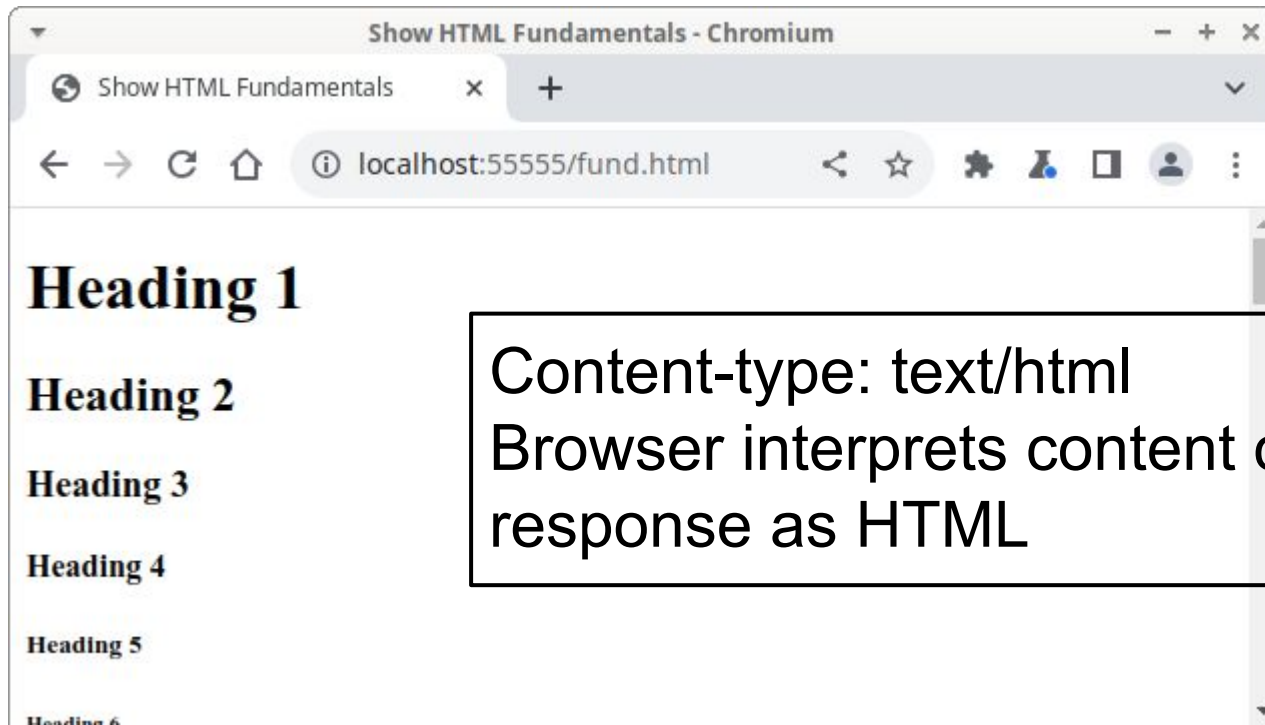Browser interprets content of response as plain text

# HTML

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

# HTML

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

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



Content-type: text/html
Browser interprets content of response as HTML

# HTML

- ## See **forms.html**

  - `<form action="`*`someurl`*`">...</form>`
    - Defines a *form*
    - Browser does not render

  - `<input type="submit"` `value="`*`label`*`">`
    - Often nested in `form` element
    - Browser renders as button with label *`label`*
    - User clicks on button => browser sends request specified by *`someurl`*
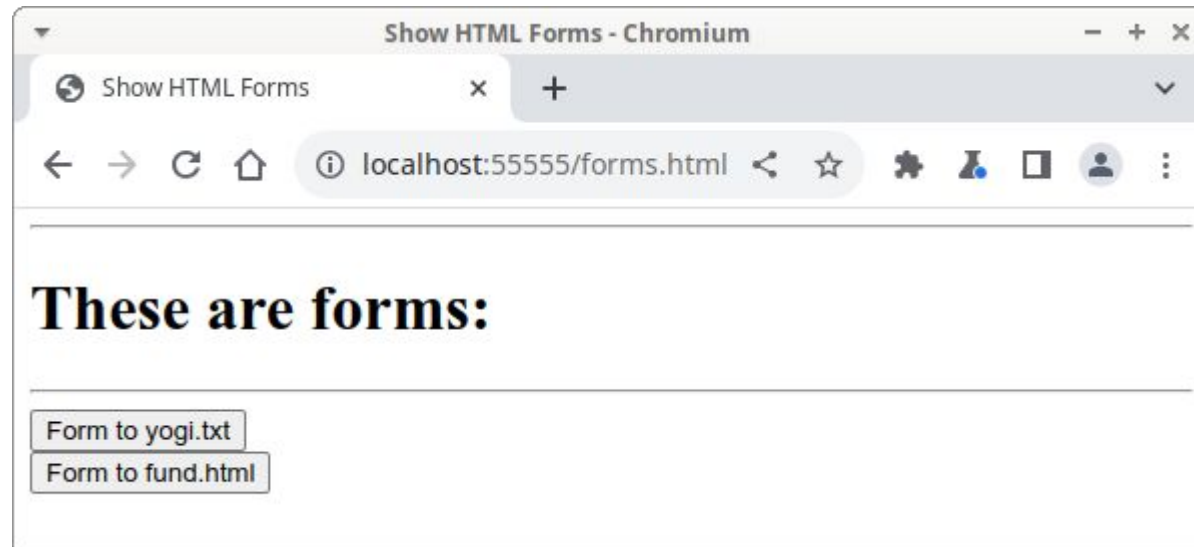
# HTML

- See **forms.html**

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

# HTML

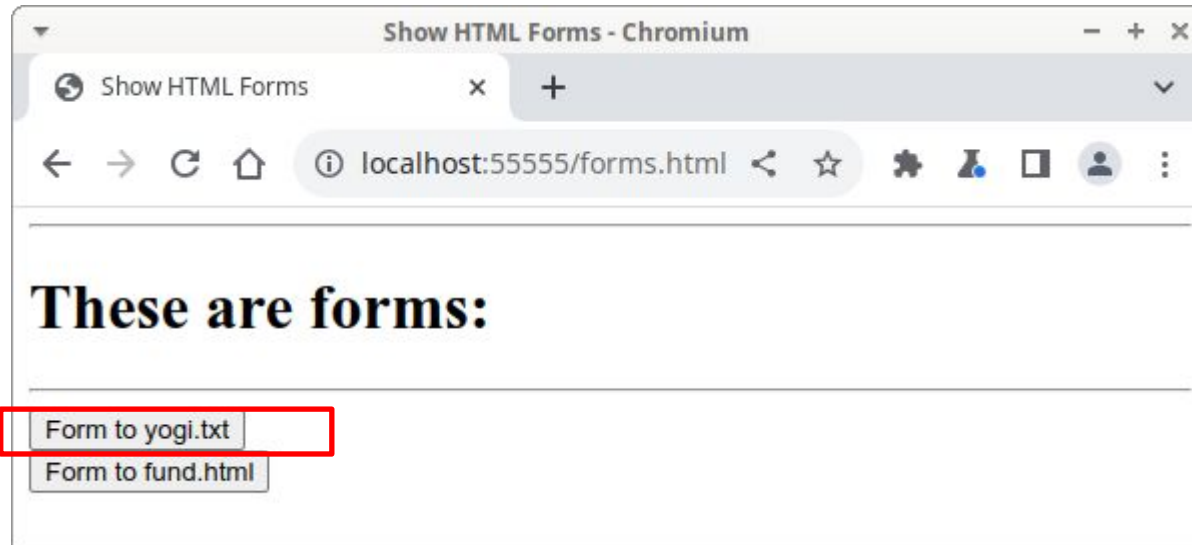- See **forms.html** (cont.)

  Use localhost as abbreviation for
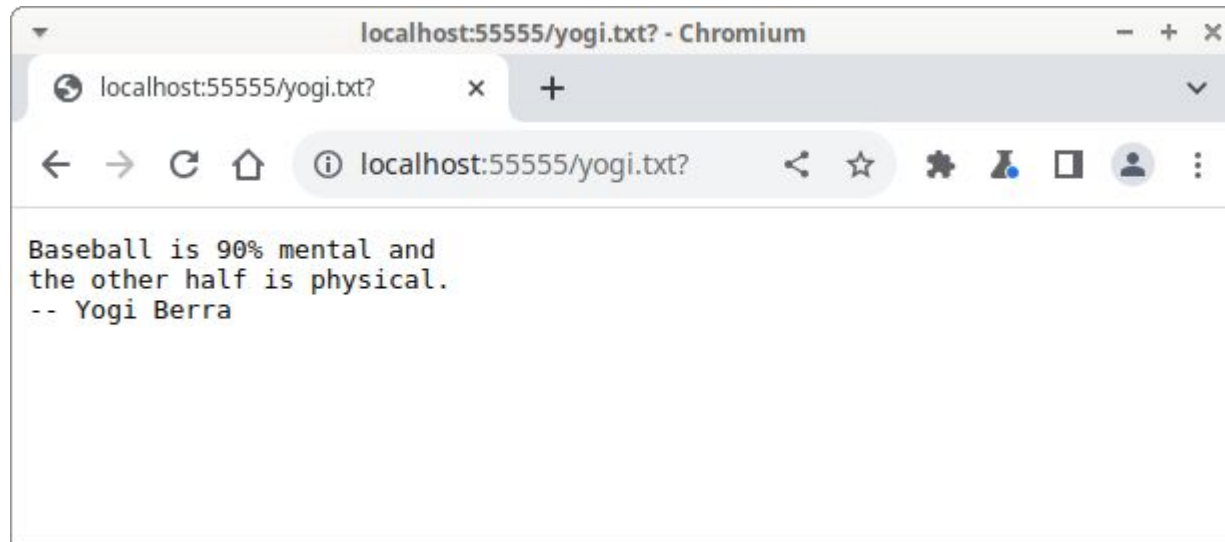  URL of current computer

# HTML

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

# HTML

- See **<u>forms.html</u>** (cont.)



Content-type: text/plain
Browser interprets content of response as plain text

# HTML

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

# HTML

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



Content-type: text/html
Browser interprets content of response as HTML
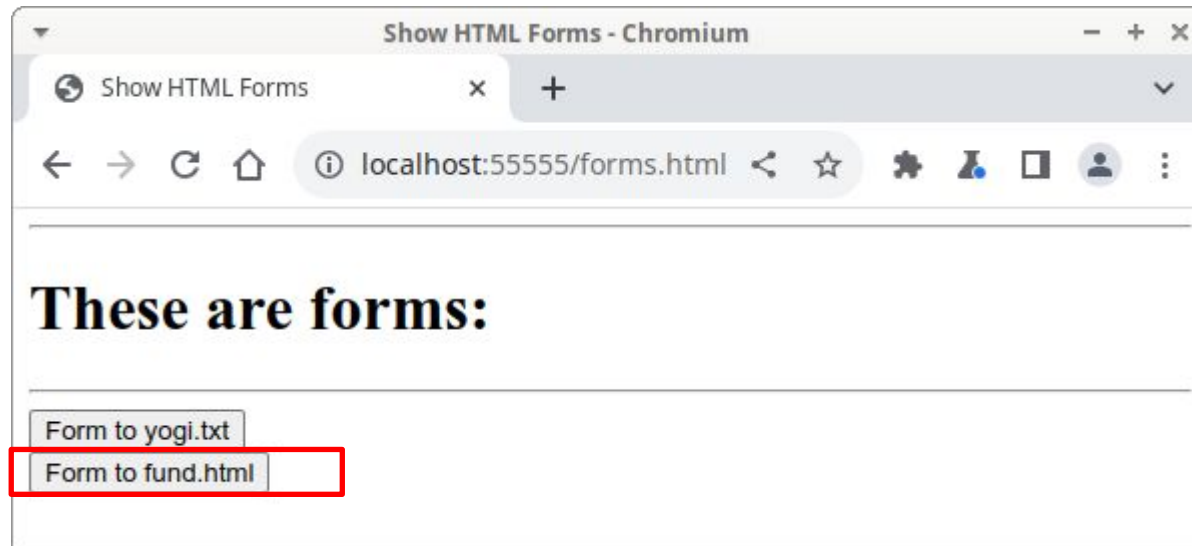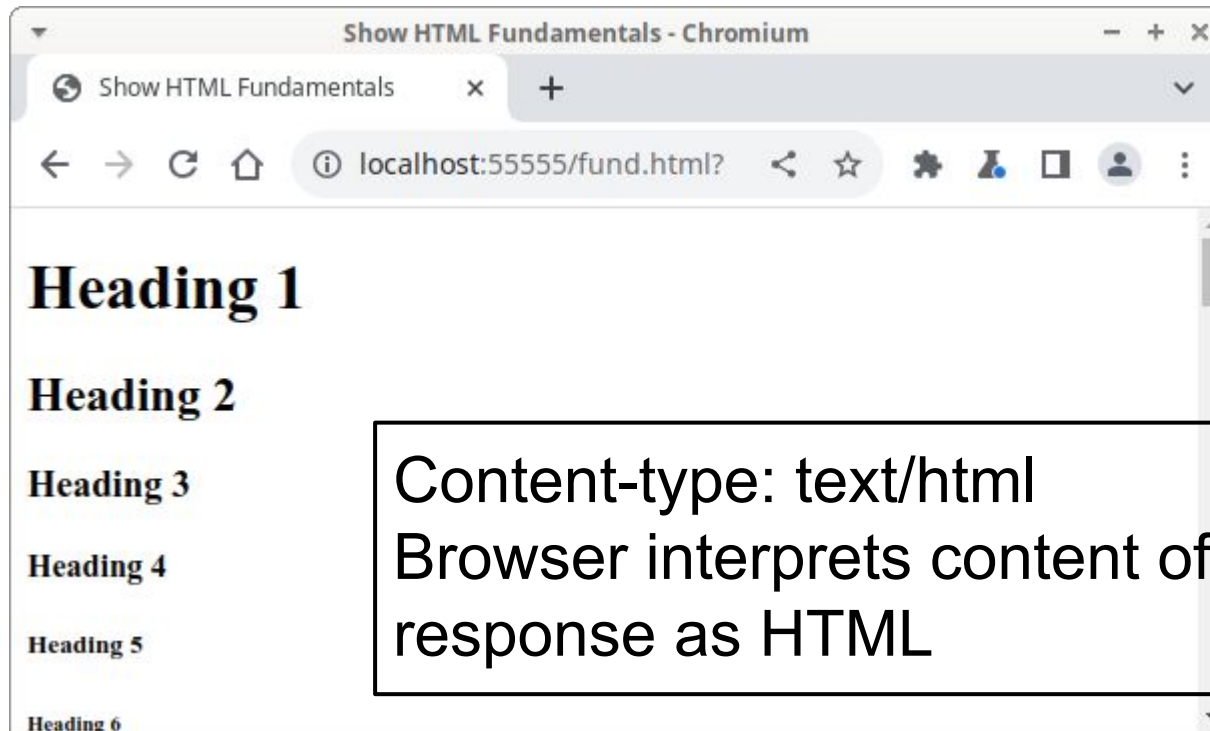
# HTML

- **Review**…

- **Question**: How to issue HTTP request?
- **Answer 1**: Telnet
- **Answer 2**: Your own program
- **Answer 3**: Browser
  - Enter URL
  - Click on HTML page link
  - Click on HTML input element of type submit

# Agenda

- HTTP
- URLs
- HTML
- **The World Wide Web**

# The World Wide Web

- Note:
  - Link or form in a HTML doc could reference another HTML doc
  - And so…

| HTTP | + | HTML docs with links and forms | = | Directed graph |
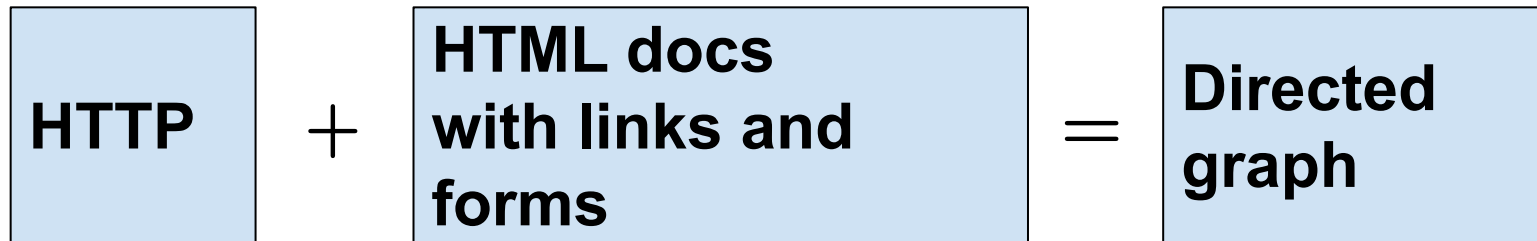
# The World Wide Web

- The ***World Wide Web***
    - A directed graph
        - Nodes: HTML docs
        - Edges: links and forms

# Aside: Examining Raw HTML

- To examine raw/uninterpreted HTML doc in browser:
    - Chrome & firefox:
        - Right click → view page source
- Good learning/debugging tool

# Summary

- We have 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

# 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 | **78%** |
| 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 | **83%** |
| 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