CS 83R : Server-Side Ruby Web Programming

Howard A. Stahl

Agenda

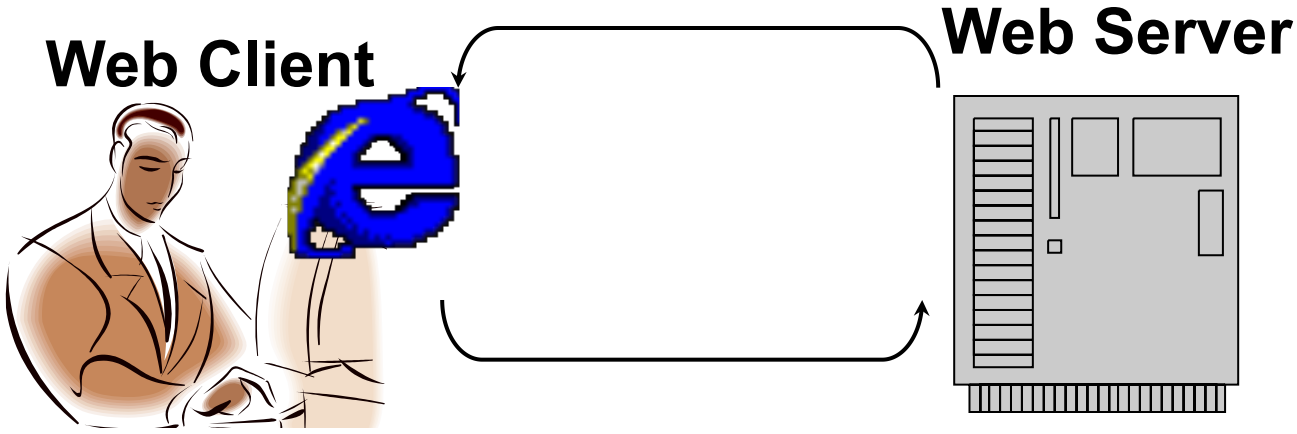
- Introducing The HTTP Protocol
  - Headers
  - GET
  - POST

Understanding HTTP

- HTTP Is The Dominant Standard For Information Interchange Over The Internet
- HTTP Is A Protocol
  - A Protocol Is A Set Of Rules That Make Communication On A Network More Efficient
  - Resides In Layer 7 Of ISO/OSI Model
  - The Protocol Is Very Precise And Rigid, But Any Program Following The Rules Can Participate

HTTP Protocol

- HTTP Supports Various Methods
  - GET, POST, HEAD, PUT, DELETE & others
- Any Application Can Serve As A Client Or Server
  - It Just Has To Speak Or Process The Protocol Correctly…

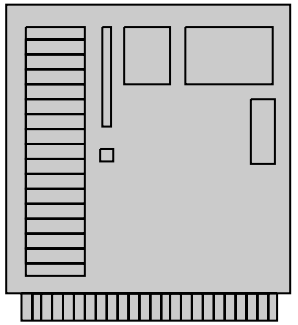


HTTP Low-Level Details

- HTTP 1.0 Communication Model

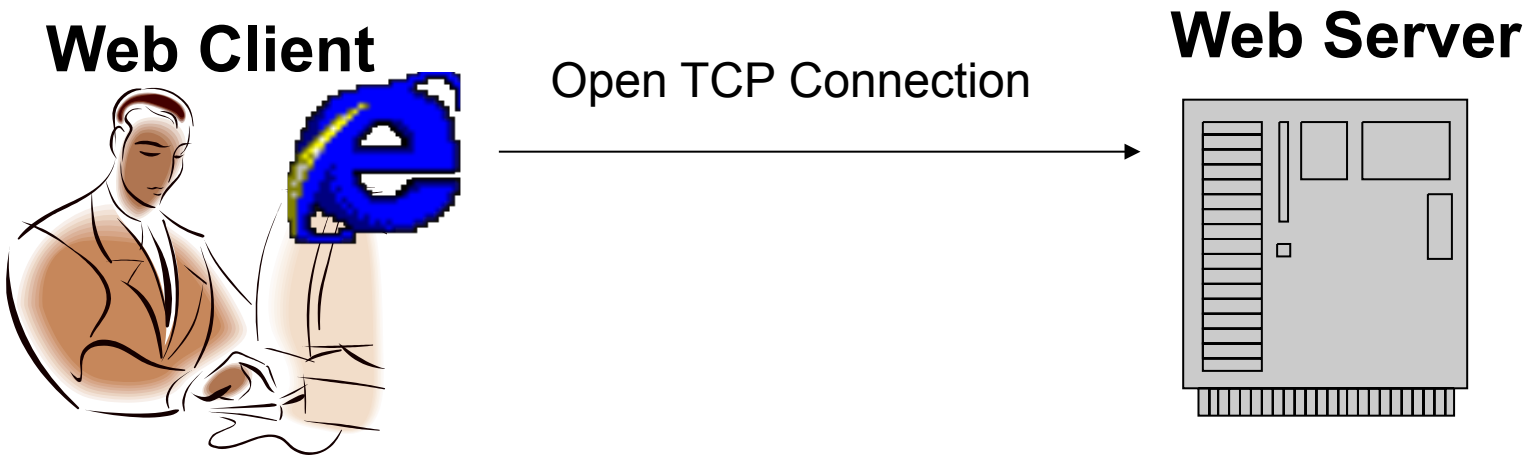
Web Client

Web Server



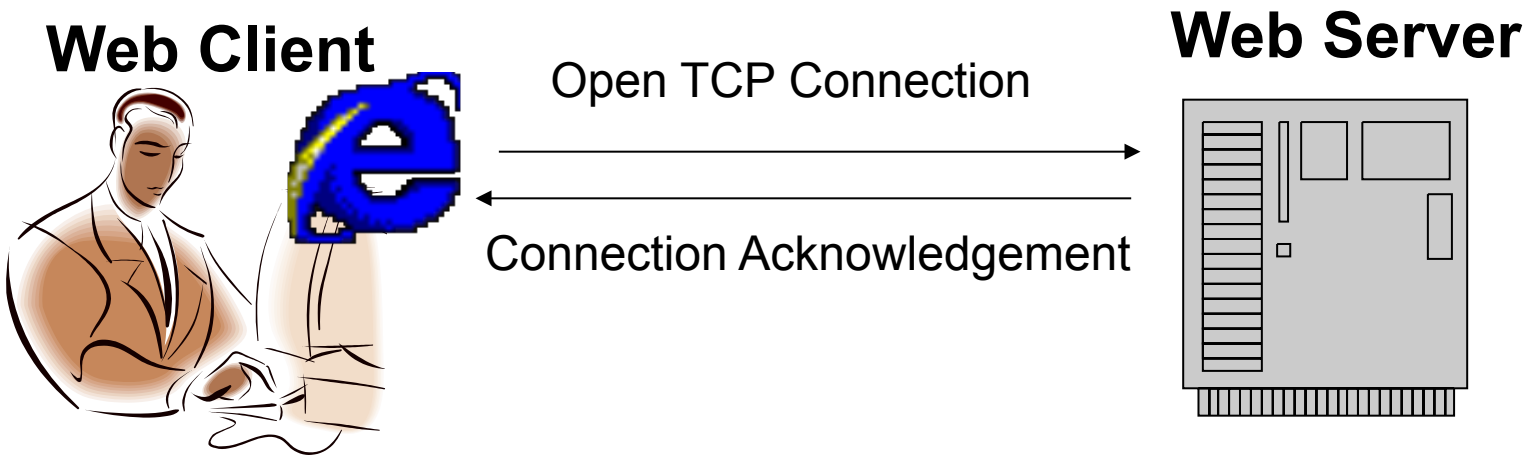
HTTP Low-Level Details

- HTTP 1.0 Communication Model



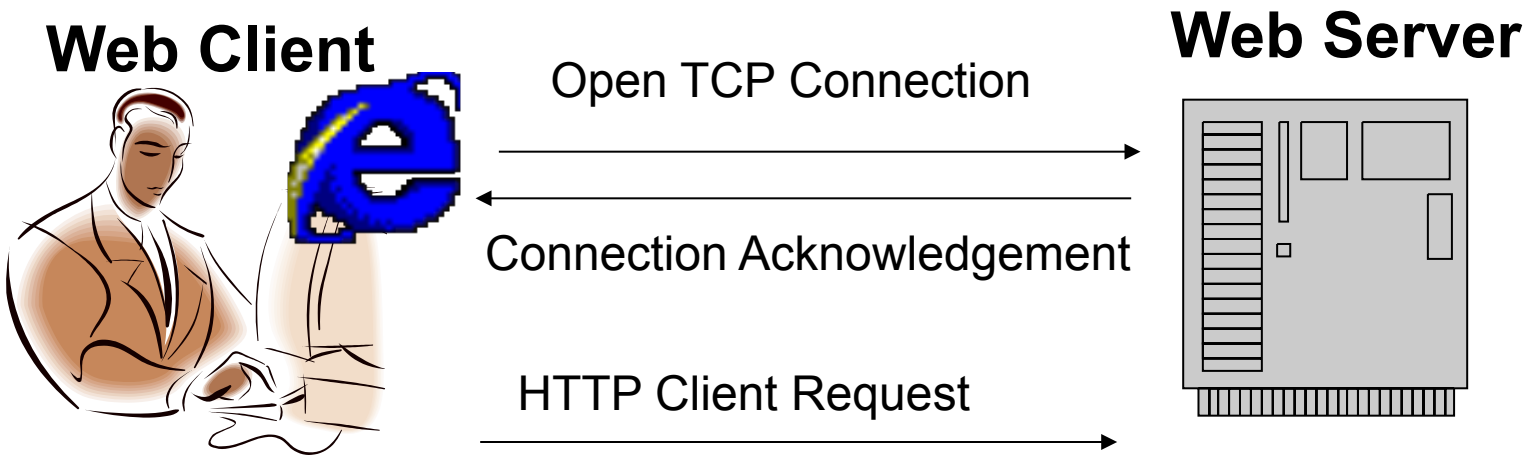
HTTP Low-Level Details

- HTTP 1.0 Communication Model



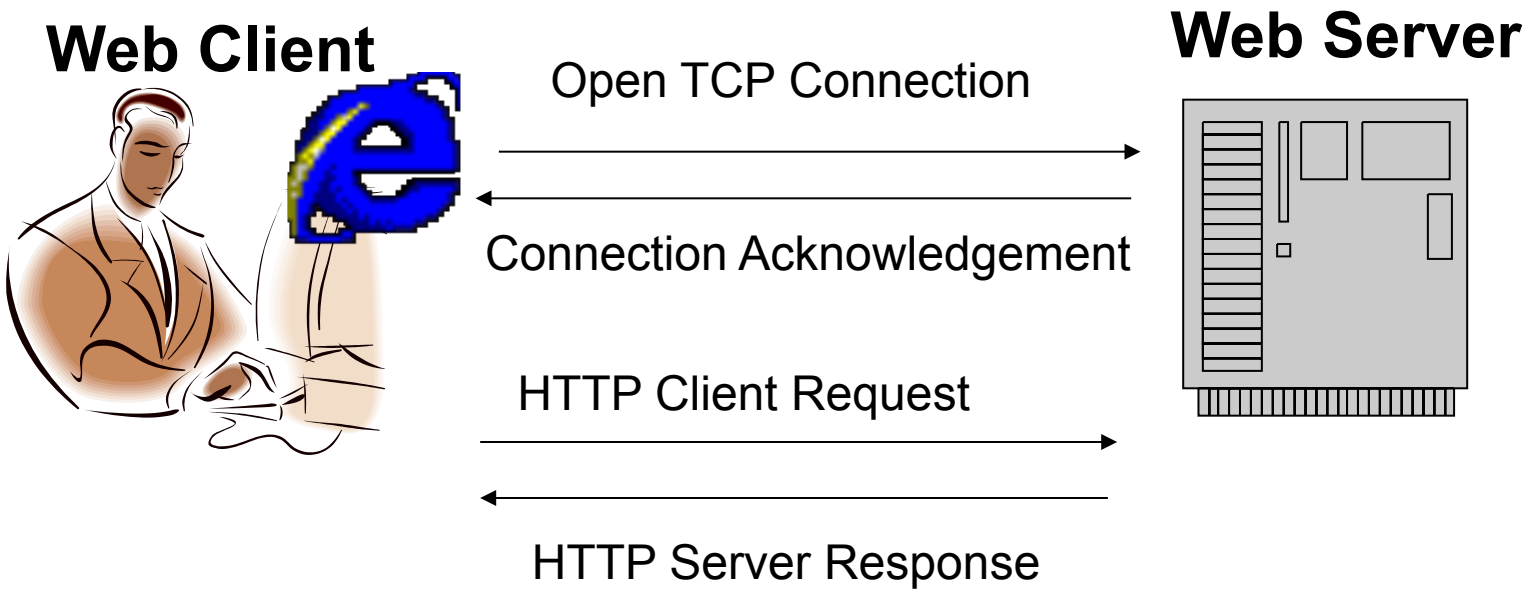
HTTP Low-Level Details

- HTTP 1.0 Communication Model



HTTP Low-Level Details

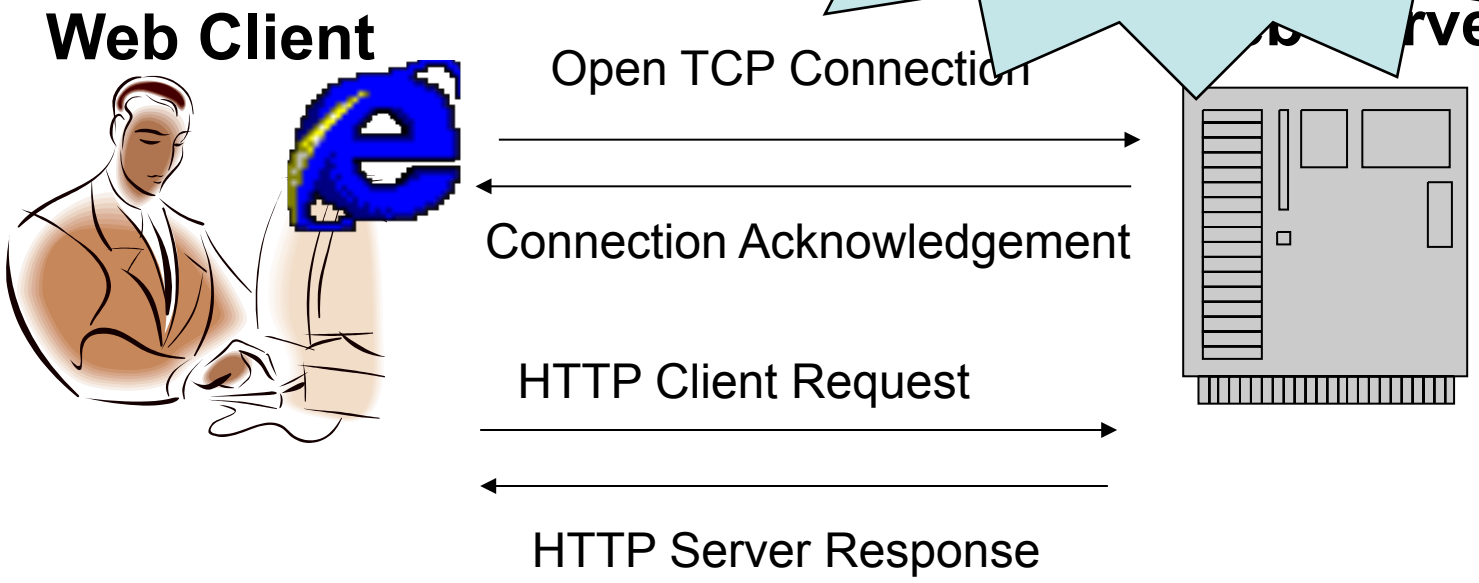
- HTTP 1.0 Communication Model



HTTP Low-Level D…

- HTTP 1.0 Communi…

One "Page View" May Result In Countless Connections…



HTTP Low-Level D…

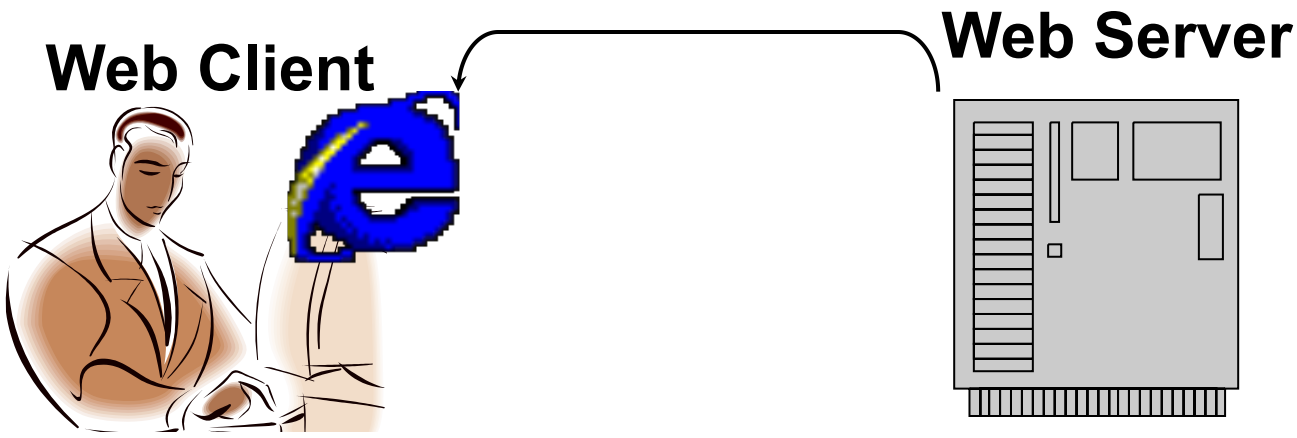
- HTTP 1.0 Communi…

One "Page View" May Result In Countless Connections…

HTTP 1.1 Allows For Persistent Connections. Connections Remain Open Until Connection: close

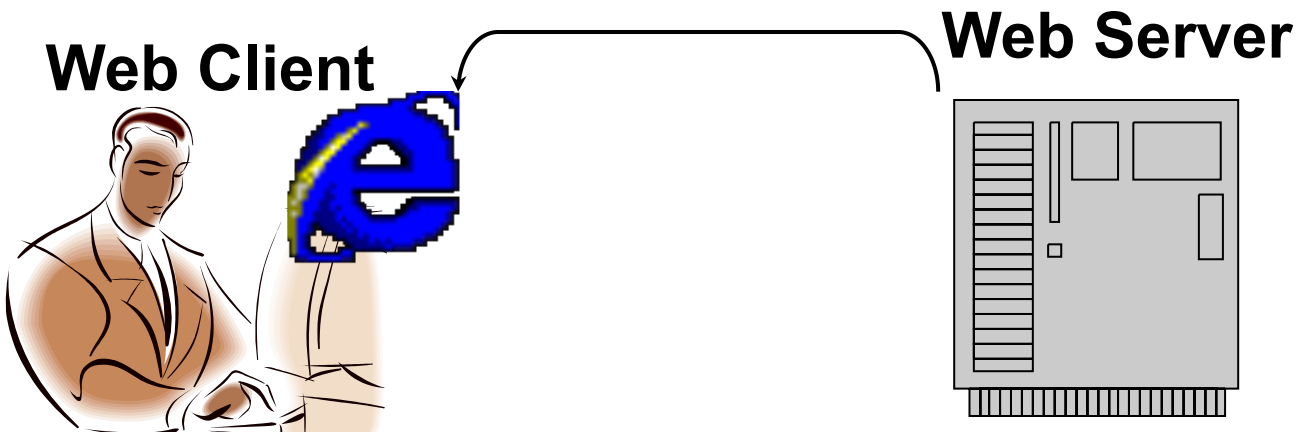
HTTP GET

- http://localhost:8080/index.html
  - Results In A GET Command To The Server From The Client
  - If Found, The Server Returns The Document



HTTP GET

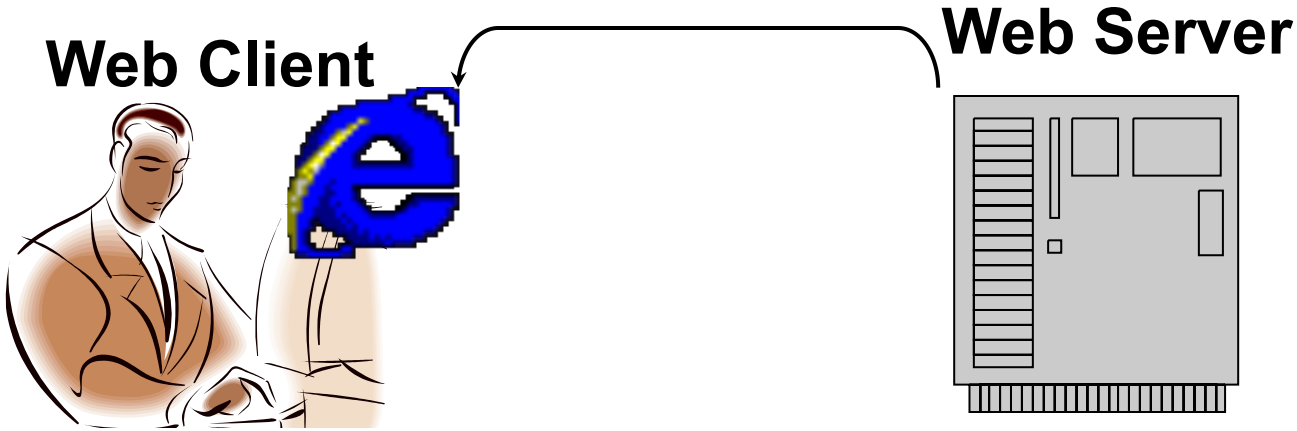
- http://localhost:8080/index.html
  - Results In A GET Command To The Server From The Client
- What Does The Server Actually Receive?



HTTP GET

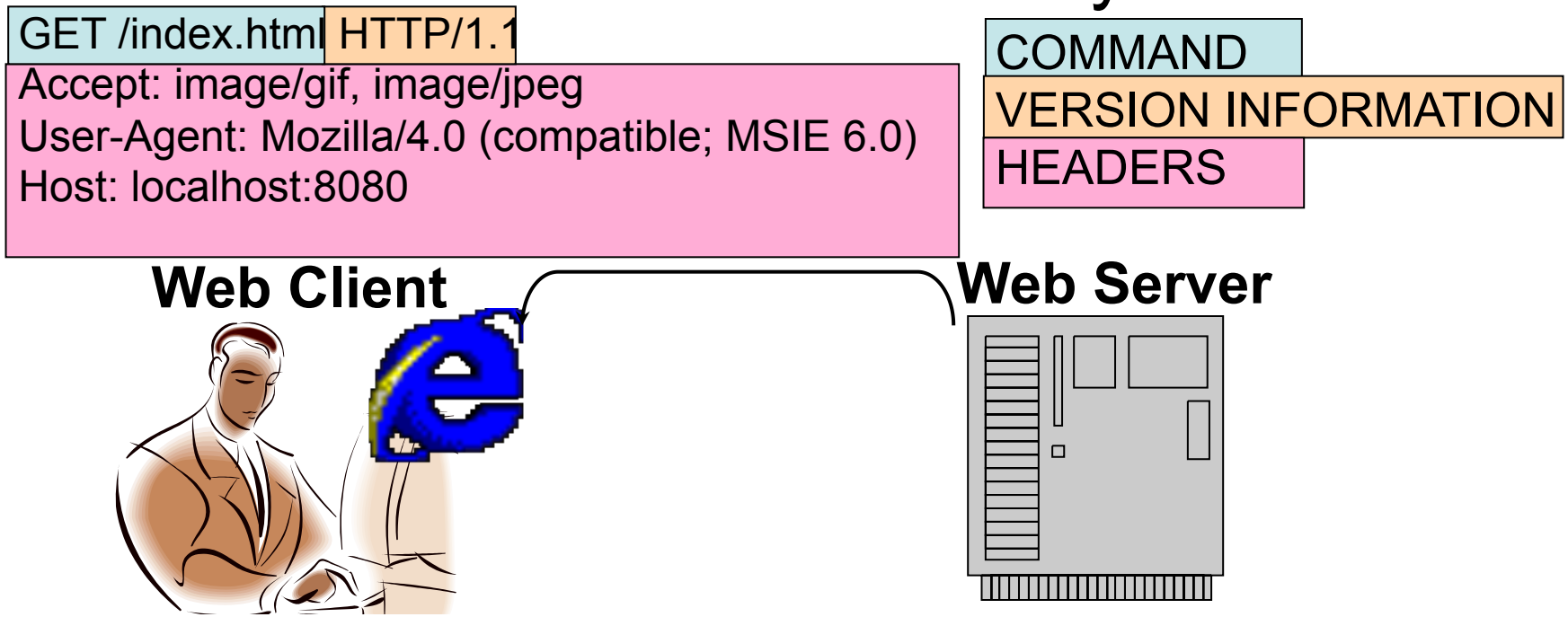
- http://localhost:8080/index.html
  - Results In A GET Command To The Server From The Client
- What Does The Server Actually Receive?

```
GET /index.html HTTP/1.1
Accept: image/gif, image/jpeg
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: localhost:8080
```



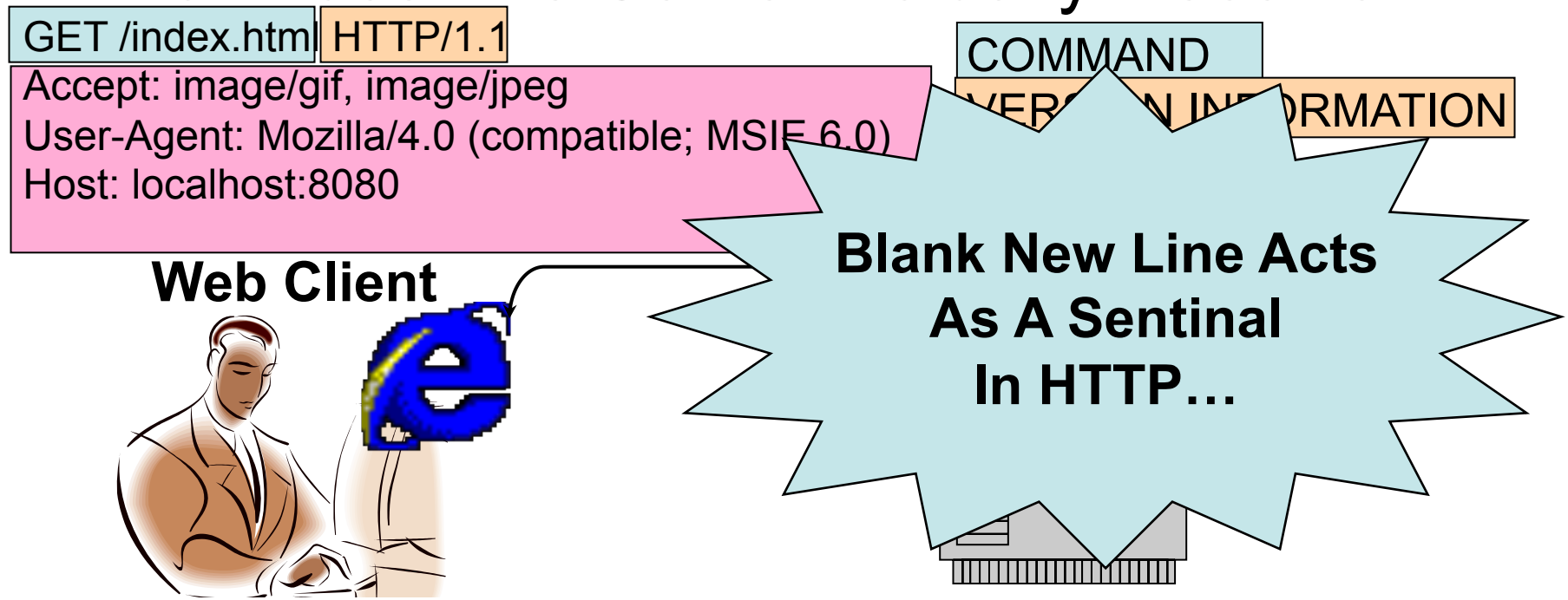
HTTP GET

- http://localhost:8080/index.html
  - Results In A GET Command To The Server From The Client
- What Does The Server Actually Receive?



HTTP GET

- http://localhost:8080/index.html
  - Results In A GET Command To The Server From The Client
- What Does The Server Actually Receive?



HTTP Is A Text-Based Protocol

- Before HTTP, Most Networking Protocols Were Binary-Based Information Exchanges

HTTP Is A Text-Based Protocol

- Before HTTP, Most Networking Protocols Were Binary-Based Information Exchanges

Binary Mode

Textual Mode

HTTP Is A Text-Based Protocol

- Before HTTP, Most Networking Protocols Were Binary-Based Information Exchanges

Binary Mode	Textual Mode
32.123456	32.123456

HTTP Is A Text-Based Protocol

- Before HTTP, Most Networking Protocols Were Binary-Based Information Exchanges

Binary Mode

32.123456

Stored In A 32-bit Double Variable

Textual Mode

32.123456

Stored In A String 9 Characters Long

HTTP Is A Text-Based Protocol

- Before HTTP, Most Networking Protocols Were Binary-Based Information Exchanges

Binary Mode	Textual Mode
32.123456	32.123456
Stored In A 32-bit Double Variable	Stored In A String 9 Characters Long
= 32 Bits Of Network Traffic	= 72 Bits Of Network Traffic

HTTP GET

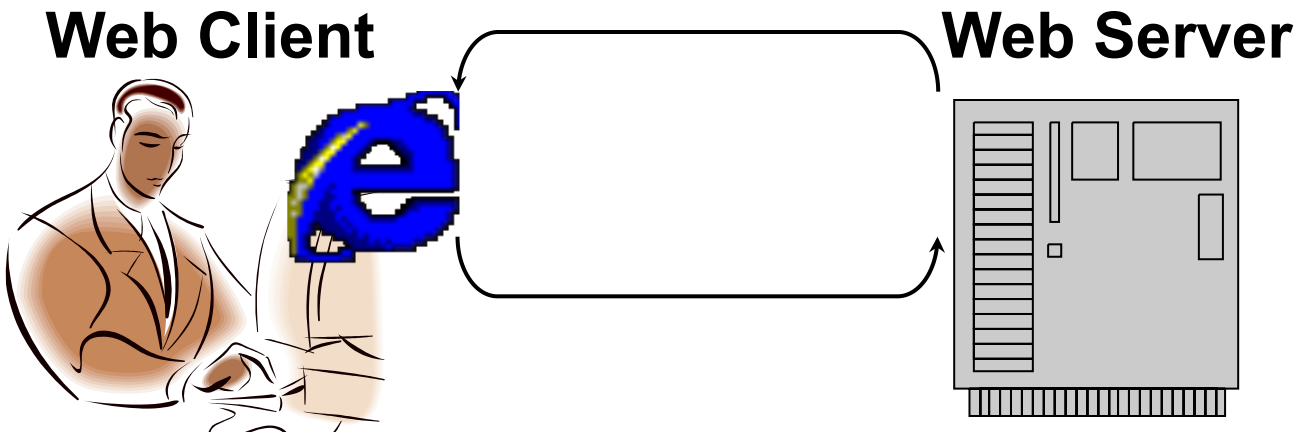
- The Command Is What The Non-Technical Person Expects The Server To Receive…
- The Version Information Is Fairly Hidden From View
  - Clients Typically Speak One Version, Servers Can Respond Potentially In Different Versions To Different Clients
  - Everything Used To Be 1.0, Now It's 1.1…
- The Headers Are Very Hidden From View

HTTP GET

- Understanding Headers Is Key To Understanding How HTTP Actually Works
  - Headers Communicate Information About The Client's Capabilities, So That The Server Responds Appropriately
  - Headers Are Magically Names And Values
  - Headers Supply "Extra Information" (aka Client Meta-Data) To The Server
- A Blank NewLine Is The Sentinel Which Marks The End Of All The Headers

HTTP GET

- http://localhost:8080/index.html
  - Results In A GET Command To The Server From The Client
- What Does The Client Actually Receive?

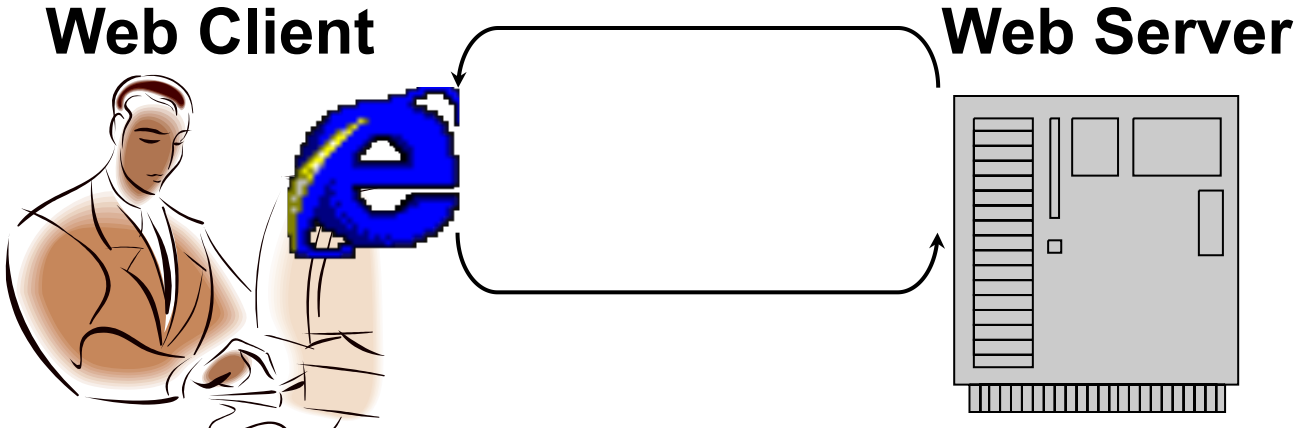


HTTP GET

- http://localhost:8080/index.html
  - Results In A GET Command To The Server From The Client
- What Does The Client Actually Receive?

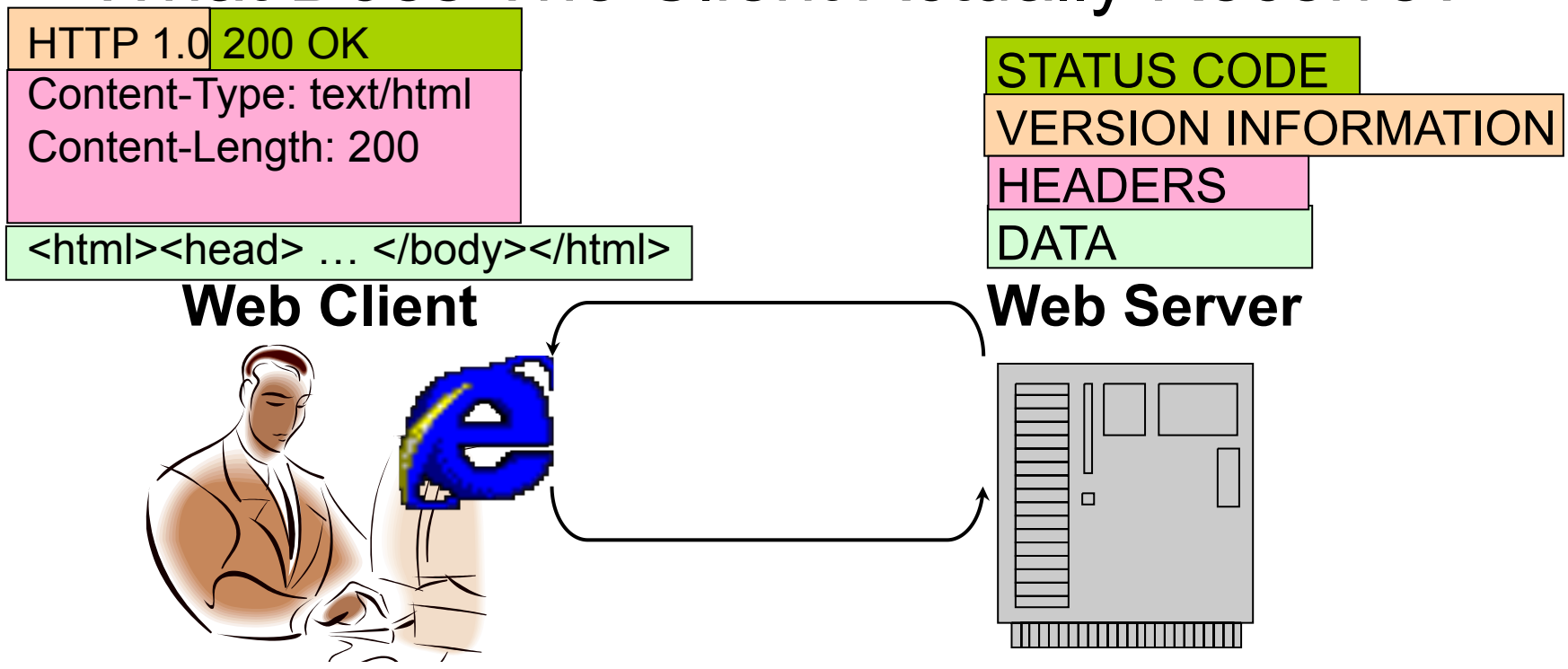
HTTP 1.0 200 OK
Content-Type: text/html
Content-Length: 200

<html><head> … </body></html>



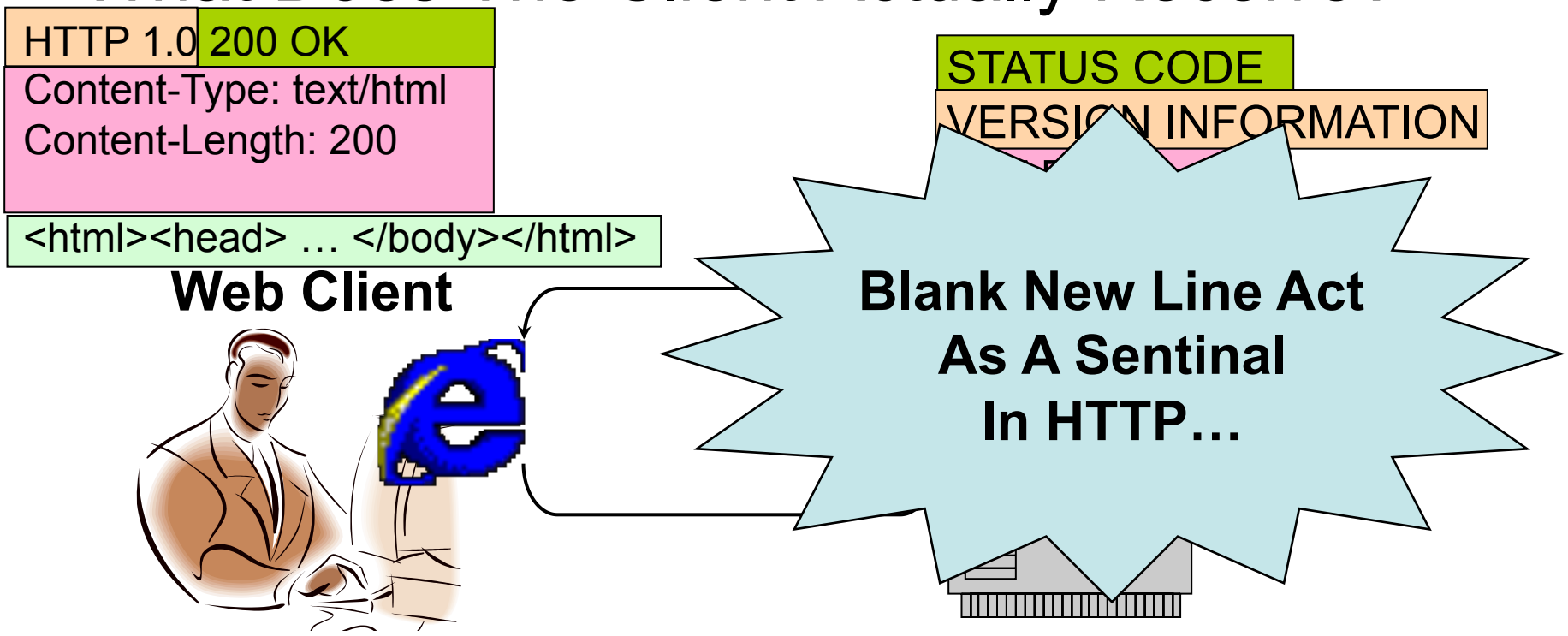
HTTP GET

- http://localhost:8080/index.html
  - Results In A GET Command To The Server From The Client
- What Does The Client Actually Receive?



HTTP GET

- http://localhost:8080/index.html
  - Results In A GET Command To The Server From The Client
- What Does The Client Actually Receive?



HTTP GET

- The Data Is What The Non-Technical Person Expects The Server Send…
- Headers Play A Key Role In HTTP
  - Headers Help The Client Interpret The Data
  - Content-Type Describes The MIME Type Of The Returned Data
  - Content-Length Describes How Many Bytes Should Be Expected In The Returned Data
- A Blank NewLine Is The Sentinel Which Marks The End Of All The Headers

HTTP GET

- A Status Code Of 200 Means Success
- There Are Many Other Status Codes That Indicate Various Kinds Of Failures
  - These Days, Most Browsers Display Standard “Error Pages” When Processing Bad Status Codes
  - Most Common Is 404 File Not Found

HTTP GET

- A Status Code Of 200 Means Success
- There Are Many Other Status Codes That Indicate Various Kinds Of Failures
  - These Days, Most Browsers Display Standard "Error Pages" When Processing Bad Status Codes
  - Most Common Is 40[illegible]

HTTP POST

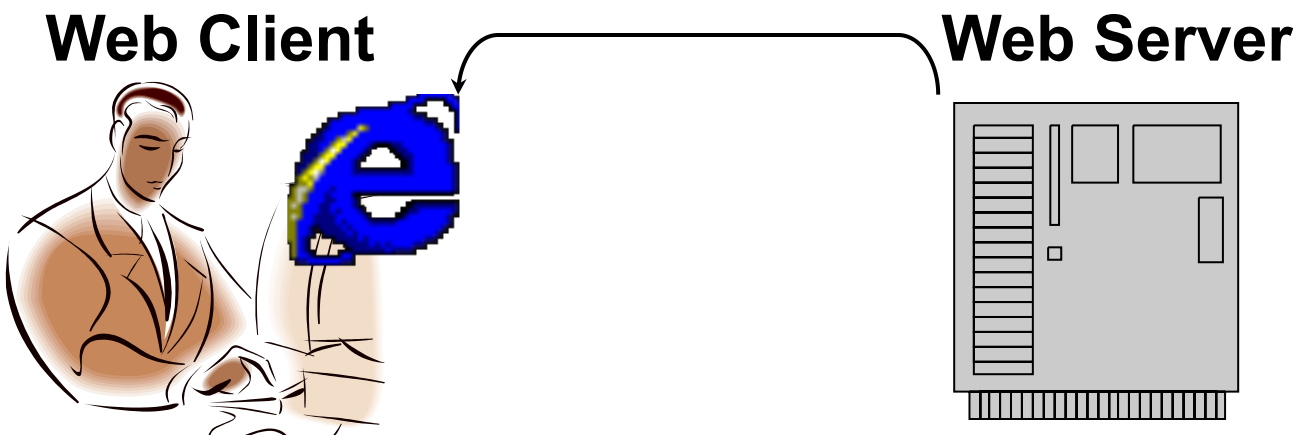
- Most Web Programmers Deal With The POST Command When Processing Web Forms
- The POST Command Allows A Payload Of Data To Travel With The Request From The Client To The Server
  - Typically, Encoded Form Data Travels In The Payload
  - Form Data Is Available To Server Side CGI

HTTP POST

- What Does The Server Receive?

```
POST /path/file HTTP/1.1
Host: localhost:8080
Content-Type: text/html
Content-Length: 150

Name1=Value&Name2=Value2
```



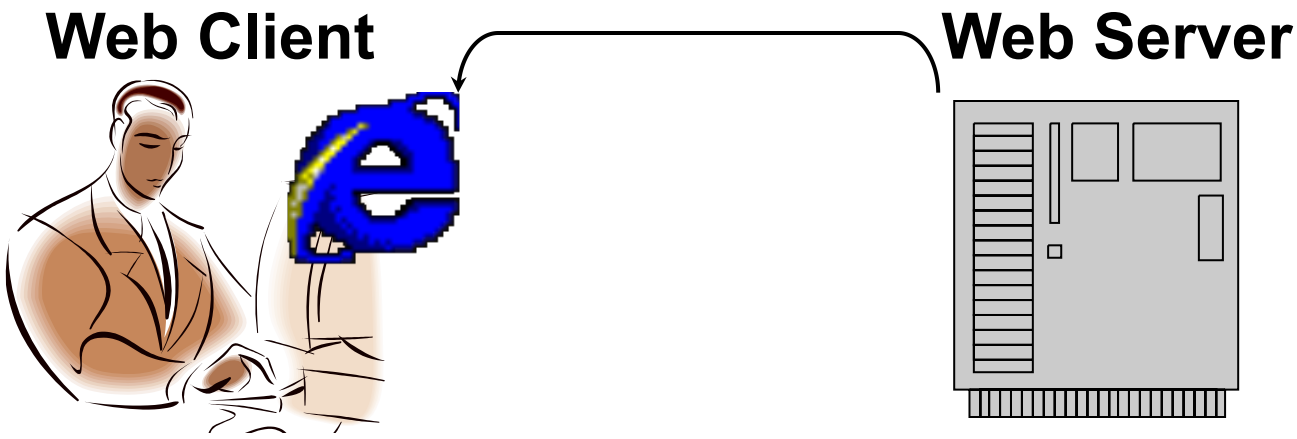
HTTP POST

- What Does The Server Receive?

```
POST /path/file HTTP/1.1
Host: localhost:8080
Content-Type: text/html
Content-Length: 150

Name1=Value&Name2=Value2
```

COMMAND
VERSION INFORMATION
HEADERS



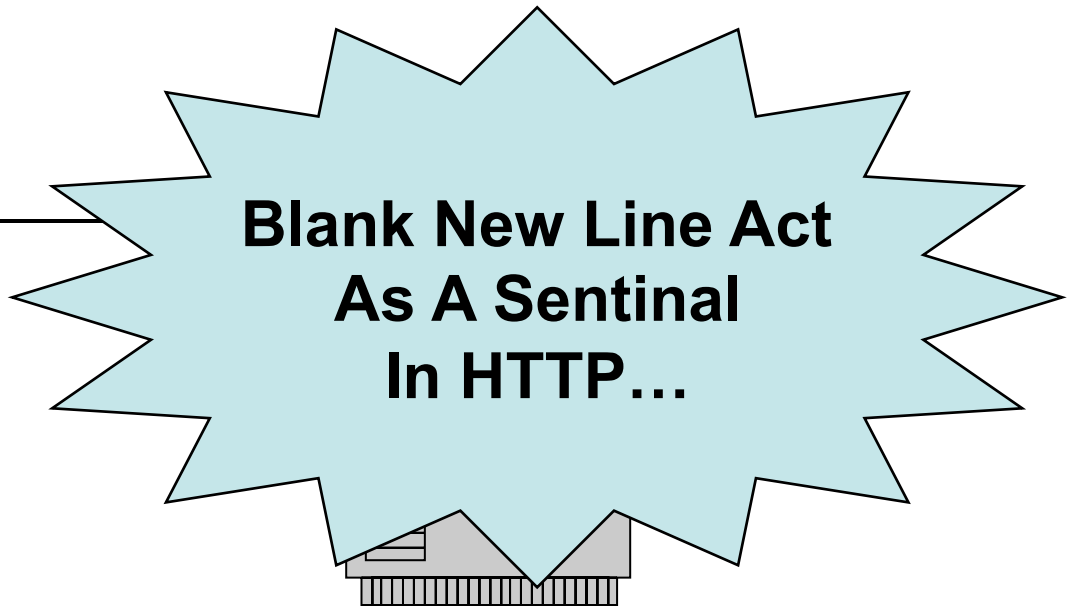
HTTP POST

- What Does The Server Receive?

POST /path/file HTTP/1.1
Host: localhost:8080
Content-Type: text/html
Content-Length: 150

Name1=Value&Name2=Value2

COMMAND
VERSION INFORMATION
HEADERS



Available HTTP Headers

- “Conditional” GET Requests Can Include:
  - If-Modified-Since
  - If-Unmodified-Since
  - If-Match
  - If-None-Match
  - If-Range
- All These Headers Allow The Server To Interact With Available Cached Pages, Preventing Unnecessary File Transfers

Available HTTP Headers

- "Conditional" GET Requests:
  - If-Modified-Since
  - If-Unmodified-Since
  - If-Match
  - If-None-Match
  - If-Range
- All These Headers Allow The Server To Interact With Available Cached Pages, Preventing Unnecessary File Transfers

PUT, DELETE and POST <u>Always</u> Invalidate The Cached Entry…

Available HTTP Headers

- "Conditional" GET Re
  - If-Modified-Since
  - If-Unmodified-Since
  - If-Match
  - If-None-Match
  - If-Range
- All These Headers A
  Interact With Availab
  Preventing Unnecessa

PUT, DELETE and POST <u>Always</u> Invalidate The Cached Entry…

We'll Investigate These Headers In More Detail Later On…

Summary

- HTTP Protocol
  - Headers
  - GET
  - POST