Csci 4131 Internet Programming

Lecture 11, Feb 21st
Spring 2024

Instructor: Dr. Dan Challou

Logistics – Csci 4131 Lecture 11, Feb 21st

- ***No zyBooks assignments this week***
- ***Hw3 is due next Sunday 3/3 (at 11:59pm)***
- ***Homework 3 is requires a significantly more complex solution than the previous 2 homework assignments***
- ***If you have not already read the assignment requirements specification; obtained your Google API key; finished much of the google maps JavaScript API tutorial, and have parts of the Assignment Working – YOU ARE BEHIND***

Reading/Tutorials – Upcoming

- HTTP Protocol
  - https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html
  - **And see course Schedule on Canvas (In the Resources Module)**

Necessary Preparations for enabling your understanding of HTTP: Getting Setup for Python 3.x – which you will use to do HW4

- https://www.python.org – to download python to your machine so you can develop and run HW Assignment 4 – note, our target machines are ubuntu, and our examples have been developed and tested on the cse labs machines. That is where we will be testing **and GRADING** your programs.

- https://docs.python.org/3/ - documentation on newest version of Python
- https://docs.python.org/3/howto/sockets.html
- https://docs.python.org/3/howto/sockets.html

- - documentation on the socket library – which will help you understand EchoClient and EchoServer python programs (which I'll post on Canvas for your review and refactoring for HW Assignment 4)

Learning or Refreshing your Python

- https://docs.python.org/3/tutorial/
- https://www.learnpython.org/

Upcoming: Node.js

Introduction to Node.js (Building a Webserver in JavaScript)

- https://www.w3schools.com/nodejs/
- https://codeburst.io/the-only-nodejs-introduction-youll-ever-need-d969a47ef219

Last Time (2/20):

Closures and Race Conditions, Revisited and Wrapped up

Google Maps overview and review

Started HTTP Protocol

Today

HTTP, URL’s, and ISO Protocols + Interfaces between layers

HTTP great in detail

Gearing up to build an HTTP Server in Python

Review Lecture 10, Exercise 1:

Refactor the HTML and JavaScript below so the text **Hello World** is Displayed in the text box when the browser loads the page racecondition.html

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Race Condition</title>
        <script language=javascript>
                var textobj = document.getElementById("stuff");
                textobj.value = "Hello World";
        </script>
</head>
<body>
        <input type="text" id="stuff">
</body>
</html>
```

- See: rfix.html

Summary

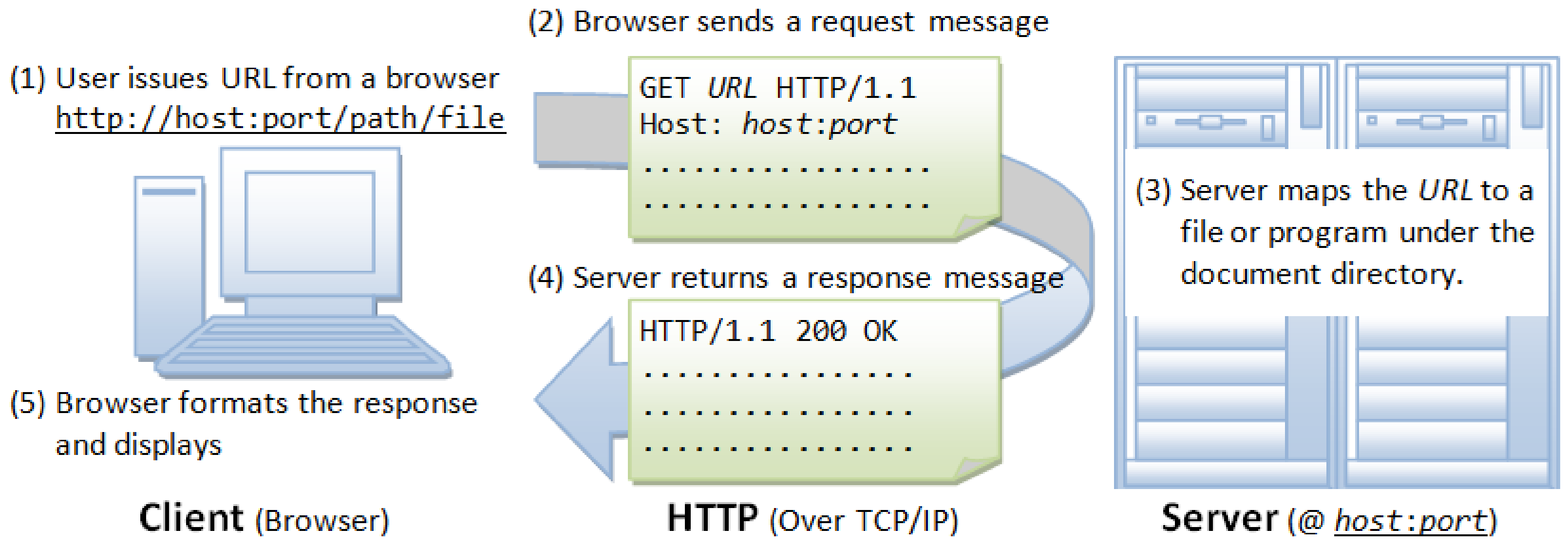
- Make sure you don't have statements that reference DOM elements outside of script functions that you load in the header, and
- then associate the functions with the objects and events by writing a function that is called when the ***window***, ***document***, or ***body*** **onload** event occurs
  - https://www.w3schools.com/jsref/event_onload.asp
- **Or???**
- Place the script(s) as the last item(s) in the document's body
- ***Are there other solutions?***
- **Of course**!

Questions?

HW3 - The Stock API

- Demo
- Key Concepts –
  - Include stocks.js in your html file before other external js files with methods that use it (remember to update your server to return the file stocks.js when it is requested!!!)
  - Examples are in the README.md file – run them to see how it works. Example – regular stock time series retrieval
    - https://github.com/wagenaartje/stocks.js/blob/master/README.md
  - You are using a fetch to get the data – see section 8.15 in your zyBook!!!!

Recall, the Hypertext Transfer Protocol (HTTP) – how info moves over the world wide web


(1) User issues URL from a browser
http://host:port/path/file
(2) Browser sends a request message
GET URL HTTP/1.1
Host: host:port
(3) Server maps the URL to a file or program under the document directory.
(4) Server returns a response message
HTTP/1.1 200 OK
(5) Browser formats the response and displays
Client (Browser)
HTTP (Over TCP/IP)
Server (@ host:port)

URI’s, URL’s, URN’s

- URL scheme is based on DNS (Domain Name System – translates names into IP addresses). It uses DNS to identify the Internet host of the resource.
- URLs belong to a more general class of naming scheme called Uniform Resource Identifiers (URI).
- URI also defines a location independent naming scheme called Uniform Resource Names (URNs). See URI:

http://www.w3.org/hypertext/WWW/Addressing/Addressing.html

URI’s, URN’s, URL’s

- https://geekflare.com/difference-between-url-uri-and-urn/

- https://www.geeksforgeeks.org/difference-between-url-uri-and-urn-in-java/

Recall URL’s

- http://www.w3.org/Addressing/URL/Overview.html

- URL’s are a naming scheme for referencing resources in the Internet. See RFC 1738.
- The URL enables resources to be accessed without knowing the specifics of their underlying protocol, such as FTP or HTTP.

It is a location-dependent scheme. i.e., a URL name becomes invalid if the resource is relocated to another host or moved to a different part of the file system.

URL’s

- Recall, a URL has the following syntax:
  protocol://*hostname*:*port*/*path-and-file-name*

- There are 4 parts in a URL:
  - *Protocol*: The application-level protocol used by the client and server, e.g., HTTP, FTP, and telnet.
  - *Hostname*: The DNS domain name (e.g., www.test101.com) or IP address (e.g., 192.128.1.2) of the server.
  - *Port*: The TCP port number that the server is listening for incoming requests from the clients.
  - *Path-and-file-name*: The name and location of the requested resource, under the server document base directory.

URL

A URL can be in one of the two forms:

- **Absolute or Complete URL**
  - Specifies the complete access path for the named resources in the Internet.
- **Relative or Partial URL**
  - Meaningful only in the context of some other URL.
  - Used when the referenced resource is on the same host machine as the referring resource.

THINK/PAIR/SHARE

Write down an complete/absolute URL:

Write down a relative URL Share em!!!!

Absolute/Complete URL

The specification of an absolute URL contains the following information:

•Scheme: Protocol to be used to access the resource: e.g., ftp, http, mailto (name others???)

•**hostname or domain name** of the server that contains the resource (resolves to an IP address)

•If needed, specification of the server's port number.

•The directory path within which the resource is contained.

•The **name of the file** representing that resource.

•Some specific named component within the resource, such as a **named "anchor" within an HTML document**.

•**Query parameters** to be passed to the resource.

Example of a URL with a Query string in it

https://en.wikipedia.org/w/index.php?title=Query_string&action=edit

Source: https://en.wikipedia.org/wiki/Query_string

Example

- Example of a URL with encoded query with it.
- http://www.cs.umn.edu/admissions/application.cgi?param1=value1&param2=value2
- Parameters (param 1 and param 2) to be passed to the Common Gateway Interface (CGI) program
- This (name, value) query will be passed to the CGI program as the QUERY_STRING environment variable

Examples continued

- Example of URL with inclusion of single parameter to be passed to the resource.

- http:// www.cs.umn.edu /admissions/application.php?**someValue**

- **SomeValue** - Command line parameter to be passed to the PHP script

Think/Pair/Share: What is the URL of the following file?

- The file: **myContacts.html**
- With hostname: **cs.umn.edu**
- Accessed by the: **http protocol**

- Also, is this a complete/absolute or partial/relative URL?

Please jot your answers down, and then share them

URL Port Numbers

- Recall, a URL has the following syntax:

  protocol://*hostname*:**port**/*path-and-file-name*

  <u>BUT the Port number can be omitted if the server is running on the default port for that service.</u>

- For example,:

  HTTP servers (port 80)
  HTTPS (port 443),
  FTP (port 21),
  Telnet (port 23).

- Protocol can be:
  - http, telnet, ftp, mailto, others

Examples of URLs that use protocols other than HTTP / HTTPS

Telnet URL

- **telnet hostName[:port]**

- For example: to Telnet to **csel- kh1262-01.cselabs.umn.edu**

  % **telnet csel-kh1262-01.cselabs.umn.edu**

- HTML:

<a href="telnet csel-kh1262-01.cselabs.umn.edu"> Telnet to csel-kh1262-01.cselabs.umn.edu</a>

FTP URL

ftp host[port]

For example to ftp to csel-kh1262-01.cselabs.umn.edu

```
% ftp csel-kh1262-01.cselabs.umn.edu
```

HTML:

```
<a href="ftp csel-kh1262-01.cselabs.umn.edu">
ftp to csel-kh1262-01.cselabs.umn.edu </a>
```

Exercise 1: **Submit your response to the Lecture 11, Exercise 1 Submission Link in week 6 module**

Please raise virtual thumb when done

Specify which of the following items is a URI, URL***, and/or*** URN (URN is defined as Universal Resource Name).

a) isbn:0451450523

b) http://www.cs.umn.edu/academics/classes.php

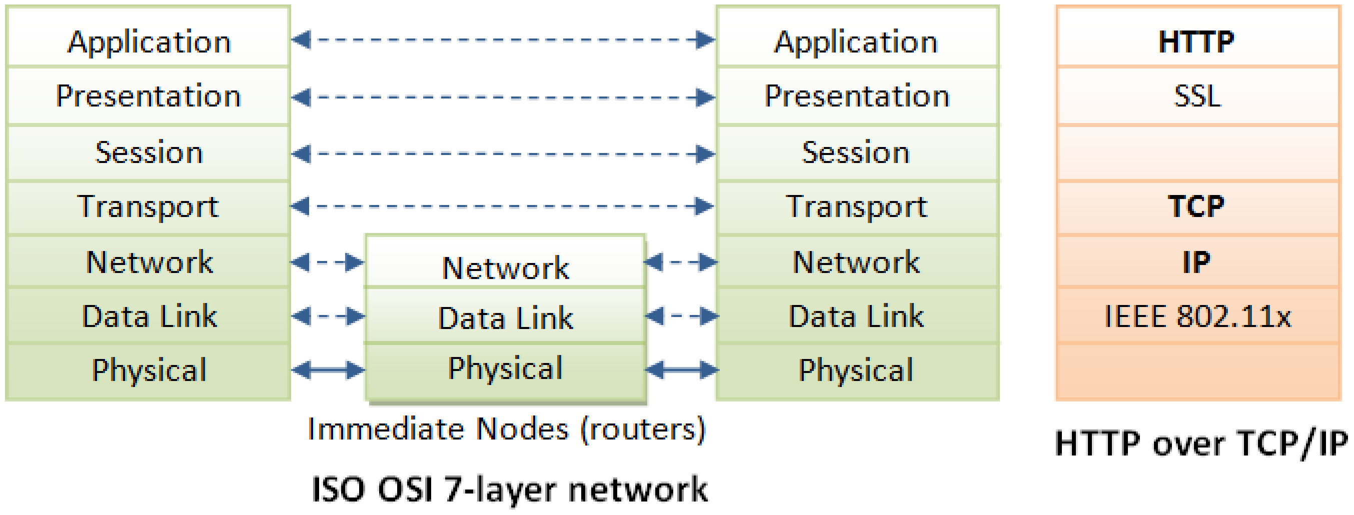
c) telnet://195.0.1.12:80/

d) mailto:Jim.Smith@acme.com

e) tel: 19146286389

HTTP relies on protocols below it to function - which are usually are built on TCP/IP

- HTTP is a client-server application-level protocol.
- It typically runs over a presentation layer protocol with a TCP/IP connection underneath, as illustrated below.



- (***HTTP need not run on TCP/IP. It only presumes a reliable transport. Any transport protocols that provide such guarantees can be used.***)
- See: https://en.wikipedia.org/wiki/Transport_layer for a nice discussion of other possible protocols

TCP/IP

- TCP/IP (Transmission Control Protocol/Internet Protocol) is a set of transport and network-layer protocols for machines to communicate with each other over the network.

- TCP (Transmission Control Protocol) is a transport-layer protocol, responsible for establishing a connection between two machines.

- TCP consists of 2 protocols: TCP and UDP (User Datagram Package).

- TCP is *reliable*: each packet has a sequence number, and an acknowledgement is expected. A packet will be re-transmitted if it is not received by the receiver. Packet delivery is guaranteed in TCP.

- UDP does not guarantee packet delivery => not reliable.
  - But, UDP has less network overhead and can be used for applications such as video and audio streaming, when reliability is not as critical.

What is the correct version of the sentence below? **(Share)**

- Teh quik brwon fox jumes over teh lzay dog

- Similarly, you will still get the meaning of an audio or video transmission if blips like this occur during the transmission
- You will notice the lack of quality in the transmission however

TCP / **IP** Continued

- IP (Internet Protocol) is a network-layer protocol (network addressing and routing).

- In an IP network, each machine is assigned an unique IP address (e.g., 165.1.2.3), and the IP software is responsible for routing a message from the source IP to the destination IP.

- In IPv4 (IP version 4), the IP address consists of 4 bytes, each ranges from 0 to 255, separated by dots, which is called a *quad-dotted form*. This numbering scheme supports up to 4G addresses on the network. IPv6 supports more addresses (16 bytes worth)

- Since memorizing a 12 digit number is difficult for most of the people, an english-like host / domain name, such as www.test101.com is used instead.

- The DNS (Domain Name Service) translates the domain name into the IP address (via distributed lookup tables).

- **A special IP address 127.0.0.1 always refers to your own machine. It's domain name is "localhost" and can be used for *local loopback testing. DNS translate localhost to 127.0.0.1***

TCP / IP Continued

- TCP *multiplexes* applications within an IP machine.
- For each IP machine, TCP supports (multiplexes) up to 65536 ports (or sockets), from port number 0 to 65535.
- An application, such as HTTP or FTP, runs (or listens) at a particular port number for incoming requests.
- Port 0 to 1023 are pre-assigned to popular protocols:
  - HTTP at 80,
  - HTTPS at 443,
  - FTP at 21,
  - Telnet at 23,
  - SMTP (Simple Mail Transfer Protocol) at 25,
  - NNTP (Nework News Transfer Protocol) at 119,
  - and DNS at 53.
  - Port 1024 and above are available to the users.

TCP / IP Final Thoughts / Summary

- TCP port 80 is pre-assigned to HTTP, as the default HTTP port number.
- This does not prohibit you from running an HTTP server at other user-assigned port number (1024-65535) such as 8000, 8080 (e.g., when testing a new server).
- You can also run multiple HTTP servers in the same machine on different port numbers.
- When a client issues a URL without explicitly stating the port number, the browser will connect to the default port number 80 of the host.
- The request from the client needs to explicitly specify the port number in the URL if they want to request something from a server that is not listening to the default url.

  e.g. **http://www.test101.com:8000/docs/index.html will request**

  docs/index.html

  from the server listening at port 8000

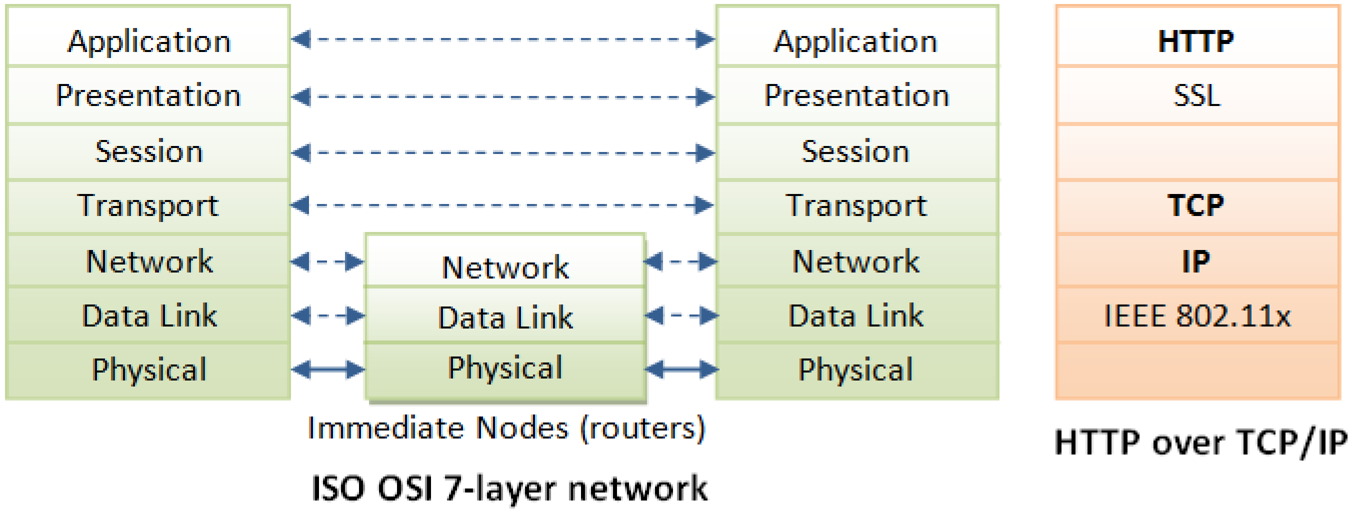
- **SUMMARY:**

  For TCP/IP to function, it needs to know:

  (a) IP address or hostname, (b) Port number.

HTTP can be built on an Interface to Presentation Layer

- HTTP is a client-server application-level protocol.
- It typically uses the protocols below it in the 7 layer model below, with TCP/IP underneath (*but any guaranteed Transport Layer Protocol will work*), as illustrated below.



- ***What mechanism can you name that functions on the Presentation Layer in the 7-layer ISO OSI 7 – layer network Model??***
- **Please venture an answer in Chat**

HW 4 Concept (Foreshadowing)

- You will update your Python server to obtain the appropriate file obtaining the file from the relative URL passed into it and the Multipurpose Internet Main Extension (MIME) type
- No more hardcoding to recognize each file
- Update the server to save information from post requests!!!!?
- And, perhaps update the server in other ways

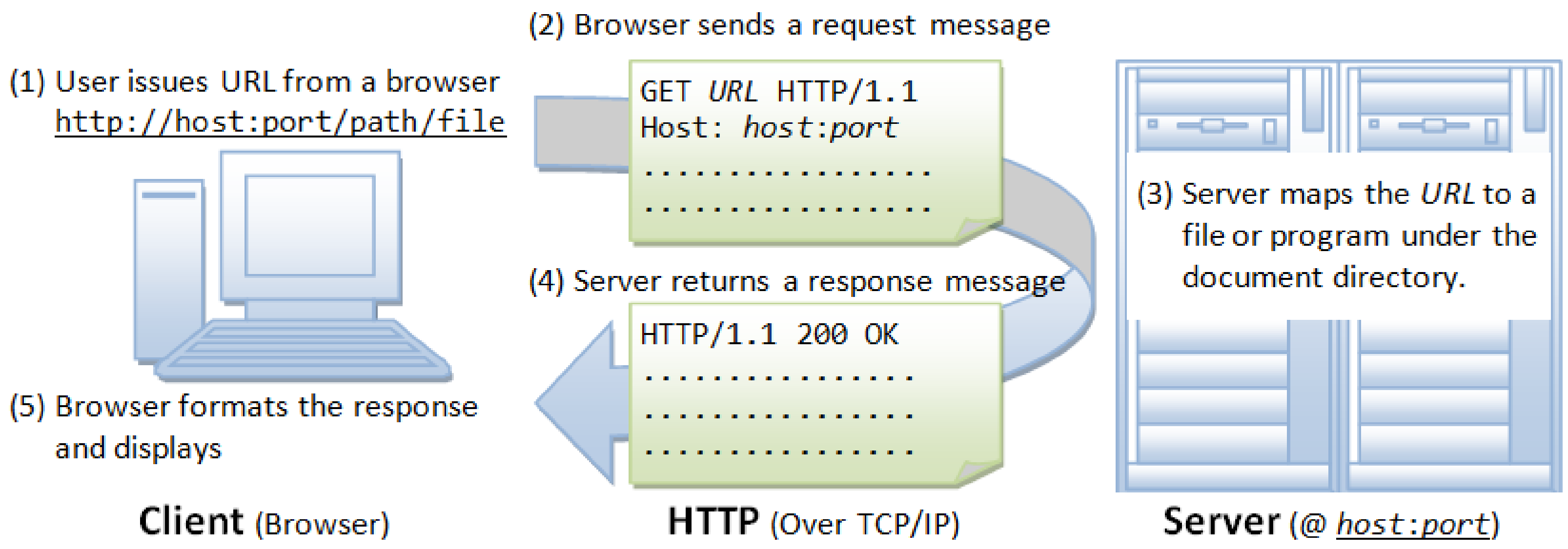
Details, Details – You'll need to understand them to build your Server

https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

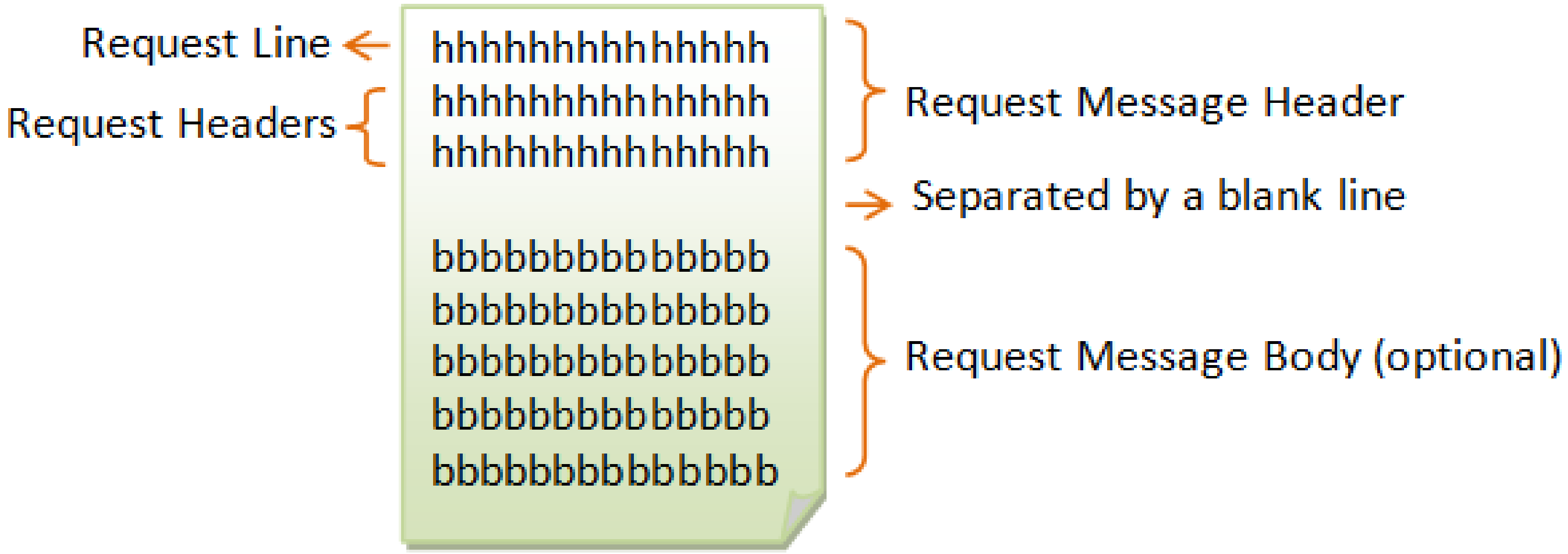
http://www.w3c.org/Protocols/

- Do at least the first reading,
- Refer to the second official Reference for clarification on the details of HTTP 1.1

Recall, yet again, how the HTTP Protocol Works

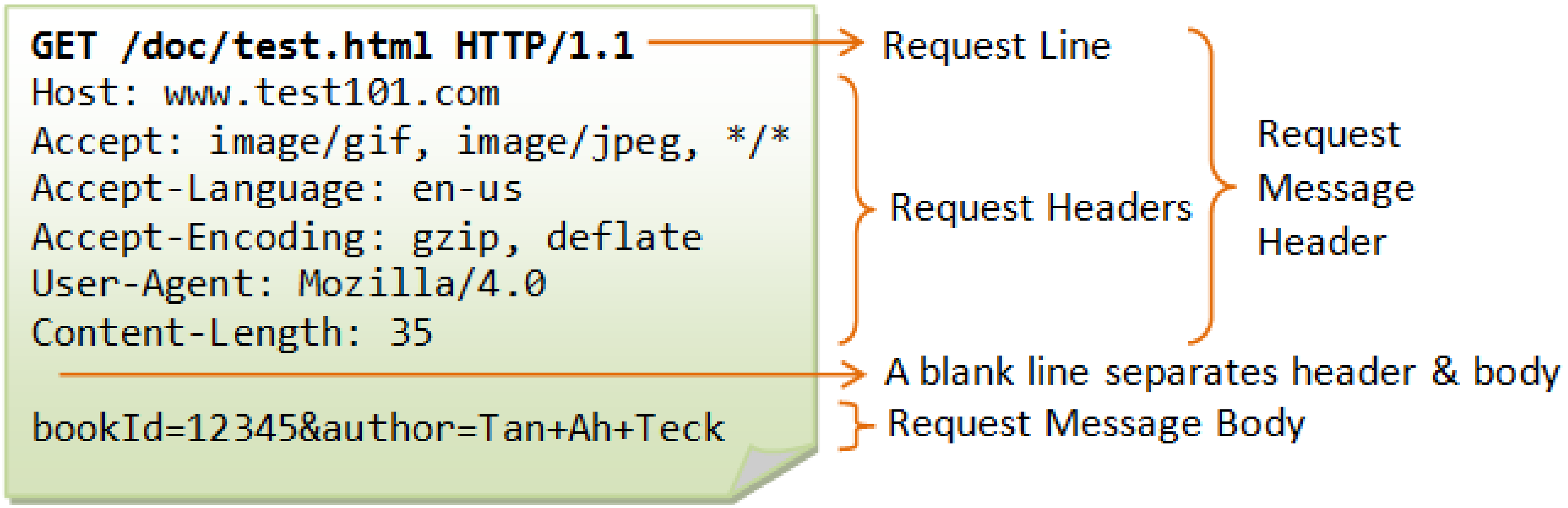

(1) User issues URL from a browser
http://host:port/path/file
(2) Browser sends a request message
GET URL HTTP/1.1
Host: host:port
(3) Server maps the URL to a file or program under the document directory.
(4) Server returns a response message
HTTP/1.1 200 OK
(5) Browser formats the response and displays
Client (Browser)
HTTP (Over TCP/IP)
Server (@ host:port)

Format of HTTP Request Message



HTTP Request Message

HTTP Request Message Example



```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

http://www.test101.com/doc/test.html?bookId=12345&author=Tan-Ah-Tek

HTTP Request – The Request Line (First Line in the Request)

The first line is the REQUEST LINE, and it contains three items:

1. Name of the requested operation.
2. Request-URL (relative URL - specifying the resource).
3. HTTP version.

In HTTP, message communication is built upon MIME (Multipurpose Internet Message Extension) format.

HTTP Request Methods

- GET: A client can use the GET request to get a resource from the web server.
- HEAD: A client can use the HEAD request to get the header that a GET request would have obtained. Since the header contains the last-modified date of the data, this can be used to check against the local cache copy.
- POST: Used to post data up to the web server.
- PUT: Ask the server to store the data.
- DELETE: Ask the server to delete the data.
- TRACE: Ask the server to return a diagnostic trace of the actions it takes.
- OPTIONS: Ask the server to return the list of request methods it supports.
- CONNECT: Used to tell a proxy to make a connection to another host and simply reply the content, without attempting to parse or cache it. This is often used to make SSL connection through the proxy.
- Other extension methods

And That’s All Folks: Next Time

- More on HTTP
- Intro to Node.js (The server language we will use to build a more functional server at a more abstract level. - ?