

# SOC2110: Internet (Web) Programming

Dr. Sarvar Abdullaev

# Course Overview

- Internet infrastructure and basics of HTTP
- Basic HTML and CSS
- Back-end development with PHP
- Database design with MySQL
- Cookies, Sessions and Authentication
- JavaScript and utilities
- HTML5, CSS3 and tools
- Web Services
- Front-end development with AngularJS

# Course Logistics

- **Instructor:**
  - Dr. Sarvar Abdullaev, [s.abdullaev@inha.uz](mailto:s.abdullaev@inha.uz)
  - Office hours:
    - Wednesday: 14:00-15:00
  - Room B411.
- 1.5 hours lecture + 1.5 hours lab
- **Assessment:**
  - Group Design Project: 30%
  - Mid-term exam: 30%
  - Final exam: 30%
  - Lab Participation (Attendance): 10%

# Course Objectives

- At the end of this class you will be able to:
  - Design and implement a professional website
  - Author web pages using HTML
  - Make stylistic decisions with CSS
  - Create interactive websites with JavaScript, jQuery and AJAX and XML
  - Use PHP for server-side programming
  - Use Laravel to build server-side web applications based on MVC architecture
  - Use ReactJS to build client-side web applications

# Course Objectives (cont.)

- At the end of this class you will be able to:
  - Understand the client-server programming model and apply this to your designs
  - Create your own web programming portfolio
  - Speak the web programming lingo
  - Have fun with web programming!

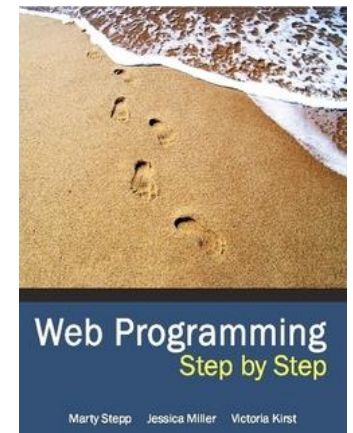
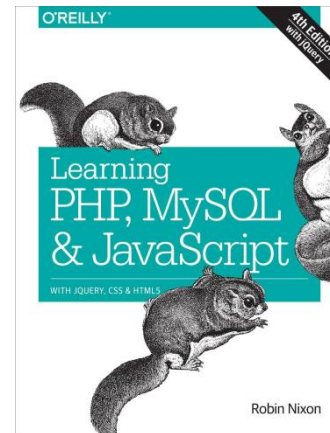


# Lectures and Labs

- Lectures will contain loads of hands-on coding. This means:
  - You are recommended to use your laptops during the lecture
  - You will have a lab exercise after every lecture to practice what we have learned
  - Completing Lab Exercises on time contribute to your participation mark (max 10% of your final mark).
  - You will post your questions on the discussion board before each lecture. If you do not post any questions, I assume you have understood everything. Therefore...
  - You are expected to ask questions while doing lab exercises
  - You may be asked in lab to explain material to your classmates

# Literature

- **Core textbook:**
  - “Web Programming Step by Step” by Marty Stepp, Jessica Miller, Victoria Kirst
- **Additional references:**
  - “PHP, MySQL and JavaScript with jQuery, CSS and HTML5” by Robin Nixon
  - Other documentation links



# Design Project

- You will have it assigned in week 9 immediately after mid-term.
- However you can start working on that now.
- Design and implementation of a professional website:
  - Professional Style
  - Interactive
- I will post the list of topics in eClass system after mid-term.
- You should complete the project in teams of 4-6 people
- I reserve the right not to grade every member of the team with the same mark for the design project if I suspect someone in the team freeloading.



# Design Project (sneak-peek)

- Your project should have five out of the following features (choose and document these):
  - 1. Use a Server-Side Framework** - use a technology other than HTML/CSS on the server.
  - 2. AJAX** - use AJAX to turn your web pages into dynamic web applications.
  - 3. Web Service** - use an external web service, mashed up with your own application to create something even better.

# Design Project (sneak-peek)

**4. Design & Evaluate** - think carefully about how users will use your site, design a great interface, and evaluate it with real people.

**5. Go Mobile** - create a version of your project designed to go mobile.

**6. Server-Side Processing** - do processing on the server to prepare for user requests in advance.

**7. Multimedia** – use sound or video to enhance the user experience.

# Introduction to Web Programming

Lecture 1

# Overview

- Internet and Web Infrastructure
- HyperText Transfer Protocol (HTTP) Basics
- Client-side and Server-side web programming

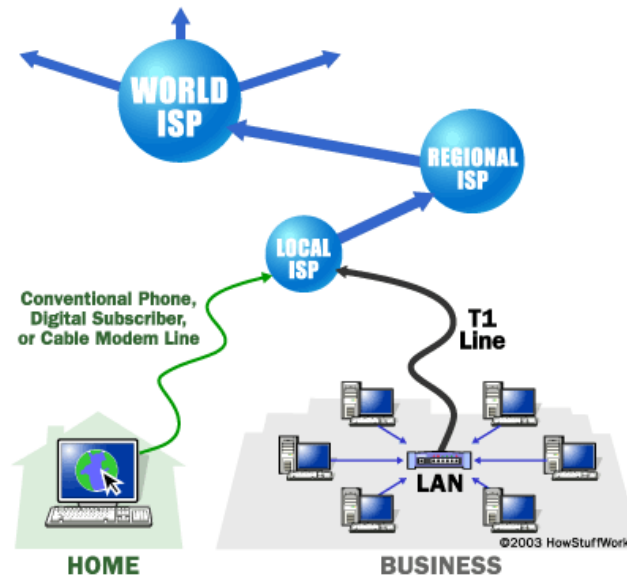
# What is the internet?

- A “series of tubes”
- How many Internets are out there?
- Is Google one of them?



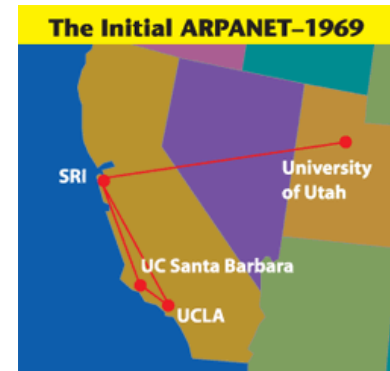
# What is the internet?

- A collection of computer networks that use a protocol to exchange data
- Is the World Wide Web (WWW) and the internet the same?



# Brief history

- Began as a US Department of Defense network called ARPANET (1960s-70s)
- Packet switching (in the 60s)
- E-mail is born on 1971
- TCP/IP beginning on 1974 (Vinton Cerf)
- USENET (1979)
- By 1987: Internet includes nearly 30,000 hosts



# Brief history (cont.)

- WWW created in 1989-91 by Tim Berners-Lee
- Popular web browsers released:
  - Netscape 1994
  - IE 1995
- Amazon.com opens in 1995
- Google January 1996
- Wikipedia launched in 2001
- MySpace opens in 2003
- Facebook February 2004





# The future of the internet?



# Key aspects of the internet

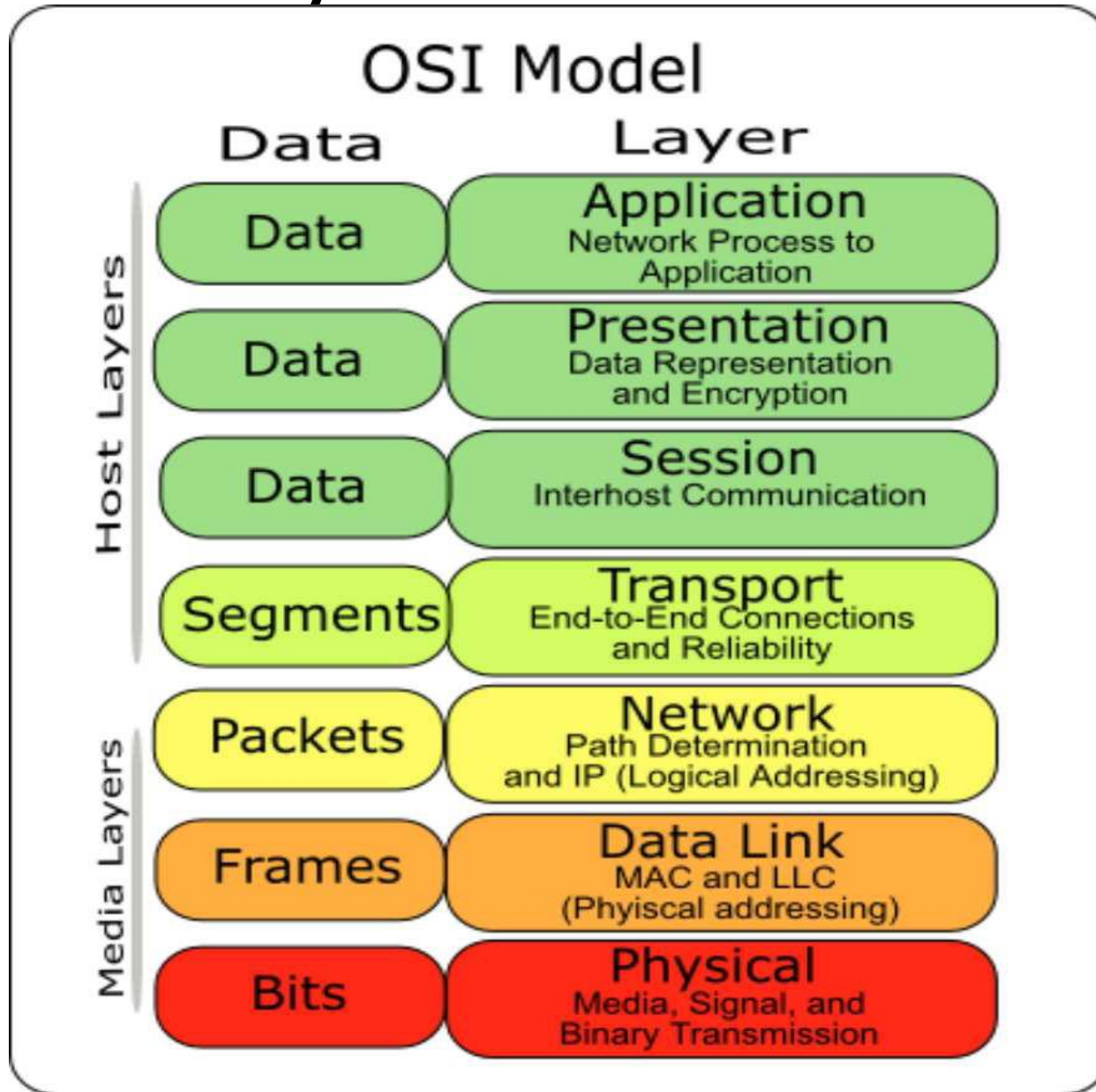
- Sub-networks are independent
- Computers can dynamically join and leave the network
- Built on open standards
- Lack of centralized control (mostly)
- Everyone can use it with simple, commonly available software

# People and organizations

- Internet Engineering Task Force (IETF): internet protocol standards
- Internet Corporation for Assigned Names and Numbers (ICANN): decides top-level domain names
- World Wide Web Consortium (W3C): web standards



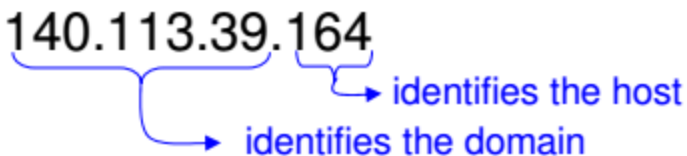
# Layered architecture



# Internet Addressing

- Each machine on a network must have a unique address: for the Internet, this is called the Internet Protocol (IP) address
  - For IPv4, an IP address is a 32-bit identifier for a machine
  - For IPv6, an IP address is 128-bits
- IP address is often written in dotted decimal notation:

– IPv4 example: 140.113.39.164



identifies the host  
identifies the domain

– IPv6 example: fe80::3153:525f:6964:8d84

or

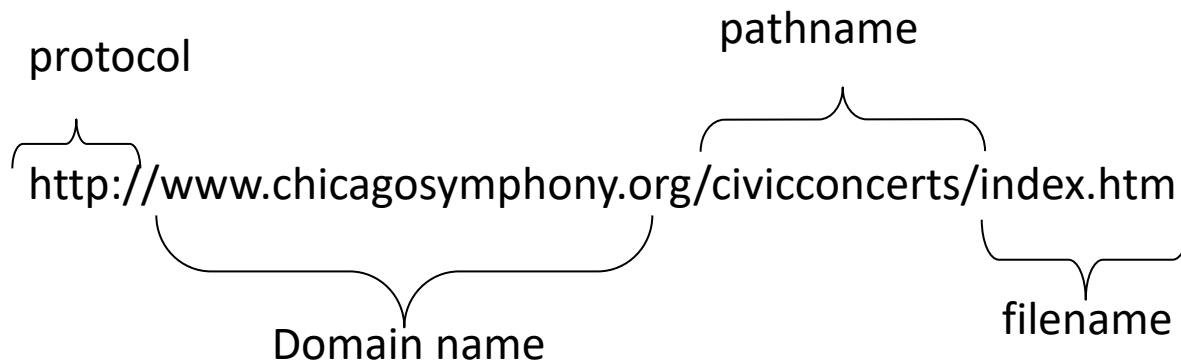
fe80:0000:0000:0000:3153:525f:6964:8d84

# Domain Names

- IP addresses are difficult for human to remember
- Each IP may have an equivalent mnemonic address, which is composed of a domain name and a host name (e.g. eclass.inha.uz)
- Domain name is the part assigned by a registrar
  - Top level domain (TLD) is the classification of domain owner (e.g. .com, .uz)
  - A domain name server (DNS) on the network translates the mnemonic addresses to binary IP addresses
- Host name is assigned by domain administrator
- Domain owner must run a DNS in order for other computers to find him/her
- Or else, he/she has to hire a DNS maintained by an ISP.

# Uniform Resource Locators

- The IP address and the domain name each identify a particular computer on the Internet.
- However, they do not indicate where a Web page's HTML document resides on that computer.
- To identify a Web pages exact location, Web browsers rely on Uniform Resource Locator (URL).
- URL is a four-part addressing scheme that tells the Web browser:
  - What transfer protocol to use for transporting the file
  - The domain name of the computer on which the file resides
  - The pathname of the folder or directory on the computer on which the file resides
  - The name of the file



# What is Web?

- The **Web (World Wide Web)** consists of information organized into Web pages containing text and graphic images.
- It contains hypertext links containing the URLs of other web resources.
- A collection of linked Web pages that has a common theme or domain name is called a **Web site**.
- The main page that all of the pages on a particular Web site are organized around and link back to is called the site's **home page (aka index page)**.



# Web Servers

- Web server: software that listens for web page requests
  - Apache 2
  - Microsoft Internet Information Server (IIS)
  - Nginx
  - Gunicorn
  - Mongrel
  - Jetty



# Application Server

- Software framework that provides an environment where applications can run
  - Apache
  - Glassfish
  - WebSphere
  - WebLogic

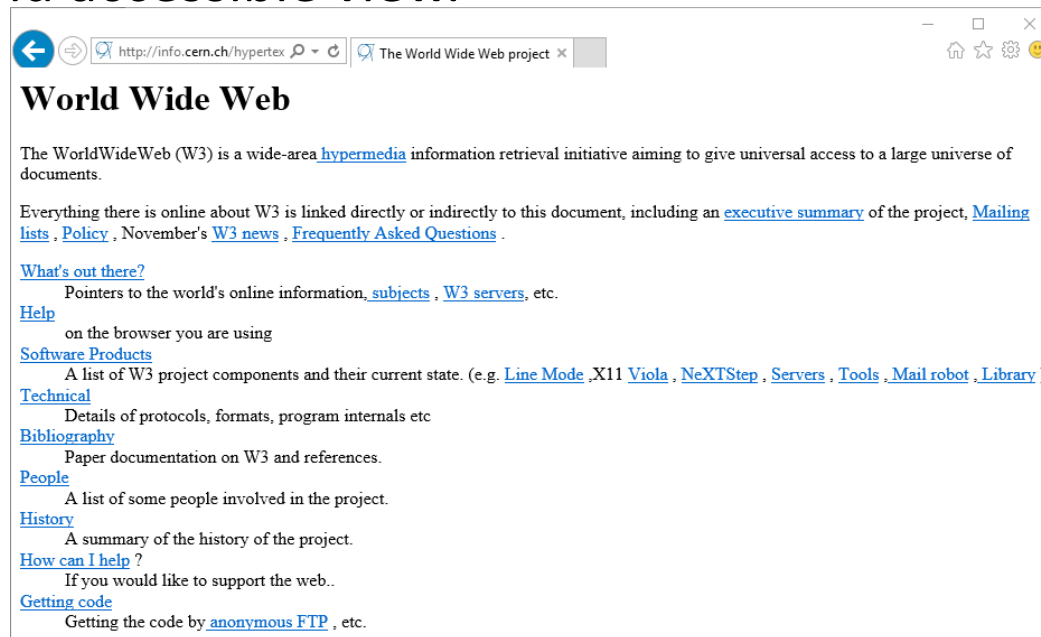


# Web Browser

- Web browser: fetches/displays documents from web servers
  - Mozilla Firefox
  - Microsoft Internet Explorer (IE)
  - Apple Safari
  - Google Chrome
  - Opera

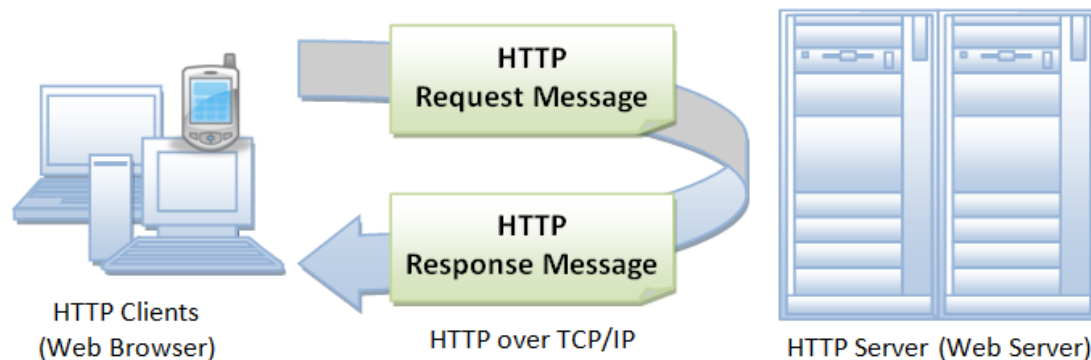
# First Web Site Ever

- Sir Timothy John Berners-Lee created his first web page on Aug 6, 1991 at CERN
- <http://info.cern.ch/hypertext/WWW/TheProject.html> is Internet's first web page.
- First web browsers like Mosaic (1993) and Netscape Navigator (1994) have been created to render such web pages into readable and accessible view.



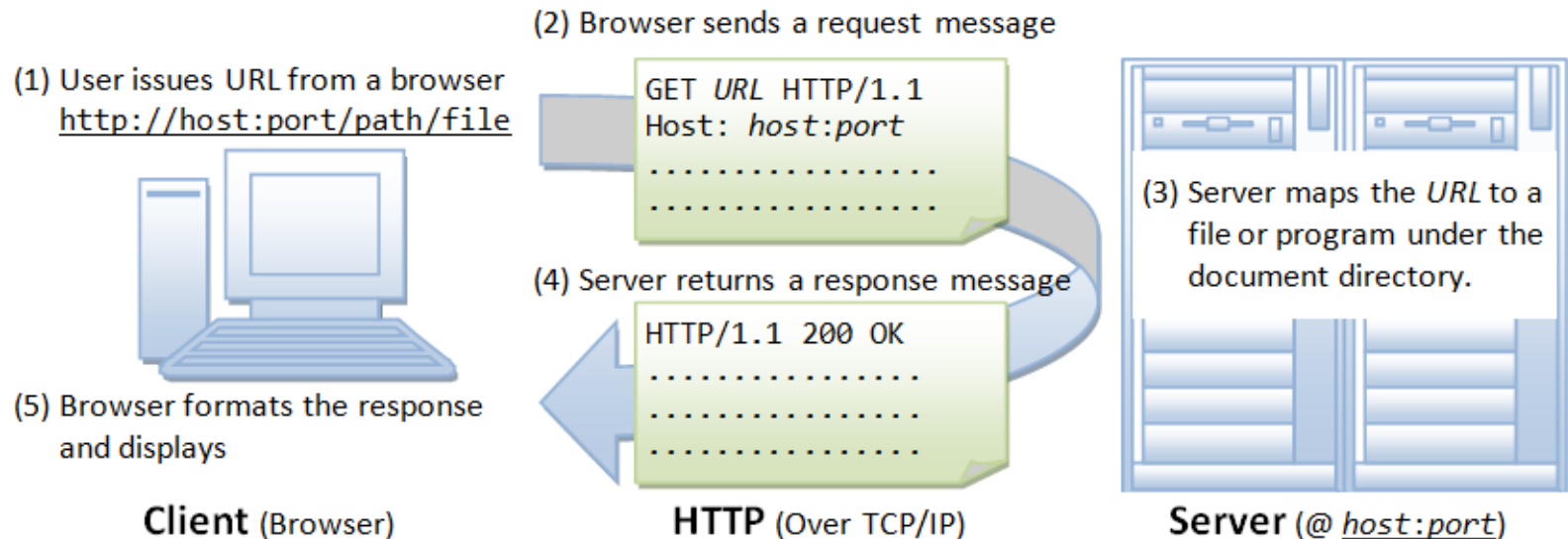
# HyperText Transfer Protocol

- HTTP is an application level protocol used for accessing web pages over the Internet
- HTTP is **connectionless** - client has to **pull** information from the server (not the other way around, where server could **push** information down to the client).
- HTTP is **stateless** – current request does not know what has been done in previous requests (thus does not maintain state)
- HTTP is **media independent** - it permits negotiating of data type and representation to be transmitted. Hence lots of freedom in transferring data with different formats.
- RFC-2616 specifications determine the standard for HTTP/1.1



# Client-Server Communication

- Whenever user types in a URL to get a web resource, browser turns URL into a *request message* and sends it to the HTTP server.
- HTTP server interprets the request message, and returns an appropriate *response message*, which is either the resource



# HTTP Protocol - Request

- For example, user wants to access:  
<http://www.nowhere123.com/doc/index.html>
- Following request message is sent:

```
GET /docs/index.html HTTP/1.1
Host: www.nowhere123.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
(blank line)
```

# HTTP Protocol - Response

- The request message reaches the server, the server can take either one of these actions:
  - Server reads the request, finds the corresponding file and returns the file to client
  - Server reads the request, finds the corresponding program, executes it, and returns the output to client
  - Server returns an error message

```
HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 08:56:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
ETag: "10000000565a5-2c-3e94b66c2e680"
Accept-Ranges: bytes
Content-Length: 44
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

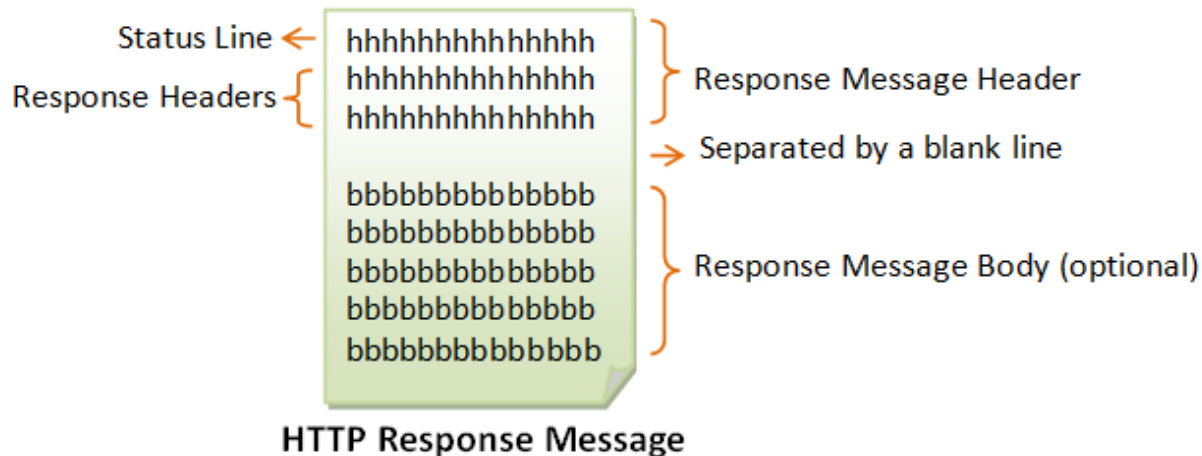
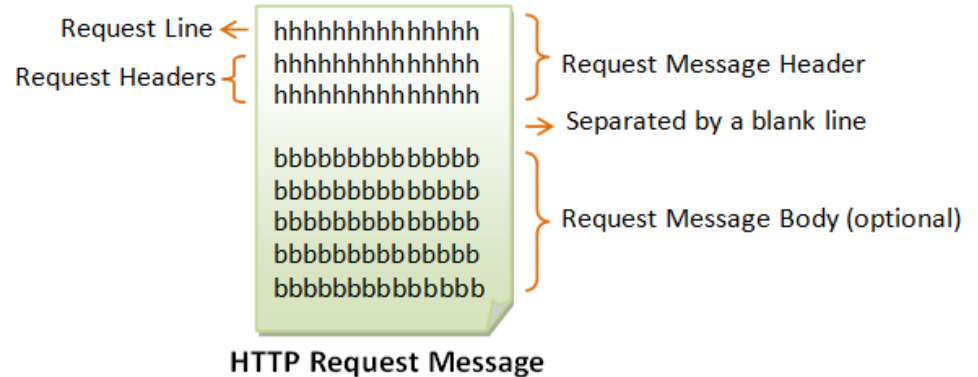
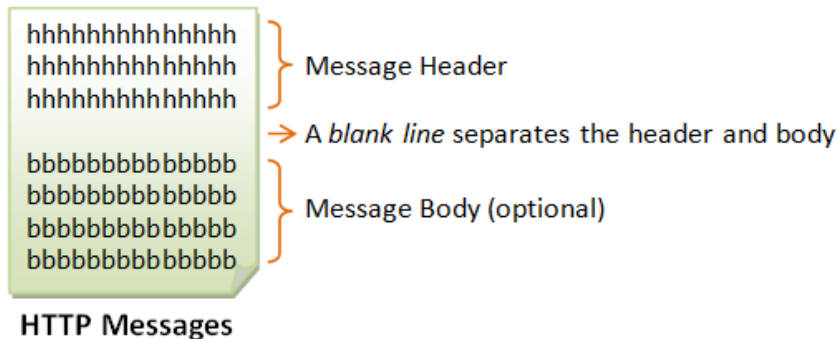
<html><body><h1>It works!</h1></body></html>
```



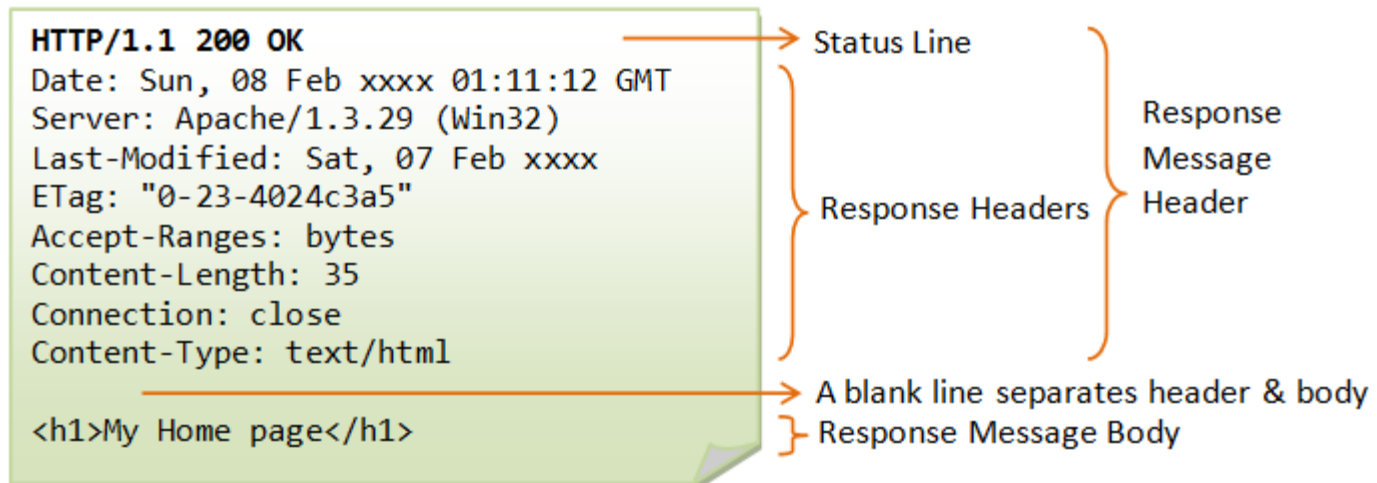
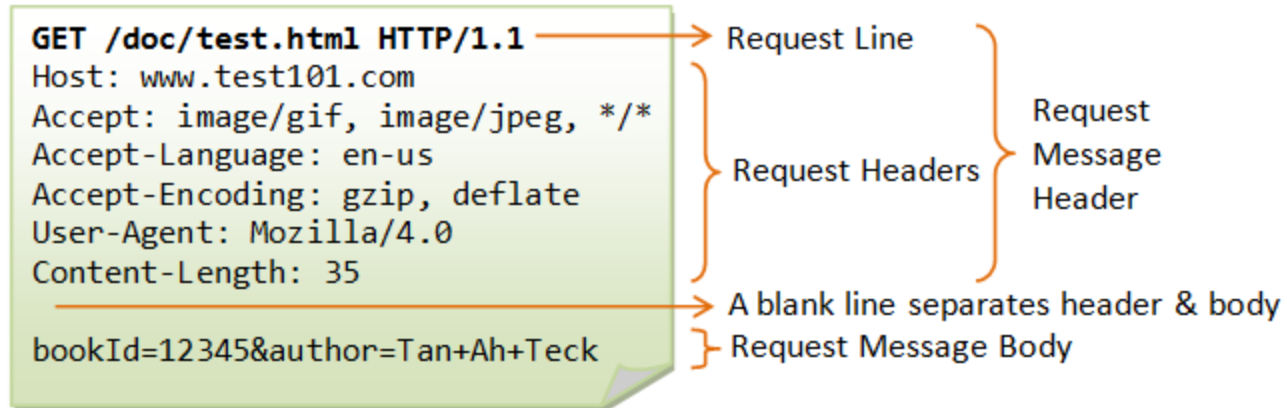
# HTTP Request/Response Headers

- While sending the request or returning the response message, some additional HTTP headers can be enclosed to the message.
- For example, *Content-Type* header identifies the format of the payload data being transmitted.
- There are multiple media types recognized by browsers:
  - text/plain
  - text/html
  - image/gif
  - image/jpeg
  - audio/mpeg
  - video/mpeg
  - application/msword
  - application/pdf

# HTTP Messages



# HTTP GET Request/Response



# HTTP Request Methods

- GET: A client can use the GET request to get a web resource from the 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.

# Testing HTTP Requests

- Telnet is a useful network utility for manually “talking” with an application using some protocol.

```
> telnet
telnet> help
... telnet help menu ...
telnet> open 127.0.0.1 8000
Connecting To 127.0.0.1...
GET /index.html HTTP/1.0
(Hit enter twice to send the terminating blank line ...)
... HTTP response message ...
```

- You can use standard libraries in Java or C# to establish Socket connection to server, and programmatically generate requests according to HTTP protocol.

# HTTP Response Statuses

The status code is a 3-digit number:

- **1xx (Informational):** Request received, server is continuing the process.
  - For ex, 100 Continue - server received request and in the process of generating response
- **2xx (Success):** The request was successfully received, understood, accepted and serviced.
  - For ex, 200 OK – request is fulfilled
- **3xx (Redirection):** Further action must be taken in order to complete the request.
  - For ex, 301 Move Permanently – the resource has been permanently moved to new location. The URL of new location is given in response header Location.
- **4xx (Client Error):** The request contains bad syntax or cannot be understood.
  - For ex, 404 Not Found – the resource requested cannot be found on server
- **5xx (Server Error):** The server failed to fulfill an apparently valid request.
  - For ex, 500 Internal Server Error – there is an error in server-side program responding the request
- Check [https://www.tutorialspoint.com/http/http\\_status\\_codes.htm](https://www.tutorialspoint.com/http/http_status_codes.htm) for more statuses.

# Request Headers

- **Host:** *domain-name* - HTTP/1.1 supports virtual hosts.
- **Accept:** *mime-type-1, mime-type-2, ...* - to tell the server the MIME types it can handle and it prefers.
- **Accept-Language:** *language-1, language-2, ...* - to tell the server what languages it can handle or it prefers.
- **Accept-Charset:** *Charset-1, Charset-2, ...* - to tell the server which character sets it can handle or it prefers. Examples of character sets are ISO-8859-1, ISO-8859-2, ISO-8859-5, BIG5, UCS2, UCS4, UTF8.
- **Accept-Encoding:** *encoding-method-1, encoding-method-2, ...* - to tell the server the type of encoding it supports. The common encoding methods are "x-gzip (.gz, .tgz)" and "x-compress (.Z)".
- **Connection:** **Close | Keep-Alive** - to tell the server whether to close the connection after this request, or to keep the connection alive for another request. HTTP/1.1 uses persistent (keep-alive) connection by default. HTTP/1.0 closes the connection by default.
- **Referer:** *referrer-URL* - to indicate the referrer of this request.

# Request Headers

- **User-Agent: *browser-type*** - Identify the type of browser used to make the request.
- **Content-Length: *number-of-bytes*** - Used by POST request, to inform the server the length of the request body.
- **Content-Type: *mime-type*** - Used by POST request, to inform the server the media type of the request body.
- **Cache-Control: *no-cache* | ...** - to specify how the pages are to be cached by proxy server.
- **Authorization:** Used by the client to supply its credential (username/password) to access protected resources.
- **Cookie: *cookie-name-1=cookie-value-1, cookie-name-2=cookie-value-2, ...*** - The client uses this header to return the cookie(s) back to the server, which was set by this server earlier for state management.
- **If-Modified-Since: *date*** - Tell the server to send the page only if it has been modified after the specific date.



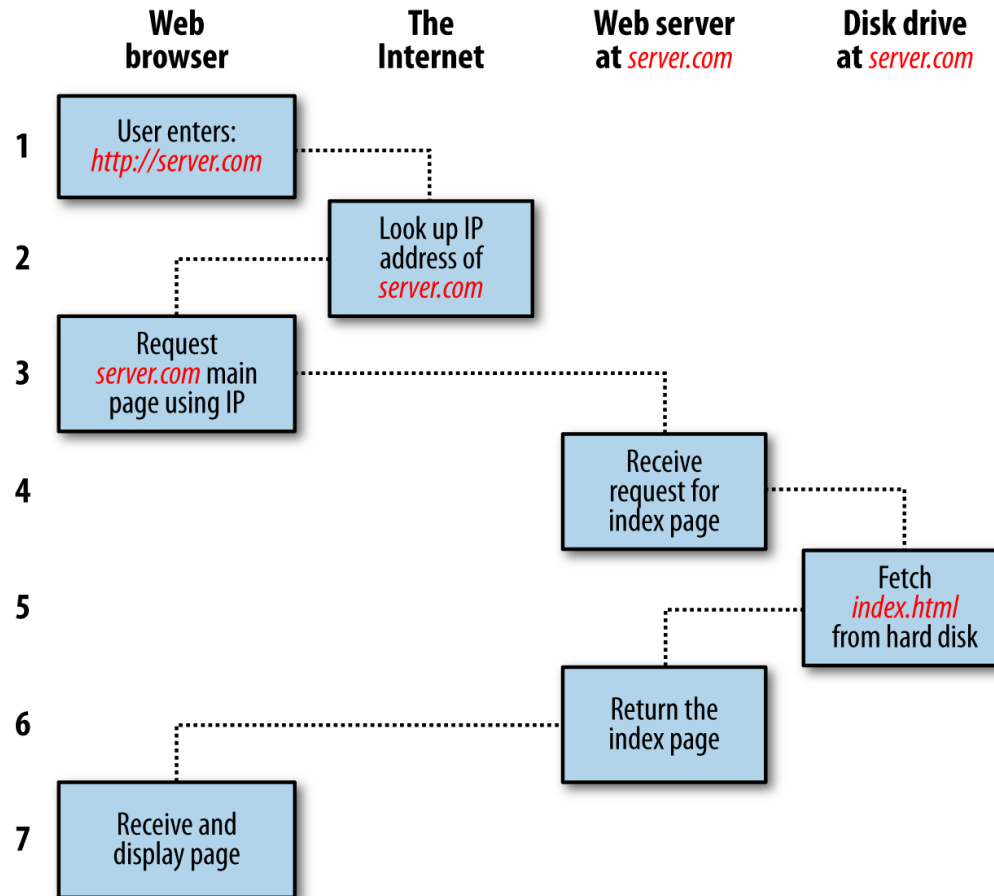
# Learn More about HTTP

- Very thorough tutorial with samples:
  - [https://www.ntu.edu.sg/home/ehchua/programming/web\\_programming/HTTP\\_Basics.html](https://www.ntu.edu.sg/home/ehchua/programming/web_programming/HTTP_Basics.html)
- Good for quick reference and recap:
  - <https://www.tutorialspoint.com/http/index.htm>

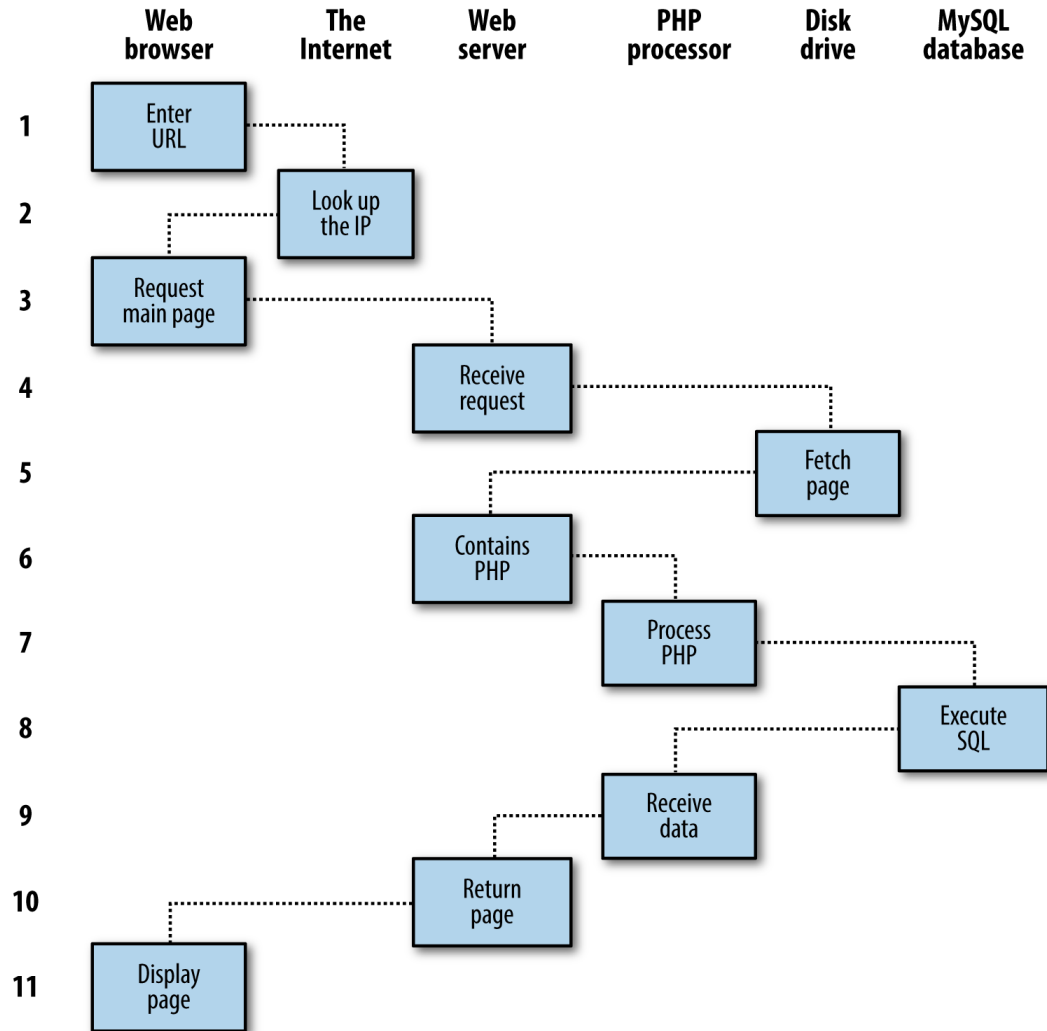
# Web Programming: Static VS Dynamic

- HTML documents are usually static
- The contents can only be changed manually
- There are needs for dynamic documents
  - **Search results**
  - **Database access**
  - **Context sensitive reply**
- Static
  - **page appears exactly as it was encoded, nothing changes**
- Dynamic
  - **page is compiled, or able to be changed**

# Static Pages



# Dynamic Pages



# Dynamic Web pages

- Applications executed by the server at run-time to process client input or generate document in response to client request
- Generating dynamic Web pages requires programming

# Scripts: Server-Side VS Client-Side

- Server-side
  - the first type possible on the Web
  - action occurs at the server
- Client-side
  - generally easier to implement
  - may be prepared and implemented offline
  - action occurs on the client side (browser)

# Client-Side Scripting

- Client side scripts are embedded inside HTML document. They are interpreted by browser.
- When Web browser encounters a script, it calls a scripting interpreter, which parses and deciphers the scripting code.
- Provide response to questions and queries without interventions from the server
  - Validate user data
  - Calculate expressions
  - Link to other applications

# Client-Side Scripting

- Client side advantages
  - Faster response time
  - Better animation
  - Simpler server programs
- Client side disadvantages
  - Longer load time
  - Browser compatibility
  - Complexity in web page design



# JavaScript

- JavaScript (most common)

- a scripting language for Web pages, developed by Netscape in 1995
- JavaScript code is embedded directly in HTML (interpreted by browser)
- good for adding dynamic features to Web page, controlling forms and GUI

- **Advantage**

- Easy to learn and use
- Wide browser support
- Protection of local resources

- **Disadvantage**

- Browser compatibility issues
- Not object oriented
- Unable to gain access to local resources

# Java Applet

- can be server-side or client-side
  - can define small, special-purpose programs in Java called applets
  - provides full expressive power of Java (but more overhead)
  - applets are included in Web pages using special HTML tags
  - interpreted by the Java Virtual Machine embedded in the browser
  - good for more complex tasks or data heavy tasks, such as graphics
- 
- |   |   |
|---|---|
| <ul style="list-style-type: none"><li>• <b>Advantage</b><ul style="list-style-type: none"><li>– High functionality</li><li>– Object oriented and full graphics functionality</li><li>– Protection of local resources</li><li>– Wide Browser support</li><li>– With Java2, be able to gain access to local resources with signed applets</li></ul></li></ul> | <ul style="list-style-type: none"><li>• <b>Disadvantage</b><ul style="list-style-type: none"><li>– JVM compatibility issues</li><li>– Difficulty to install and configure for local access</li><li>– Loading time and performance may be poor for large application.</li><li>– Some browsers forbid</li></ul></li></ul> |
|---|---|

# Server-Side Scripting

## Advantages

- Allows creation of dynamic web pages
- Modifies HTML code on the server before sent to client
- Uses databases such as Access and Oracle
- Responds to user input

## Disadvantages

- More complicated then HTML (with debugging)
- Slower to load on the server
- Harder to learn
- Web server must be enabled

# Examples of Server-Side Scripts

- **CGI (Common Gateway Interface)**
  - A standard for interfacing external applications with information servers, such as HTTP or Web servers
  - CGI program is any program designed to accept and return data that conforms to the CGI specification
  - CGI program can be written in any language that allows it to be executed on the system, such as: C/C++, Fortran, PERL, TCL, Any Unix shell, Visual Basic, AppleScript etc.
- **PHP (Hypertext Preprocessor )**
  - Widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

# Examples of Server-Side Scripts

- **Active Server Pages** (Microsoft)
  - ASP is a Microsoft Technology and run on IIS (Internet Information Server) & PWS (Personal Web Server)
- **Java Server Pages** (Sun)
  - Sun's solution for developing dynamic web sites
  - JSP enable the developers to directly insert java code into jsp file, this makes the development process very simple and its maintenance also becomes very easy