Module 1

Web programming fundamentals

Working of web browser

- A browser is a software application used to locate, retrieve and display content on the World Wide Web, including Web pages, images, video and other files.
- As a client/server model, the browser is the client run on a computer that contacts the Web server and requests information.
- The Web server sends the information back to the Web browser which displays the results on the computer or other Internet-enabled device that supports a browser.

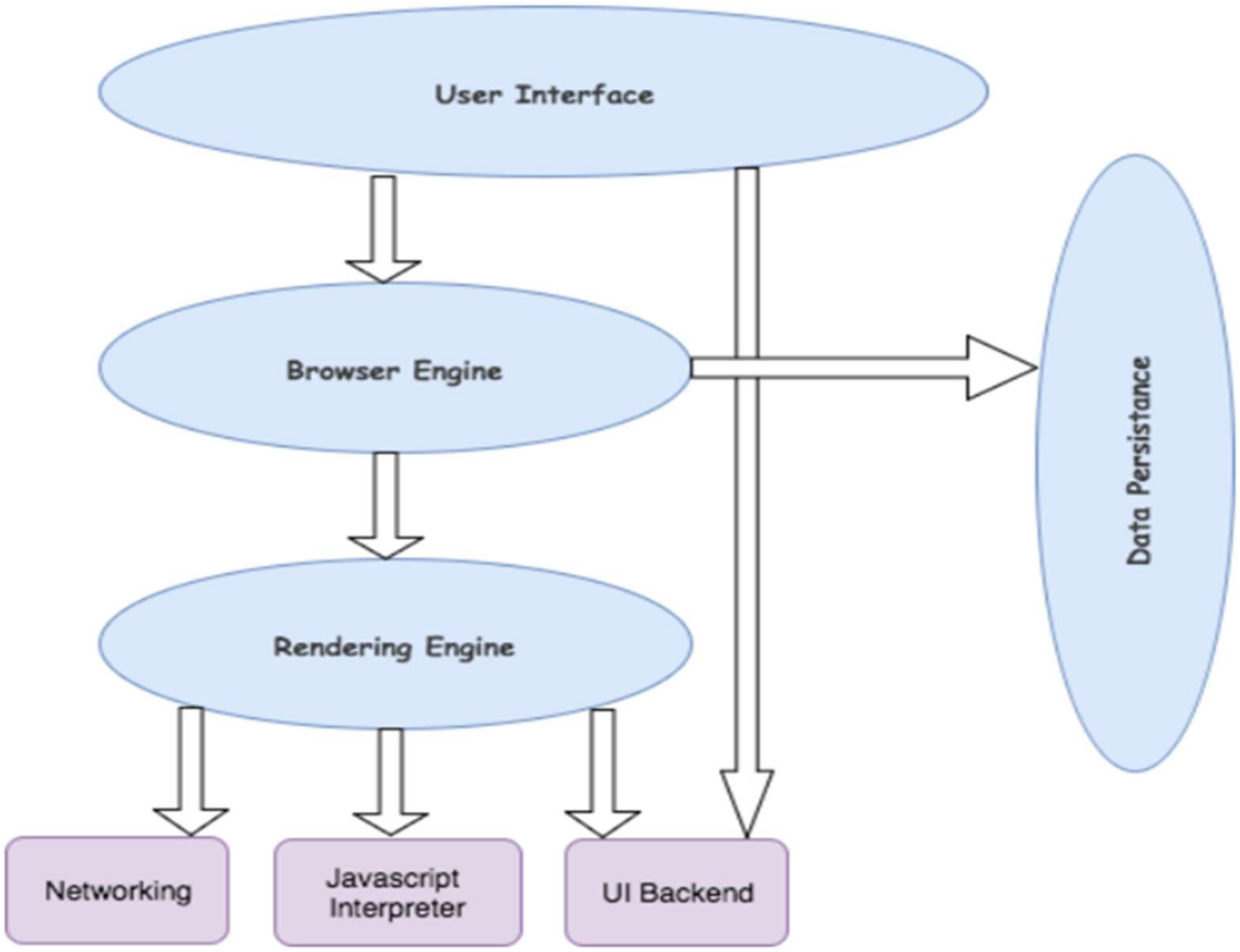
Working of web browser

- Today's browsers are fully-functional software suites that can interpret and display HTML Web pages, applications, JavaScript, AJAX and other content hosted on Web servers.
- Many browsers offer plug-ins which extend the capabilities of the software so it can display multimedia information (including sound and video), or the browser can be used to perform tasks such as videoconferencing, to design web pages or add anti-phishing filters and other security features to the browser.

Working of web browser

- A browser is a group of structured codes which together performs a series of tasks to display a web page on the screen. According to the tasks they perform, these codes are made as different components.

High-level architecture of browser



Main components of the browser

High-level architecture of browser

- **The Rendering Engine**:

  The rendering engine, as the name suggests is responsible for rendering the requested web page on the browser screen. The rendering engine interprets the HTML, XML documents and images that are formatted using CSS and generates the layout that is displayed in the User Interface. However, using plugins or extensions, it can display other types data also. Different browsers user different rendering engines:

  * Internet Explorer: Trident
  * Firefox & other Mozilla browsers: Gecko
  * Chrome & Opera 15+: Blink
  * Chrome (iPhone) & Safari: Webkit

High-level architecture of browser

- **The User Interface**: The user interface is the space where User interacts with the browser. It includes the address bar, back and next buttons, home button, refresh and stop, bookmark option, etc. Every other part, except the window where requested web page is displayed, comes under it.
- **The Browser Engine**: The browser engine works as a bridge between the User interface and the rendering engine. According to the inputs from various user interfaces, it queries and manipulates the rendering engine.

High-level architecture of browser

- **Networking**: Component of the browser which retrieves the URLs using the common internet protocols of HTTP or FTP. The networking component handles all aspects of Internet communication and security. The network component may implement a cache of retrieved documents in order to reduce network traffic.

High-level architecture of browser

- **JavaScript Interpreter:** It is the component of the browser which interprets and executes the javascript code embedded in a website. The interpreted results are sent to the rendering engine for display. If the script is external then first the resource is fetched from the network. Parser keeps on hold until the script is executed.

High-level architecture of browser

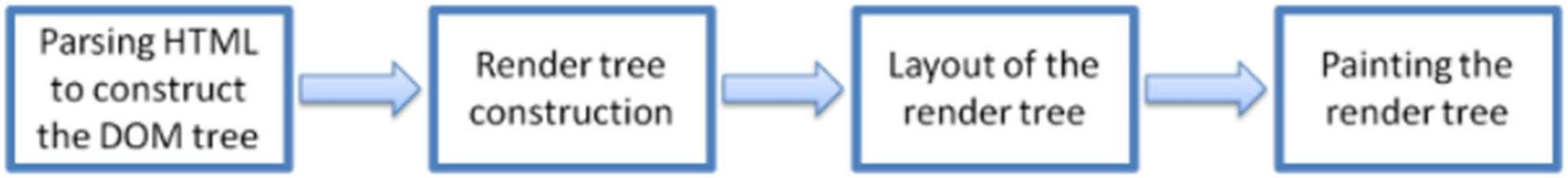
- **UI Backend**: UI backend is used for drawing basic widgets like combo boxes and windows. This backend exposes a generic interface that is not platform specific. It underneath uses operating system user interface methods.
- **Data Persistence/Storage:** This is a persistence layer. Browsers support storage mechanisms such as local Storage, IndexedDB, WebSQL and FileSystem. It is a small database created on the local drive of the computer where the browser is installed. It manages user data such as cache, cookies, bookmarks and preferences.

High-level architecture of browser

- ***An important thing to note here is that in web browsers such as Google Chrome each tab runs in a separate process(multiple instances of rendering engine).***

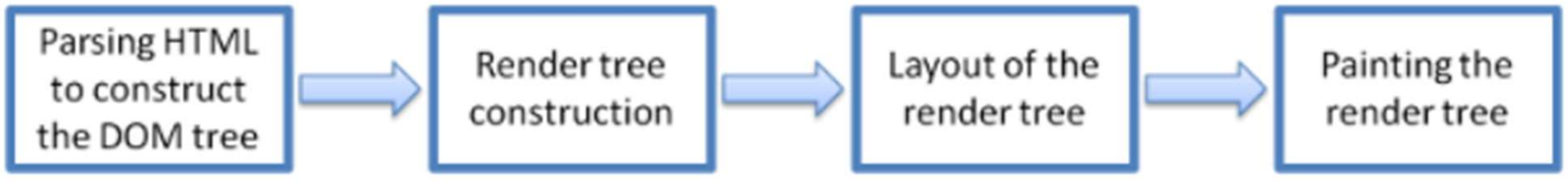
High-level architecture of browser

- **Rendering engine**
- The networking layer will start sending the contents of the requested documents to the rendering engine in chunks of 8KBs.


Parsing HTML to construct the DOM tree
Render tree construction
Layout of the render tree
Painting the render tree

Rendering engine basic flow

High-level architecture of browser


Parsing HTML to construct the DOM tree
Render tree construction
Layout of the render tree
Painting the render tree

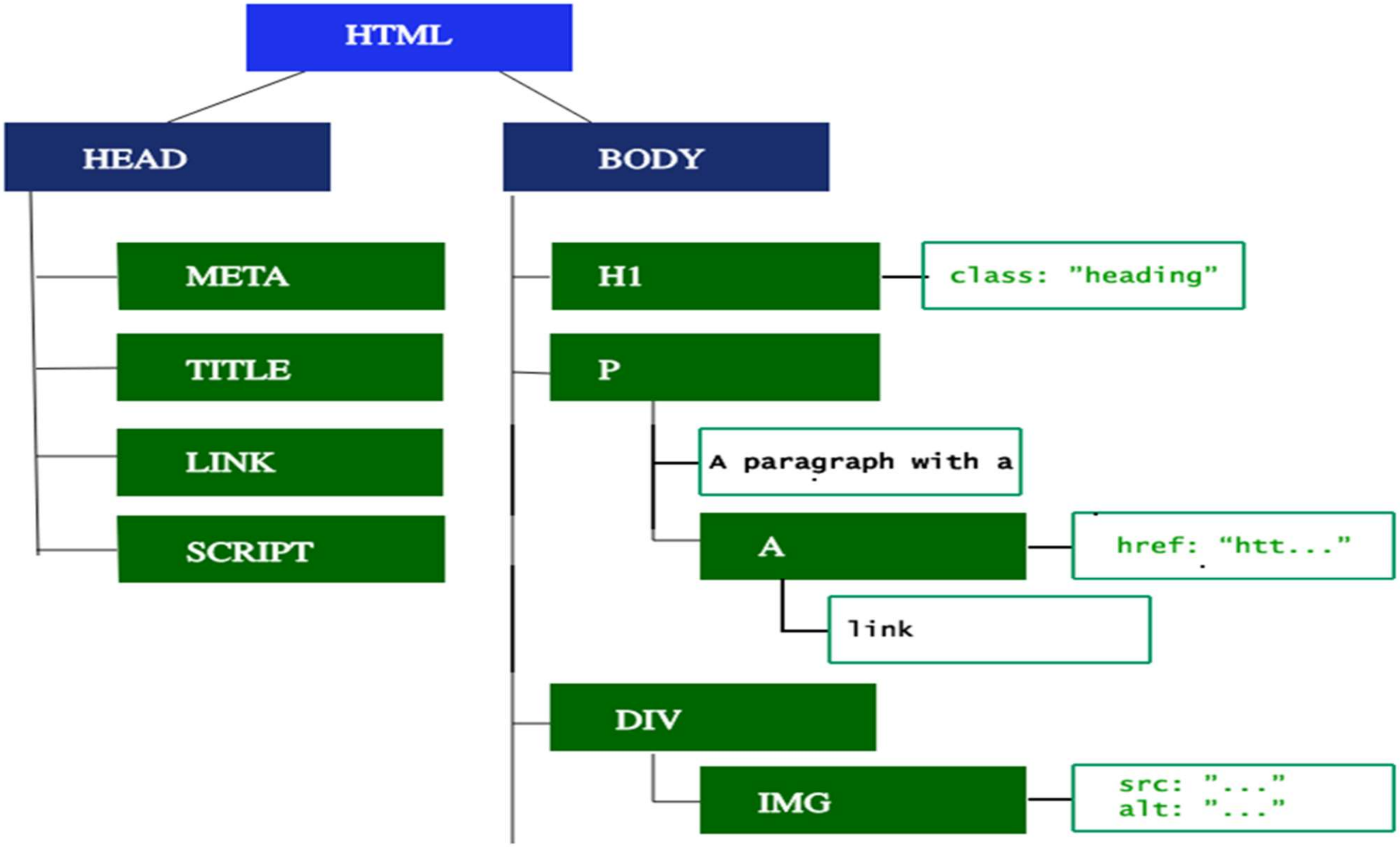
Rendering engine basic flow

- The rendering engine parses the chunks of HTML document and convert the elements to DOM nodes in a tree called the "**content tree**" or the "**DOM tree**".
- It also parses both the external CSS files as well in style elements.

DOM tree

- The DOM tree describes the content of the document.
- The **<html>** element is the first tag and root node of the document tree. The tree reflects the relationships and hierarchies between different tags. Tags nested within other tags are child nodes. The greater the number of DOM nodes, the longer it takes to construct the DOM tree.

DOM Tree


HTML
HEAD
BODY
META
TITLE
LINK
SCRIPT
H1
class: "heading"
P
A paragraph with a
A
href: "htt..."
link
DIV
IMG
src: "..."
alt: "..."

High-level architecture of browser

- While the DOM tree is being constructed, the browser constructs another tree, the **render tree**.
- This tree is of visual elements in the order in which they will be displayed.
- It is the visual representation of the document. The purpose of this tree is to enable painting the contents in their correct order.
- Firefox calls the elements in the render tree "frames".
- WebKit uses the term renderer or render object.

High-level architecture of browser

- After the construction of the render tree, it goes through a "**layout process**" of the render tree.
- When the renderer is created and added to the tree, it does not have a position and size.
- **The process of calculating these values is called layout or reflow.** This means giving each node the exact coordinates where it should appear on the screen.
- The position of the root renderer is 0,0 and its dimensions are the viewport–the visible part of the browser window. All renderers have a "layout" or "reflow" method, each renderer invokes the layout method of its children that need layout.

High-level architecture of browser

- The next stage is **painting**. In the painting stage, the render tree is traversed and the renderer's "paint()" method is called to display content on the screen. Painting uses the UI backend layer.
- The rendering engine always tries to display the contents on the screen as soon as possible for better user experience. It does not wait for the HTML parsing to complete before starting to build and layout the render tree. It parses and displays the content it has received from the network, while rest of the contents stills keeps coming from the network.

Web Technologies

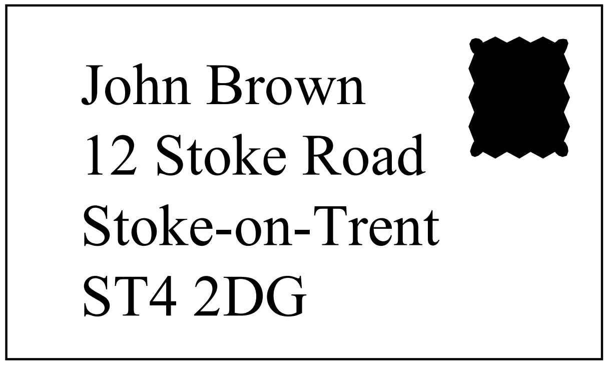
URL

URL Defined

- A Uniform Resource Locator (URL) is a standard way developed to specify the location of a resource available electronically.

What are URLs?

- URLs make it possible to direct both people and software applications to a variety of information, available from a number of different Internet protocols.
- A URL is like your complete **mailing address**: it specifies all the information necessary for someone to address an envelope to you.


John Brown
12 Stoke Road
Stoke-on-Trent
ST4 2DG

What are URLs? (contd.)

- In short, a URL is a very convenient and brief way to direct people to a file or other electronic resource.
- Learning how to interpret, use and construct URLs will assist your exploration of the Internet.

URL Types

- There are two forms of URL as listed below:
- **Absolute URL**
- **Relative URL**

- **ABSOLUTE URL:**
- Absolute URL is a complete address of a resource on the web. **This completed address comprises of protocol used, server name, path name and file name.**
- **http:// www.tutorialspoint.com / internet_technology /index.htm.**

RELATIVE URL

- Relative URL is a partial address of a webpage. Unlike absolute URL, the **protocol and server part are omitted from relative URL.**

- Relative URLs are used for internal links i.e. to create links to file that are part of same website as the WebPages on which you are placing the link.

General URL syntax

http://www.house.gov/house/House_Calender.html

Protocol/Scheme examples:

- http
- ftp
- news
- gopher

Web Technologies

Scheme

< Protocol/Scheme >:<scheme-dependent-information>

- Tells you what type of resource we are trying to reach and/or what mechanism to use to obtain it.
- Examples:
  - **http** (Hyper Text Transfer Protocol)
  - **ftp** (File Transfer Protocol)
  - **news** (News protocol)

Scheme Dependent Information

<scheme>:<scheme-dependent-information>

- This information is detailed with each scheme
- Most schemes include the:
  - **Machine** making the file available
  - **"Path"** to that file
- Example (for HTTP):

$$\text{Scheme}\ \Big\}\ \textbf{http://}\underbrace{\textbf{www.7sport.net}}_{\text{Machine}}\textbf{/}\underbrace{\textbf{7sport/index.htm}}_{\text{Path}}$$

URL Example (explained)

Scheme $\Big\}$ $\textbf{http://}\underbrace{\textbf{www.7sport.net}}_{\text{Machine}}\textbf{/}\underbrace{\textbf{7sport/index.htm}}_{\text{Path}}$

- **http** is the **scheme**
  hyper text transfer protocol
- **two slashes** (//) separate the scheme from the machine/domain name
- **www.7sport.net** is the **machine/domain name**
- **single slash** (/) separates the name from the path
- Finally **7sport/index.htm** is the **path**.

URL Example (explained)

$$\text{Scheme}\;\Big\}\;\textbf{http://}\underbrace{\textbf{www.7sport.net}}_{\text{Machine}}\textbf{/}\underbrace{\textbf{7sport/}}_{\text{Path}}$$

- **sometimes the path will end in a slash (/)**
- **this indicates that the URL is not pointing to a specific file**
- in this case the server returns the **"default" page**
  - homepage.html
  - home.html
  - welcome.html
  - default.html

HTTP URL Example (explained)

$$\underbrace{\textbf{http:}}_{\text{Scheme}}\textbf{//}\underbrace{\textbf{www.w3schools.com}}_{\text{Machine/Domain Name}}\textbf{/}\underbrace{\textbf{html/html\_forms.asp}}_{\text{Path}}$$

- **http** is the **scheme**
  hyper text transfer protocol
- **two slashes** (//) separate the scheme from the machine/domain name
- **www.w3schools.com** is the **machine/domain name**
- **single slash** (/) separates the name from the path
- Finally **html/html_forms.asp** is the **path**.

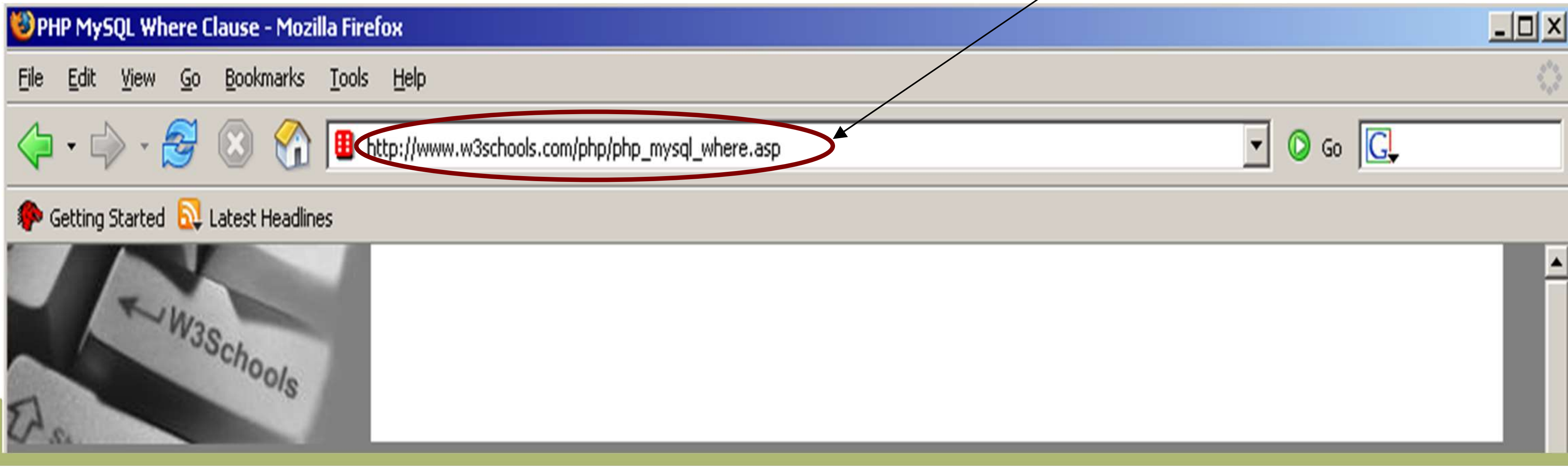
FTP URL Example (explained)

$$\text{Scheme}\ \}\ \textbf{ftp://}\underbrace{\textbf{garbo.uwasa.fi}}_{\text{Machine}}\textbf{/}\underbrace{\textbf{pc/doc-net/}}_{\text{Path}}$$

- **ftp** is the **scheme**
  file transfer protocol
- **two slashes** (//) separate the scheme from the machine/domain name
- **garbo.uwasa.fi** is the **machine/domain name**
- **single slash** (/) separates the name from the path
- Finally **pc/doc-net** is the **path**.

Using URLs

- You can double click on a URL (link) and if your system is configured properly the appropriate application will be launched to obtain the resource.
- You can also copy the URL and paste it into the application which you use to get to the resource.


PHP MySQL Where Clause - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.w3schools.com/php/php_mysql_where.asp
Go
Getting Started
Latest Headlines
W3Schools

Troubleshooting URLs

Reasons for not being able to access URLs:

– the remote machine refuses the connection

– the site is very busy (e.g. peak hours of use)

– you have misspelled the URL

– the file was moved

– if all else fails you can try looking up the hierarchy by sequentially removing the file name first, and then the last directory in the path.

Domain Name System

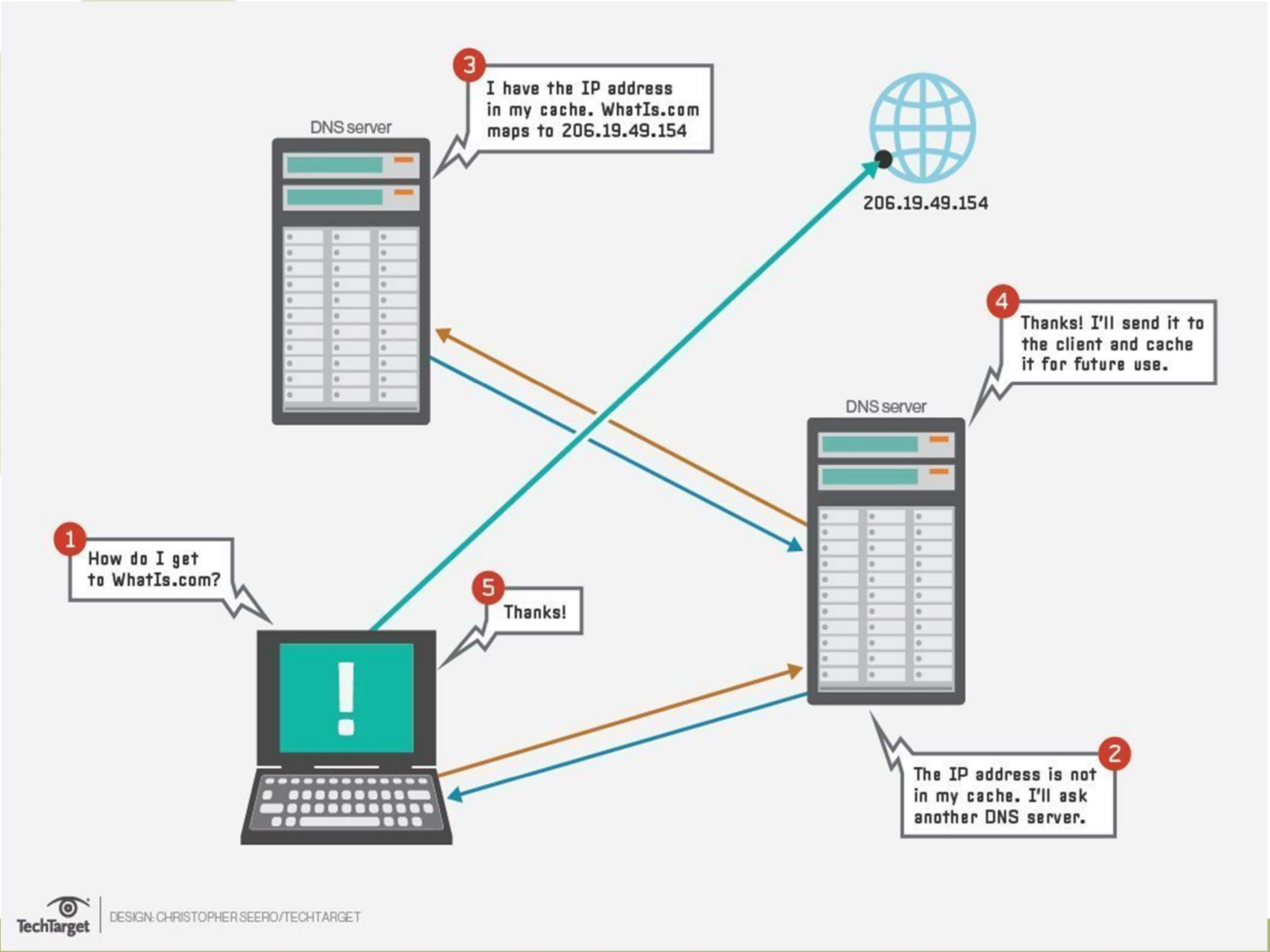
- **Domain Name System** helps to resolve the host name to an address. It uses a hierarchical naming scheme and distributed database of IP addresses and associated names

IP Address

- IP address is a unique logical address assigned to a machine over the network. An IP address exhibits the following properties:
- IP address is the unique address assigned to each host present on Internet.
- IP address is 32 bits (4 bytes) long.
- IP address consists of two components: network component and host component.

- Each of the 4 bytes is represented by a number from 0 to 255, separated with dots. For example 137.170.4.124

- IP address is 32-bit number while on the other hand domain names are easy to remember names. For example, when we enter an email address we always enter a symbolic string such as webmaster@tutorialspoint.com.



DNS server
DNS server
1 How do I get to WhatIs.com?
2 The IP address is not in my cache. I'll ask another DNS server.
3 I have the IP address in my cache. WhatIs.com maps to 206.19.49.154
4 Thanks! I'll send it to the client and cache it for future use.
5 Thanks!
206.19.49.154
TechTarget
DESIGN: CHRISTOPHER SEERO/TECHTARGET

Domain Name System Architecture

- The Domain name system comprises of **Domain Names, Domain Name Space, Name Server** that have been described below:
- **Domain Names**
- **Domain Name is a symbolic string associated with an IP address.** There are several domain names available; some of them are generic such as **com, edu, gov, net** etc, while some country level domain names such as **au, in, za, us**etc.

The following table shows the Generic Top-Level Domain names:

Domain Name	**Meaning**
Com	Commercial business
Edu	Education
Gov	U.S. government agency
Int	International entity
Mil	U.S. military
Net	Networking organization
Org	Non profit organization

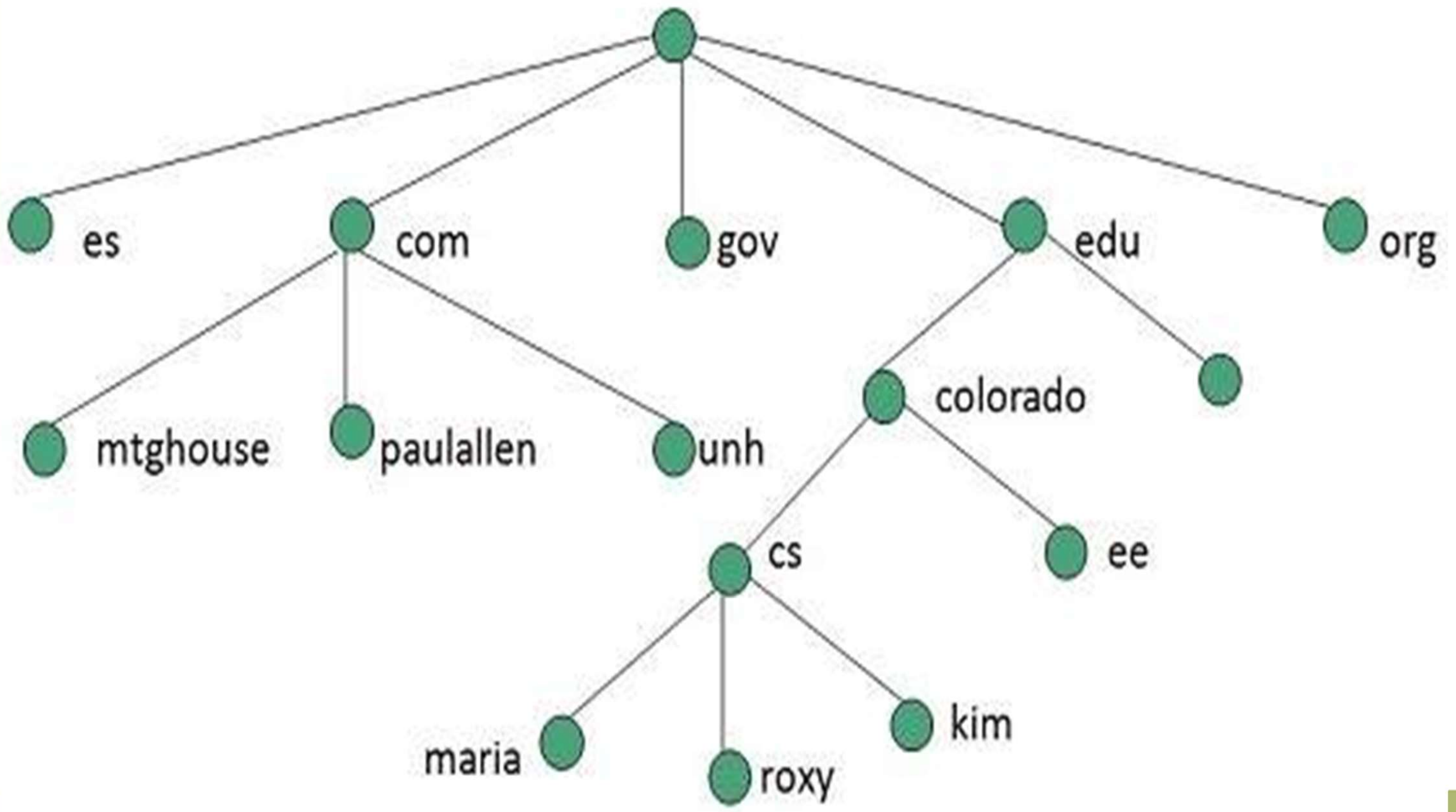
The following table shows the **Country top-level** domain names:

Domain Name	Meaning
au	Australia
in	India
cl	Chile
fr	France
us	United States
za	South Africa
uk	United Kingdom
jp	Japan
es	Spain
de	Germany
ca	Canada
ee	Estonia
hk	Hong Kong

Domain Name Space

- **The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127),** with a root at the top. The following diagram shows the domain name space hierarchy:

Domain name space hierarchy



Name Server

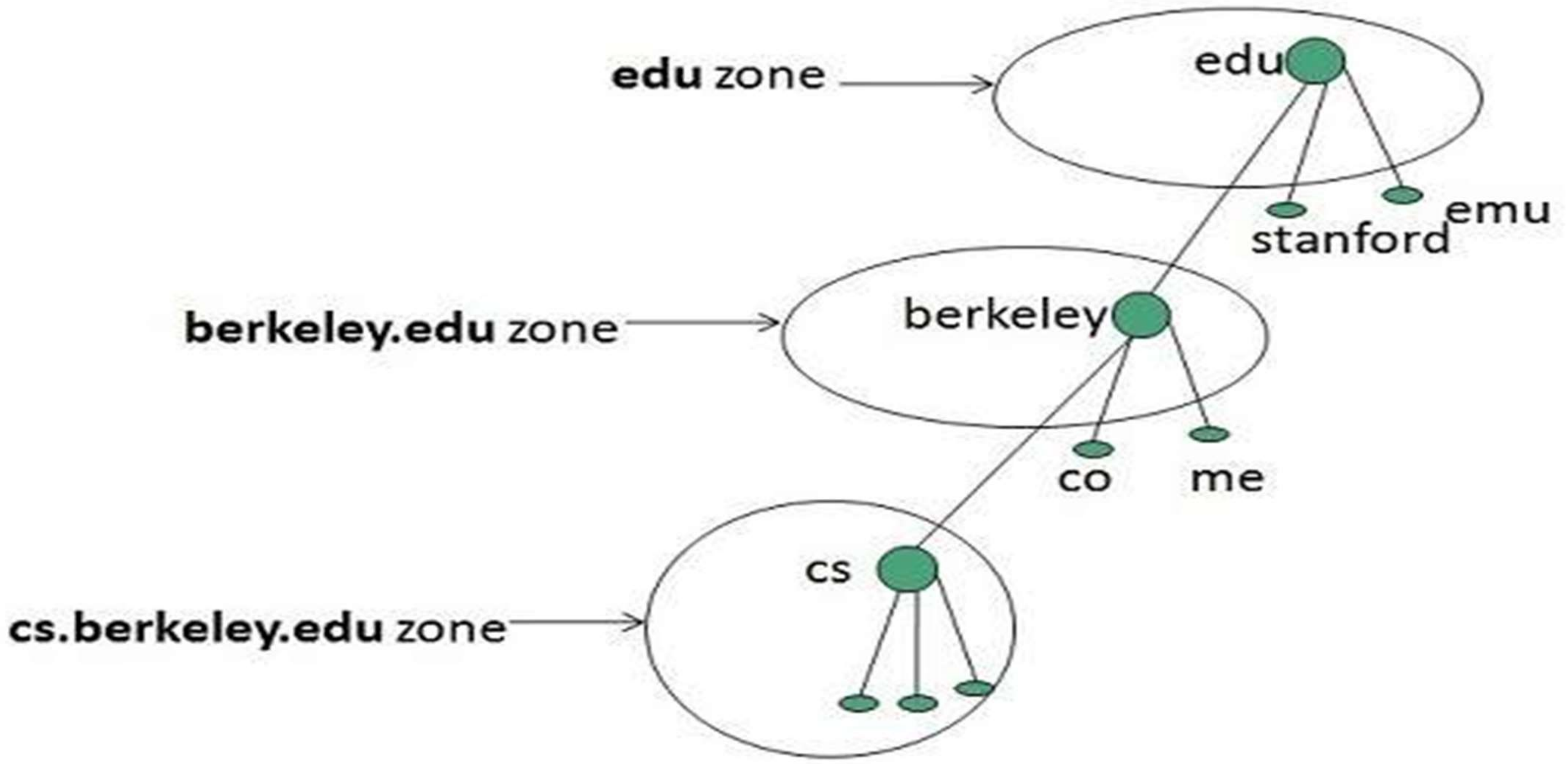
- **Name server contains the DNS database.** This database comprises of various names and their corresponding IP addresses. Since it is not possible for a single server to maintain entire DNS database, therefore, the information is distributed among many DNS servers.
- Hierarchy of server is same as hierarchy of names.
- The entire name space is divided into the zones

TYPES OF NAME SERVERS

- Following are the three categories of Name Servers that manages the entire Domain Name System:
- Root Server
- Primary Server
- Secondary Server

- **ROOT SERVER**
- Root Server is the top level server which consists of the entire DNS tree. It does not contain the information about domains but delegates the authority to the other server
- **PRIMARY SERVERS**
- Primary Server stores a file about its zone. It has authority to create, maintain, and update the zone file.

Zone is collection of nodes (sub domains) under the main domain. The server maintains a database called zone file for every zone.



- **SECONDARY SERVER**
- Secondary Server transfers complete information about a zone from another server which may be primary or secondary server. The secondary server does not have authority to create or update a zone file.

Hyper Text Transfer Protocol (HTTP)

http://<host>:<port>/<path>?<searchpart>

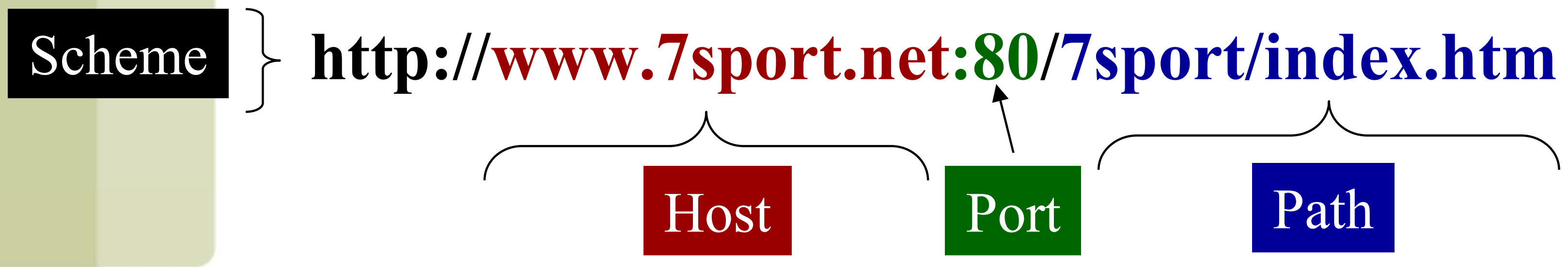
– the **host** is the Internet address of the WWW server

– the **port** is the port number to connect to (generally omitted along with the colon **:**) (defaults to standard "80")

– **path** tells the server which file you want (if file name is omitted you want the "home page")

– the **searchpart** may be used to pass information to the server (often to a CGI script) (generally omitted, along with the question mark **?**)

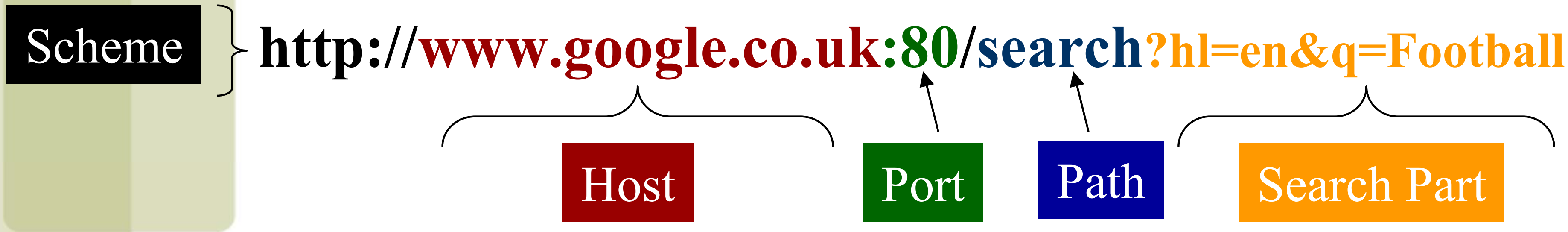
HTTP URL Detailed Example



http://www.7sport.net:80/7sport/index.htm

- **http** is the **scheme**
- **www.7sport.net** is the **host name**
- **:80** is the **port** (it can be omitted)
- Finally **7sport/index.htm** is the **path**

Another HTTP Example



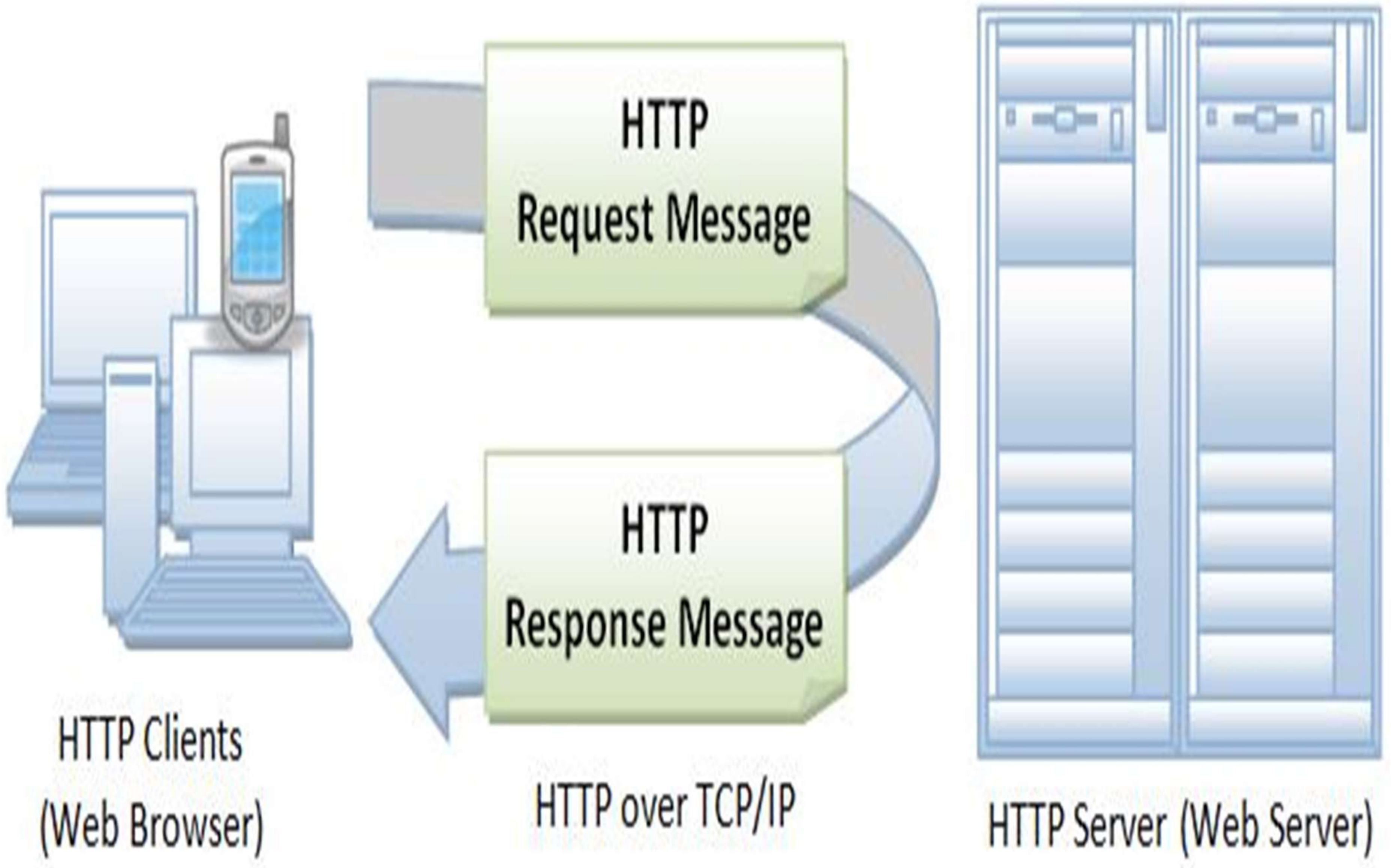
http://www.google.co.uk:80/search?hl=en&q=Football

- **http** is the **scheme**
- **www.google.co.uk** is the **host name**
- **:80** is the **port** (it can be omitted)
- **search** is the **path**
- **?hl=en&q=Football** is the search part
  - **hl=en** (search language is English)
  - **q=Football** (search keyword is Football)

HyperText Transfer Protocol (HTTP)

- HTTP is an asymmetric request-response client-server protocol as illustrated.
- An HTTP client sends a request message to an HTTP server.
- The server, in turn, returns a response message.
- In other words, HTTP is a pull protocol, the client pulls information from the server (instead of server pushes information down to the client).

Working of HTTP



HTTP

- HTTP is a stateless protocol. In other words, the **current request does not know what has been done in the previous requests.**
- HTTP permits negotiating of data type and representation, so as to allow systems to be built independently of the data being transferred.

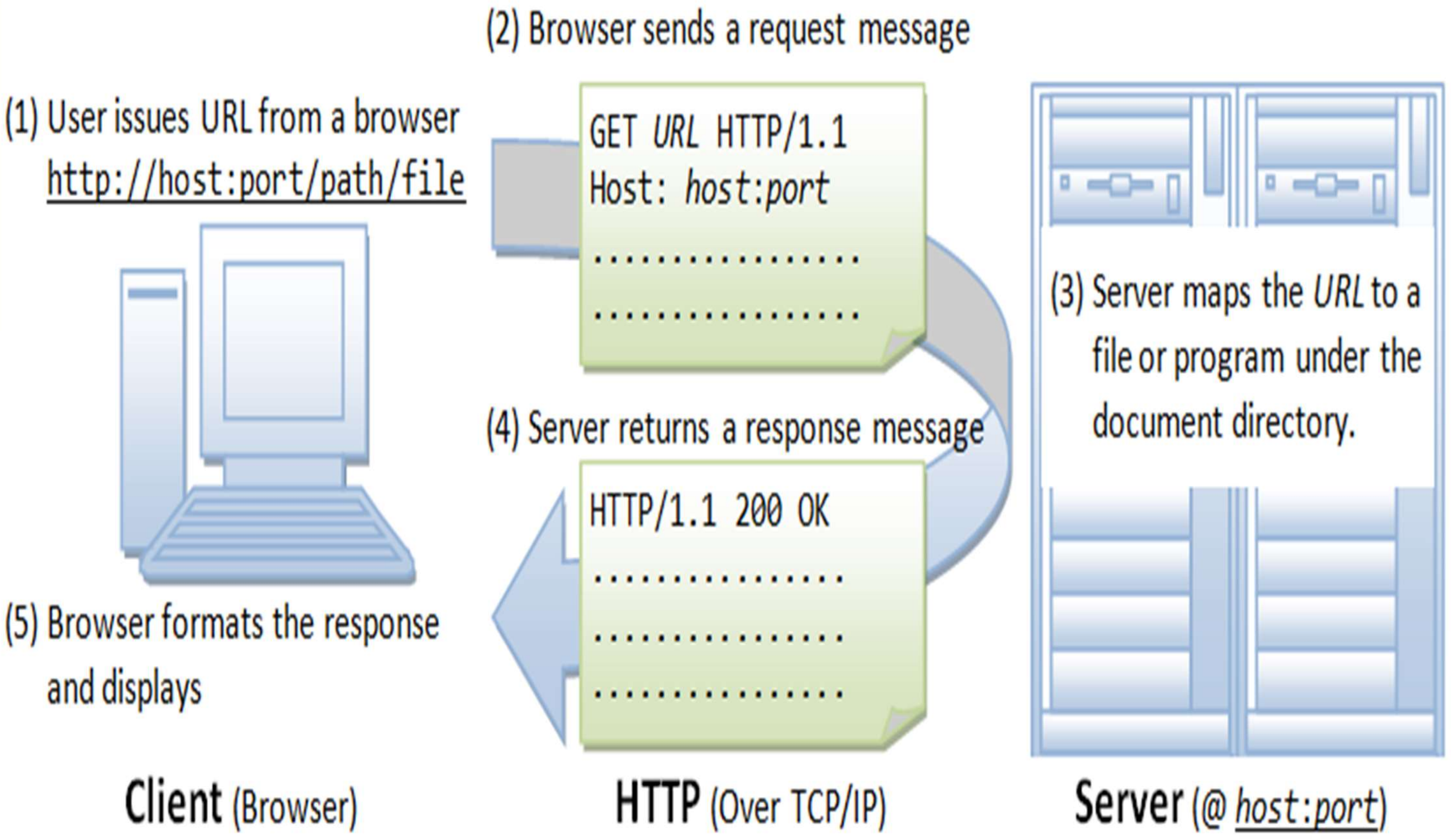
- Whenever you issue a URL from your browser to get a web resource using HTTP, **e.g. http://www.nowhere123.com/index.html**, the **browser turns the URL into a request message and sends it to the HTTP server.**

  The HTTP server interprets the request message, and returns you an appropriate response message, which is either the resource you requested or an error message. This process is illustrated below:

HTTP Request Processing

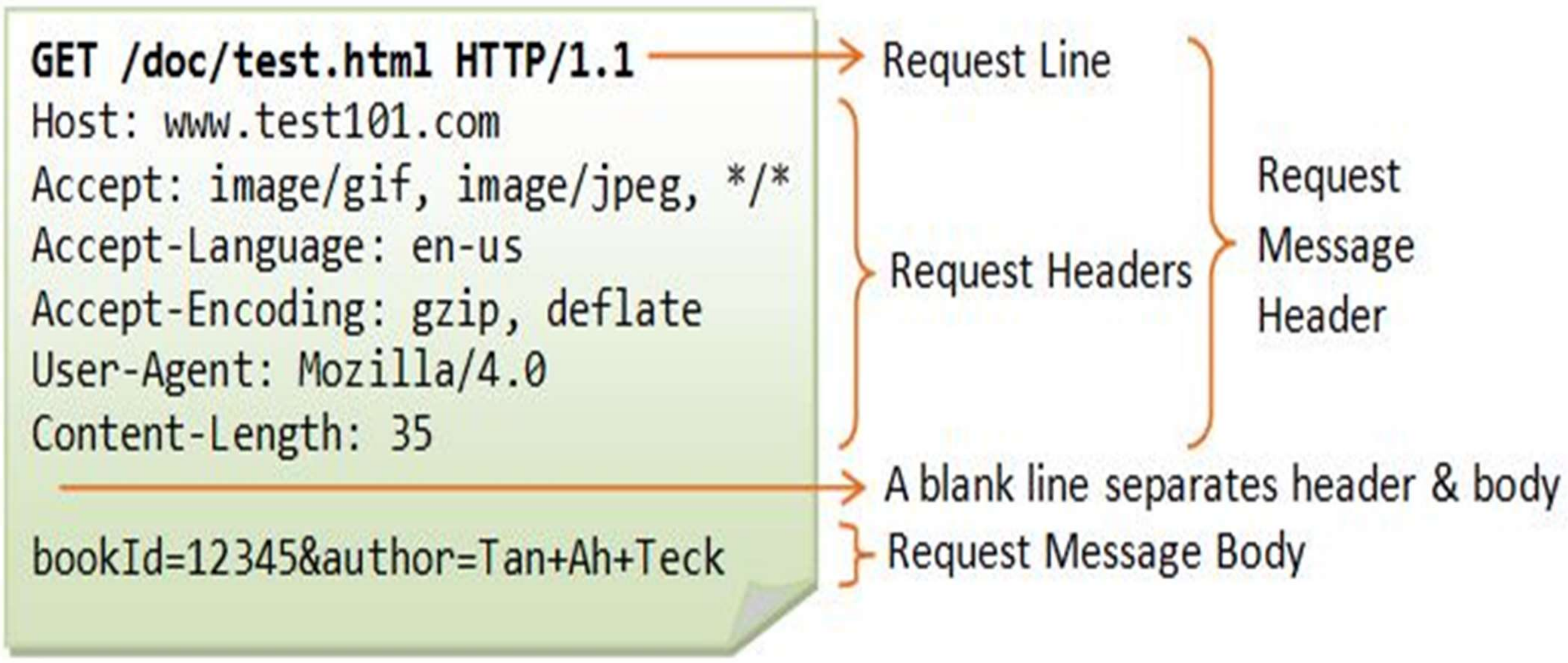
- Whenever you issue a URL from your browser to get a web resource using HTTP, **e.g. http://www.nowhere123.com/index.html**, the browser turns the URL into a request message and sends it to the HTTP server.
- The HTTP server interprets the request message, and returns you an appropriate response message, which is either the resource you requested or an error message. This process is illustrated below:

HTTP Request Processing


(2) Browser sends a request message
(1) User issues URL from a browser
http://host:port/path/file
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)

HTTP Request

- For example, the browser translated the URL http://www.test101.com/doc/index.html into the following request message:

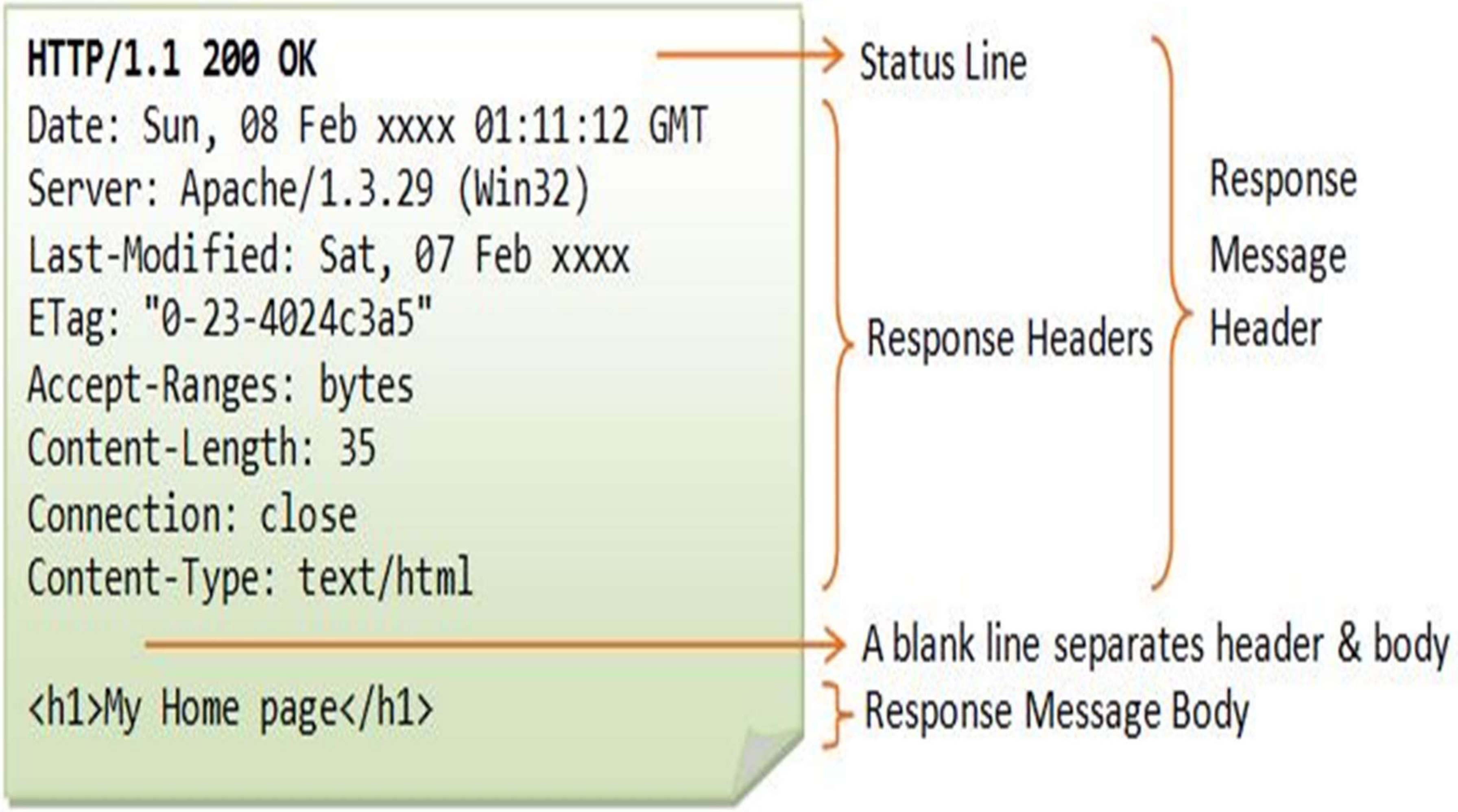


```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

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

- **When this request message reaches the server, the server can take either one of these actions:**
- **The server interprets the request received, maps the request into a file under the server's document directory, and returns the file requested to the client.**
- **The server interprets the request received, maps the request into a program kept in the server, executes the program, and returns the output of the program to the client.**
- **The request cannot be satisfied, the server returns an error message.**

HTTP response message



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

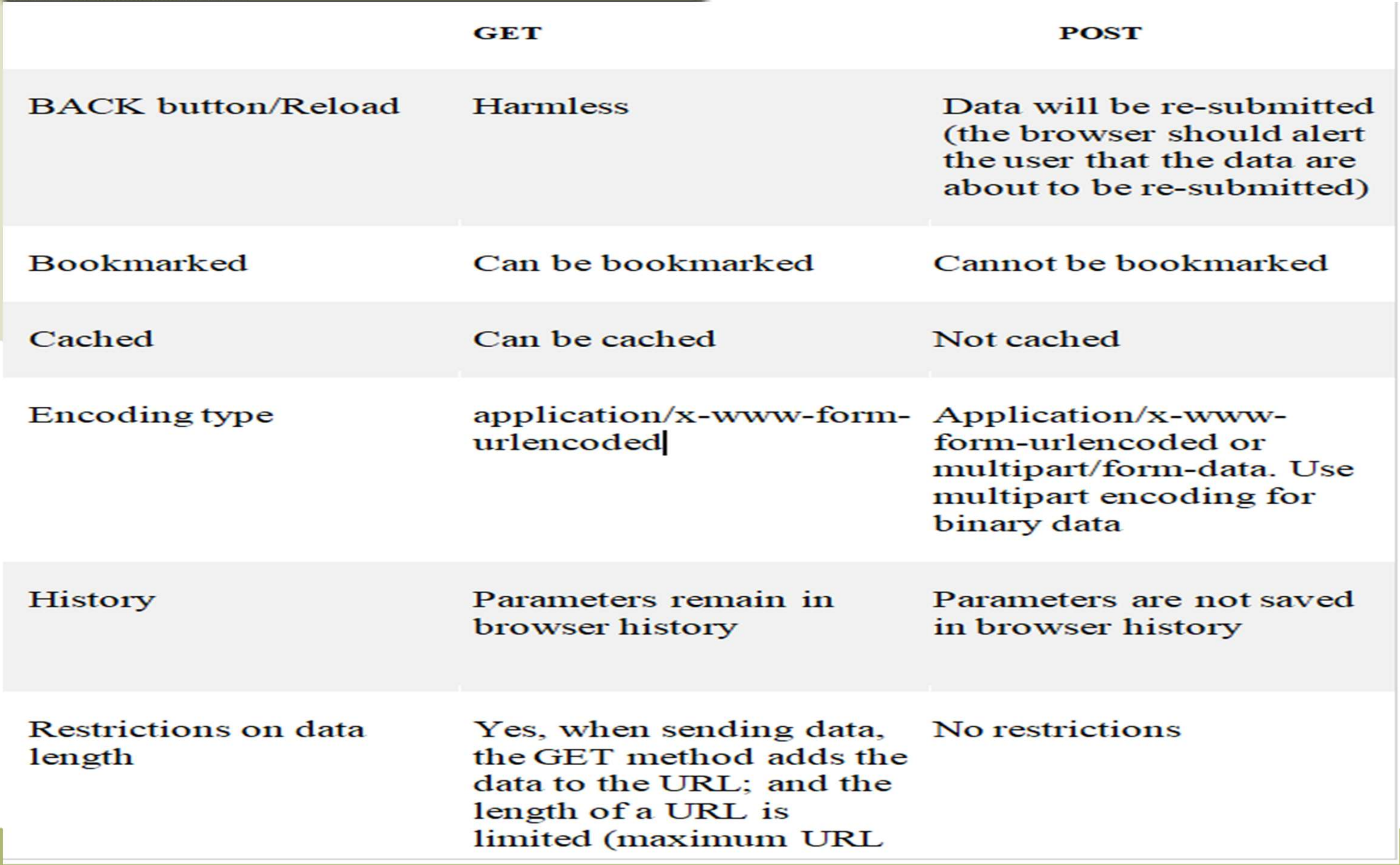
HTTP Request Methods

- HTTP protocol defines a set of request methods. A client can use one of these request methods to send a request message to an HTTP server. The methods are:
- 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.

HTTP Request Methods

- **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.**

The following table compares the two HTTP methods: GET and POST



	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	Application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL	No restrictions

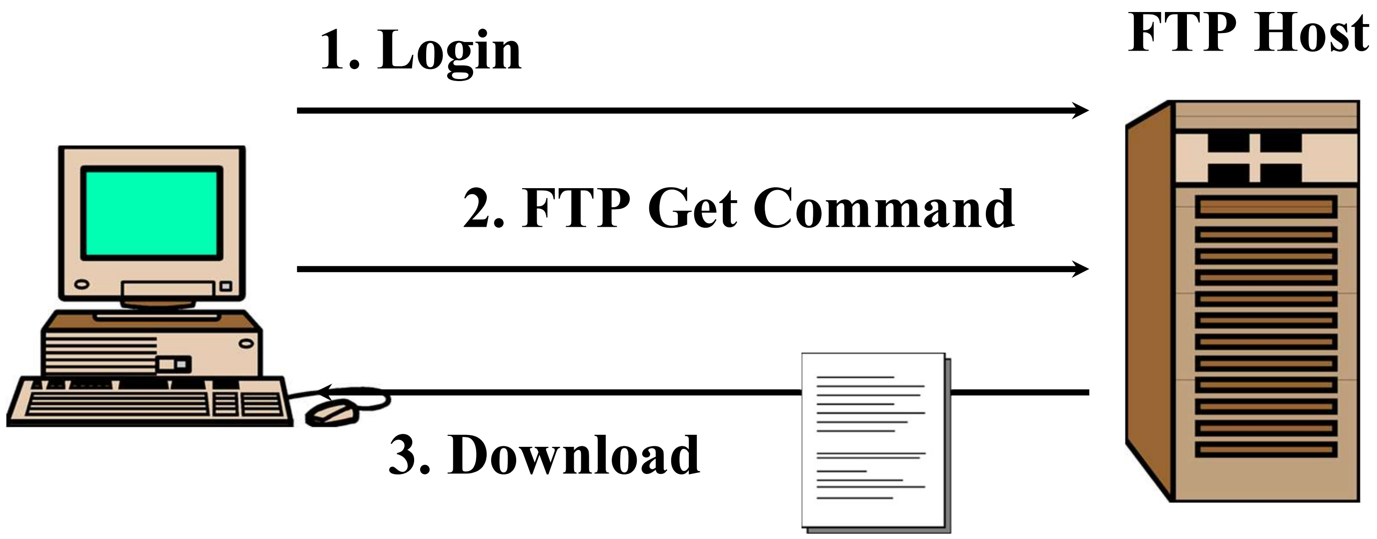
	GET	POST
	length is 2048 characters)	
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

File Transfer Protocol (FTP)

- The **File Transfer Protocol** (**FTP**) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.

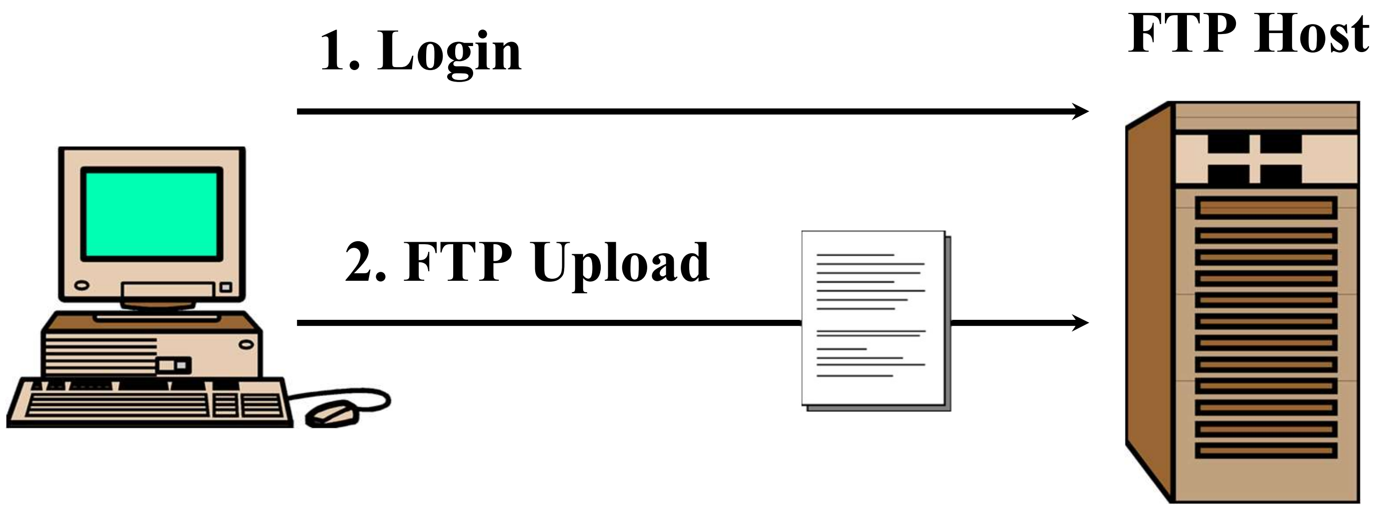
File Transfer Protocol (FTP)

- FTP host stores files
- Client logs into host
- Client program sends command to get a file
- FTP host downloads the file with error correction



File Transfer Protocol (FTP)

- User can also *upload* a file to the FTP Server
  - WWW cannot do this



File Transfer Protocol (FTP)

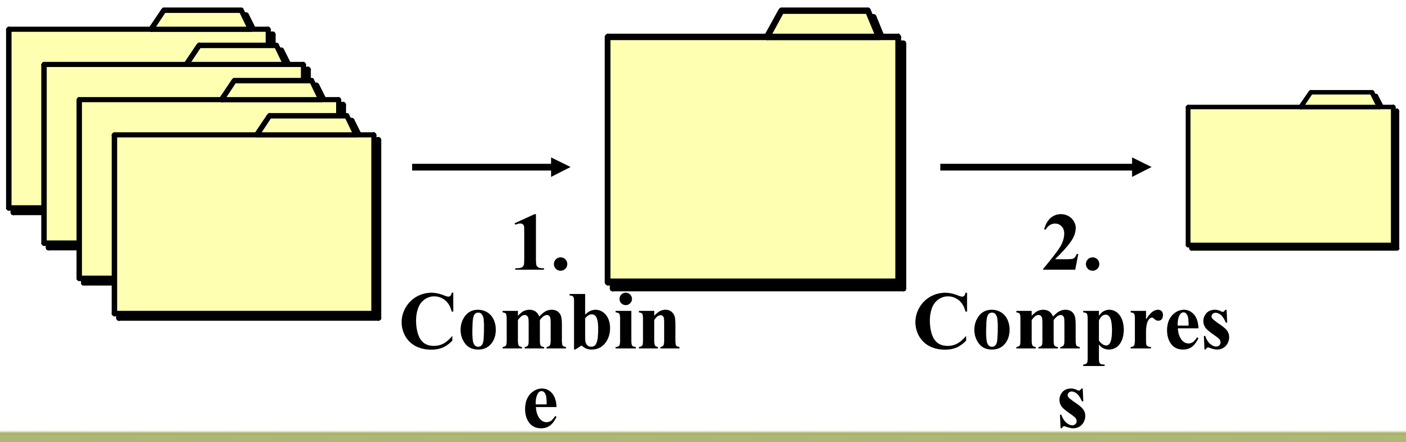
- Must Log into FTP Host Before Transfers
- Traditional FTP
  - You log into a specific account with a password
  - You can transfer to and from directories accessible to that account
- *Anonymous FTP*
  - You log in as “anonymous”
  - Give your e-mail address as password (usually optional)
  - Host gives you access to public directories
  - Usually for downloading only
  - Not truly anonymous: your internet address is known

FTP File Format Standards

- There are no standards for FTP file formats
  - Nothing like HTML
  - No limit on generality
  - No help in handling after downloads
  - You must know the file structure of the file you download
  - You must have an application program that can read that file format

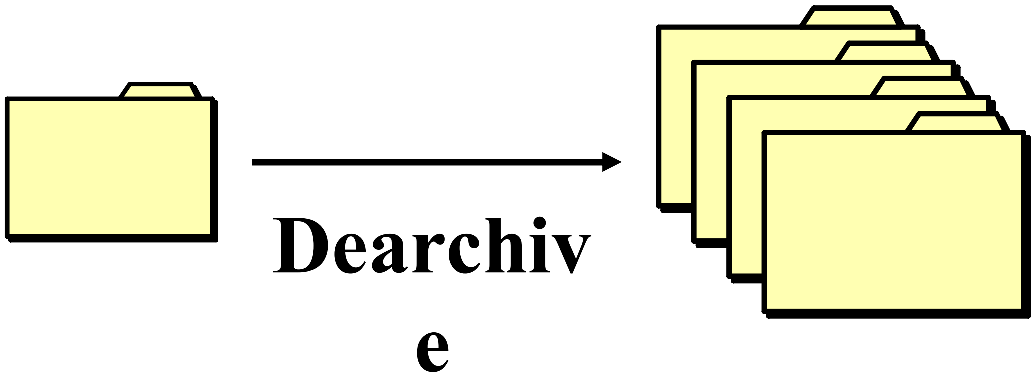
FTP Archiving

- Many FTP files are *archived*
  - Two-step process
  - First, several files are combined into one archive to avoid having to make multiple downloads
  - Second, the combined files are compressed to reduce download times



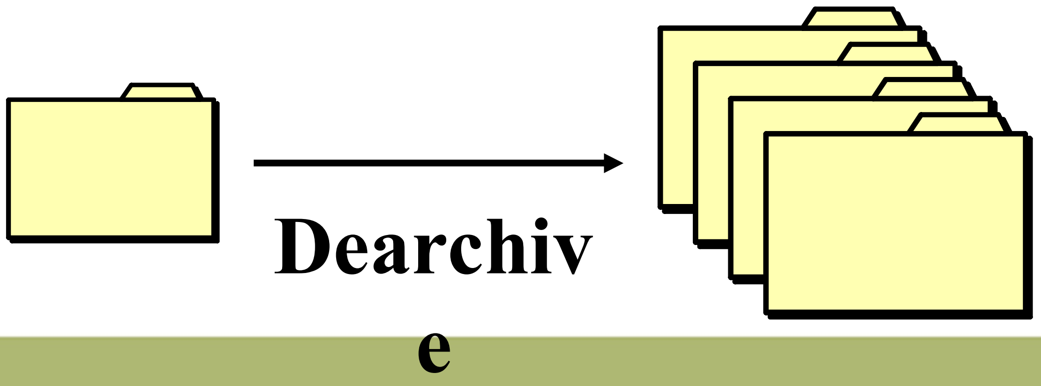
FTP Archiving

- Receiver must dearchive the files to read them
- Unfortunately, many archiving standards
  - *Zip* is the most common
  - UNIX users tend to use others (tar)
  - Some dearchiving programs handle only one archiving standards, others several



FTP Archiving

- Receiver must dearchive the files to read them
- *Self-Dearchiving Files*
  - Usually, you need a dearchiving program to dearchive
  - However, some archives have the extension .exe
  - Really programs; Dearchive themselves
  - Executing the archive file dearchives its files
  - These are programs, so be wary of viruses!



What is cross browser compatibility?

- Making your website behave consistently across browsers is called cross browser compatibility. Whether it is just look and feel or its behavior, a website can be made to function similarly across browsers with cross browser compatibility.
- **Ensuring cross browser compatibility is an important step in the web design process that many web designers tend to bypass or compromise on.**

Cross browser compatibility issues

- You know you have a cross browser compatibility problem when you receive complaints from your visitors that some of your website functions are not working.
- An example of this is when someone visits your site in say **Internet Explorer** and the layout of the site displays your content differently than when you are in a different browser.
- The cause of this is typically due to **"*unsupported*" tags** that are "*deprecated*" and are not recommended for use or you have Bad HTML, CSS or JavaScript code that was incorrectly written.

How should I fix compatibility issue?

- The first step in fixing a browser compatibility problem is to check the code to see if it passes "*validation*".
- Code validation will show you where your website has incorrectly coded HTML tags, missing tags, non W3c compliant tags, and more. You can validate your website for free at the **W3c** site. See the following links on this.
- World Wide Web Consortium(W3c)

How should I fix compatibility issue?

- **HTML Validator**
  http://validator.w3.org/
- **CSS Validator**
  http://jigsaw.w3.org/css-validator/
- Once you validate your HTML and have no errors on your site, You should see the browser look more uniform.
- You will want to continue looking through your website code and fixing code problems until all browsers look the same.
- This process will require you to be able to check your website in each browser.

How can I test for browser compatibility?

- There are different ways you can test your site for cross browser compatibility. You can install all the known browsers on your computer and then view your site in each browser as you develop it. The downfall to this is you will not be able to see what the older versions of a browser looks like.

- You can get programs that check your website for you.
- To **check different Internet Explorer browser** versions you **can use "*IE Tester*".** Or you can try a universal browser compatibility checker like what's at BrowserShots. Below are links to different software that checks your website browser compatibility
- **BrowserShots** http://browsershots.org/
- **IE Tester** http://www.my-debugbar.com/wiki/IETester/HomePage
- **Adobe Browser Lab** https://browserlab.adobe.com/en-us/index.html
- **Quirks Mode** http://www.quirksmode.org/compatibility.html

What is Transport Layer Security (TLS)?

- Transport Layer Security, or TLS, is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet.
- A primary use case of TLS is encrypting the communication between web applications and servers, such as web browsers loading a website.
- TLS can also be used to encrypt other communications such as email, messaging, and voice over IP (VoIP).
- TLS was proposed by the Internet Engineering Task Force (IETF), an international standards organization, and the first version of the protocol was published in 1999. The most recent version is TLS 1.3, which was published in 2018.

What is the difference between TLS and SSL?

- TLS evolved from a previous encryption protocol called Secure Sockets Layer (SSL), which was developed by Netscape.
- TLS was proposed by the Internet Engineering Task Force (IETF), an international standards organization
- TLS version 1.0 actually began development as SSL version 3.1, but the name of the protocol was changed before publication in order to indicate that it was no longer associated with Netscape. Because of this history, the terms TLS and SSL are sometimes used interchangeably.

What is the difference between TLS and HTTPS?

- HTTPS is an implementation of TLS encryption on top of the HTTP protocol, which is used by all websites as well as some other web services. Any website that uses HTTPS is therefore employing TLS encryption.
- **Additional Alerts added**
- **Modification to hash calculations**
- **Protocol version 3.1 in ClientHello, ServerHello**

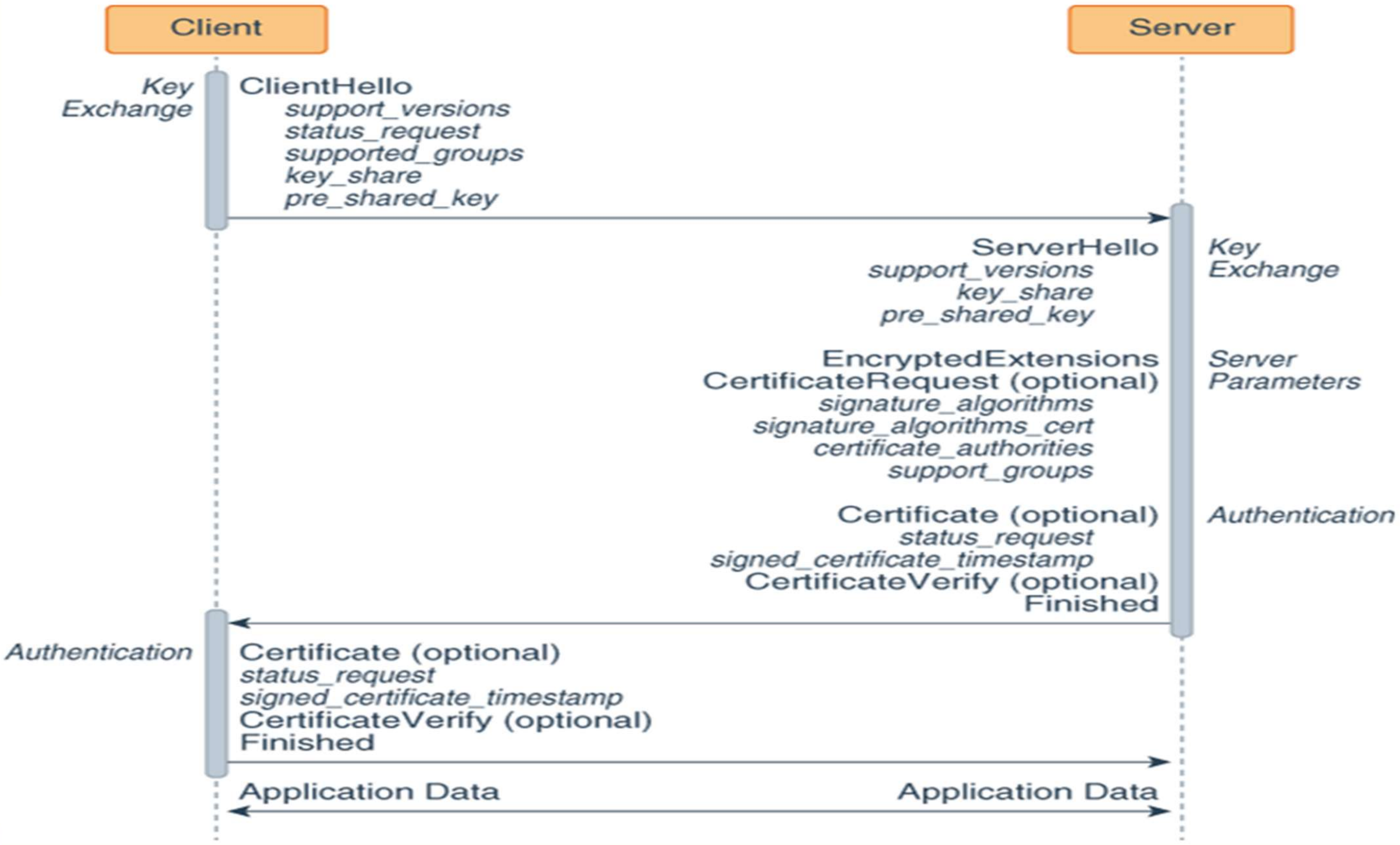
Why should businesses and web applications use the TLS protocol?

- TLS encryption can help protect web applications from data breaches and other attacks.
- Additionally, TLS-protected HTTPS is quickly becoming a standard practice for websites.

What does TLS do?

- There are three main components to what the TLS protocol accomplishes: Encryption, Authentication, and Integrity.
- **Encryption:** hides the data being transferred from third parties.
- **Authentication:** ensures that the parties exchanging information are who they claim to be.
- **Integrity:** verifies that the data has not been forged or tampered with.

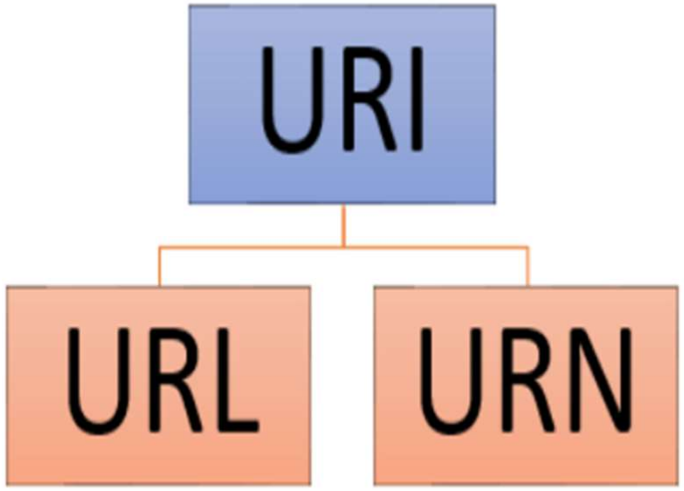
TLS 1.3 Handshake



- **Key exchange:**
  - The client sends a ClientHello message to server.
  - The server processes the ClientHello message and determines the appropriate cryptographic parameters for the connection. It then responds with its own ServerHello message, which indicates the negotiated connection parameters. For TLS 1.3, the ServerHello message determines the key and cipher options only. Other handshake parameters may be determined later.
- **Server parameters:** The server sends two messages to establish server parameters:
  - EncryptedExtensions: This message contains responses to ClientHello extensions that are not required to determine the cryptographic parameters, other than those that are specific to individual certificates.

  - **CertificateRequest (optional): If certificate-based client authentication is desired, then the server sends this message, which contains** the desired parameters for that certificate. This message is omitted if client authentication is not desired.
- **Authentication:**
  - The server sends these authentication messages:
    - **Certificate (optional):** This message contains the authentication certificate and any other supporting certificates in the certificate chain. This message is omitted if the server is not authenticating with a certificate.
- **Note: The Certificate message can contain a raw key instead of a certificate.**
    - **CertificateVerify (optional):** This message contains a signature over the entire handshake using the private key corresponding to the public key in the Certificate message. This message is omitted if the server is not authenticating with a certificate.
    - **Finished:** a MAC (Message Authentication Code) over the entire handshake.
  - The client responds with its own Certificate, CertificateVerify, and Finished messages.
  - The Certificate message is omitted if the server did not send a CertificateRequest message.
  - The CertificateVerify message is omitted if the client is not authenticating with a certificate.
- The client and server can now securely send application data to each other.

URI



Types of URI

As mention in the above figure, there are two types of URI:

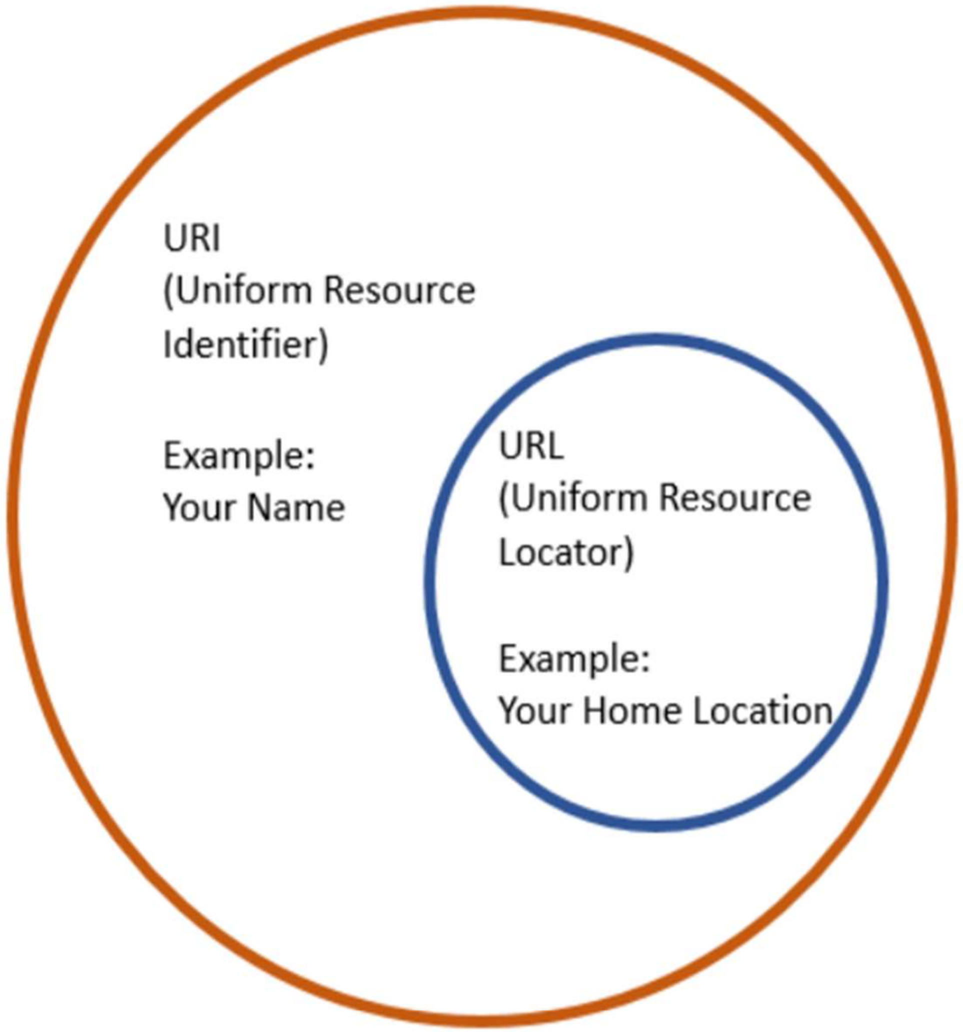
1. **URL:** URL specifies a location on the computer network and technique for retrieving it.
2. **URN:** Uniform Resource Name (URN) is an internet resource that specifies URN scheme.

URI Vs URL

KEY DIFFERENCES:

- URL is a subset of URI that specifies where a resource exists and the mechanism for retrieving it, while URI is a superset of URL that identifies a resource
- The main aim of URL is to get the location or address of a resource whereas the main aim of URI is to find a resource.
- URL is used to locate only web pages, on the other hand, URI in used in HTML, XML and other files.
- URL contains components such as protocol, domain, path, hash, query string, etc. while, URI contains components like scheme, authority, path, query, etc.
- Example of URL is : https://google.com while example of URI is :urn:isbn:0-486-27557-4.

Ven Diagram of URIs and URL



REST API

REST

- REST is a set of architectural constraints, not a protocol or a standard. API developers can implement REST in a variety of ways.
- When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text. JSON is the most generally popular programming language to use because, despite its name, it's language-agnostic, as well as readable by both humans and machines.

API

- An API is a set of definitions and protocols for building and integrating application software.
- It’s sometimes referred to as a contract between an information provider and an information user—establishing the content required from the consumer (the call) and the content required by the producer (the response).
- For example, the API design for a weather service could specify that the user supply a zip code and that the producer reply with a 2-part answer, the first being the high temperature, and the second being the low.