COMP2406A

Lecture Notes and Summary
Intro to Web Apps and HTTP

Main lecture content –

- HTTP protocol and how it forms the backbone of the web
- Request and response model of HTTP
- Statelessness of HTTP and its benefits
- Structure of URLs
- Caching mechanism and how it benefits communication over the web
- Various other web terminologies

This course differs from the ones is first year because those dealt with desktop applications with interaction taking place mainly through the console. It mostly involved one user and no interacting resources. This course involves two parties, client and the server. Multiple users and lots of interacting resources need to be handled. Dynamic responses are also implemented in web applications with resources being constantly updated and the program responding to the user's actions. The main protocol underlying all web apps and the one that facilitates all of this interaction is the HTTP.

Some of the issues that weren't important during programming of desktop application that will pop up during this course is scalability, latency and uncertainty. Latency comes into play because in web applications data is transferred over long distances with servers and clients being far away and it is necessary to ensure getting and sending resources take as less time as possible. Scalability is also important to consider when the application has lots of users and writing code for the servers should ensure it'll be able to handle the larger number of users and scale well. Understanding this well will help you when designing your final project and in explaining in your final report how robust your application is.

Generally, with web applications, it is good to minimize the data being sent over the internet. Asynchronous programming also helps with ensuring applications are not stuck on a single request and can process other requests simultaneously which also allows scalability. Furthermore, it is also good to transfer some of the processing to the client. HTTP has features that enable these things in a way.

HTTP – Hyper Text Transfer Protocol
It is the protocol that allows clients and servers to communicate with each other. When clients type into the browser or request information, server decides what to send back, the rules under which this is done is HTTP. It follows a request/response model, where the user/client requests a resource and the server sends the response.

Requests and responses contain headers which specify details and metadata about it. They also contain bodies, which make up the main content. HTTP is stateless which means the server

does not store information about that specific interaction and the client. All information required for request is specified in the request itself. Statelessness is good because it offers scalability, when millions of users are making requests to the server, the server would run out of memory if it stored their info and additionally, this ensures if a server fails, any other server can continue the process as no extra information about the client is needed.

HTTP uses plain text for request and response headers, this allows it to be human-readable and thus easy to debug. However, the body containing the main data can hold binary. Since headers are stored in plain text, it is not secure, for this HTTPS is required which enables sensitive data to be transferred.

HTTP is an application layer protocol, the lower layers with complex processes are abstracted away and are not necessary for us. Separation of concerns(client-server) will also recur throughout the course, keep an eye out for it as it'll be important for the project.

DNS – Domain Name System
Maps domain names to IP addresses, removes need for us to remember specific IP addresses to access it on the web. They go from most specific to most general, example - people.scs.carleton.ca

URL – Uniform Resource Locator
Represents the location and name of resource on the web. It allows users to request specific resources. This is useful to understand so we can understand what queries or actions clients are making to our servers in the project.

Parts of a URL –
https://google.com/#q=express

https: - protocol used
google.com – hostname
#q=express – fragment

http://bing.com/search?q=grunt&first=9

https: protocol used
bing.com – host name, it specifies the server, full domain name
/search – path, first part of the URL your server cares about as it uniquely identifies resources within the application
?q=grunt&first=9 – query string, collection of key value pairs, important to understand, also when using forms in HTML

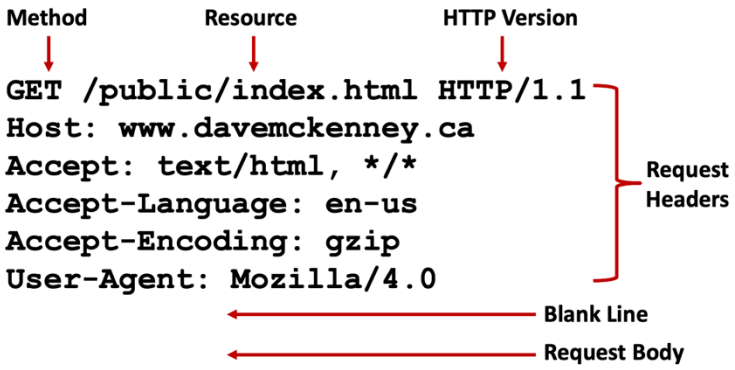
Localhost – refers to the computer we're currently on, the servers we make for our project will be on our own computers, so it'll be localhost

FTP – File Transfer Protocol

Ports are not always included in URL's, default port is 80 and the ones we should use 3000 when using JavaScript. It allows the server to direct traffic to a certain program.

Format of an HTTP request-

- Request line – domain name, port, URL
- Header – more info about request
- Blank line – shows end of header
- Message body(optional)



There are different types of request methods-

- GET – retrieve resources
- POST – send data to server to create or update
- HEAD – similar to GET, only header
- PUT – store new resource, used in RESTFUL web design (later in the course)
- DELETE – remove resource

GET and POST are most common

GET URL's include query strings
Example - /test/demo_form.php?value1=val1&value2=val2

/test/demo_form.php – the PHP file/database its requesting resources from

GET request can be cached which means once it's requested by the client, it can be store locally so if it is requested again, it can be retrieved quickly without needing to transfer data over the internet which can cause delays. It can also be bookmarked to revisit later. It has length restrictions as we're only asking for data. It's not good for transferring sensitive data as it's stored in plain text and in the browser history. Express handles caching later in the course. Different caching mechanisms can check if the data has been modified, and it will not send the cached version.

POST method

Example POST:

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com

name1=value1&name2=value2
```

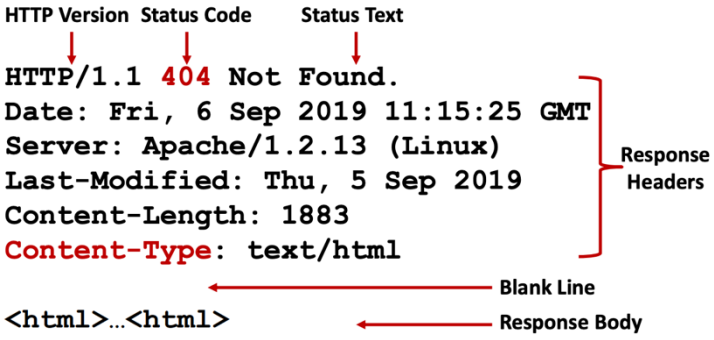
Data is stored in the body, it is not cached as we're sending data, no length restriction and not stored in browser history

Some common request headers-

- Accept – what content type to accept, client specifying what type of data it wants, will come up a lot in the course
- Accept-Encoding – specifies compression to reduce size of files sent over the web
- Authorization – used for logged in
- Content-Length
- Content-Type – what type of data client is sending – image, video, JSON data, HTML

HTTP Response

After client makes request, server processes it, and then sends a response. Structure is quite similar to requests. Important to understand to get how client-server will communicate.



Content-Type will allow the client to understand how it'll handle the incoming data.
Status codes are machine readable; it shows the result of the request. We'll automate these responses when coding servers, allows us to use logical operators with the status code to handle results appropriately. Later in the course, we'll learn about handling multiple data types.

While this chapter doesn't involve any programming, and it can be quite easy to overlook this chapter, it is imperative to understand the fundamental workings behind the client-server communication. Even though, we've covered only client-side programming till now in this course, we will start covering the server side more. It'll be important to understand how the requests and responses work between client and server to ensure it's easy to implement in your final project when the time comes.

For more clarification on the chapter, check out the chapters 1.2 and 1.3 on the Zybooks we have access to for this course.

Here's a link to see all the list of status codes-
https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Here's a fantastic link for understanding the finer details of HTTP Request/Responses
http://blog.catchpoint.com/2010/09/17/anatomyhttp/

Fun link to explore a list of top-level domain names-
https://en.wikipedia.org/wiki/List_of_Internet_top-level_domains