Foundational Technologies - HTTP

Tools

- Browser - web developer tools
- curl - a command-line HTTP client: tool to transfer data to or from a server, using any of the supported protocols such as HTTP, FTP, TFTP etc https://curl.se/
- Postman - an API platform for building and using APIs

HTTP

- Hypertext Transfer Protocol (HTTP) is an application-layer protocol in the Internet protocol suite for transmitting hypermedia documents, such as HTML.
- It operates as a client-server protocol, with requests typically initiated by web browsers.
- Web browser communicates with a web server
- Mobile app communicates with a server to populate app data

Clients and servers communicate by exchanging individual messages (as opposed to a stream of data). The messages sent by the client, usually a Web browser, are called requests and the messages sent by the server as an answer are called responses.

Use case

User enters a URL on a web browser which initiates a HTTP transfer, the web browser creates a HTTP request and uses the network to send it to the server as the server is part of the URL. Server creates a HTTP response and uses the same network connection to send it to the browser. Browser renders the HTML code to display the page in the browser. User views the web page contained in the HTTP response.

HTTP Data Exchange

- HTTP is a stateless application-level protocol and it requires a reliable network transport connection to exchange data between client and server

Request and Response Messages through Connections

- Data is exchanged through a sequence of request–response messages which are exchanged by a session layer transport connection

Persistent Connections

- HTTP persistent connection or HTTP keep-alive or HTTP connection reuse is the concept of using a single TCP connection to send and receive multiple HTTP requests and responses, as opposed to opening a new connection
- IN HTTP.1.1 a connection can be reused for more than one request/response
- Persistent connections reduce request latency because the client does not need to re-negotiate the TCP 3-Way-Handshake connection after the first request has been sent
- Connection becomes faster with time due to TCPs slow-start mechanism

Curl

- command-line tool and library for transferring data with URLs. It is a versatile and widely used tool in the world of web development and networking. Curl allows you to make HTTP, HTTPS, FTP, FTPS, SCP, SFTP, and many other types of network requests to transfer data between your local system and remote servers or resources.

Commands:

- curl -v https://www.spartaglobal.com
- Network IP address we are sending address to obtained by Domain name service (DMS) to create connection between hostname and IP address

```
riggy@riggy-Latitude-5310:~$ curl -v https:/www.spartaglobal.com
*   Trying 178.62.25.232:443...
* Connected to www.spartaglobal.com (178.62.25.232) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
*  CAfile: /etc/ssl/certs/ca-certificates.crt
*  CApath: /etc/ssl/certs
* TLSv1.0 (OUT), TLS header, Certificate Status (22):
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS header, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS header, Finished (20):
```

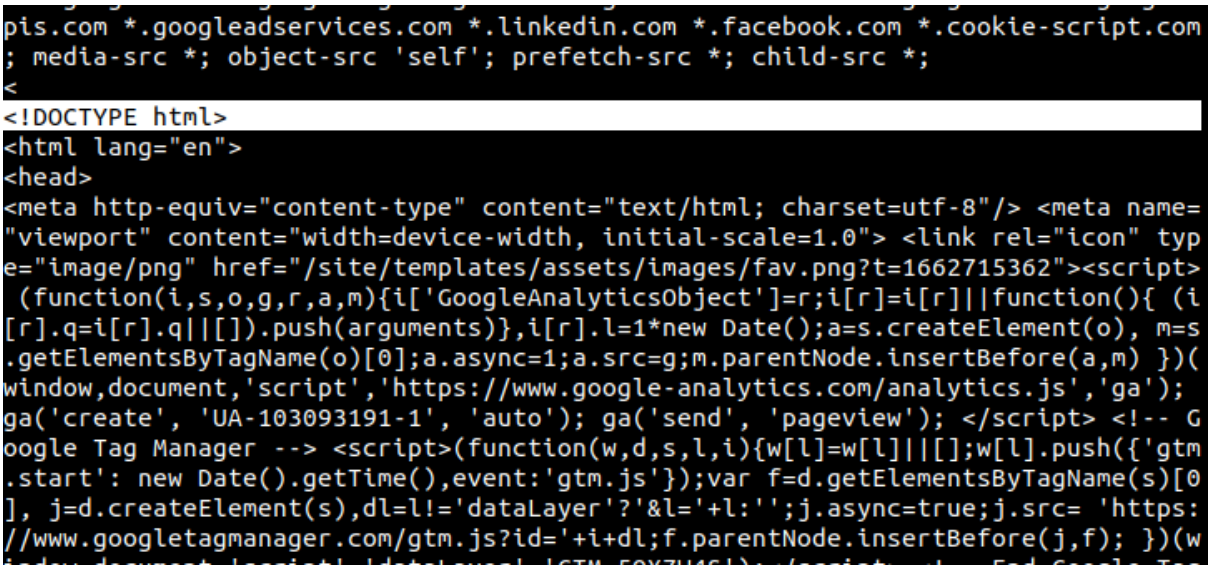
- **HTTP request** (outgoing messages)
- GET is a HTTP method and version of HTTP
- Header include host information for processing the request
- User Agent header creates the HTTP request
- The Accept request HTTP header indicates which content types, expressed as MIME types, the client is able to understand. */* indicates accepting anything
- Blank line signals the end of HTTP request which is sent to the server

```
* Using Stream ID: 1 (easy handle 0x558c8b094e90)
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
> GET / HTTP/2
> Host: www.spartaglobal.com
> user-agent: curl/7.81.0
> accept: */*
>
* TLSv1.2 (IN), TLS header, Supplemental data (23):
```

- **HTTP response**
- Information returned with the response is dependent on the method used in the request
- HTTP version and 200 HTTP status code indicates successful
- Server header displays name of the server that generated the response
- Date header
- Content Type header provides the client with the actual content type of the returned content such as text/html, charset shows encoding method
- Connection: keep-alive is an instruction that allows a single TCP connection to remain open for multiple HTTP requests/responses. By default, HTTP connections close after each request.
- Set-Cookie HTTP response header is used to send a cookie from the server to the user agent, so that the user agent can send it back to the server later
- Expires header indicates whether it is possible to obtain information on request from the browser cache
- The Cache-Control HTTP header field holds directives (instructions) — in both requests and responses — that control caching in browsers and shared caches (e.g. Proxies, CDNs).
- The Pragma header is used to pass directions along with the message. These directions could be almost anything, but often they are used to control caching behaviour

```
* TLSv1.2 (IN), TLS header, Supplemental data (23):
< HTTP/2 200
< server: nginx
< date: Mon, 04 Sep 2023 19:45:28 GMT
< content-type: text/html; charset=utf-8
< vary: Accept-Encoding
< set-cookie: wires=2jk2ivtf5dkgcs95lu8s2bjhdc; path=/; secure; HttpOnly;
te=Lax
< expires: Thu, 19 Nov 1981 08:52:00 GMT
< cache-control: no-store, no-cache, must-revalidate
< pragma: no-cache
< x-powered-by: ProcessWire CMS
< x-frame-options: SAMEORIGIN
< x-xss-protection: 1; mode=block
< x-content-type-options: nosniff
< referrer-policy: origin-when-cross-origin
< permissions-policy: autoplay=*, camera=(), display-capture=(), fullscre
eolocation=(), microphone=(), web-share=self
```

- In HTML, the doctype is the required "<!DOCTYPE html>" found at the top of all documents. Its purpose is to ensure that the browser makes a best-effort attempt at following the relevant specifications, rather than using a different rendering mode that is incompatible with some specifications.
- The <body> HTML element represents the content of an HTML document. There can be only one <body> element in a document.



- Developer tools in the browser can be used to see GET requests and Server request headers
- The browser is always the entity initiating the request
- Use Ctrl Shift I to access web developer tools

HTTP Overview
https://developer.mozilla.org/en-US/docs/Web/HTTP

- HTTP Components: HTTP operates on top of the TCP/IP stack and consists of various components, including requests, responses, headers, and messages. It is designed to be extensible and versatile.
- Client-Server Communication: In HTTP, clients (usually web browsers) send requests to servers, which respond with appropriate resources. This communication is based on individual messages rather than a continuous data stream.
- Web Documents: A web document is composed of various resources like text, layout information, images, videos, scripts, etc. HTTP fetches these resources to construct a complete web page.
- User-Agent: The user-agent, typically a web browser, initiates HTTP requests. It also processes responses, combining resources to render web pages.
- Web Servers: Servers handle client requests by serving documents or generating them dynamically. Servers can be single machines or a collection of machines that distribute the workload.
- Proxies: Proxies are intermediary entities that can perform various functions, including caching, filtering, load balancing, authentication, and logging, between clients and servers.
- Stateless Nature: HTTP is a stateless protocol, meaning there's no connection link between successive requests. However, stateful sessions can be created using HTTP cookies to maintain context across requests.

- Transport Layer Control: HTTP relies on the underlying transport layer, typically TCP, which is connection-based. It requires establishing a TCP connection before exchanging request/response pairs.
- HTTP Flow: The typical HTTP flow involves opening a TCP connection, sending an HTTP request, receiving a response, and optionally reusing or closing the connection. HTTP pipelining and persistent connections improve efficiency.
- HTTP Messages: HTTP messages can be human-readable in earlier versions and are encapsulated into frames in HTTP/2 for optimization. Messages include requests (with methods, paths, headers, and bodies) and responses (with version, status codes, headers, and bodies).
- APIs Based on HTTP: Common APIs based on HTTP include XMLHttpRequest and Fetch for data exchange between user agents and servers. Server-sent events allow one-way communication from the server to the client.

HTTP request methods

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource

GET - The GET method is used to request a representation of a specific resource from the server. It is primarily used for retrieving data from the server. When you enter a URL in your browser and press Enter, it typically sends a GET request to the server to fetch the web page associated with that URL.

POST - The POST method is used to submit data to the server to create or update a resource. It often leads to a change in the server's state or triggers side effects. For example, when you submit a form on a website, the data is sent to the server using a POST request.

PUT - The PUT method is used to update or replace the current representation of a resource with the request payload. It essentially replaces the existing resource with the new data provided in the request. PUT is idempotent, meaning multiple identical requests should have the same effect as a single request.

DELETE - The DELETE method is used to request the removal of a specific resource from the server. It instructs the server to delete the resource identified by the URL. It's important to use this method with caution as it permanently removes the resource.

PATCH - The PATCH method is used to apply partial modifications to a resource. It is typically used when you want to update only a portion of a resource rather than replacing it entirely. It's a more efficient way to make small changes to a resource.

More HTTP request methods

HEAD - The HEAD method is similar to GET in that it requests information from the server about a specific resource. However, it only asks for the response headers and not the actual content of the resource. This is useful when you want to check if a resource exists or retrieve metadata about it without downloading the full content.

CONNECT - The CONNECT method is used to establish a network connection to a resource identified by the target URI. It is often used for setting up secure communication through proxies, such as when establishing an HTTPS tunnel.

OPTIONS - The OPTIONS method is used to retrieve information about the communication options available for a specific resource. It allows the client to inquire about the methods, headers, and other capabilities supported by the server for the given resource.

TRACE - The TRACE method is used for diagnostic purposes. It performs a loop-back test by sending a request to the target resource and then having the server return the same request in the response. This helps in debugging and understanding how requests are processed by intermediate servers.

HTTP Status Codes
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
https://http.cat/
Response codes:
1xx - informational **100 = Continue**
2xx - successful **200 = OK**
3xx - redirection **301 = Moved Permanently**
4xx - client error **404 = Not Found**
5xx - server error **500 = Internal Server Error**

HTTP Cookies

- An HTTP cookie (web cookie, browser cookie) is a small piece of data that a server sends to a user's web browser.
- The browser may store the cookie and send it back to the same server with later requests.
- Used to tell if two requests come from the same browser—keeping a user logged in, for example.
- It remembers stateful information for the stateless HTTP protocol. Widely used for marketing. Cookies help in maintaining state, tracking user behaviour, and personalising user experiences on websites.

HTTPS

- HTTPS (HyperText Transfer Protocol Secure) is an encrypted version of the HTTP protocol. It uses TLS (Transport layer security) or its predecessor SSL (Secure Sockets Layer) to encrypt all communication between a client and a server.
- This secure connection allows clients to safely exchange sensitive data with a server, such as when performing banking activities or online shopping.
- Lock icon shows HTTPS certificate, certificate authority, privacy details, and protocols, encryption algorithms, TLS version, and technical details used for security

HTTPS Request (Client Perspective):

- Client Request: When a secure website (HTTPS) is visited, the web browser initiates a request to the server, seeking access to a web page. This request typically includes details such as the URL being accessed and any supplementary information.
- Encryption: Prior to dispatching the request, the web browser encrypts the data using the public key provided by the server. This step guarantees that only the server, armed with its private key, possesses the capability to decipher the request.

HTTPS Response (Server Perspective)

- Implementation of HTTPS Is managed on server-side, therefore the server is configured with certificates and keys in order to allow HTTPS encryption, then all communication is encrypted.

- Server Receives Request: The server receives the request in an encrypted form from the web browser.
- Decryption: Employing its private key, the server decrypts the request to comprehend the nature of the request.
- Server Processes Request: The server fulfils the request, whether it involves retrieving a web page, conducting a transaction, or executing any other task.
- Response Encryption: Should the server need to transmit a response (such as the requested web page), it encrypts the response data using the public key associated with the web browser before dispatching it.

HTTPS Response (Client Perspective)

- Client Receives Response: The web browser receives the response from the server in an encrypted form.
- Decryption: The web browser utilises its private key to decrypt and access the content of the response.
- Display or Action: The web browser proceeds to display the webpage or take the requisite action based on the response, which may involve presenting a secure page or executing a transaction.

TLS

- TLS (Transport Layer Security) is a protocol that secures internet communication. It encrypts data, ensures its integrity, and verifies the identity of servers and clients, making online interactions secure and private.
- Begins with handshake where client and server agree on encryption methods and exchange keys, all data is encrypted, TLS adds digital signatures to packets, allowing the recipient to verify that data has not been tampered with, TLS uses digital certificates to authenticate servers, ensuring their identity, TLS establishes a secure, private channel for data exchange, protecting against data manipulation and eavesdropping.

Difference between POST, PUT, and PATCH

POST	PUT	PATCH
Requests that the server accepts the data enclosed in the body of the request	Requests that the enclosed entity is stored under the supplied URI. If the URI refers to an existing resource, that resource is modified and then put back	Requests that an existing resource is modified with the enclosed partial data. Does not modify the entire data
A new entry is provided in the request body and added to a database table	Said to be idempotent - sending a request multiple times should be equivalent to a single request modification	Said to be non-idempotent - trying a request N times, results in having N resources with N different URIs created on the server
If the URI does not point to an existing resources, then the resource is created with that URI	If the resource does not exist, then it is created	If the resource does not exist, an error is returned