

# WEB FUNDAMENTALS

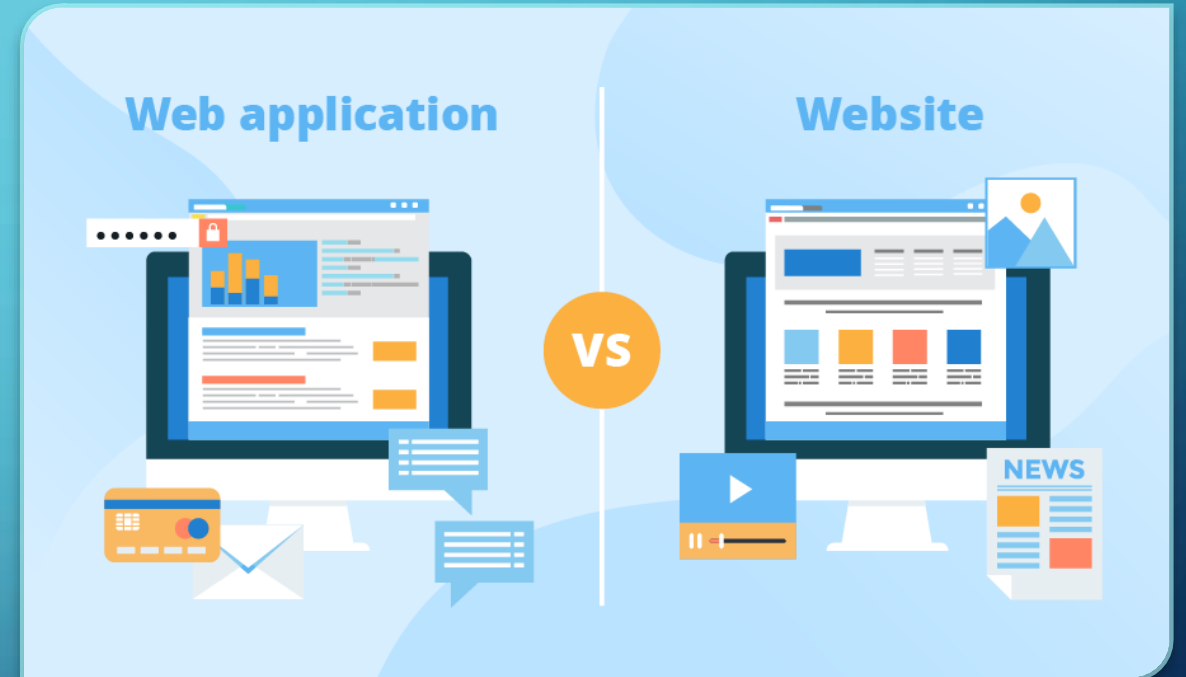
ICTDBS507/ICTPRG537 – DATA & SECURITY BUNDLE

# WHAT WE'RE COVERING

- Websites vs Web Applications
- Servers
- Http & Https – Requests & Responses
- URLs & Query Parameters
- Addresses & Ports

# WEBSITES VS WEB APPLICATIONS

What we often consider as websites are actually broken into 2 main categories, websites and web applications. Although they may look the same on the surface and how we initially access them is the same, they are actually quite different.



# WEBSITES

- Contains static content
- Page content is readable but not directly interactable.
- May still have forms and simple authentication but they don't affect the sites content directly.
- Generally easier to implement and deploy.







# WEB APPLICATIONS

- Content is designed for user interactions
- Will change content shown based upon user inputs and access.
- Commonly have a database component.
- Higher complexity in development and deployment.
- Generally require authentication to fully access content/functionality.
- Most common type of modern web pages.



# SERVERS

Regardless of whether you are developing a website or a web application. It needs to be hosted (stored) on a server to allow users to access.

Servers are used to serve content to a client device across a network such as the internet when it is requested.

They run on hardware that can be either hosted locally or remotely.

They can return a variety of different types of data from text to images and video.

There are different types of servers depending upon what their purpose is and how it is used.

# WEB SERVERS

- Web Servers focus on providing the infrastructure required to allow for hosting a web site or web application
- Web Servers may run alongside DB servers and application servers on the same hardware, or on their own hardware
- Common Web Servers include Internet Information Services (IIS), Apache HTTP Server, NGINX (Engine X)
- Web Servers can handle requests before they reach an application



©DESIGNALIKIE



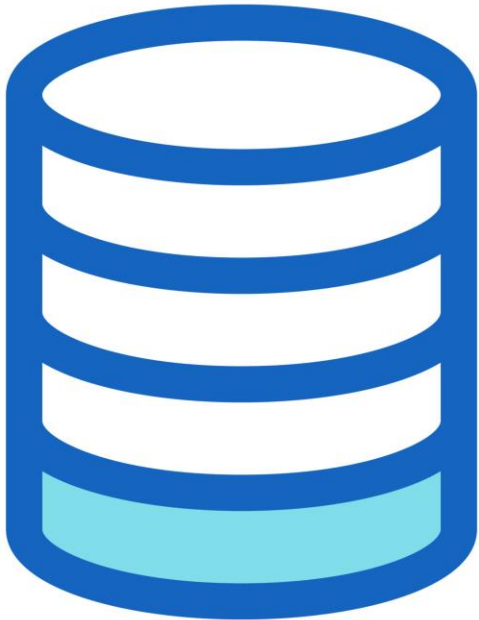
# APPLICATION SERVERS



- Application Servers work with, or in place of an operating system
- Manage the communication and interconnectivity between a range of different applications on a server
- An application server might manage the underlying infrastructure for IIS and SQL Server
- Windows Server is an example of an Operating System focused on providing an Application Server



# DATABASE SERVERS



- Database Servers are often referred to as Database Management Systems
- DBMS manage local and remote connections, working with requests and responses differently to HTTP
- Will generally require authentication
- Can exist on the same hardware as a web or application server, or on its own dedicated hardware

# CONNECTING TO A SERVER/SITE

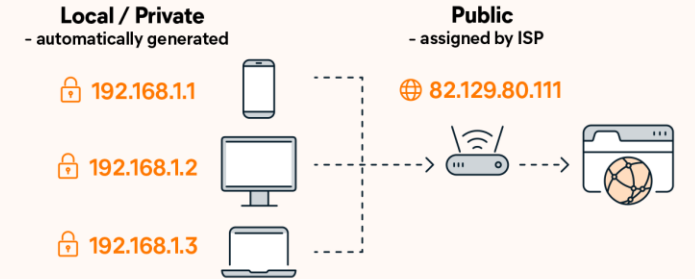


- When you type the address of a website, this is used to specify the domain name of the website which is used to locate it.
- It locates the desired site by using the DNS (Domain Name System), which is like the phonebook of the internet which stores all website names and locations.
- Using DNS the provided address is used to locate the IP Address of the site's domain and provides it to the machine to help find the desired address location.
- This system is better than using the IP Addresses directly because URL addresses are easier to remember and don't need to be changed if the hosting location of the web address changes.

# IP ADDRESSES

- Every device on the internet has an IP (Internet Protocol) Address which defines a unique identifier for that device on the internet (or intranet)
- The IP address is a numeric identifier consisting of 4 sets of numbers (called octets) between 0-255 and separated by dots. Example: 197.168.1.54
- One device, such as a web server, may be hosting multiple web applications, and each of them will share the same IP Address from that device.
- 'Localhost' points to the IP Address of '127.0.0.1', this is the internal IP address of your local machine used in development
- Visual Studio uses IIS Express when running a web application locally on this address

## Types of IP Addresses



### Static



- permanent
- used by servers or other important equipment

### Dynamic



- occasionally changes
- used for consumer equipment

### IPv4

192.168.5.18

- numeric dot-decimal notation

4.3 billion addresses

- addresses must be reused and masked

### IPv6

50b2:6400:0000:0000:  
6c3a:b17d:0000:10a9

- alphanumeric hexadecimal notation

7.9x10<sup>28</sup> addresses

- every device can have a unique address

## IP Address Security

### VPNs protect you against:

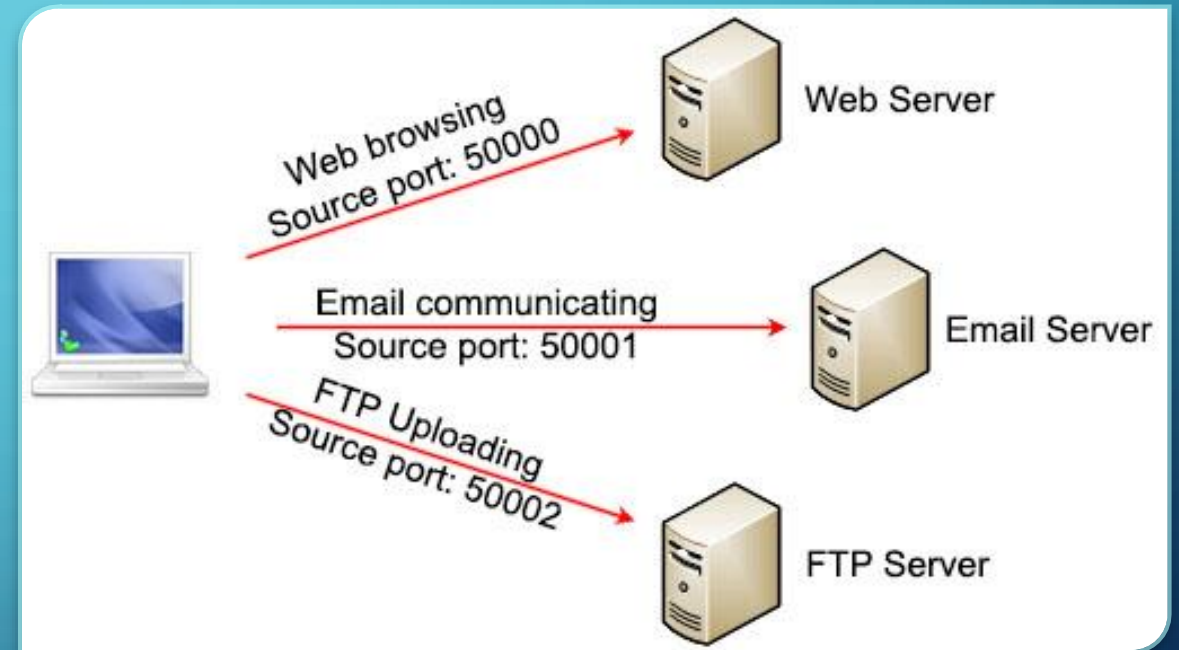
Ad tracking, Hackers, Government surveillance, Wi-Fi snoops



# PORTS

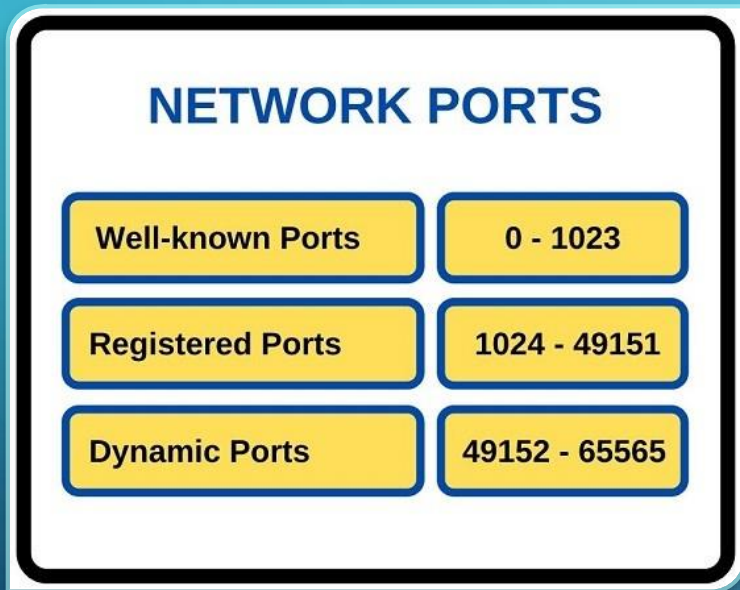
Because more than once piece of software/website/web application can potentially be running on the one machine and be sharing the same IP address, we need a way of determining which one is the intended target of any messages or data that is passed to the device. This is where port numbers come into play.

Each application on the device will need its own Port on the local machine which helps the device know where data should be sent once received from the internet.





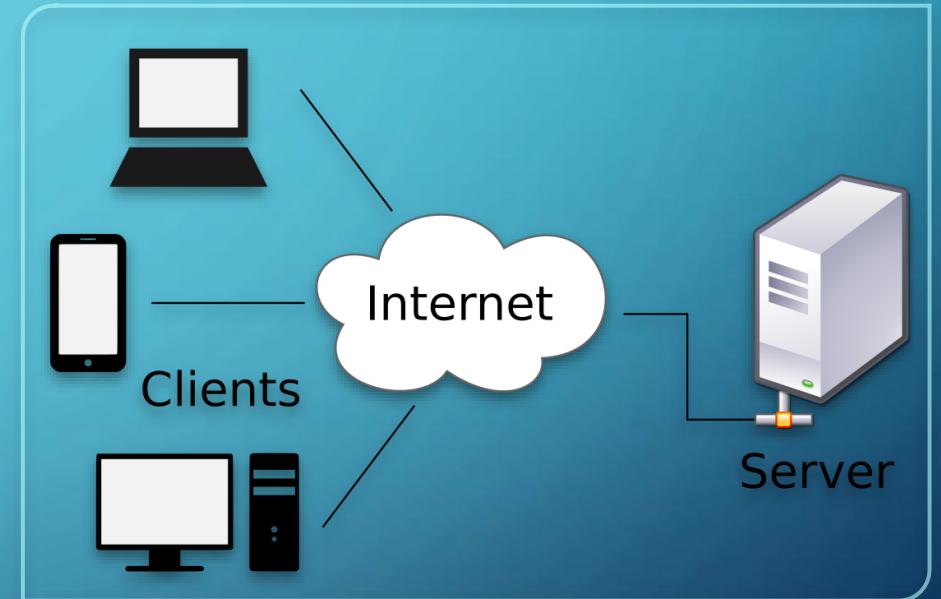
# PORTS



- Port numbers range between 0-65,535
- Port numbers between 0-1023 are considered restricted and are generally reserved for operating system services and prominent software companies such as SQL, Apple and Microsoft.
- Ports above 1023 are generally available for use but some numbers can be already registered to particular software already and may cause conflicts if you are not careful.
- If you allow for dynamic port selection, available ports are determined and assigned when an application is initialized.
- Port settings can be found in properties -> launchsettings.json

# SERVING CONTENT

- When a client request is made, it is received by a server and 'Handled'. On a webserver, this request may be routed to one or more web applications or sites.
- An appropriate response is calculated and can be returned in a variety of ways.
- A Web API may return JSON or XML, representing an object or data.
- SSR (Server-side rendering) web applications may return HTML, CSS + JS representing a web page that contains data



# REQUESTS

- A request is generally initiated by a client attempting to retrieve a resource on a server or trigger an event based on accessing an endpoint.
- A request is a combination of:
  - The destination (target / endpoint) of the request – Where is the request going?
  - The method of the request – Browsers loading a webpage use the GET method by default
  - The request headers – a description of the Requester + any metadata for the request or the requests content
  - The content of the request – generally POST or PUT requests contain data that is being sent to the server, get requests may contain data in the form of a query string

# REQUEST DESTINATION

- A webserver generally has a base address (<https://en.wikipedia.org/>) that every request appends further detail too.
- A request will be to a specific Endpoint (/wiki/Web\_server) that when combined with the base address forms a URI – Unique Resource Identifier
- A complete request URI ([https://en.Wikipedia/wiki/web\\_Server](https://en.Wikipedia/wiki/web_Server))



# REQUEST HEADERS

- Headers provide extra detail to the webserver about the request
- Headers demonstrated when loading [https://en.wikipedia.org/wiki/Web\\_server](https://en.wikipedia.org/wiki/Web_server) in a browser:
- Headers can determine how a Server might handle the request

```
:authority: en.wikipedia.org
:method: GET
:path: /wiki/Web_server
:scheme: https
accept: text/html,application/xhtml+xml,application/signed-exchange;v=b3;q=0.9
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
cache-control: max-age=0
```

Headers starting with ':' are new to HTTP2, and replace presenting this information in a single string

**Accept:** the content the client is happy to receive in a response, these are listed as MIME Types. The server matches an appropriate MIME Type and responds with that format

Http Headers Reference:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

# HTTP REQUEST METHODS

- **GET** – used to describe a request that is asking to receive information
  - [https://en.wikipedia.org/wiki/Web\\_server](https://en.wikipedia.org/wiki/Web_server) - GET will return the webpage
- **POST** – used to describe a method that is sending information – these requests will generally contain a body and a less specific Request URI
- **PUT** – used to describe a method that is updating information – will contain a body (the data to be updated) and a very specific URI (the location of the existing data to be updated)
- **DELETE** – used to signal to the webserver that the specified resource (via the URI) should be deleted

# HTTP & HTTPS

- HTTP(Hypertext Transfer Protocol) is an application protocol designed to define a standardised way of communicating between different applications, servers, and clients on the internet.
- As a request-response protocol, HTTP gives users a way to interact with web resources such as HTML files by transmitting hypertext messages between clients and servers.
- HTTP utilizes specific request methods in order to perform various tasks.
- HTTPS (Hypertext Transfer Protocol Secure) layers additional security onto the system by encrypting data transmissions to prevent data from being compromised.
- It is recommended to use HTTPS instead of HTTP whenever you can.



# REQUEST CONTENT

- A PUT or POST request may contain data in a specific format or (MIME Type) that can be interpreted by the server
- A GET request *can* contain a body, but it is bad practice to do so
- A GET Request can include query strings to provide more detail in a request (if the webserver accepts query parameters)



# QUERY/URL PARAMETERS

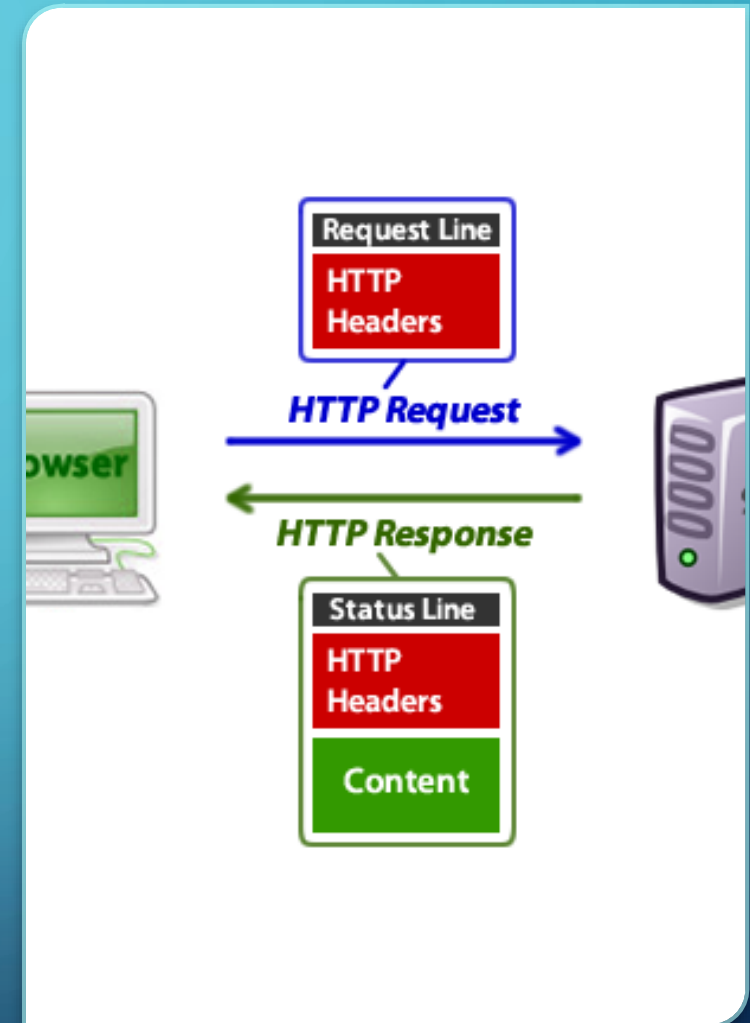
- Query parameters can provide more detail with a GET request:
- Using Wikipedias search functionality to search for 'Query Parameters':
  - <https://en.wikipedia.org/w/index.php?search=query+parameters&title=Special%3ASearch&go=Go&ns0=1>
  - The basaddress + endpoint are: <https://en.wikipedia.org/w/index.php>
  - ? Is used to signify a parameter, & is used to include (and separate) another parameter
  - The first parameter is: [search=query+parameters](#)
  - The second is: [title=Special%3ASearch](#)
  - The third is: [go=Go](#)
  - The fourth is: [ns0=1](#)

# URL ENCODING

- URL - Uniform Resource Locator can only contain characters in the ASCII character
- Non-ASCII characters must be encoded to allow for transmission
- Encoding replaces a character with a '%' symbol and a hexadecimal representation of the original character (Example: %24 is the encoding for the \$)
- Spaces are also encoded to either a '+' or '%20'
- A GET Request can include query strings to provide more detail in a request (if the webserver accepts query parameters)
- Url Encoding:[https://www.w3schools.com/tags/ref\\_urlencode.ASP](https://www.w3schools.com/tags/ref_urlencode.ASP)
- <https://en.wikipedia.org/wiki/Percent-encoding>

# RESPONSE

- Once a request has been sent to a server, response from the server will be sent to the client.
- A response is a combination of:
  - A Status Code – indicating the outcome of the request
  - Response Headers – providing detail around the response and metadata for any content returned from the server
  - Content – any content requested from the server, navigating to a website returns HTML content to a browser



# STATUS CODES

- Response codes are used to describe any event during the request -> response cycle and are categorized by 5 ranges:
  - 1xx – Information, not success or failure
  - 2xx – Success, 200 represents OK, 201 represents Created
  - 3xx – Redirection, when there is a system redirecting the request/response
  - 4xx – Client Error – an issue with the request
    - Indicates an incorrect, or poorly formatted request
  - 5xx – Server Error – an issue with processing the request and sending the response
    - Can indicate a correct request, but a fault in the server that has prevented the request from being processed
- 418 – I'm a teapot <https://datatracker.ietf.org/doc/html/rfc2324>

# RESPONSE HEADERS

- Headers provide extra detail to the client about the response
- Headers demonstrated when receiving a response from [https://en.wikipedia.org/wiki/Web\\_server](https://en.wikipedia.org/wiki/Web_server) in a browser:
- Response Headers describe the server, and the content returned

## ▼ Response Headers

```
age: 80384
cache-control: private, s-maxage=0, max-age=0, must-revalidate
content-encoding: gzip
content-language: en
content-type: text/html; charset=UTF-8
date: Mon, 10 Jan 2022 08:47:01 GMT
last-modified: Mon, 10 Jan 2022 08:42:20 GMT
```

**Age:** The time, in seconds, that the object has been in a proxy cache.

**Cache-control:** specifies whether the response can be cached, how long for and any action to take when the cached data age is greater than the set max-age

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

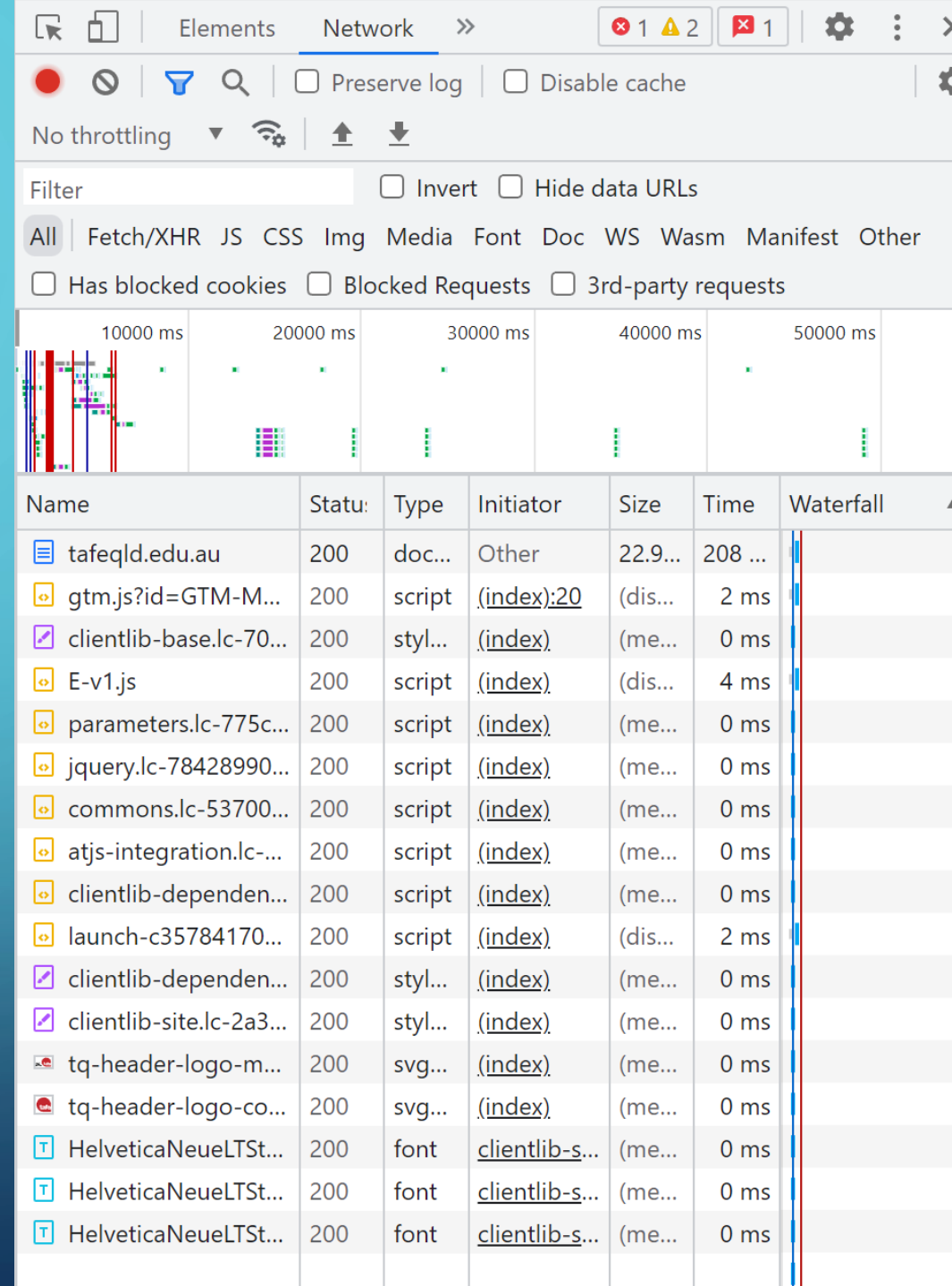


# CONTENT

- Acceptable content format is defined in the Request headers, while the returned format is described in the Response headers.
- Will conform to a MIME type (Multipurpose Internet Mail Extensions)
  - [https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics\\_of\\_HTTP/MIME\\_types/Common\\_types](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types)
- Primary (default) MIME types are:
  - text/plain – for anything in a textual format
  - Application/octet-stream – for anything that the browser should not execute directly (application files, binary data)
- Web Pages are generally returned as 'text/html' while structured application data (POCO's, Data Models) can be returned as XML or JSON application/xml + application/json
- Content can be Encoded, or compressed to reduce file size, but the type of encoding or compression must be specified using the **Accept-Encoding** and **Content-Encoding** Headers

# VIEWING REQUEST / RESPONSE DETAILS

- Chrome Dev tools can show request / response data
  - F12 or right click -> inspect
- Selecting the Network Tab and refreshing a web page shows relevant request / response data
- Each individual file listed has been sent from the server as a HTTP response, triggered by request



The screenshot shows the Chrome DevTools Network tab. At the top, there are tabs for 'Elements' and 'Network'. The 'Network' tab is active, showing a list of resources and a waterfall chart. The resources list includes various files like 'tafeqld.edu.au', 'gtm.js?id=GTM-M...', 'clientlib-base.lc-70...', 'E-v1.js', 'parameters.lc-775c...', 'jquery.lc-78428990...', 'commons.lc-53700...', 'atjs-integration.lc-...', 'clientlib-dependen...', 'launch-c35784170...', 'clientlib-dependen...', 'clientlib-site.lc-2a3...', 'tq-header-logo-m...', 'tq-header-logo-co...', and three instances of 'HelveticaNeueLTSt...'. The waterfall chart shows the timing of these requests.

Name	Status	Type	Initiator	Size	Time	Waterfall
tafeqld.edu.au	200	doc...	Other	22.9...	208 ...	
gtm.js?id=GTM-M...	200	script	(index):20	(dis...	2 ms	
clientlib-base.lc-70...	200	styl...	(index)	(me...	0 ms	
E-v1.js	200	script	(index)	(dis...	4 ms	
parameters.lc-775c...	200	script	(index)	(me...	0 ms	
jquery.lc-78428990...	200	script	(index)	(me...	0 ms	
commons.lc-53700...	200	script	(index)	(me...	0 ms	
atjs-integration.lc-...	200	script	(index)	(me...	0 ms	
clientlib-dependen...	200	script	(index)	(me...	0 ms	
launch-c35784170...	200	script	(index)	(dis...	2 ms	
clientlib-dependen...	200	styl...	(index)	(me...	0 ms	
clientlib-site.lc-2a3...	200	styl...	(index)	(me...	0 ms	
tq-header-logo-m...	200	svg...	(index)	(me...	0 ms	
tq-header-logo-co...	200	svg...	(index)	(me...	0 ms	
HelveticaNeueLTSt...	200	font	clientlib-s...	(me...	0 ms	
HelveticaNeueLTSt...	200	font	clientlib-s...	(me...	0 ms	
HelveticaNeueLTSt...	200	font	clientlib-s...	(me...	0 ms	



QUESTIONS?

