Edinburgh Napier
UNIVERSITY

ADVANCED WEB TECHNOLOGIES SET09103 LECTURE 02 (WEEK 2)

HTTP

Dr Simon Wells
s.wells@napier.ac.uk
http://www.simonwells.org

TL/DR

- HTTP is an agreement for text-based communication between clients & servers for retrieving, altering, & deleting resources (amongst other things)

- *The* basic protocol that underpins the web

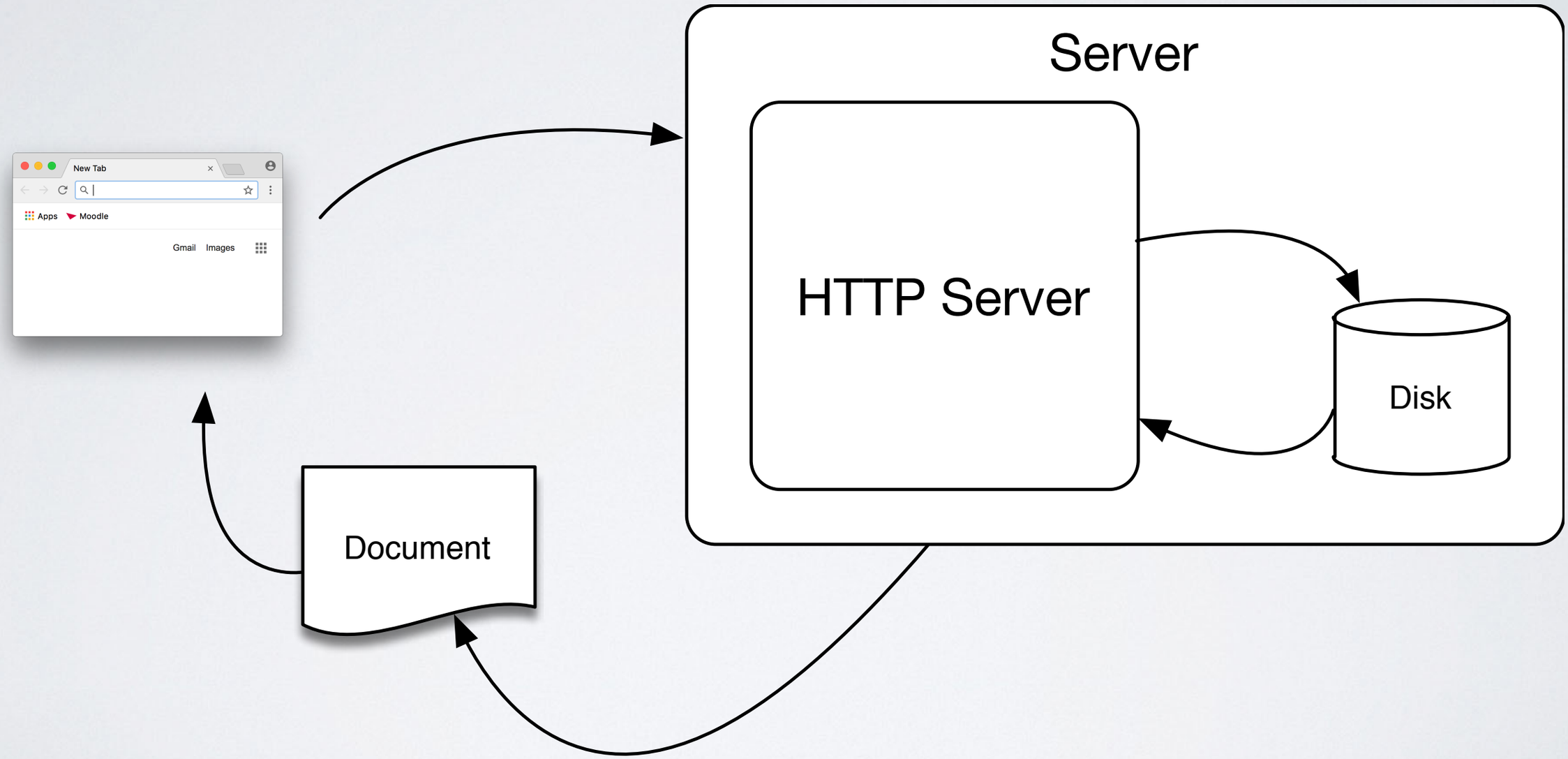
Edinburgh Napier UNIVERSITY

OVERVIEW

- Hypermedia & hypertext
- The HTTP protocol:
  - Basics,
  - Statelessness,
  - Networking (relationship to TCP/IP)
  - Requests & Responses
  - URLs
  - HTTP Verbs
  - HTTP Status Codes
  - HTTP Message Formats

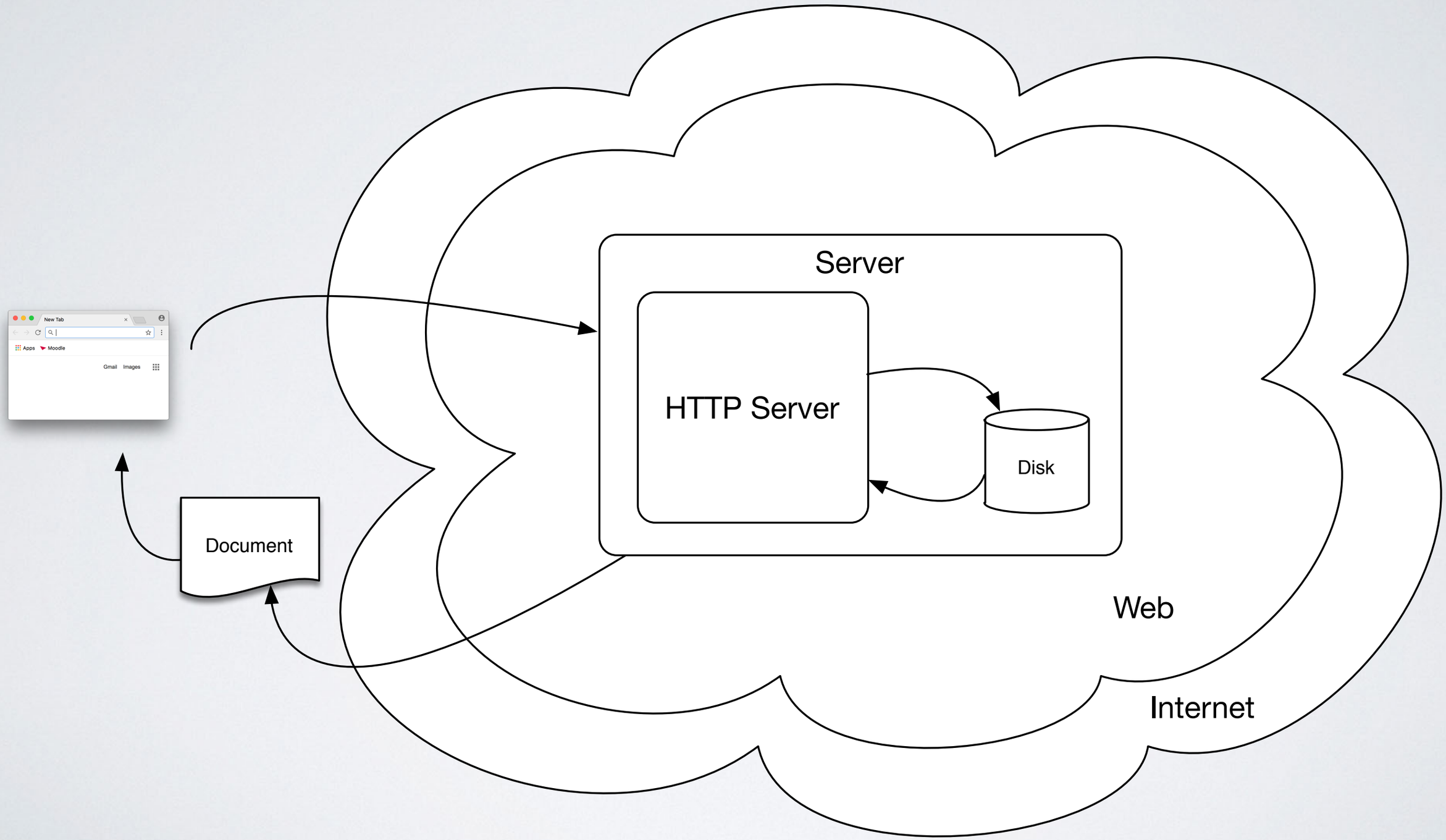
CLIENT SERVER ARCHITECTURES

Edinburgh Napier UNIVERSITY



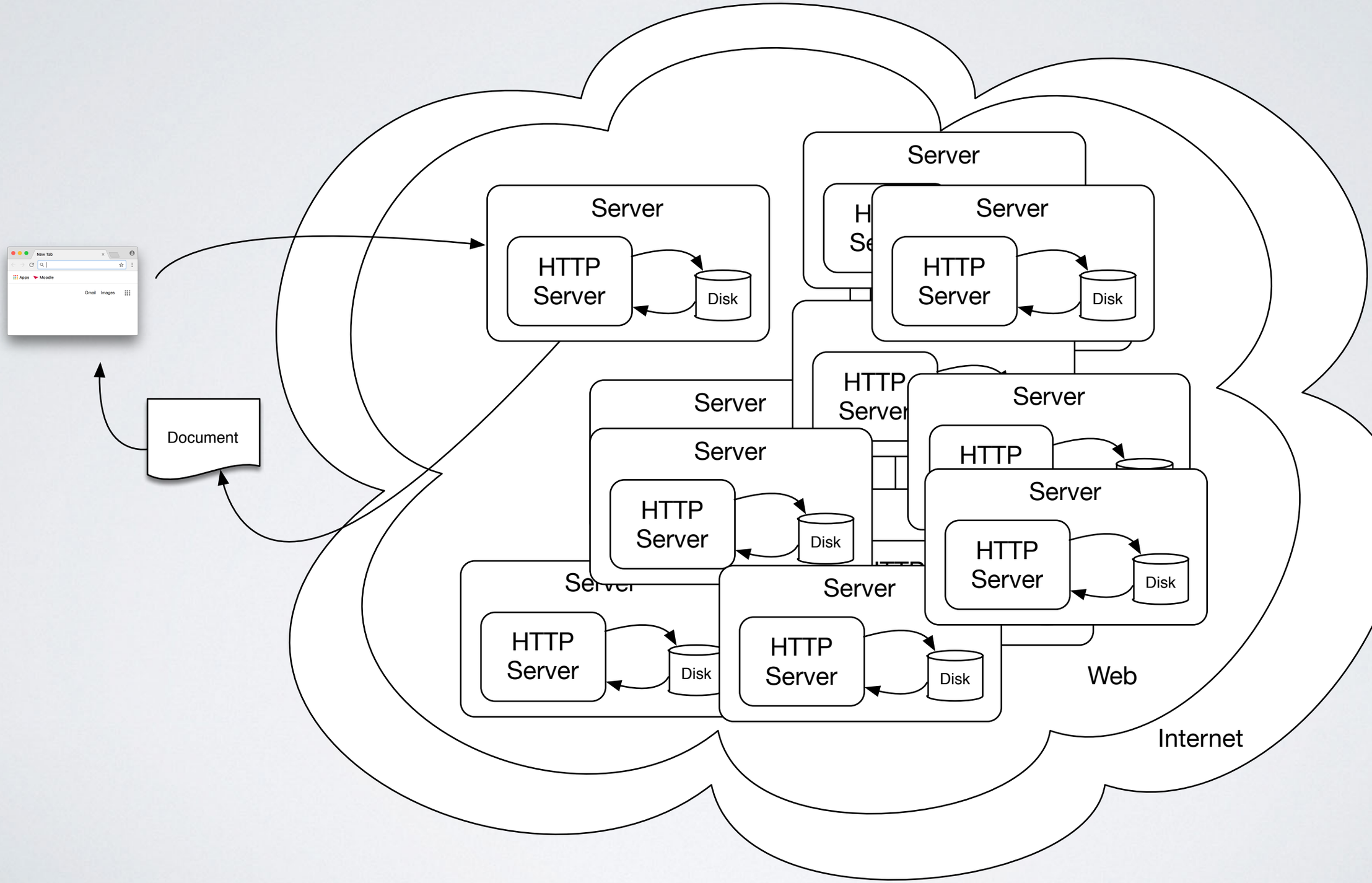
Edinburgh Napier
UNIVERSITY

WEB ARCHITECTURE



Edinburgh Napier UNIVERSITY

WEB SCALE



Edinburgh Napier UNIVERSITY

NOT JUST WEB SERVERS…

- Actually more complicated than just a bunch of web servers:
  - Each server may actually be part of a cluster of machines with incoming messages distributed between them (load balancing
  - Datastores
  - Additional protocol servers: Web server might palm off certain jobs to dedicated machines, e.g. media streaming
  - Plus: Web caches, DNS, Time servers — **anything else…?**

Edinburgh Napier UNIVERSITY

HYPERTEXT

- Text displayed on screen, e.g. computer, with references (hyperlinks) to other text - assumption is that the user can immediately follow the link or else reveal other text progressively
- Coined by Ted Nelson (Xanadu project) in 1963 (also coined related term hypermedia)
- Underlying concept defining structure of WWW
- Pages written in HTML contain hyperlinks to resources at other web locations (URLs)

> By now the word "hypertext" has become generally accepted for branching and responding text, but the corresponding word "hypermedia", meaning complexes of branching and responding graphics, movies and sound – as well as text – is much less used. Instead they use the strange term "interactive multimedia": this is four syllables longer, and does not express the idea of extending hypertext.
>
> — Nelson, *Literary Machines*, 1992

Edinburgh Napier UNIVERSITY

HYPERTEXT TRANSFER PROTOCOL (HTTP)

- Core protocol that underpins the World Wide Web
- Current version is **HTTP/1.1** - just adds a few extra features to v1.0 (e.g. persistent connections. fine-grained caching headers, chunked transfer-coding)
- Defines:
  - how messages (primarily text) are formatted and transmitted
  - what actions web-servers & browsers should perform in response to a given message
- *NB. Other core standard of the web is HTML*
- Stateless, application layer protocol for communicating between distributed systems

Edinburgh Napier UNIVERSITY

HTTP/2

- Mechanism for clients & servers to negotiate the specific protocol to use (e.g. HTTP/1.1, HTTP/2.0, or others)
- High level compatibility with HTTP/1.1 (methods, status codes, URIs, most header fields)
- Designed to decrease latency (time to load pages) by
  - Binary protocol & header compression
  - Server Push
  - Pipelining & Multiplexing requests over TCP
- Supported by major browsers & servers & makes up ~40% of web sites

STANDARDS

- HTTP/1/1 Originally defined in RFC2616 (1999)]
  - Now a series of RFC: 723x
    - RFC7230: HTTP/1.1, part 1: Message Syntax and Routing
    - RFC7231: HTTP/1.1, part 2: Semantics and Content
    - RFC7232: HTTP/1.1, part 4: Conditional Requests
    - RFC7233: HTTP/1.1, part 5: Range Requests
    - RFC7234: HTTP/1.1, part 6: Caching
    - RFC7235: HTTP/1.1, part 7: Authentication
  - HTTP/2 is RFC 7540
- Developed & coordinated by the Internet Engineering Task Force (IETF) & the World Wide Web Consortium (W3C)
- Process of discussion culminates in Request For Comments (RFC) documents

Edinburgh Napier UNIVERSITY

HTTP BASICS

- Allows for communication between hosts and clients and supports many network configurations
  - How? By not assuming very much about any particular underlying system & not keeping state (stateless)
- Communication uses TCP/IP (but can work over any reliable transport) and runs, by default, over port **80**
  - *NB. Other ports can be used, e.g. we are using 5000 for our web-apps during development*

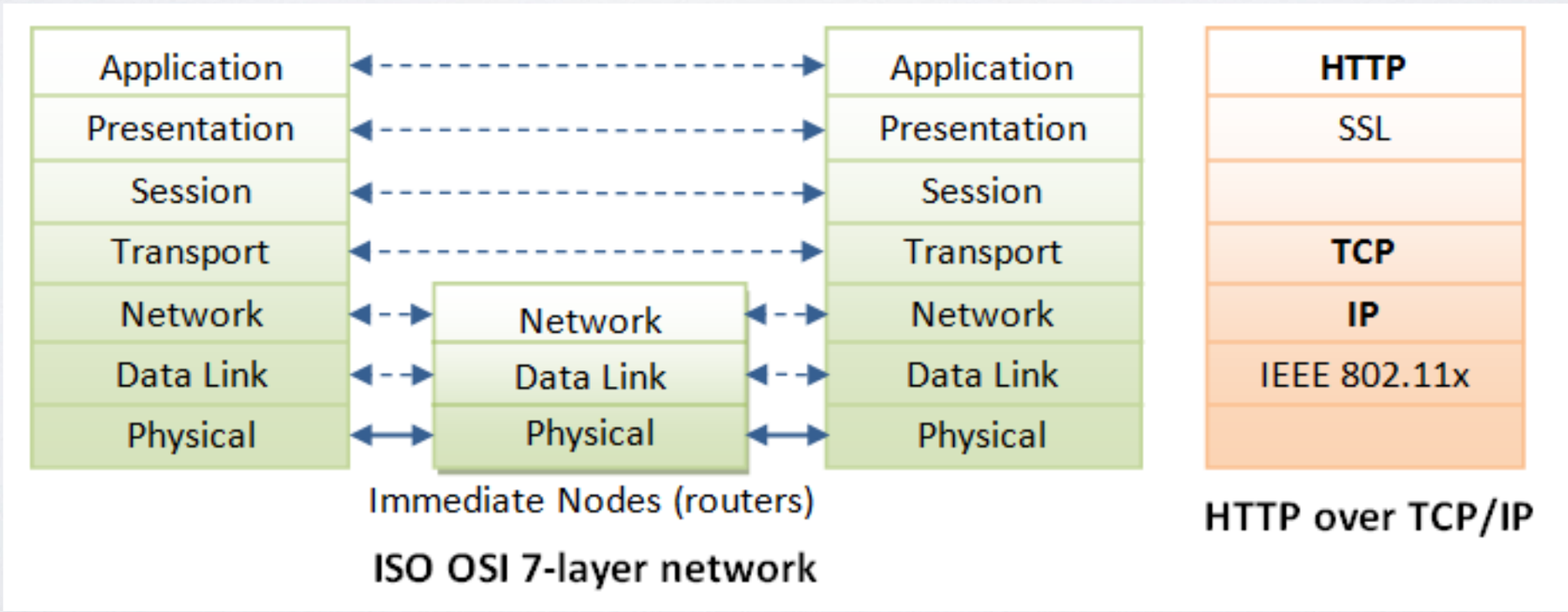
Edinburgh Napier UNIVERSITY

STATELESS PROTOCOL

- Intrinsically stateless
- Each request is processed without any knowledge of previous pages requested
- *Both advantage & disadvantage*
- Maintaining state is very useful: remember what you were doing before, personalise experience - change web to fit your habits, experience, & expectations
- Many stateful alternatives have been produced:
    - Alternative server APIs, e.g. NSAPI - Netscape Server API, ISAPI - MIcrosoft IIS Server API, Cookies
- & work-arounds:
    - e.g. URL encoding (& other methods for client to provide state information back to server)
    - XMLHTTPRequest (AJAX), Java, Javascript, ActiveX, web-sockets, &c.

RELATIONSHIP TO WIDER NETWORK STACK

- HTTP operates at the Application layer (the top layer)
- Doesn't care too much about what is below so long as it can transmit data and is reliable

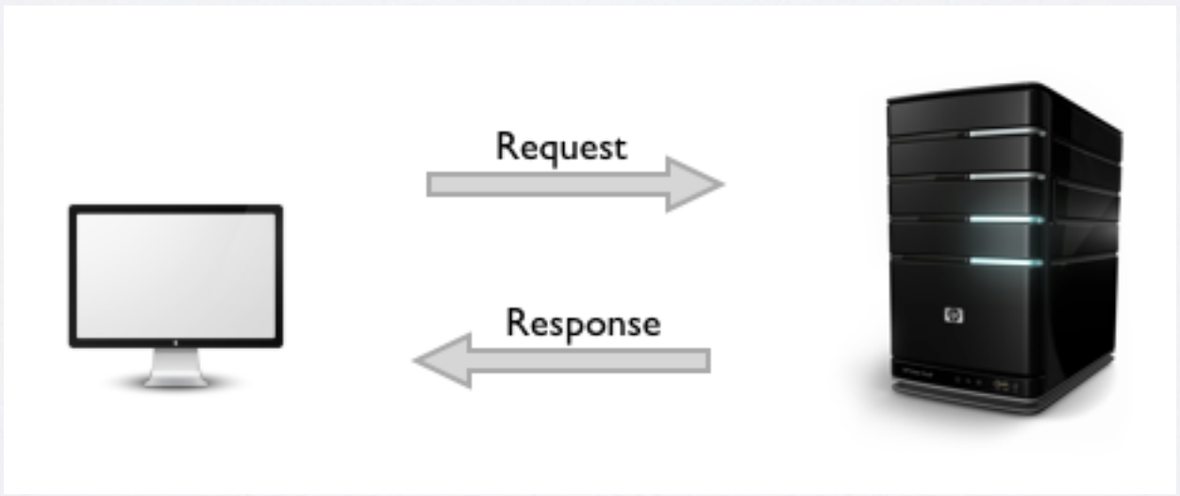


Edinburgh Napier UNIVERSITY

REQUESTS & RESPONSES

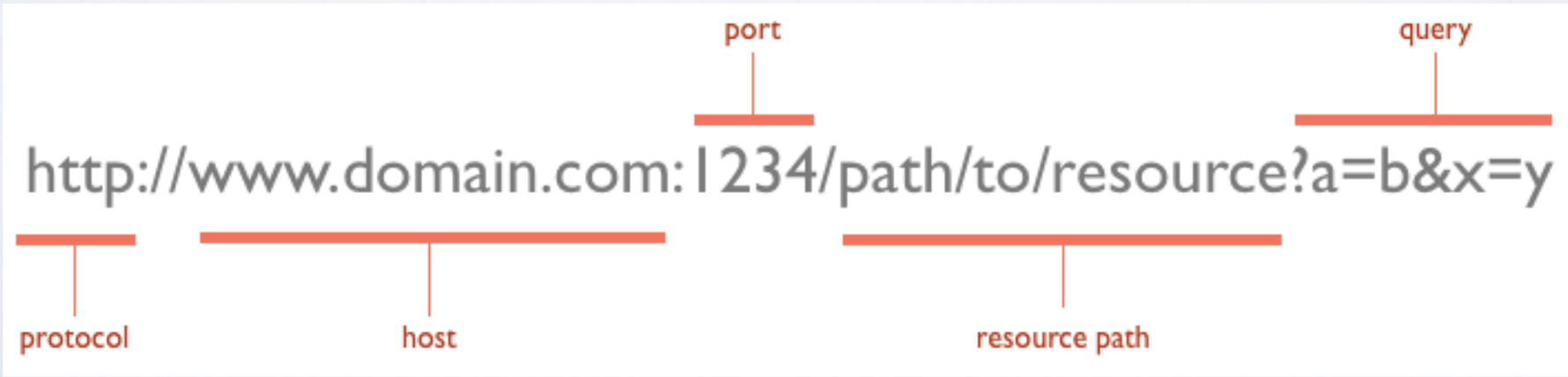
- Communication occurs between a **host (server)** and a **client** via a **request/response** pair

- Client initiates HTTP **request** message and the host *services* the request and returns a **response** message



URLS

- The request message is the heart of communications via the web
- Requests are sent via **Uniform Resource Locators (URLs)**
- **Protocol** is usually http (but can also be https)
- **Port** defaults to 80 but can be set explicitly in the URL
- **Resource Path** is the local path to the resource on the server



Edinburgh Napier UNIVERSITY

HTTP VERBS

- URLs identify resources on the server with which we can interact
- By default we **GET** resources (Web as data retrieval)
- Verbs describe actions so verbs are used to define the ways in which an HTTP host can be interacted with
- HTTP formalises a set of verbs that capture the essentials of communication:. The four most popular are:
  - **GET** fetch an existing resource
  - **POST** - create a new resource. Post requests always carry a payload that specifies the data for the new resource
  - **PUT** - update an existing resource. Payload must contain the updated data for the resource
  - **DELETE** - remove the resource
- Browsers support GET & POST. PUT & DELETE are ofter repackaged as POST requests where the payload makes explicit the action to perform, e.g. creating, updating, or deleting

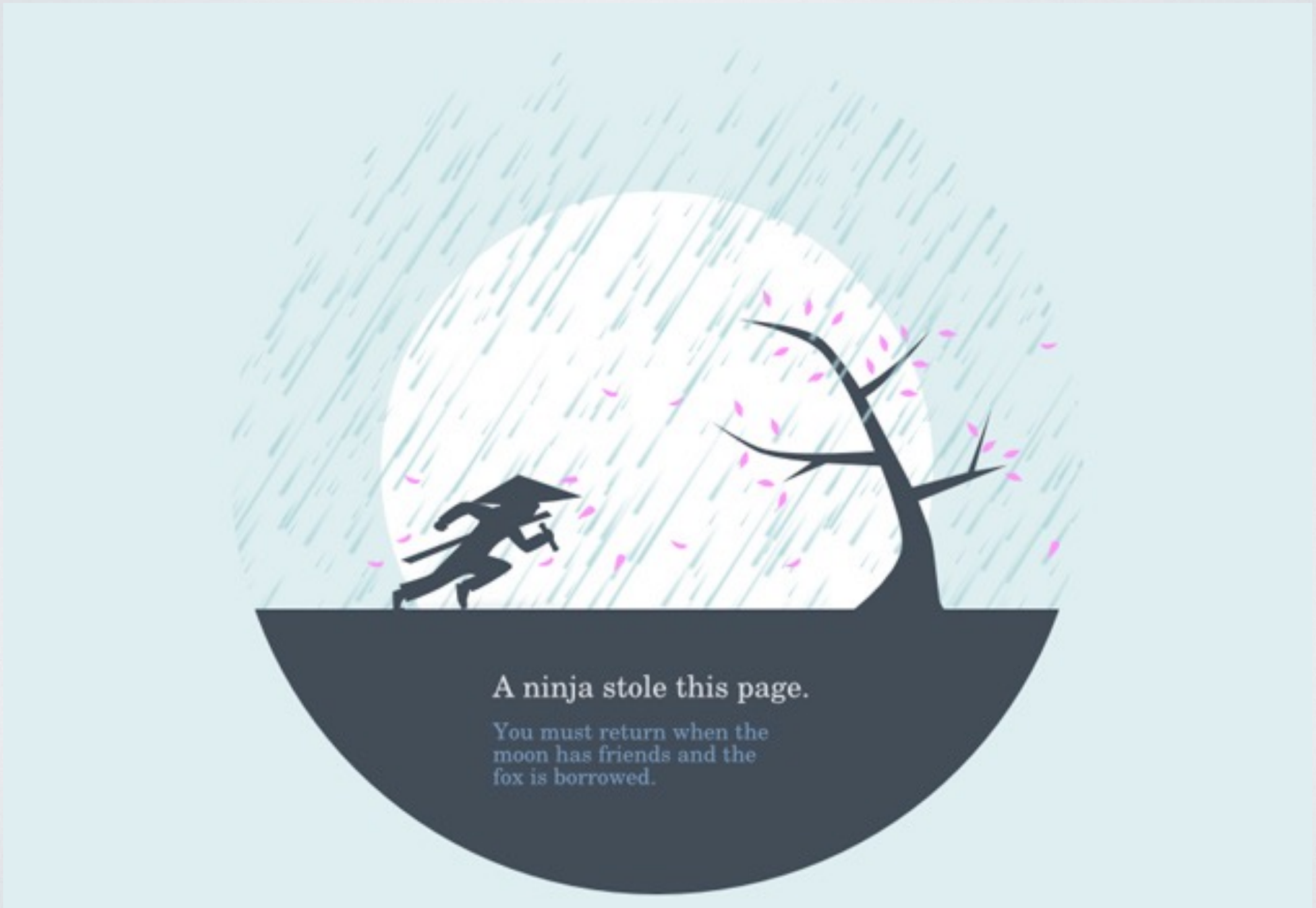
Edinburgh Napier UNIVERSITY

OTHER HTTP VERBS

- Most APIs will support GET, POST, PUT, DELETE
    - *Why support verbs that browsers can't make use of?*
- Other verbs are also available, e.g.
- ***HEAD*** - The same as GET but retrieves just the server headers for the resource without the message body - useful to check whether a large resources has changed (using timestamps) so that you don't retrieve it all
- ***TRACE*** - retrieve hops that the request made during round trip to server, e.g. each intermediate proxy or gateways injects its IP address of DNS name into the *Via* field of the header (useful for diagnostics)
- ***OPTIONS*** - to get information about the host/API capabilities
- URLs+Verbs provide a good proportion of an API for the host that we are trying to communicate with

Edinburgh Napier
UNIVERSITY

RESPONSE/STATUS CODES



Edinburgh Napier UNIVERSITY

HTTP STATUS CODES

- Make request to a server, server processes request, returns response & status code

- Request + Verb > Response + Status Code

- Status Code is important - it tells us how to interpret the response from the server

Edinburgh Napier
UNIVERSITY

2XX: SUCCESSFUL

- Used by server to tell the client that the request was successfully processed.
- **200 OK** - Probably the most common code (except for 404) but as users we don't notice it when things are going well
- **202 Accepted** - the request was accepted but may not include the resource in the response. This is useful for async processing on the server side. The server may choose to send information for monitoring.
- **204 No Content** - there is no message body in the response.
- **205 Reset Content** - indicates to the client to reset its document view.
- **206 Partial Content** - indicates that the response only contains partial content. Additional headers indicate the exact range and content expiration information.

Edinburgh Napier
UNIVERSITY

3XX: REDIRECTION

- Require the client to take additional action
  - e.g. request a different url if the location for a resource has changed
- Help provide stability if used correctly
- **301 Moved Permanently** - the resource is now located at a new URL.
- **303 See Other** - the resource is temporarily located at a new URL. The Location response header contains the temporary URL.
- **304 Not Modified** - the server has determined that the resource has not changed and the client should use its cached copy. This relies on the fact that the client is sending ETag (Entity Tag) information that is a hash of the content. The server compares this with its own computed ETag to check for modifications.

Edinburgh Napier
UNIVERSITY

4XX CLIENT ERROR

- Used when server believes the client is at fault, e.g. requesting an invalid resource or making a bad request
- **400 Bad Request** - the request was malformed.
- **401 Unauthorized** - the request requires authentication. The client can repeat the request with the Authorization header. If the client already included the Authorization header, then the credentials were wrong.
- **403 Forbidden** - the server has denied access to the resource.
- **404 Not Found** - the resource is invalid and does not exist on the server
- **405 Method Not Allowed** - invalid HTTP verb used in the request line, or the server does not support that verb.
- **409 Conflict** - the server could not complete the request because the client is trying to modify a resource that is newer than the client's timestamp. Conflicts arise mostly for PUT requests during collaborative edits on a resource.

Edinburgh Napier UNIVERSITY

5XX: SERVER ERROR

- Class of codes that indicate a server failure whilst processing the request

- **500 Internal Server Error** - general "something bad happened" code

- **501 Not Implemented** - server doesn't yet support the requested functionality

- **503 Service Unavailable** - something on the server has failed or the server is overloaded (*NB. often the server won't even respond & the request will timeout. Timeout can usually be interpreted as a 503*)

1XX: INFORMATIONAL MESSAGES

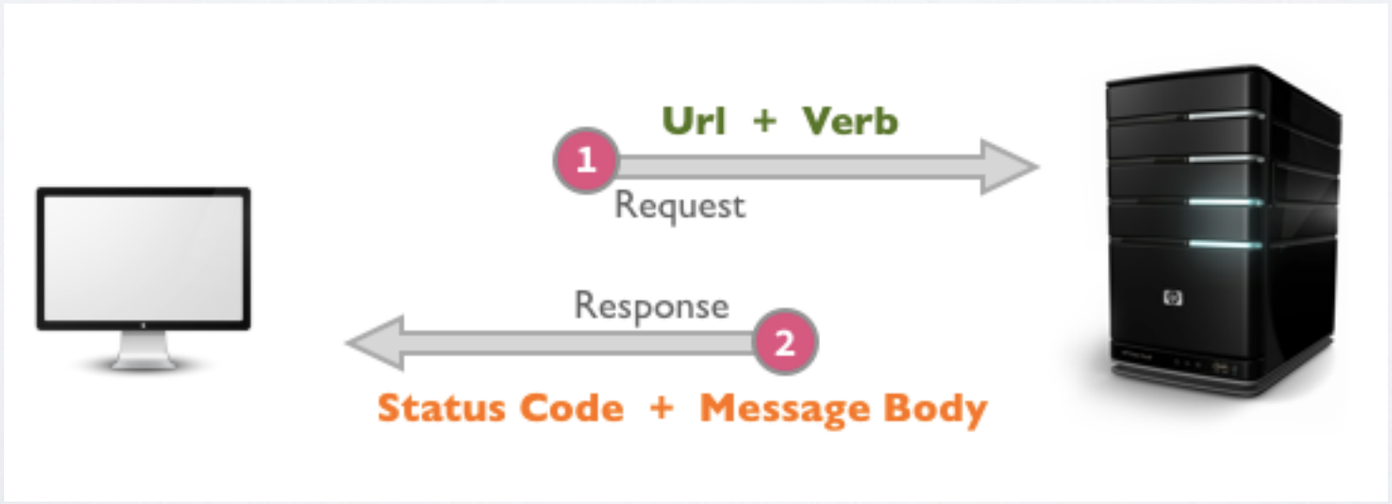
Edinburgh Napier UNIVERSITY

- Introduced in HTTP/1.1
- Purely provisional
- Ignored by HTTP/1.0 clients
- **100 Continue** - tell the client to continue sending the remainder of the request or ignore if it has already been sent

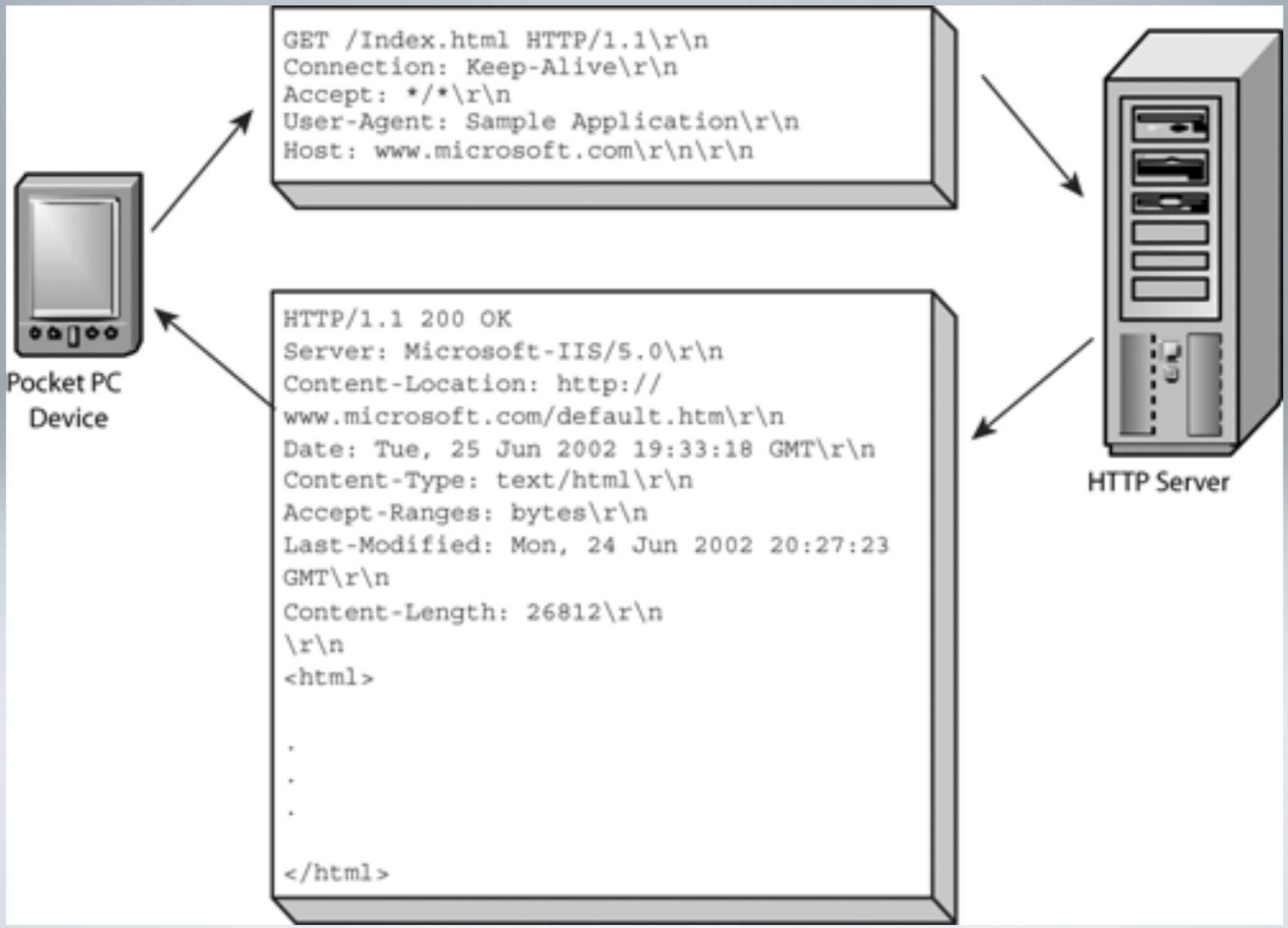
Edinburgh Napier UNIVERSITY

HTTP COMMUNICATION

- URLs + verbs + status codes are the fundamental parts that make up a request/response pair
- But what do requests & responses look like?



MESSAGE FORMATS



Edinburgh Napier UNIVERSITY

HTTPS

- Extension to HTTP for **secure** communication on the Internet
- The connection between the client & server is **encrypted** at the **transport layer** - can see where connections start & end but not there content - hence TLS (transport layer security)
- NB. Used to be Secure Sockets Layer (SSL)
- AIM:
  - Authentication of the site you're accessing
  - Protection of data transported between you & site (& vice versa)
    - i.e. prevent someone from eavesdropping on your messages or altering them en route

Edinburgh Napier UNIVERSITY

HTTP STRICT TRANSPORT SECURITY (HSTS)

- Attempt to force greater security by preventing a class of attacks on HTTP/HTTPS sites
- Protects against protocol downgrade attacks/SSL-stripping/Man-in-the-middle attacks
- Because many sites use both HTTP & HTTPS it can be unclear whether the connection should be secure or not.
- Web server declares that it will **only** allow interaction using HTTPS connections and will **never** drop back to plain HTTP
- Communicated to client in HTTP response header field named "Strict-Transport-Security"

Edinburgh Napier UNIVERSITY

WRAPPING UP

- Hypermedia & hypertext
- The HTTP protocol:
  - Basics,
  - Statelessness,
  - Networking (relationship to TCP/IP)
  - Requests & Responses
  - URLs
  - HTTP Verbs
  - HTTP Status Codes
  - HTTP Message Formats
  - HTTPS/HSTS