University of Sheffield

COM1001 SPRING SEMESTER

Professor Phil McMinn

p.mcminn@sheffield.ac.uk

Web Servers and HTTP (The HyperText Transfer Protocol)

What is a Web Server?

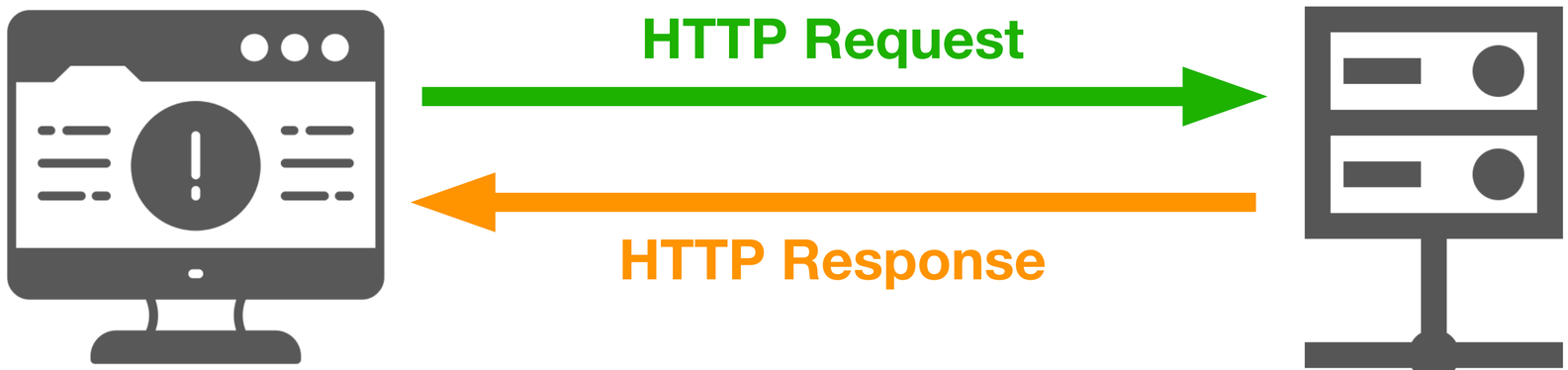
A **web server** is a **computer program** that **runs continuously** on a machine connected to the internet.

Its job is to **respond to requests for web pages from a web browser being used on another machine** somewhere else on the Internet.

It then sends the web pages to that web browser.

Web browsers and web servers have an agreed method of communicating with one another, called the HyperText Transfer Protocol (HTTP).

A user's **web browser** communicates with the **web server** hosting the page using an exchange of "**requests**" and "**responses**".



Device with web browser **Web server with web site**

The Path of Browser Request

1. The **browser** performs DNS Lookup for the **web server** based on its **domain name** (e.g. `www.sheffield.ac.uk`),
2. The **browser** sends an **HTTP Request** to the **web server**
3. The **web server** sends a **HTTP Response**, with the requested HTML file
4. The **browser** begins to render HTML
5. The **browser** sends additional requests for objects embedded in the HTML file (CSS files, images, JavaScript, etc.)

1 DNS Lookup

The web browser needs to find where the web server lives on the internet so that it can route its request.

It does this based on the domain name of the website, given in the URL of the web page.

The domain name is the the part of the URL after the `http://` or `https://` and before the next forward slash, e.g. `www.sheffield.ac.uk`

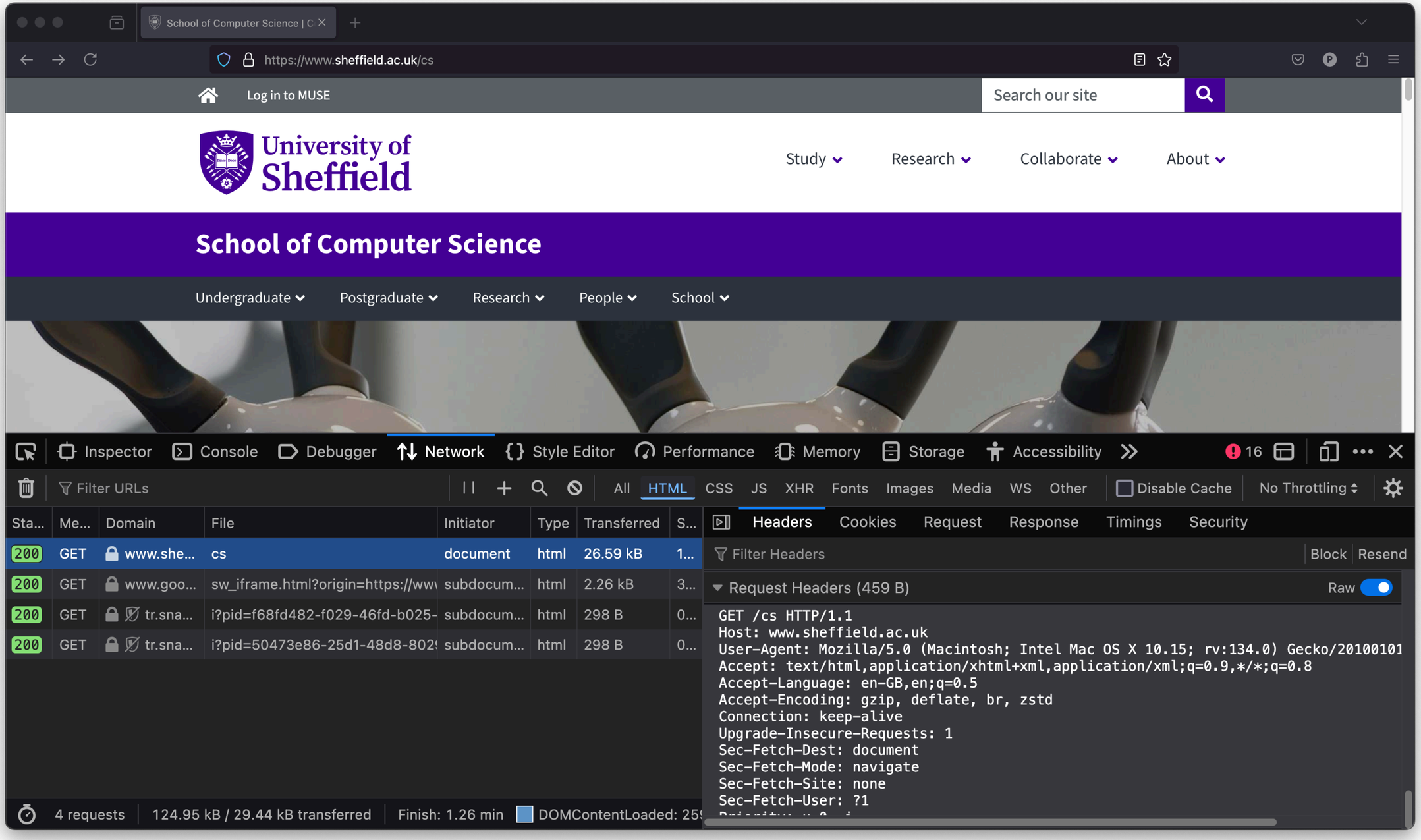
The browser then needs to convert this domain name to its **Internet Protocol address** – `143.167.2.102`

IP addresses are how computers locate each other on the Internet – domain names are a human convenience. A **domain name** like "`www.sheffield.ac.uk`" is easier for humans to remember than `143.167.2.102`

2 Browser sends HTTP Request

Once it has the IP address of the web server, a browser can now send that server a **HTTP Request**

Having the IP address means the request be faithfully routed to the server over the Internet (a bit like how the postal service works with letters and parcels).


School of Computer Science | C
https://www.sheffield.ac.uk/cs
Log in to MUSE
Search our site
University of Sheffield
Study
Research
Collaborate
About
School of Computer Science
Undergraduate
Postgraduate
Research
People
School
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
16
Filter URLs
All
HTML
CSS
JS
XHR
Fonts
Images
Media
WS
Other
Disable Cache
No Throttling
Sta...
Me...
Domain
File
Initiator
Type
Transferred
S...
200 GET www.she... cs document html 26.59 kB 1...
200 GET www.goo... sw_iframe.html?origin=https://ww subdocum... html 2.26 kB 3...
200 GET tr.sna... i?pid=f68fd482-f029-46fd-b025- subdocum... html 298 B 0...
200 GET tr.sna... i?pid=50473e86-25d1-48d8-802 subdocum... html 298 B 0...
Headers
Cookies
Request
Response
Timings
Security
Filter Headers
Block
Resend
Request Headers (459 B)
Raw
GET /cs HTTP/1.1
Host: www.sheffield.ac.uk
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:134.0) Gecko/20100101
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
4 requests
124.95 kB / 29.44 kB transferred
Finish: 1.26 min
DOMContentLoaded: 25

```
GET /cs HTTP/1.1
Host: www.sheffield.ac.uk
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:134.0) Gecko/20100101 Firefox/134.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
```

The first part of the first line of a **HTTP Request** is the **HTTP method**.

The **HTTP method** defines the type of request being made and therefore how the server will interpret it. The most important **HTTP methods** are `GET` and `POST`.

```
GET /cs HTTP/1.1
Host: www.sheffield.ac.uk
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:134.0) Gecko/20100101 Firefox/134.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
```

Secondly, we have the **resource identifier** of the **resource** being requested, which could be a web page, or any type of file (such an image, script, or document).

```
GET /cs HTTP/1.1
Host: www.sheffield.ac.uk
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:134.0) Gecko/20100101 Firefox/134.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
```

The **HTTP Request** contains a series of **Request Headers**

Most of these are not particularly important for this module.

3 Web Server sends HTTP Response

Once the server receives a **HTTP Request**, it will respond in the form of a **HTTP Response**. How it does depends on a number of things.

For example, the resource (i.e., a web page) may not actually exist.

The response consists of a series of headers (like the request did), and the **body**, which contains the resource requested (e.g., the HTML of a web page).

Here's the initial part of the **HTTP response** Sheffield University's web server sent for the **HTTP request** for `http://www.sheffield.ac.uk/cs`

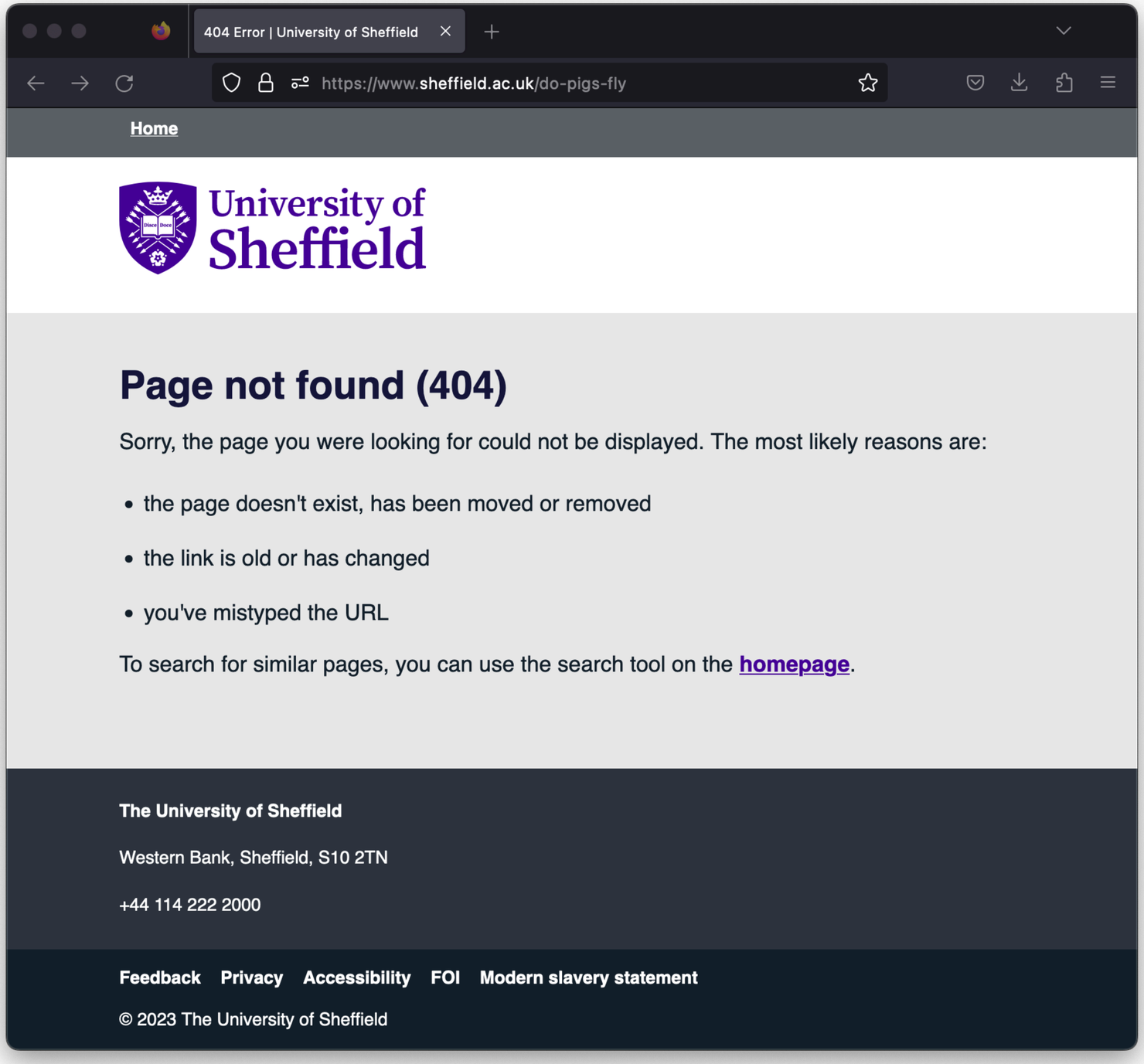
```
HTTP/3 200
accept-ranges: bytes
content-encoding: br
cross-origin-resource-policy: cross-origin
cross-origin-opener-policy: same-origin; report-to="analytics-container-tag-serving"
content-length: 1476
content-type: text/html
…
```

The most important part of the response headers is the **status code**. Ideally it sends a `200 OK`, which means success.

But it may send a `404 Not Found` or a `500 Internal Server Error`.

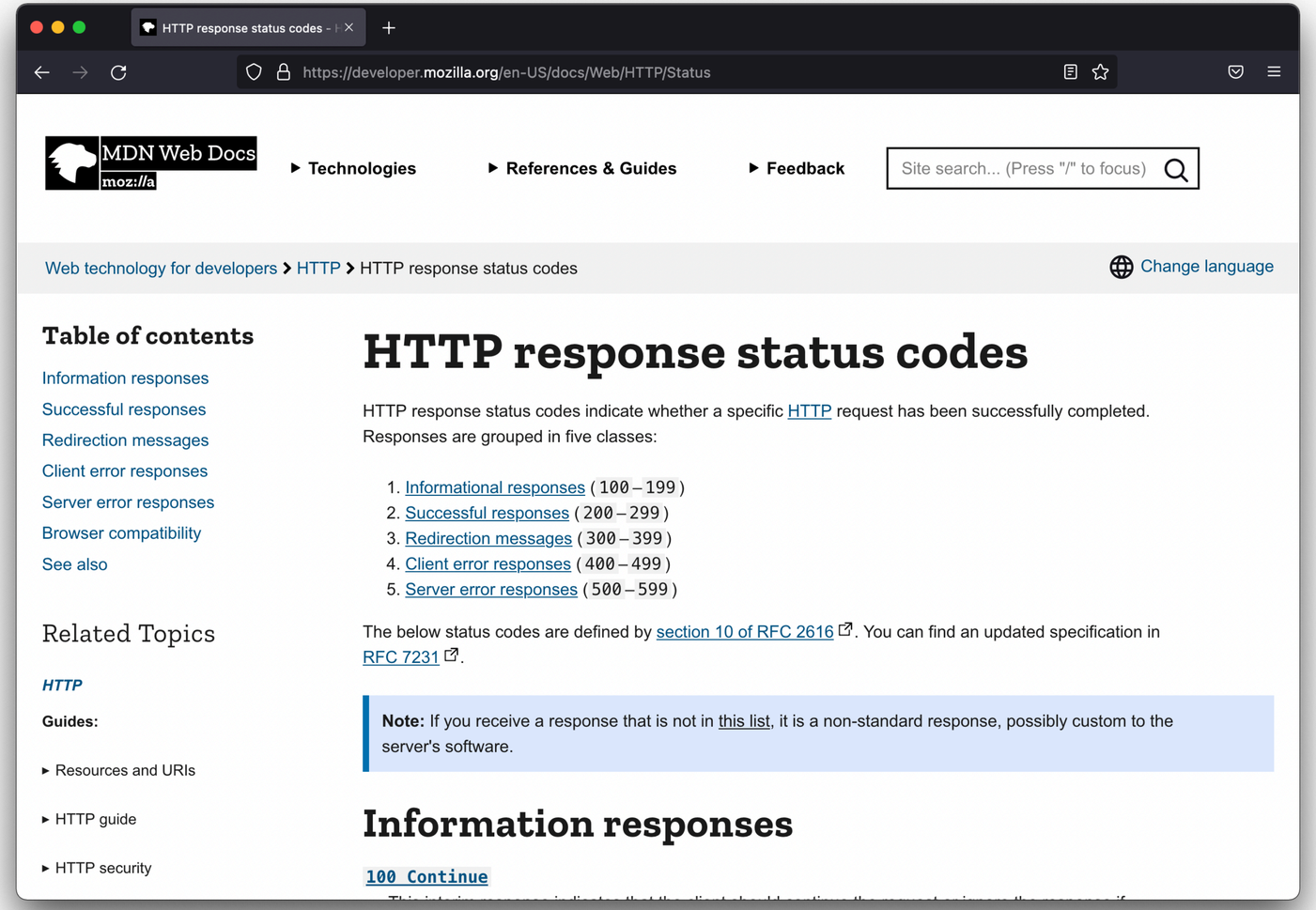
Both of these mean the requested resource cannot be sent.

```
HTTP/3 200
accept-ranges: bytes
content-encoding: br
cross-origin-resource-policy: cross-origin
cross-origin-opener-policy: same-origin; report-to="analytics-container-tag-serving"
content-length: 1476
content-type: text/html
…
```


404 Error | University of Sheffield
https://www.sheffield.ac.uk/do-pigs-fly
Home
University of Sheffield
Page not found (404)
Sorry, the page you were looking for could not be displayed. The most likely reasons are:
the page doesn't exist, has been moved or removed
the link is old or has changed
you've mistyped the URL
To search for similar pages, you can use the search tool on the homepage.
The University of Sheffield
Western Bank, Sheffield, S10 2TN
+44 114 222 2000
Feedback Privacy Accessibility FOI Modern slavery statement
© 2023 The University of Sheffield

More on Status Codes

developer.mozilla.org/en-US/docs/Web/HTTP/Status



Important HTTP Status Codes You Need to Know

200 (OK) – if the web application successfully processes the request

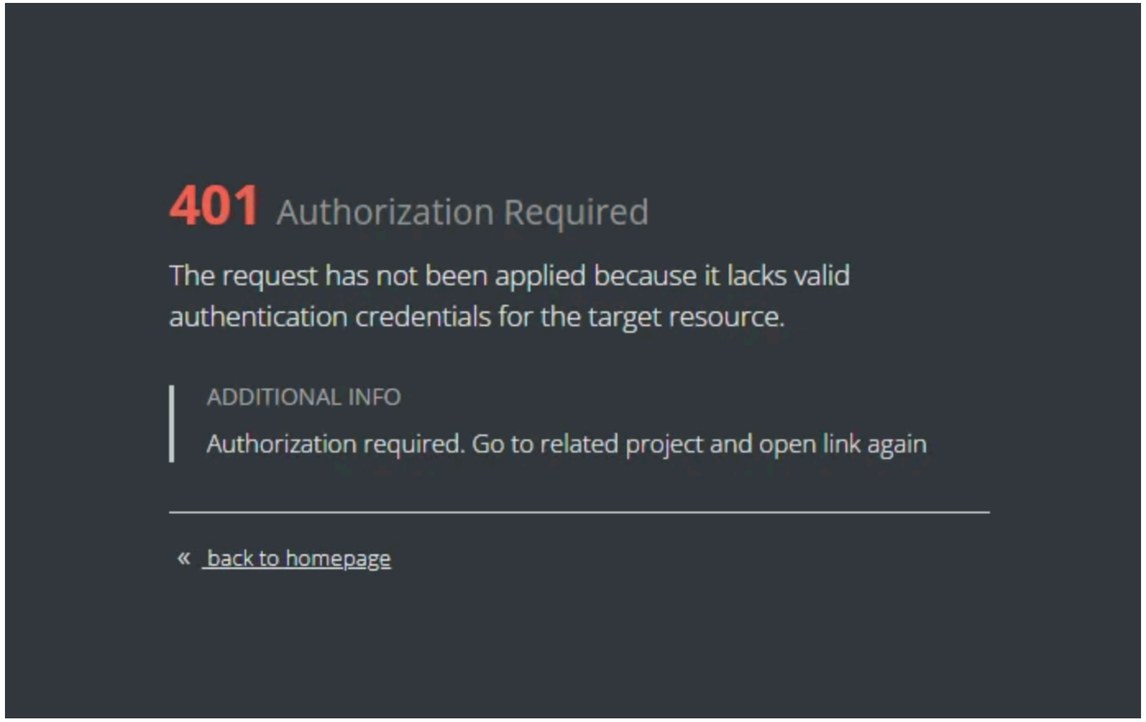
302 (Redirect) – if the web application re-directed the request (e.g., to an alternative resource ID)

404 (Not Found) – if the web application could not find the resource requested.

500 (Internal Server Error) – if the web application encountered an error while trying to process the request (e.g., its code contained a bug)

Some others

401 (Unauthorized error) – invalid credentials



Some others

418 (I’m a teapot) – the server refuses to brew coffee because it’s a teapot. Part of the **Hyper Text Coffee Pot Control Protocol**.

Quick Rules of Thumb

2xx Success codes (e.g., 200)

3xx Redirection codes (e.g., 302)

4xx Client error codes (e.g., 401, 404 …)

5xx Server error codes (e.g., 500)

See https://en.wikipedia.org/wiki/List_of_HTTP_status_codes for a full list

The Body of the HTTP Response: Static vs Dynamic Resources

Depending on the nature of the resource, there may be more work for the web server to do in generating the body of the HTTP response.

Static resources ***already exist*** before the request is made – e.g., an image files. Sometimes whole websites are static – the HTML pre-exists too. In this case, the server just needs to locate the file and send it to the browser.

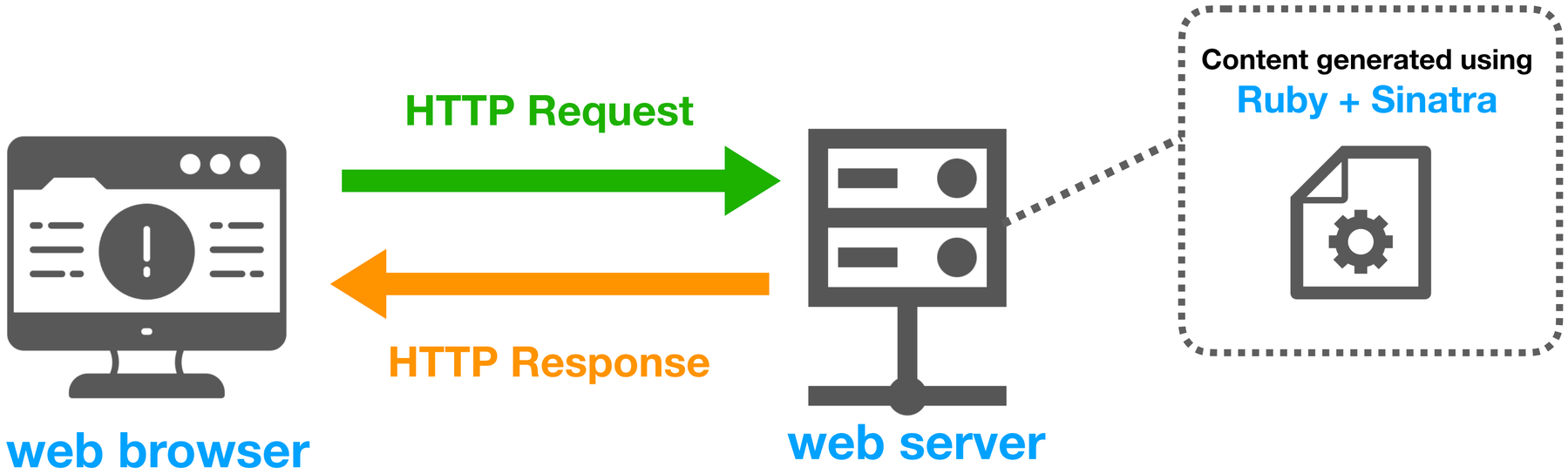
Dynamic resources ***are generated as a result of a request for them***. If a resource is dynamically generated, the **web server needs to execute some code to generate the content before it can send it.**

Dynamic Content Generation

If we can dynamically generate content, the web pages can respond to user actions and change the information appearing in the web page. This information could come from a **database**, for example.

In other words, we can write **web applications**.

In this module, we will be dynamically generating web content using **Ruby** and the help of a domain specific language called **Sinatra**.



4 The Browser Renders the HTML

Once the browser receives an HTML file, as part of the body of the HTTP Response, it can then process it and render it onto the screen.

5 The Browser Sends Additional Requests

During processing of the HTML file, the browser may find that it needs to request additional files (e.g., images and scripts, etc.) and will send additional HTTP Requests for those.

Why is all of this important?

Our **Sinatra applications** will need to respond to **HTTP requests** and generate content to form the **HTTP response**.

So its important to know a bit about what constitutes a HTTP request and a response – although we need not be concerned with all the details.

Summary

To understand how to write a web application we need to have an understanding of how web browsers and servers communicate using HTTP (HyperText Transfer Protocol).

- Browsers send **HTTP requests** to a web server, which a web application processes, sending back an appropriate **HTTP response**.

Two important parts of the HTTP request are the HTTP method being used and the identifier of the resource being requested.

- The most important HTTP methods are `GET` and `POST`.

Two important parts of the HTTP response include its body (the HTML of a web page) and its status code.

- Important codes include **200 (OK)**, **302 (Redirect)**, **404 (Not Found)**, **500 (Internal Server Error)**.