HTTP - Request/ Response

HTTP - Documentation

- HTTP/1.1 is defined by RFC2616 of the IETF
  - https://tools.ietf.org/html/rfc2616
  - This is THE document for all your questions about HTTP
  - Today we'll discus topics in sections 4, 5, and 6
- RFC
  - Request For Comments
  - Submit an RFC for public discussion or to publish information
- IETF
  - Internet Engineering Task Force
  - Adopts some RFC's as Internet standards

New Lines

- A new line character in an HTTP request must be:
  - **"\r\n"**
  - Carriage return (From the days of typewriters)
  - New line
  - In the documentation this is referred to as a CRLF
  - CRLF == Carriage Return Line Feed
- Also from the documentation

```
The line terminator for message-header fields is the sequence CRLF.
However, we recommend that applications, when parsing such headers,
recognize a single LF as a line terminator and ignore the leading CR.
```

Request

- We'll use this simple request as an example

- What specific resource is being requested?

```
GET / HTTP/1.1
Host: cse312.com
```

The Request Line

- The first line of the request is always the request line
- The request line has 3 values separated by spaces
  - The request type (ex. GET/POST)
  - The path of the request (ex. "/")
  - The HTTP Version
    - We'll always use HTTP/1.1 in this course
    - You can assume the request uses HTTP/1.1 in your assignments without checking this string

```
GET / HTTP/1.1
Host: cse312.com
```

The Request Line

- Parse the request line by looking for the 2 space characters
    - Separate the values and check the strings
- Typically: When the root path "/" is requested, serve your home page
    - By convention, web servers look for index.html to server
- If the url contained a different path, it will appear in the request line

```
GET / HTTP/1.1
Host: cse312.com
```

```
GET /lecture HTTP/1.1
Host: cse312.com
```

Headers

- Following the request line are any number of headers
- HTTP Headers
  - Key-Value pairs
  - Key and value separated by a colon ":"
- Each header will be a new line
- To parse, look for the colon ":" and read the key and value

```
GET / HTTP/1.1
Host: cse312.com
```

Response

- Your web server will listen for requests over the TCP sockets and respond with HTTP responses
- Send this response back to the client to serve them the requested content

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 5

hello
```

Status Line

- The first line of the response must be the status line
- Status line contains 3 values separated by spaces
  - The HTTP version
  - The status code
  - The status message (Reason phrase in docs)

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 5

hello
```

Response Headers

- The headers in the response follow the same format as request headers
- Should have at least two headers
  - Content-Type - Tells the browser how to parse this content
  - Content-Length - How much information should be read from the body

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 5

hello
```

Body

- The headers are followed by 2 new lines to indicate the beginning of the body
- The body contains the content that is being served
- The body ends with 2 new line characters

- [Requests can have bodies too]

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 5

hello
```

Response Codes

- Tells the browser the nature of the response
  - 200 means everything went well
  - 404 means the content was not found
- Include a human readable message
  - OK
  - Each code has a standard message, but different messages are accepted

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 5

hello
```

Response Codes

- Some responses don't include a body
- To redirect to another page use 301 with no body
    - Must contain a Location header for the location of the redirect
    - Location can be a relative path (Same server) or an absolute path

```
HTTP/1.1 301 OK
Location: /
```