Hyper Text Transfer Protocol (HTTP)

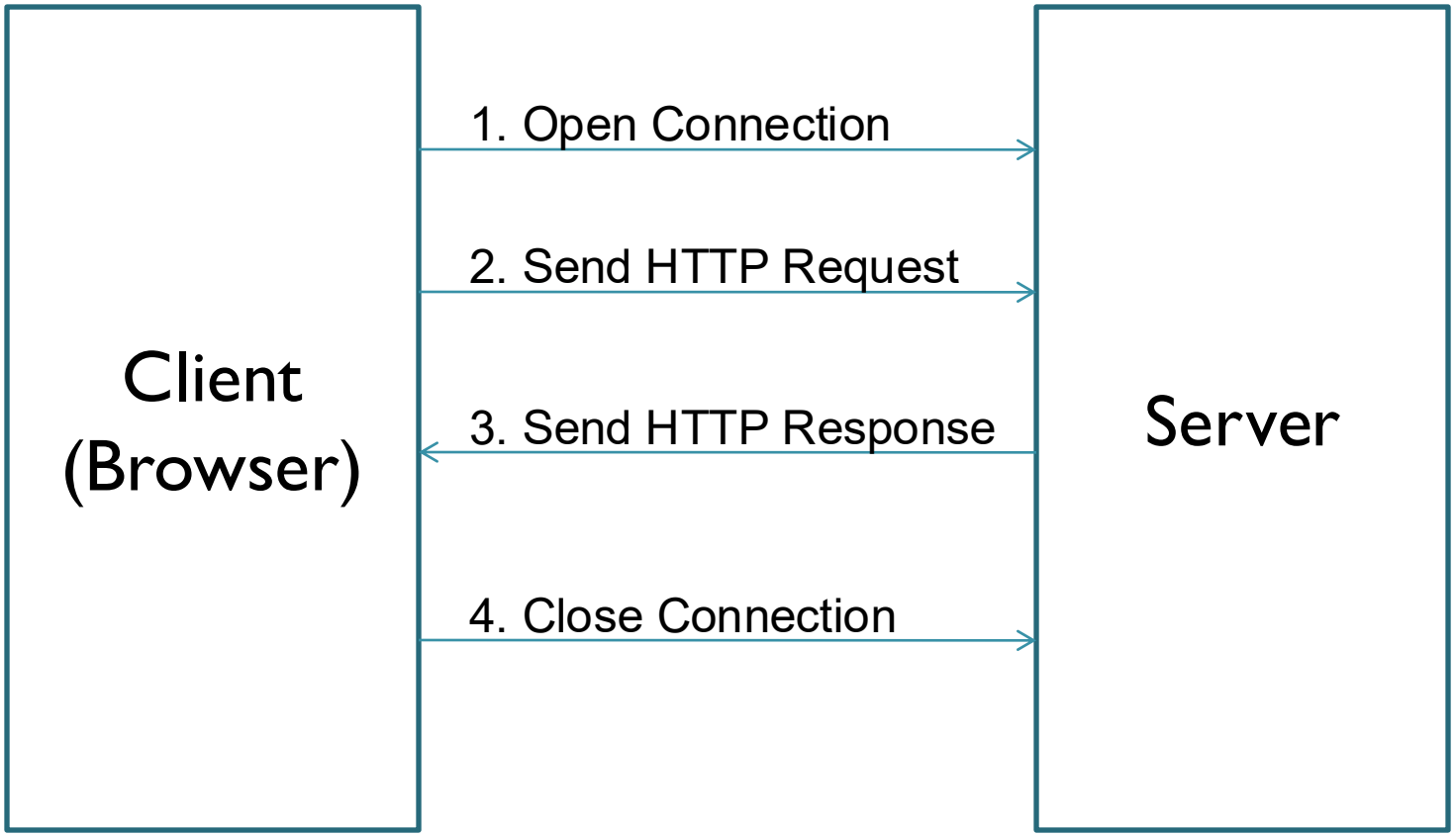
Refer for detailed notes :-

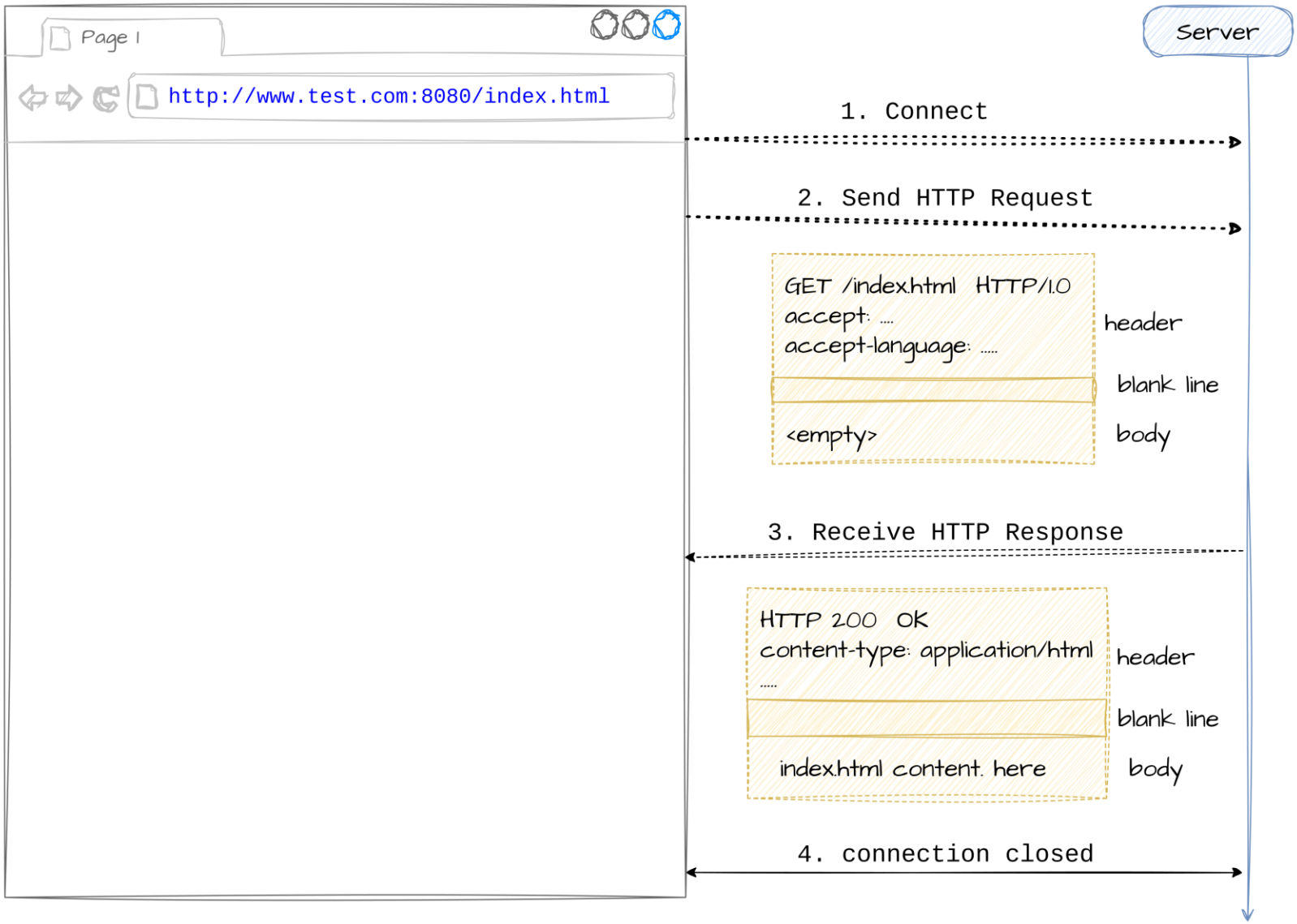
https://github.com/sagaruppuluri/EP/blob/main/Module3/Readme.md

HTTP Overview

- ✓ Hypertext Transfer Protocol is a request-response-based stateless protocol.
- ✓ A client sends an **HTTP request** for a resource
- ✓ Server returns an **HTTP response** with the desired resource.
- ✓ It is stateless because once the server sends the response it forgets about the client.

HTTP Overview




Running at: www.test.com
Listening to Port: 8080
Page 1
http://www.test.com:8080/index.html
Server
1. Connect
2. Send HTTP Request
GET /index.html HTTP/1.0
accept: ....
accept-language: .....
header
blank line
<empty>
body
3. Receive HTTP Response
HTTP 200 OK
content-type: application/html
.....
header
blank line
index.html content here
body
4. connection closed

HTTP Message

It is either a request from the client or a response from the server.

Parts of an HTTP message

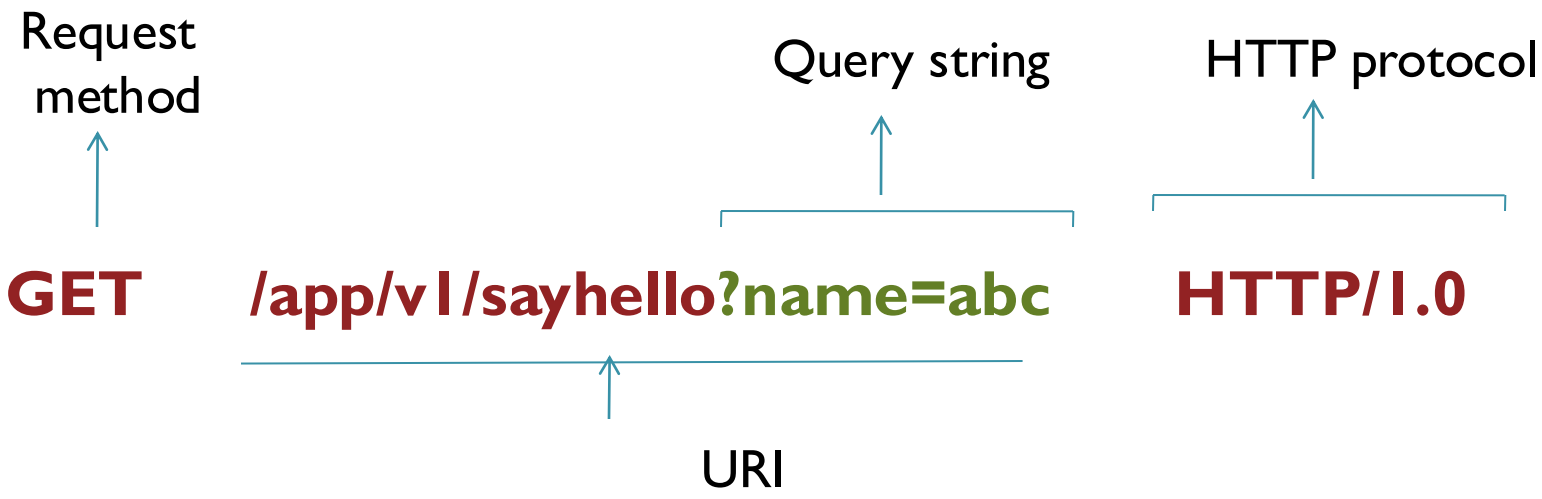
Message part	Description
The initial line	Purpose of the request or response
The Header	Contains meta-information
A blank line	
An optional message body	The main content of the request or response message.

HTTP Request Methods

- ✓ GET – Retrieving resource
- ✓ POST – Creation of resource
- ✓ PUT – Updating Existing resource
- ✓ DELETE – Deleting a resource
- ✓ PATCH – Similar to PUT but for Partial updates
- ✓ HEAD – Same as GET but only for getting headers
- ✓ OPTIONS – Communication options e.g., such as what are the allowed methods for an URI.

GET Method

✓ The HTTP GET method is used to retrieve a resource.
✓ Normally requests a *passive resource*
✓ May be used for an *active resource* if there are few or no parameters.
Example :-



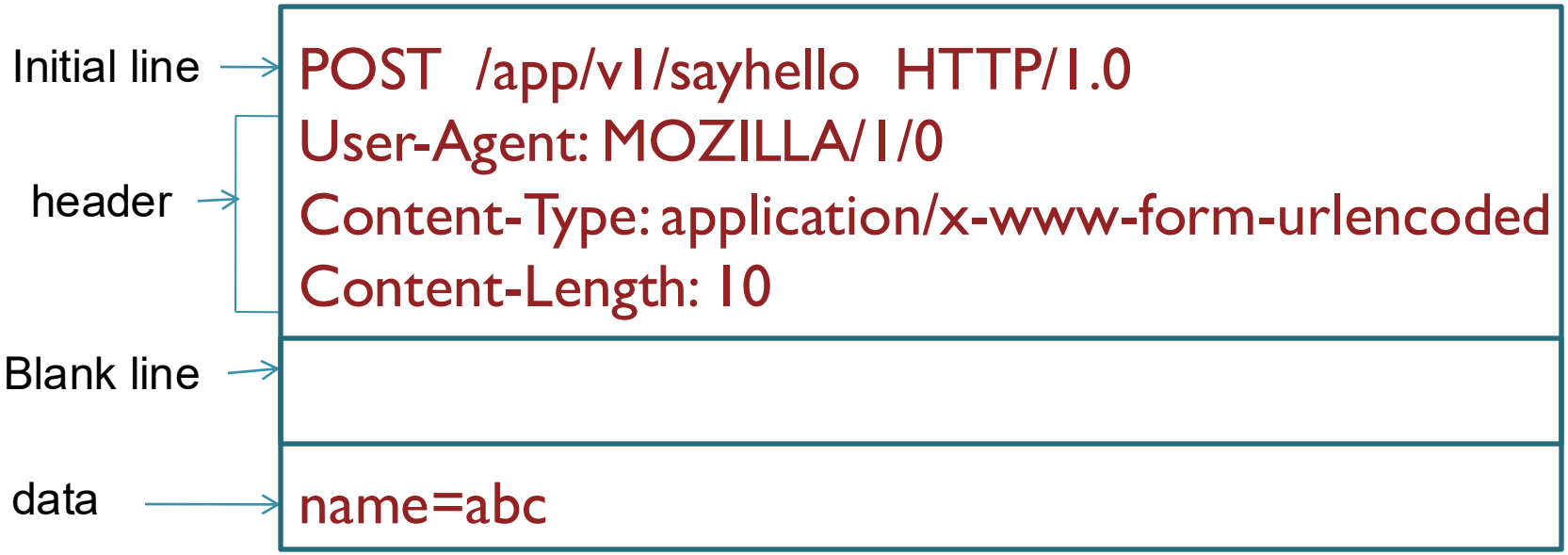
POST Method

✓ A POST request is used to send data to the server in order to be processed.

✓ The block of data is sent in the message body.

✓ Extra lines such as **Content-Type** and **Content-Length** are present in the header.

Example :-



HTTP Response

- ✓ It is the HTTP message sent by server to the client
- ✓ Initial line is called the **status line**, it contains
  - version
  - status code
  - description of the status code
- ✓ Typical HTTP response,

```
HTTP/1.0  200  OK
Content-Type :  text/html
Content-Length:  44

<html>
   <body>
         Hello!  abc
   </body>
</html>
```

HTTP Response (Status Codes)

100-199 :- informational; indicating that the client should respond with some other action.

200-299 :- Success status codes

300-399 :- Redirection, usually include a Location header indicating the new address.

400-499 :- Client error such as Unauthorized, Forbidden, Bad Request etc.

500-599 :- Server Error such as Internal Server Error, Service Unavailable etc.

Interpreting URL in the Address bar

General structure of the URL in the address bar

protocol :// DNS or IP : portno /uri ? querystring

e.g.

http://www.demo.com: 7001/score ? htno=1

When no port number is specified
80 (default) for HTTP
443 (default) for HTTPS

https://www.demo.com/score

Equivalent to https://www.demo.com:443/score

Relative URL

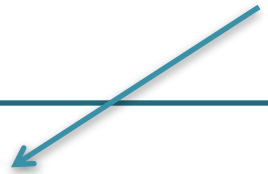
Relative URL is formed using the existing URL in the address bar.

e.g.

Address	http://www.sample.com:8080/college/home.html

With in the page

 courses



When you click the link, home.html will be replaced with courselist.html,

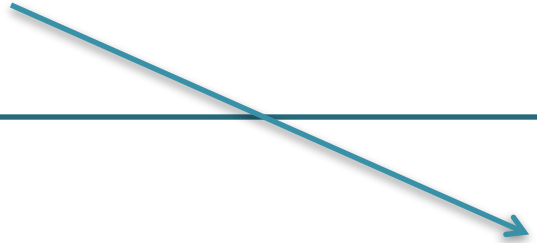
http://www.sample.com:8080/college/**courselist.html**

Relative URL (continued)

Address	http://www.sample.com:8080/college/home.html

With in the page

 courses



Relative URL that starts with / replaces the content from the root (i.e. from / after portno).

http://www.sample.com:8080/**univ/courselist.html**

Relative URL (continued)

You can also use .. (dot dot) notation in relative url.

Address	http://www.sample.com:8080/college/**cs/home.html**

With in the page

 IT dept

When you click the link, .. in the relative URL is like previous directory, or in other words one step behind the current. ,

http://www.sample.com:8080/college/**it/home.html**