**HTTP Request Methods**

**GET**

Retrieves information identified by the Request-URI in a form of an entity (RFC 1945, 2616).

The primary way of retrieving information and resources. It ‘request transfer of a current selected representation for a target resource’ (RFC 7231).

Syntax: GET /example/index.html

**HEAD**

It is the same as the GET method except that the server must not return the payload in the response. Instead, it only return the headers much like if it had been a Get method (RFC 1945, 2616, 7231).

In other words, it returns the metadata of the response (classroom notes)

Syntax: HEAD /example/index.html

**POST**

Submit an HTML form data to be posted on the server. It requests the target resource to process the representation in the message body (classroom notes).

“Used to request that the destination server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request Line.” (RFC 1945, 2616)

Syntax: POST /example/index.html

**PUT**

Creates/replaces a representation of the target resource enclosed in a request payload. It is possible that this method can create side effects in the server (MDN web docs & classroom notes).

“Requests that the state of the target resource be created or replaced with the state defined by the representation enclosed in the request payload.” (RFC 7231 and classroom notes)

NOTE: The difference between the POST and PUT methods is that PUT is idempotent. This means that calling the method repeatedly does the same exact effect whereas the POST method, being not idempotent, may cause additional effects when repeatedly called, otherwise known as a side effect (MDN web docs & classroom notes).

Syntax: PUT /newExample.html HTTP/1.1

**DELETE**

Deletes a specified resource in the original server (Classroom notes).

“RFC 7231: ‘Requests that the origin server remove the association between the target resource and its current functionality.” (RFC 7231 and classroom notes)

Syntax: DELETE /oldExample.html HTTP/1.1

**CONNECT**

Starts a two-way communications with the requested resource, otherwise known as a tunnel, which is used with proxies to establish a TLS session with an origin server. (MDN web docs and RFCs)

\* Tunnel – used to create a direct network link between two locations

\* Tunneling – “port forwarding”

A method of transferring encapsulated data from a private network and protocol information using a public network (Udemy).

Syntax`: CONNECT www.example.net:80 HTTP/1.1

**OPTIONS**

Requests information about the communications options available for the target resource but does not specify a resource action. (RFC 7231 and MDN web docs)

Syntax: OPTIONS /example/index.html HTTP/1.1

OPTIONS \* HTTP/1.1 (refers to the entire server)

**TRACE**

Requests a remote, application-level loop-back test of the request message along the path to the target resource. (RFC 7231 and MDN web docs)

Syntax: TRACE http://www.example.com HTTP/1.1

**PATCH**

Requests a partial modification to the resource identified in the request entity. It is also possible for side-effects to occur when using this method. (RFC 5789 and MDN web docs)

Syntax: PATCH /new.txt HTTP/1.1

**References**

Classroom notes

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

https://blog.udemy.com/http-tunneling/

https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP\_Basics.html

RFC 1945, 2616, 7231, 7540, 5789