**WEB TECH LET’S GO!**

**World Wide Web**

> Collection of web resources and applications that provide access to ??? through the Internet.

> Established around 1990

> Invented by Sir Timothy Berners-Lee

**Hypertext Transfer Protocol**

> Application-level protocol with thee lightness and speed necessary for distributed collaborative hypermedia information system.

> In use by www since 1990

> Jointly developed by W3C and IETF

> HTTP 0.9 (1991)

* Only has GET method
* Response is only html
* Disconnects after receiving response

> HTTP 1.0 (RFC 1945 - 1996)

* GET, HEAD, POST method

> HTTP 1.1 (RFC 2068 - January 1997; RFC 2616 - June 1999; RFC 7230-7235 - June 2014)

* PUT, TRACE, OPTIONS, DELETE, CONNECT method
* HEADER is long and cannot be compressed

> HTTP 2.0 (RFC 7540 - May 2015)

* HEADER compression is provided
* Multiplexing concurrency
* Server push (server can anticipate in advance)

\*Uniform Resource Identifiers - builds on the discipline of reference

\*Uniform Resource Locators - indicating on which a method is to be applied

\*Multipurpose Internet Mail Extension - messages are passed in a format similar to that used by Internet mail

\*Connection - transport layer virtual circuit established between two application program

\*Message -basic unit of HTTP communication consists of structured ??? of octet

\*Requests - HTTP request message

\*Response - HTTP response message

\*Resource - network data object

\*Entity - representation/rendition of data resource

\*Client - application program establishes connections for sending requests

**HTTP Fundamentals**

> Runs on TCP/IP using TCP port by default

> HTTP is a stateless communication protocol (does not keep information between client request)

**HTTP Functionalities**

> Cache control

> MIME specifications

> Language & character specifications (ASCII)

> Content/transfer coding

> Content negotiations (allows preferences among format types)

> Client-server protocol negotiation (relies on protocol)

> Persistent connections (provides reuse for connections unlike former single connections/requests)

> Request pipelining/multiplexing

> Authentication/authorization (securing requests)

**Uniform Resource Identifiers**

> Begins with scheme (i.e. http, https)

> Authority

* Host/Domain name
* Port number

> Path to source (document root on the server)

> Query (key)

> Fragment identifier (subsections within the page)

Example:

http://jpl:eng@qcine.org:69/info/idnumber.php?id=1021#addr

Red - scheme

Orange - host

Yellow - port

Green - path to source

Blue - query

Purple - fragment identifier

|  |  |
| --- | --- |
| **HTTP Request Message** | **HTTP Response** |
| > Request Line   * Method * Target (resource identifier) * Protocol version   > Headers (general, request, entity)  > Empty line (CRLF)  > Message body (payload) | > Status Line   * Protocol version * Status code * Reason phrase   > Message headers  > Empty line (CRLF)  >Message body |

**HTTP Request Message**

> GET - obtain target resource

> POST - process on origin metadata about the target resource without having data

> HEAD - used to obtain metadata about the target resource without

> PUT - create/replace target resource

> DELETE - delete resource

> OPTIONS - to determine requirements/capabilities of serves to the target resource

> TRACE - loopback (echo/repeats request)  
> CONNECT - establish a tunnel to the server (end-to-end virtual connecting)

\*Methods in RED are required to be supported by the server

**Method Properties**

> Safe - read-only method, no changes in the server [GET, HEAD, OPTION, TRACE]

> Idempotent - no matter how many times the request is made, the expected behavior from the server is the same [GET, HEAD, OPTION, TRACE, PUT, DELETE]

\*Safe methods are idempotent but not vice versa

> Cacheable - the response for a method is allowed to be for future use [GET, HEAD, POST]

**HTTP Status Code**

> Informational (1xx)

* 100 Switching protocols
* 101 Continue

> Success (2xx)

* 200 OK
* 201 Created
* 202 Accepted
* 203 Non-authoritative
* 204 No content
* 205 Reset content
* 206 Partial content

> Redirection (3xx)

* 300 Multiple choices
* 301 Moved permanently
* 302 Found
* 303 See other
* 304 Not modified
* 305 Use proxy
* 306 (Unused but reserved)
* 307 Temporary redirect

> Client Error (4xx)

* 400 Bad request
* 401 Unauthorized
* 402 Payment required
* 403 Forbidden
* 404 Not found
* 405 Method not allowed

> Server Error (5xx)

* 500 Internal Server Error message is suitable
* 501 Not Implemented
* 502 Bad Gateway server
* 503 Service Unavailable a temporary state
* 504 Gateway Timeout
* 505 HTTP Version Not Supported

**HTTP Message Headers**

> General headers

> Request headers

> Entity

**HTTP Extension**

> HTTP is extended by defining new request methods, message headers or status codes

> Web Distributing And Versioning (authorization)