Questions 1 – HTTP

1. **What kind of information HTTP header is delivering?**

   The purpose of the HTTP header fields is to provide required information about the HTTP request or response or about the data sent in the message body. The type of header fields included in a HTTP message partially depend on the type of the message. While some fields, so called general-headers, are applicable for both requests and responses, client request-headers are only appropriate for requests and server response-headers for responses. Fourth header field category is entity-headers, which provide meta information about the entity body – if it's present – or about the resource identified by the request.

   All HTTP headers follow the same generic format of a field name followed by a semicolon and the value of the field.

   ```
   field-name ":" [ value of the field ]
   ```

   If you are writing your own custom web client and server, you can create custom fields in addition to the pre-existing ones.

   Some of the topics the information delivered in the header cover are:

   - Authentication
   - Caching
   - Client hints
   - Conditionals
   - Connection management
   - Content negotiation
   - Controls
   - Cookies
   - Downloads
   - Proxies

2. **What is purpose of status codes and what kind of values they have?**

   Status codes are 3-digit integers placed in the HTTP response server sends a client. They indicate whether a specific HTTP request sent to the server has been successfully completed or whether an error or some other notable event occurred. Responses, and hence the status

codes, can be divided into five different classes: informational responses, successful responses, redirects, client errors and server errors. The first digit of a status code indicates the class of the status code while the other two have no specific role.

The status code classes

- 1xx: Informational
    - These codes mean that the request has been received and the process is in progress.
    - For example, code 101 means that the server is switching protocols.
- 2xx: Success
    - This class of codes say that the action in the request was successfully received, understood and accepted.
    - For example, code 200 simply means “OK”, the request is okay.
- 3xx: Redirection
    - This group of response code tell the client that completing the request requires further action to be taken.
    - For example, code 301 tells the client that the requested page has permanently moved to a new URL.
- 4xx: Client error
    - These codes tell the client that the requested either contained incorrect syntax or cannot be fulfilled.
    - A well-known example of a status code belonging to this class is 404, “page not found”, which means that the server cannot find the requested page.
- 5xx: Server error
    - This group of codes mean that the server failed to fulfill a requested that appeared to be valid.
    - For example, code 505 indicates that the server doesn’t support the HTTP version the client is using.

3. HTTP is stateless protocol, what does it mean?

HTTP being a stateless protocol means that the server and the client are aware of each other only during the current request. Each request is treated as an independent transaction and after the request is over, both the client and the server forget about each other. The protocol doesn’t require server to retain information about each client for the duration of multiple requests.

4. List request methods defined in HTTP?

- **GET**
  GET is the primary method for to retrieving documents and is used to fetch information from a specified URI. The GET requests can only supply data as parameters in the URI or as cookies in the cookie request header. Request using GET should only request data and not have any side-effects of it.

- **HEAD**
  The HEAD request method is otherwise identical to GET but it's asking for a response that doesn't have a response body, only status line and the header section. Hence this request method is useful for getting meta information written in response headers without needing to transfer the entire content.

- **POST**
  This request method is used for sending data to the server and it often causes side effects or a change of state on the server.

- **PUT**
  This method allows user to upload new files to the web server or update existing ones. If the URI specified in the request refers to an existing resource, it is modified. Otherwise server creates a new resource with the given URI.

- **DELETE**
  Deletes the resources specified by given URI.

- **CONNECT**
  The CONNECT method establishes a TCP/IP tunnel connection to the server. This usually done in order to facilitate SSL-encrypted communication (ex. HTTPS).

- **OPTIONS**
  The method returns a list of HTTP methods that the servers supports for a specified URL. Functionality of the server can be checked using this method by requesting "*" instead of a specific resource.

- **TRACE**
  This method simply echoes the received client request so that a client can see if and what changes or additions have been made by intermediate servers. The method is used mainly for debugging purposes.

5. Show format of a HTTP request and a possible response to it.

HTTP request messages are in the following format:

- A request line, which consist of name of the request method used (GET, POST…), request URI and the protocol version.
- Zero or more header fields followed by CRLF (the end-of-line marker used in HTTP)
- An empty line which indicates the end of the header fields
- Possibly a message body

Example of a HTTP request:

```
GET http://rajala.me/projects.html HTTP/1.1
Host: rajala.me
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:52.0)
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fi-FI,fi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
If-Modified-Since: Sun, 16 Apr 2017 17:03:34 GMT
Cache-Control: max-age=0
```

HTTP response's format then looks almost identical:

- A status line, which consist of the HTTP version and a status code accompanied by a phrase associated with the said status code.
- Zero or more header fields followed by CRLF (the end-of-line marker used in HTTP)
- An empty line which indicates the end of the header fields
- Possibly a message body

Example of a HTTP response to the preceding request:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Cache-Control: max-age=600
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Sun, 16 Apr 2017 17:17:30 GMT
Expires: Sun, 16 Apr 2017 17:27:30 GMT
Last-Modified: Sun, 16 Apr 2017 17:03:34 GMT
Server: GitHub.com
X-GitHub-Request-Id: DA88:25A0:BB3C34F:F6BA406:58F3AA2C

<!DOCTYPE html>
<html lang="en">
<head>
      <title>Projects</title>
      <meta charset="utf-8">
</head>
<body>
      <h1>Projects</h1>
      <p>Page under construction</p>
</body>
</html>
```

6. What is HTTPS, and what does it require compared to normal HTTP?

HTTPS (Hyper Text Transfer Protocol Secure) is a secure version of HTTP. This means that unlike in HTTP protocol, which is not encrypted and vulnerable to eavesdropping and man-in-the-middle attacks, in HTTP all communication between the client and the server is encrypted and the protocol is designed to withstand the mentioned attacks. HTTPS protocol combines HTTP with TLS/SSL protocol which is used for encryption of the communication.

The TLS/SSL protocol uses asymmetric “keys” to encrypt the communication, a public and a private key. Any data encrypted with a public key can only be decrypted with private key and vice-versa. The private key should be carefully protected and only accessible to the owner of the private key while the public key is meant to be given anyone who needs to be able to decrypt data encrypted with the private key.

Which protocol HTTP or HTTPS is being used, is visible in the URL as HTTPS URLs begin with “https://” and HTTP URLs with “http://”. By default, HTTPS also uses port 443 while HTTP uses port 80.

Sources

- HTTP – Overview. Tutorials Point [article]. Retrieved 26.3.2017 from https://www.tutorialspoint.com/http/http_overview.htm
- HTTP – Status Codes. Tutorials Point [article]. Retrieved 26.3.2017 from https://www.tutorialspoint.com/http/http_status_codes.htm
- HTTP Gallery. HttpWatch [article]. Retrieved 26.3.2017 from https://www.httpwatch.com/httpgallery/methods/
- Hypertext Transfer Protocol. Wikipeia [article]. Retrieved 26.3.2017 from https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_message
- HTTP Request Methods. Mozilla Developer Network [article]. Retrieved 26.3.2017 from https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
- HTTP – Methods. Tutorials Point [article]. Retrieved 26.3.2017 from https://www.tutorialspoint.com/http/http_methods.htm
- What is HTTPS? Instant SSL [article]. Retrieved 16.4.2017 from https://www.instantssl.com/ssl-certificate-products/https.html
- HTTPS. Wikipedia [article]. Retrieved 16.4.2017 from https://en.wikipedia.org/wiki/HTTPS
- HTTP – Responses. Tutorials Point [article]. Retrieved 16.4.2017 from https://www.tutorialspoint.com/http/http_responses.htm
- HTTP – Requests. Tutorials Point [article]. Retrieved 16.4.2017 from https://www.tutorialspoint.com/http/http_requests.htm
- HTTP Headers for Dummies. TutsPlus [article]. Retrieved 16.4.2017 from https://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039
- List of HTTP Header Fields. Wikipedia [article]. Retrieved 16.4.2017 from https://en.wikipedia.org/wiki/List_of_HTTP_header_fields
- HTTP Headers. Mozilla Developer Network [article]. Retrieved 16.4.2017 from https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers