1. HTTP and basic working of the web.

What is HTTP? How does it exchange messages between servers and clients? Read about Request and Response HTTP Message.

HTTP gives users a way to interact with web resources such as HTML files and establishes a communication between them by transmitting hypertext messages between clients and servers. HTTP uses specific request methods to accomplish this task, mainly GET and HEAD methods.

The exchange of messages takes place through a four-step process:

1. A website URL starting with "http://" is entered in a web browser from a computer (client). The browser can be a Chrome, Firefox, Edge, Safari, Opera or anything else.

2. Browser sends a request sent to the web server that hosts the website. (REQUEST HTTP MESSAGE)

3. The web server then returns a response as a HTML page or any other document format to the browser.(RESPONSE HTTP MESSAGE)

4. Browser displays the response from the server to the user.

What are the different HTTP methods that can be specified in a request? Document the common use-case for a few of them.

Different methods that can be specified in a request are :

GET requests a data from specific resource

HEAD requests a specific resource without the body content,. HEAD requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body.

POST is used to send data to a server to create/update a resource.

PUT directly modifies an existing web resource that means, when the current representations of target source have to be replaced with request payload, this method is used.

DELETE gets rid of a specified resource

TRACE shows users any changes or additions made to a web resource. This is used when user need to perform a message loop-back test along the path to the target resource.

OPTIONS shows users which HTTP methods are available for a specific URL. So whenever there is a need to show the options of communication resources for the target source, this method is used.

CONNECT converts the request connection to a transparent TCP/IP tunnel. Thus, we can say that whenever a tunnel has to be established to the server, thus method is used.

PATCH partially modifies a web resource

Find out how to see the request and responses being sent by your web browser. Generally, it is the networks tab in the developer console. Go to any website, take any request and examine the headers. What is the User-Agent header for the request?

Two key points in its working are:

1. HTTP is connectionless, i.e. after making request, the client disconnects, when he response is ready, the server re-establish the connection and deliver the response

2. HTTP is stateless, i.e. during the processing of request, client and server know each other but when the process is complete and the connection needs to be re-established later, they need to provide information to each other from scratch (anew) and then the connection is handled as the very first one.

After you type the URL into your browser, your browser will extract the http part and recognize that it is the name of the network protocol to use. Then, it takes the domain name from the URL, and asks the internet Domain Name Server to return an Internet Protocol (IP) address.

Now the client knows the destination's IP address. It then opens a connection to the server at that address, using the http protocol as specified. It will initiate a GET request to the server which contains the IP address of the host and optionally a data payload. The GET request contains the following text:

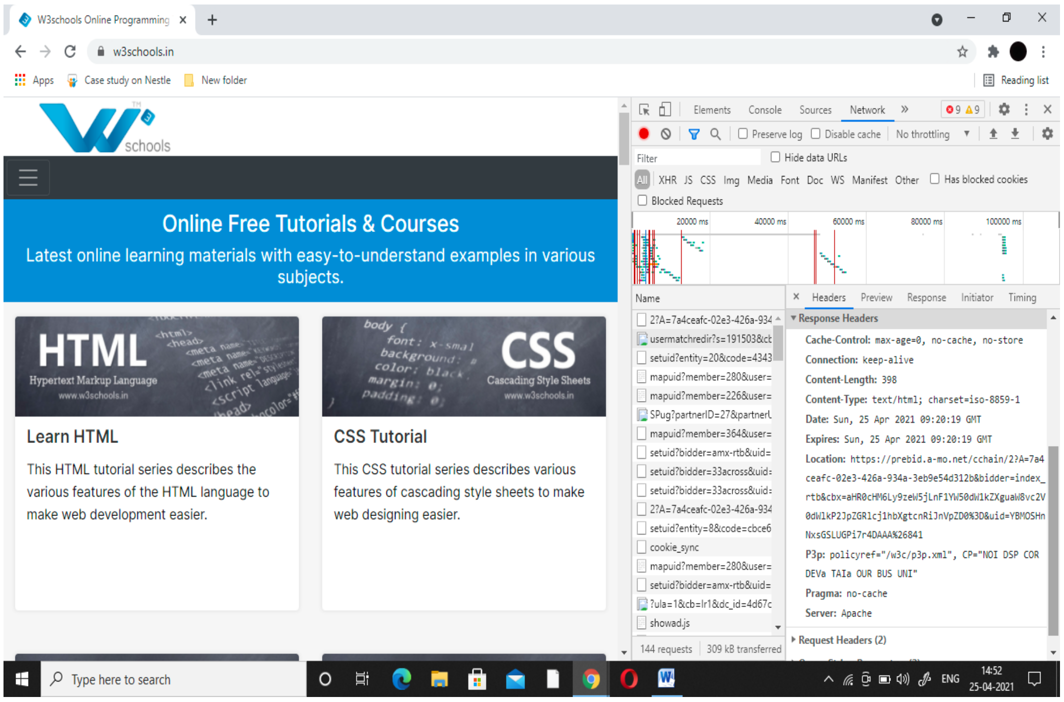
```
GET / HTTP/1.1
Host: www.abc.com       // here abc.com is the domain name
```

This identifies the type of request, the path on www.abc.com and the protocol "HTTP/1.1." The second line of the request contains the address of the server . There may be additional lines as well depending on what data your browser chooses to send.

If the server is able to locate the path requested, the server might respond with the header:

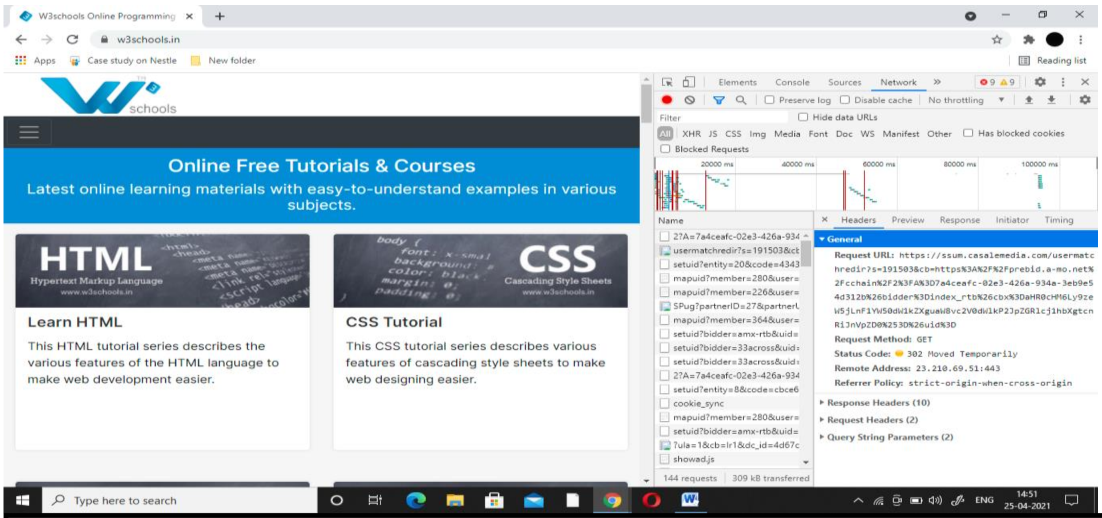
```
HTTP/1.1 200 OK
Content-Type: text/html
```

This header is followed by the content requested,.The first line of the header, HTTP/1.1 200 OK, is confirmation that the server understands the protocol that the client wants to communicate with (HTTP/1.1), and an HTTP status code signifying that the resource was found on the server. The third line, Content-Type: text/html, shows the type of content that it will be sending to the client.


W3schools Online Programming
w3schools.in
Apps
Case study on Nestle
New folder
Reading list
schools
Online Free Tutorials & Courses
Latest online learning materials with easy-to-understand examples in various subjects.
HTML
Hypertext Markup Language
CSS
Cascading Style Sheets
Learn HTML
This HTML tutorial series describes the various features of the HTML language to make web development easier.
CSS Tutorial
This CSS tutorial series describes various features of cascading style sheets to make web designing easier.
Elements
Console
Sources
Network
Preserve log
Disable cache
No throttling
Filter
Hide data URLs
All XHR JS CSS Img Media Font Doc WS Manifest Other
Has blocked cookies
Blocked Requests
Name
Headers
Preview
Response
Initiator
Timing
Type here to search
ENG
14:52
25-04-2021

```
2?A=7a4ceafc-02e3-426a-934
usermatchredir?s=191503&cb
setuid?entity=20&code=4343
mapuid?member=280&user=
mapuid?member=226&user=
SPug?partnerID=27&partnerU
mapuid?member=364&user=
setuid?bidder=amx-rtb&uid=
setuid?bidder=33across&uid=
setuid?bidder=33across&uid=
2?A=7a4ceafc-02e3-426a-934
setuid?entity=8&code=cbce6
cookie_sync
mapuid?member=280&user=
setuid?bidder=amx-rtb&uid=
?ula=1&cb=lr1&dc_id=4d67c
showad.js
144 requests | 309 kB transferred
```

```
▼ Response Headers
   Cache-Control: max-age=0, no-cache, no-store
   Connection: keep-alive
   Content-Length: 398
   Content-Type: text/html; charset=iso-8859-1
   Date: Sun, 25 Apr 2021 09:20:19 GMT
   Expires: Sun, 25 Apr 2021 09:20:19 GMT
   Location: https://prebid.a-mo.net/cchain/2?A=7a4
   ceafc-02e3-426a-934a-3eb9e54d312b&bidder=index_
   rtb&cbx=aHR0cHM6Ly9zeW5jLnF1YW50dW1kZXguaW8vc2V
   0dW1kP2JpZGRlcj1hbXgtcnRiJnVpZD0%3D&uid=YBMOSHn
   NxsGSLUGPi7r4DAAA%26841
   P3p: policyref="/w3c/p3p.xml", CP="NOI DSP COR
   DEVa TAIa OUR BUS UNI"
   Pragma: no-cache
   Server: Apache
▶ Request Headers (2)
```



Online Free Tutorials & Courses

Latest online learning materials with easy-to-understand examples in various subjects.

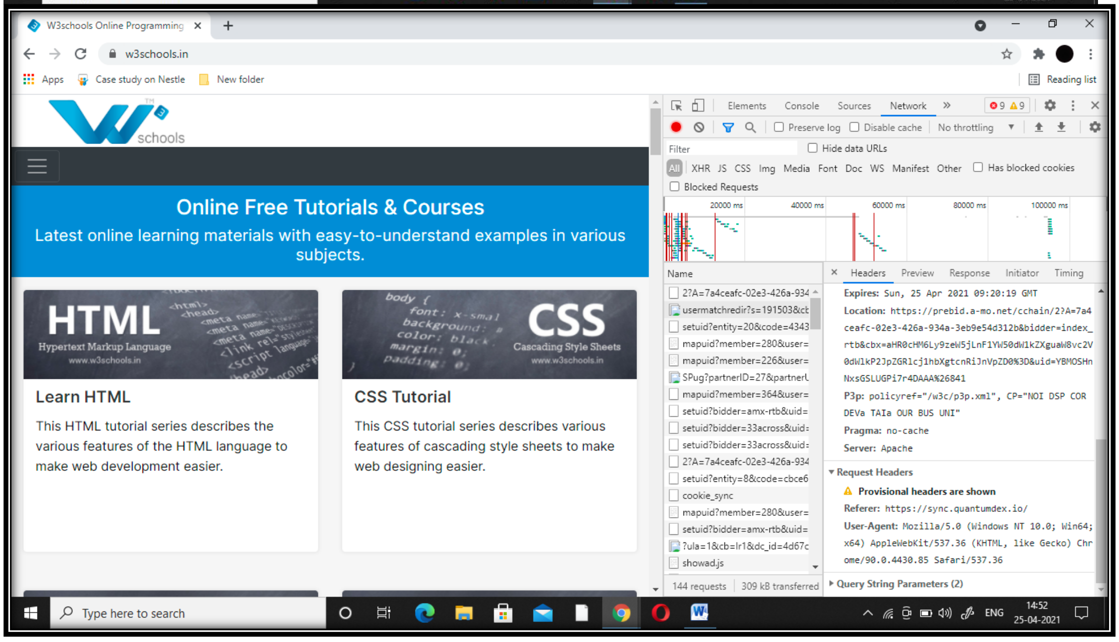
Learn HTML

This HTML tutorial series describes the various features of the HTML language to make web development easier.

CSS Tutorial

This CSS tutorial series describes various features of cascading style sheets to make web designing easier.

```
▼ General
    Request URL: https://ssum.casalemedia.com/usermatchredir?s=191503&cb=https%3A%2F%2Fprebid.a-mo.net%2Fcchain%2F2%3FA%3D7a4ceafc-02e3-426a-934a-3eb9e54d312b%26bidder%3Dindex_rtb%26cbx%3DaHR0cHM6Ly9zeW5jLnF1YW50dW1kZXguaW8vc2V0dW1kP2JpZGRlcj1hbXgtcnRiJnVpZD0%253D%26uid%3D
    Request Method: GET
    Status Code: 302 Moved Temporarily
    Remote Address: 23.210.69.51:443
    Referrer Policy: strict-origin-when-cross-origin
▸ Response Headers (10)
▸ Request Headers (2)
▸ Query String Parameters (2)
```



```
    Expires: Sun, 25 Apr 2021 09:20:19 GMT
    Location: https://prebid.a-mo.net/cchain/2?A=7a4ceafc-02e3-426a-934a-3eb9e54d312b&bidder=index_rtb&cbx=aHR0cHM6Ly9zeW5jLnF1YW50dW1kZXguaW8vc2V0dW1kP2JpZGRlcj1hbXgtcnRiJnVpZD0%3D&uid=YBMOSHnNxsGSLUGPi7r4DAAA%26841
    P3p: policyref="/w3c/p3p.xml", CP="NOI DSP COR DEVa TAIa OUR BUS UNI"
    Pragma: no-cache
    Server: Apache
▼ Request Headers
    ⚠ Provisional headers are shown
    Referer: https://sync.quantumdex.io/
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36
▸ Query String Parameters (2)
```

USER AGENT STRING (can be seen in above image) : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36

This describes the environment we are working on, and rest info in the next ques.

Most likely, the User-Agent would've been a very complicated string starting with Mozilla. Who knew that the company who makes Firefox was so prevalent.. Read about the history of the User-Agent string and document it briefly. Specifically, what does Mozilla mean in this context? What about Gecko?

The User Agent string , given above, whose various components and their roles are as follows:

Mozilla/5.0: s the general token that says the browser is Mozilla compatible, and is common to almost every browser today. (Windows NT 10.0; Win64; x64): Details of the system in which the browser is running.

AppleWebKit/537.36: The platform the browser uses.

(KHTML, like Gecko): Browser platform details. i.e it based on Gecko. Gecko includes an application-layer language based on web technologies called XUL which can be used for cross-platform GUI development (and is employed in Firefox, Thunderbird, and add-ons for both)

KHTML – originally developed for Konquerer on Linux's KDE desktop,(Konquerer is the browser for KDE that WebKit is based on).

KHTML added the words "like Gecko" so they'd get the modern pages designed for Gecko,and kept "KHTML, like Gecko" line for compatibility purposes

Chrome/90.0.4430.85 Safari/537.36:: This is used by the browser to indicate specific enhancements that are available directly in the browser or through third parties

Resources:

- https://www.w3schools.in/
- https://www.youtube.com/watch?v=iYM2zFP3Zn0
- https://en.wikipedia.org/wiki/User_agent
- https://stackoverflow.com/questions/tagged/user-agent#:~:text=In%20the%20HTTP%20protocol%2C%20a,from%20which%20the%20request%20came.
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview
- https://www.youtube.com/watch?v=wMjYDeiZ40g
- https://www.tutorialspoint.com/http/http_responses.htm
- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

2. Research and document how web cookies work [Read about HTTP Cookies, Cookie related headers, etc.]

What are cookies? How does a website set a cookie and why would it do that?

Cookies are text files or small pieces of data in the browser which stores information in order to identify your computer network. They may also be sent to server with each request. When the cookie is exchanged between computer and the network server, the server reads the ID and knows what information is more relevant for the user.

Cookies are created when a web server tells a browser to create the cookie .There are more than one method to create cookies. The instructions for creating the cookie are usually sent in an HTTP header and look something like this:

Set-Cookie: <cookie_name>=<cookie_value>

- Cookies may also be created with client-side JavaScript by using the document.cookie method.
- The Set-Cookie HTTP response header is used to send a cookie from the server to the user agent, so the user agent can send it back to the server later.

.A website will set a cookie in order to do the following tasks:

1. Session management. For example, cookies let websites recognize users and recall their individual login information and preferences, such as sports news versus politics.
2. Personalization. Customized advertising is the main way cookies are used to personalize your sessions. You may view certain items or parts of a site, and cookies use this data to help build targeted ads that you might enjoy.
3. Tracking. Shopping sites use cookies to track items users previously viewed, allowing the sites to suggest other goods they might like and keep items in shopping carts while they continue shopping.

What are the different attributes that can be applied while setting a cookie? What do they achieve?

Different Attributes that can be applied are:

- ***Secure:*** This attribute, when applied, ensures that only those cookies are sent where the request from browser is transferred over a encrypted protocol .(eg HTTPS)
- ***Max-Age***: This attribute creates persistent cookies and must be added to the Set-cookie header.
- ***Path:*** This attribute when applied expands the scope of cookie to complete website.

- ***Expires***: It maintains the state of cookie up to the specified date and time.
- ***Domain***: As clear from its name, this attribute defines the domain of cookie.
- ***Samesite Flag***: It is a relatively new attribute that ensures that cookies will only be transmitted back to the same website from which they originated.
- ***HttpOnly***: Cookies flagged as HttpOnly will be inaccessible to JavaScript within the webpage DOM and will only be transmitted back to the issuing domain.

  `

  These attributes are used to enhance the functionality of cookies. As mentioned above the role of different attributes, the usage of cookies can be made more efficient and as per the requirement.

How do cookies assist in advertising companies (such as Google) being able to track users across different sites?

When we set preferences on a website, a cookie allows the website to remember then the next time we visit. Or when we sign into a website, the website might use a cookie to recognize our browser later on, so that you don't have to sign in again. Cookies also allow websites to collect data about user activity, such as how many unique visitors a page receives per month. These actions of cookies are used to track users across different sites and keep a check on their interest.
Talking specifically about Google AdSense, it also uses cookies for advertising including serving and rendering ads, personalizing ads, limiting the number of times an ad is shown to a user, muting ads you have chosen to stop seeing, and measuring the effectiveness of ads. AdSense sends a cookie to the user's browser after any impression, click, or other activity that results in a call to our servers. If the browser accepts the cookie, the cookie is stored on the browser.

- 'NID' cookie is used for these purposes to show Google ads in Google services for signed-out users
- 'IDE' and 'ANID' are used for these purposes to show Google ads on non-Google sites

Resources

- *https://support.google.com/adsense/answer/9713?hl=en&ref_topic=1628432*
- HTTP Cookies Crash Course - YouTube
- https://www.kaspersky.com/resource-center/definitions/cookies
- What is Http Cookie | Browser Cookie | Internet Cookie - YouTube
- HTTP Headers and Cookies - YouTube
- https://en.wikipedia.org/wiki/HTTP_cookie
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie
- https://www.javatpoint.com/javascript-cookie-attributes

3. Read about Cross-Origin Resource Sharing (CORS)

• *When is the CORS mechanism required?*

Its requirement can be clearly identified by analysing its name itself (i.e. when Resource Sharing has to be done between different origins). Whenever data/resources are supposed to be shared between web applications of different "origins" (or basically domains), CORS mechanism comes in picture, as originally, this "sharing" was not possible as resource was from different origin (same origin policy). Moreover, it keeps a check on potentially malicious documents and verifies the operatives regulating this "sharing" process.

- ***What are the headers set in CORS and how do their values affect things?***

  There are two major kinds of headers:

  - **Response headers** which are sent by server as its response, which can have following different values performing the mentioned task:
    1. The Access-Control-Expose-Headers is a header which allows the server to accept those headers which are accessible to Javascript in browsers.
    2. The Access-Control-Allow-Origin header tells the browser the specific origin that is allowed access to resources. Also, using "*" , this header will tell browsers that any origin can access the resources.
    3. The Access-Control-Max-Age header indicates the duration that how long the results of a preflight request can be stored in the cache memory..
    4. The Access-Control-Allow-Credentials header indicates whether the response to the request can be exposed when the credentials flag is true
    5. The Access-Control-Allow-Methods header specifies the method or methods allowed when accessing the resource.
    6. The Access-Control-Allow-Headers header is used in response to a preflight request to indicate which HTTP headers can be used when making the actual request.
  - **Request Headers** which are sent by client as request to follow CORS Mechanism. They can have different values performing the mentioned task:

    1 .Origin header which indicates the origin of cross-site access request.

    2. Access-Control-Request –Method header is used when preflight request is being made (not the simple one) to indicate the server which http method will be used when the actual(post) request is made

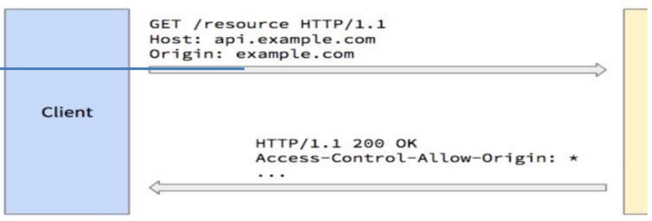
    3. The Access-Control-Request-control header is used when preflight request is being made (not the simple one) to indicate the server which http headers will be used when the actual(post) request is made

- ***What are CORS preflight requests? When are these requests sent? How does a CORS preflight request look like?***

Preflight request (unlike simple requests) asks for the server's permission to send the request i.e. they determine if the actual request is safe to send and whether the server is aware using specific methods and headers. The preflight gives the server a chance to examine what the actual request will look like before it's made. **In** preflight requests, the browser first sends an HTTP request using the OPTIONS method (using three HTTP request headers which are Access-Control-Request-Method, Access-Control-Request-Headers , and the Origin header) to the resource on the other origin. A browser may also send preflight if the request contains additional HTTP headers from the client. **A Sample preflight** request looks like:

```
const xhr = new XMLHttpRequest();

xhr.open('POST', 'https://bar.other/resources/post-here/');

xhr.setRequestHeader('X-PINGOTHER', 'pingpong');

xhr.setRequestHeader('Content-Type', 'application/xml');

xhr.onreadystatechange = handler;

xhr.send('<person><name>Arun</name></person>');
```

A Preflight Request



```
GET /resource HTTP/1.1
Host: api.example.com
Origin: example.com

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
...
```

Resources:

- https://www.youtube.com/watch?v=tcLW5d0KAYE&t=363s
- https://livebook.manning.com/book/cors-in-action/chapter-4/118
- https://geekflare.com/cors-basics/
- https://www.google.com/search?q=what+is+a+preflight+request+look+like&rlz=1C1CHBD_enIN913IN913&sxsrf=ALeKk00kMnz21Sp-DQvghdJHeMq2-PucGg:1619195827619&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjhyIHB5pTwAhWkkOYKHTOiAuwQ_AUoAXoECAEQAw&biw=1366&bih=625#imgrc=0CNgFqdbDgcYaM
- https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS