# HTTP: *Hypertext Transfer Protocol*

Dr. Öğr. Üyesi Ali Nizam

# HTTP

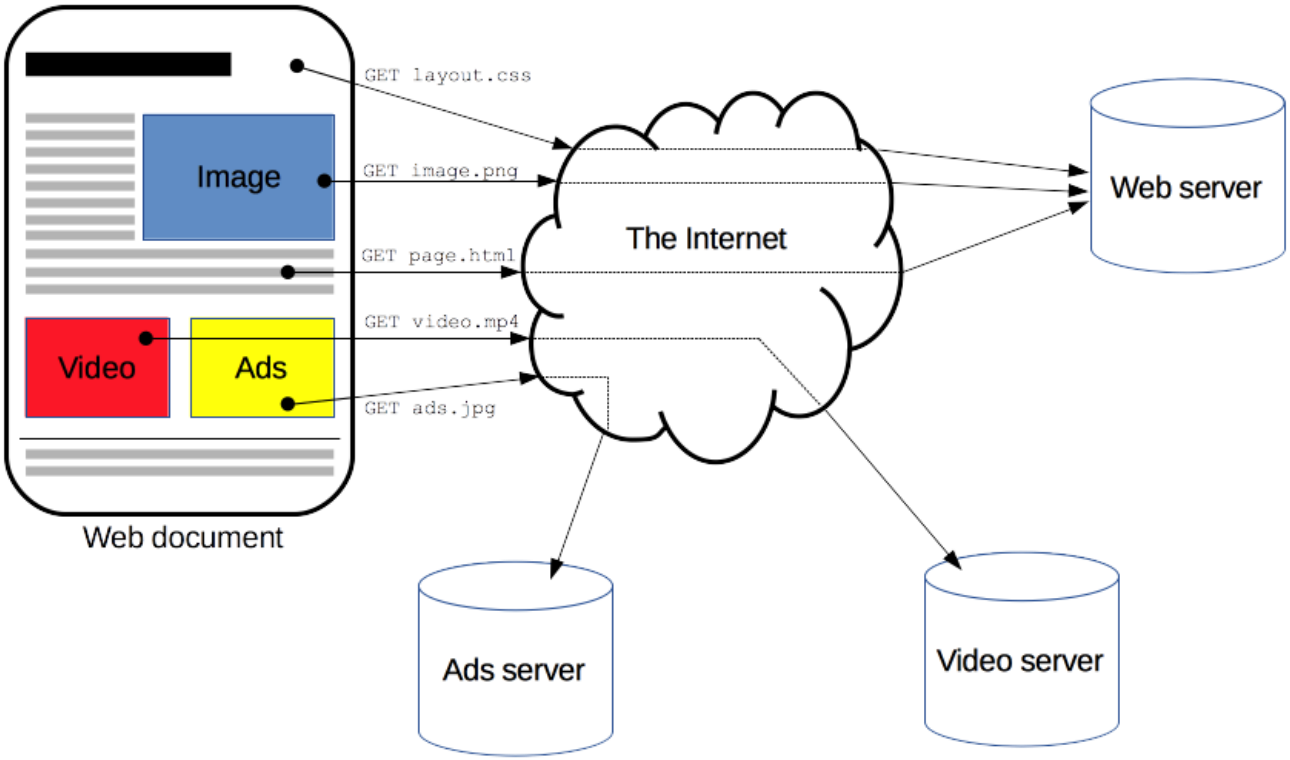


- HTTP is a protocol which allows the fetching of resources, such as HTML documents.
- It is the foundation of any data exchange on the Web and a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.
- A complete document is reconstructed from the different sub-documents fetched, for instance text, layout description, images, videos, scripts, and more.

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

# HTTP




GET layout.css
Image
GET image.png
GET page.html
GET video.mp4
Video
Ads
GET ads.jpg
Web document
The Internet
Web server
Ads server
Video server

# HTTP Flow

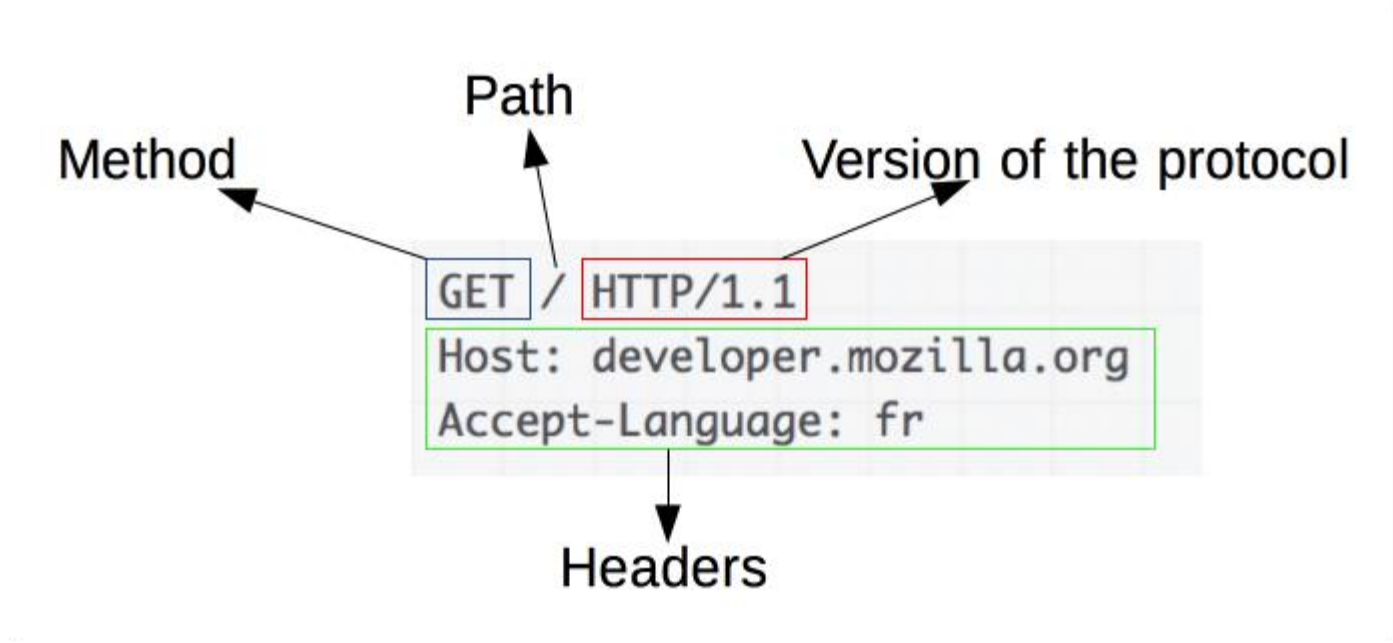


- **Open a TCP connection:** The TCP connection will be used to send a request, or several, and receive an answer. The client may open a new connection, reuse an existing connection, or open several TCP connections to the servers.
- **Send an HTTP message:** HTTP messages (before HTTP/2) are human-readable. With HTTP/2, these simple messages are encapsulated in frames, making them impossible to read directly, but the principle remains the same.
  - GET / HTTP/1.1
  - Host: developer.mozilla.org
  - Accept-Language: fr
- **Read the response sent by the server**
  - HTTP/1.1 200 OK
  - Date: Sat, 09 Oct 2010 14:28:02 GMT
  - Server: Apache
  - Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
  - ETag: "51142bc1-7449-479b075b2891b"
  - Accept-Ranges: bytes
  - Content-Length: 29769
  - Content-Type: text/html

# HTTP Messages



- Request



# Request Content



- An HTTP method, usually a verb like GET, POST or a noun like OPTIONS or HEAD that defines the operation the client wants to perform. Typically, a client wants to fetch a resource (using GET) or post the value of an HTML form (using POST), though more operations may be needed in other cases.
- The path of the resource to fetch; the URL of the resource stripped from elements that are obvious from the context, for example without the protocol (http://), the domain (here developer.mozilla.org), or the TCP port (here 80).
- The version of the HTTP protocol.
- Optional headers that convey additional information for the servers.
- Or a body, for some methods like POST, similar to those in responses, which contain the resource sent.

# HTTP request methods



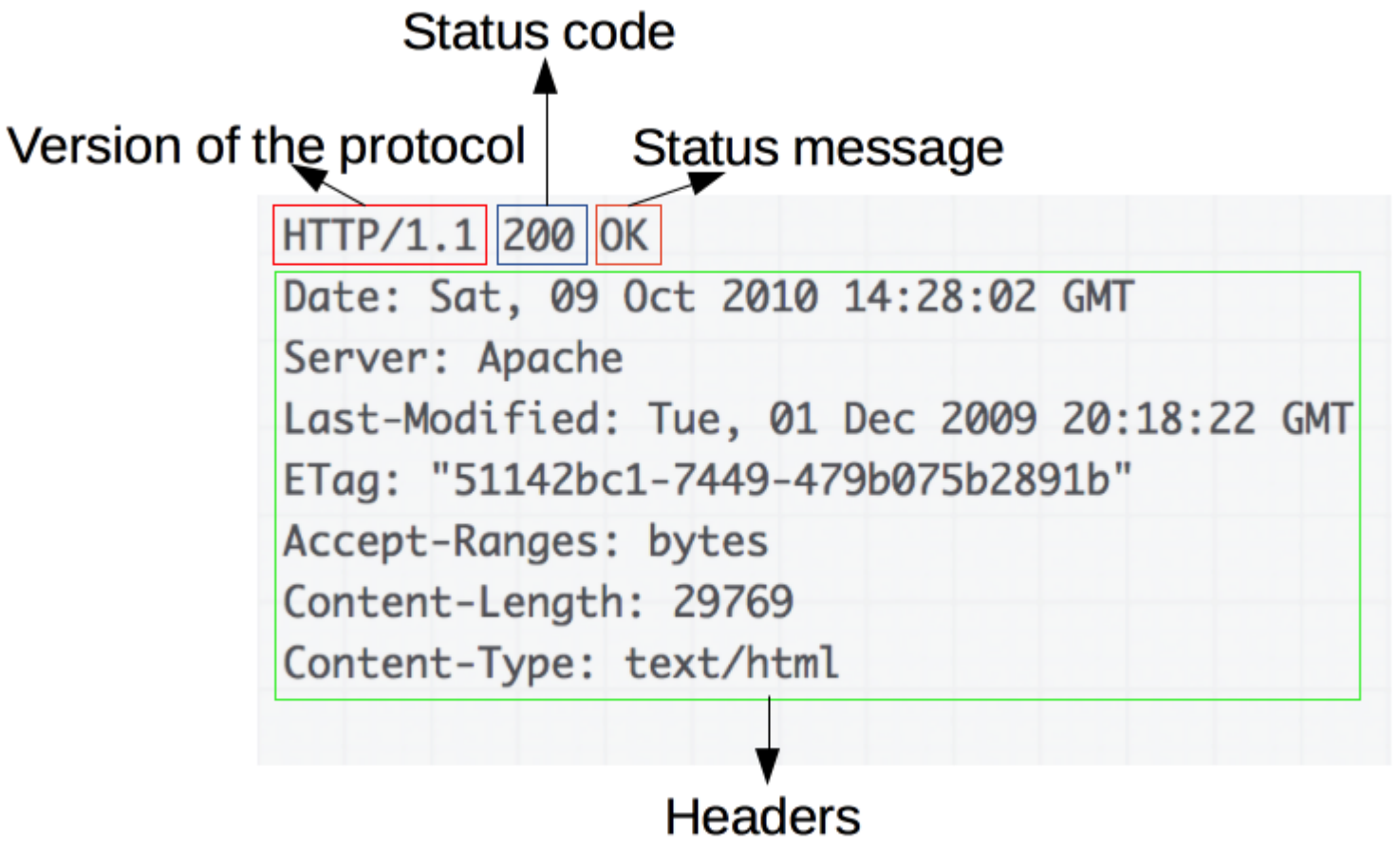
- The request methods implements a different semantic, but some common features are shared by a group of them: e.g. a request method can be safe, idempotent, or cacheable.

- GET: The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.
- HEAD: The HEAD method asks for a response identical to that of a GET request, but without the response body.
- POST: The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.
- PUT: The PUT method replaces all current representations of the target resource with the request payload.
- DELETE: The DELETE method deletes the specified resource.
- CONNECT: The CONNECT method establishes a tunnel to the server identified by the target resource.
- OPTIONS: The OPTIONS method is used to describe the communication options for the target resource.
- TRACE: The TRACE method performs a message loop-back test along the path to the target resource.
- PATCH: The PATCH method is used to apply partial modifications to a resource.

# HTTP Messages



- Responce



# Responses consist of the following elements



- The version of the HTTP protocol they follow.
- A status code, indicating if the request has been successful, or not, and why.
- A status message, a non-authoritative short description of the status code.
- HTTP headers, like those for requests.
- Optionally, a body containing the fetched resource.

# Server

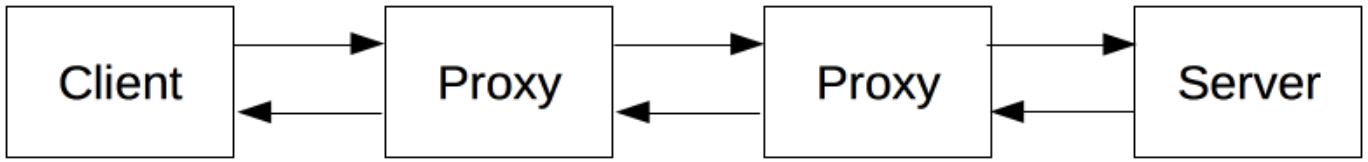


- The server serves the document as requested by the client.
- A server presents only as a single machine virtually: this is because it may actually be a collection of servers, sharing the load (load balancing) or a complex piece of software interrogating other computers (like cache, a DB server, e-commerce servers, …), totally or partially generating the document on demand.

# Components of HTTP-based systems



- HTTP is a client-server protocol: requests are sent by one entity, the user-agent (or a proxy on behalf of it).



# Proxies



- Between the Web browser and the server, numerous computers and machines relay the HTTP messages.
- perform numerous functions:
  - caching (the cache can be public or private, like the browser cache)
  - filtering (like an antivirus scan, parental controls, …)
  - load balancing (to allow multiple servers to serve the different requests)
  - authentication (to control access to different resources)
  - logging (allowing the storage of historical information)
- These are also the common features controllable with HTTP

# HTTP is stateless



- HTTP is stateless, but not sessionless
- HTTP is stateless: there is no link between two requests being successively carried out on the same connection.
- This immediately has the prospect of being problematic for users attempting to interact with certain pages coherently, for example, using e-commerce shopping baskets.
- But while the core of HTTP itself is stateless, HTTP cookies allow the use of stateful sessions. Using header extensibility, HTTP Cookies are added to the workflow, allowing session creation on each HTTP request to share the same context, or the same state.

# Client: the user-agent



- The *user-agent* is any tool that acts on the behalf of the user.
- This role is primarily performed by the Web browser; a few exceptions being programs used by engineers, and Web developers to debug their applications.

# Basic Code to Create a HTTP Call (c#)



```
// Create a request for the URL.
        WebRequest request = WebRequest.Create(
          "http://www.fsm.edu.tr/");
        // If required by the server, set the credentials.
        request.Credentials = CredentialCache.DefaultCredentials;
        // Get the response.
        WebResponse response = request.GetResponse();
        // Display the status.
        Console.WriteLine(((HttpWebResponse)response).StatusDescription);
        // Get the stream containing content returned by the server.
        Stream dataStream = response.GetResponseStream();
        // Open the stream using a StreamReader for easy access.
        StreamReader reader = new StreamReader(dataStream);
        // Read the content.
        string responseFromServer = reader.ReadToEnd();
        // Display the content.
        Console.WriteLine(responseFromServer);
        // Clean up the streams and the response.
        reader.Close();
        response.Close();
```

# Basic Code to Create a HTTP Call (JS)



```
const Http = new XMLHttpRequest();
const url='https://jsonplaceholder.typicode.com/posts’;
Http.open("GET", url);
Http.send();

Http.onreadystatechange = (e) => {
  console.log(Http.responseText)
}
```

# Questions