HTTP

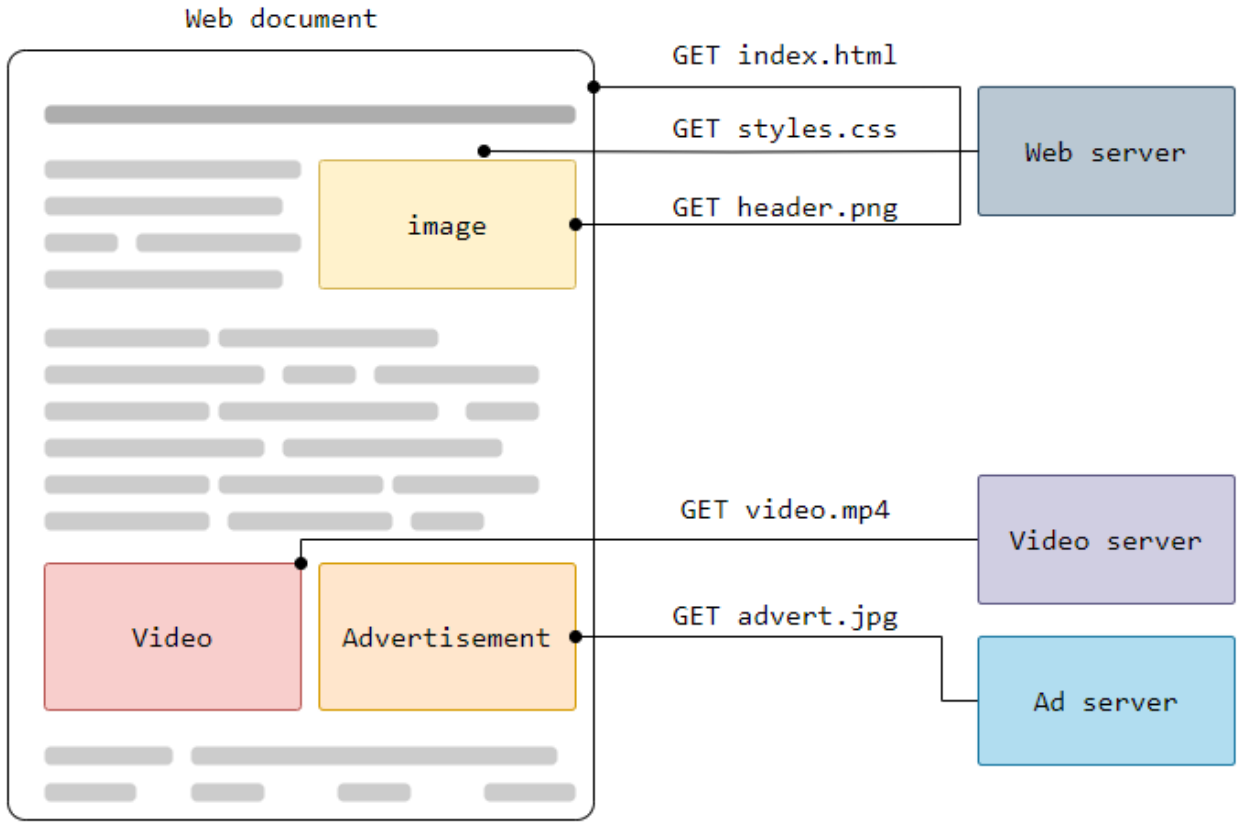
Sridhar Alagar

HTTP

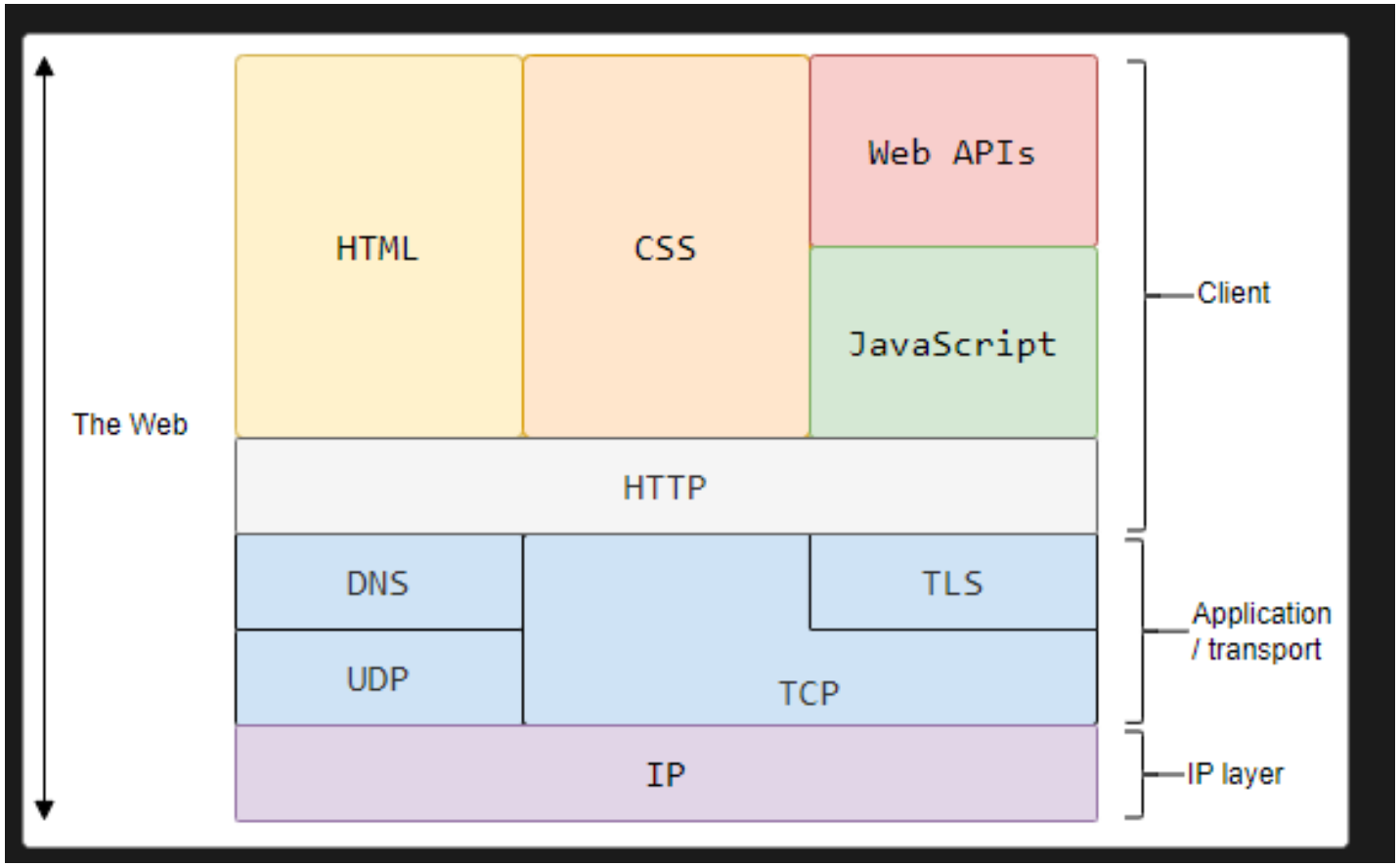
HTTP is a client-server protocol

A protocol to fetch resources by a client from a server

Client initiates a request message, and server replies with a response



HTTP is an application layer protocol



Basic aspects of HTTP

HTTP is simple and human readable

Messages can be understood by humans

HTTP is extensible

New functionality can be implemented

HTTP is stateless, but not session-less

No link between two requests

Session can be maintained using cookies

HTTP requires reliability, not connections

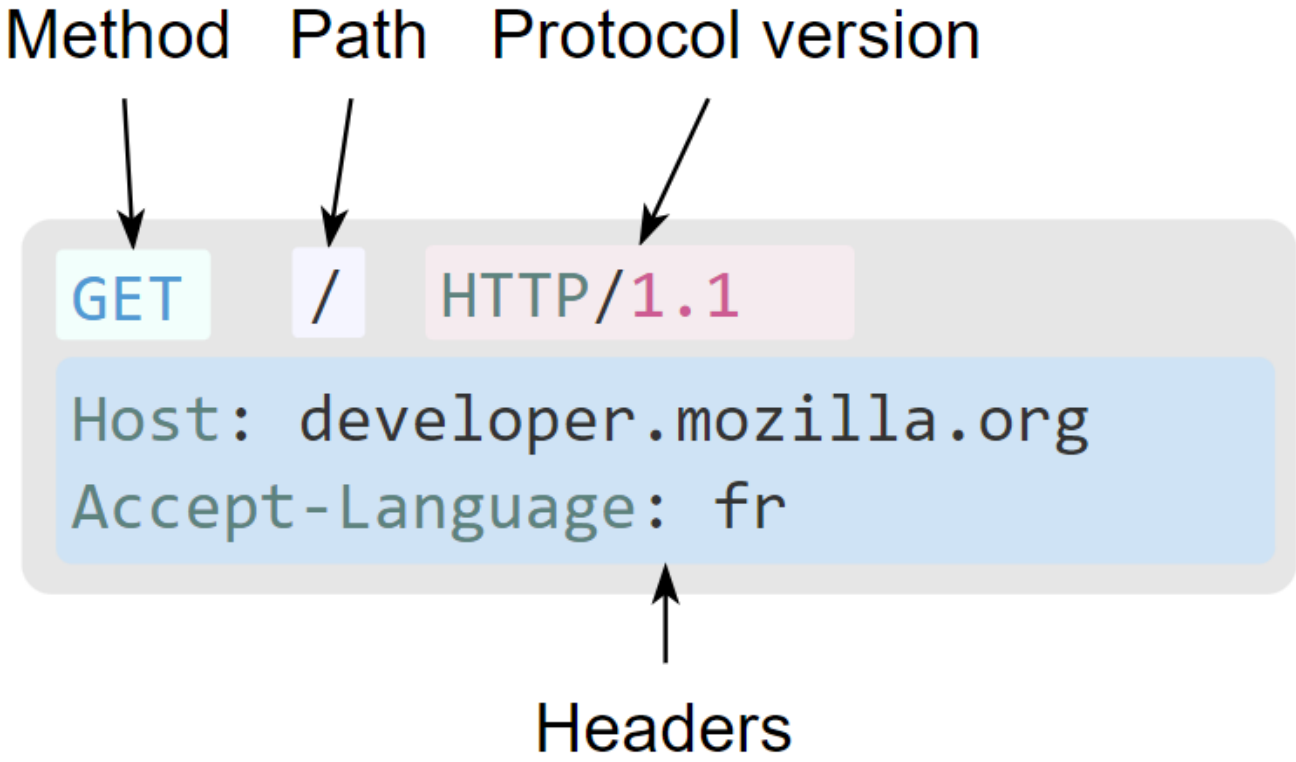
HTTP flow

A client performs the following step to communicate with a server

http://example.com:80/index.html

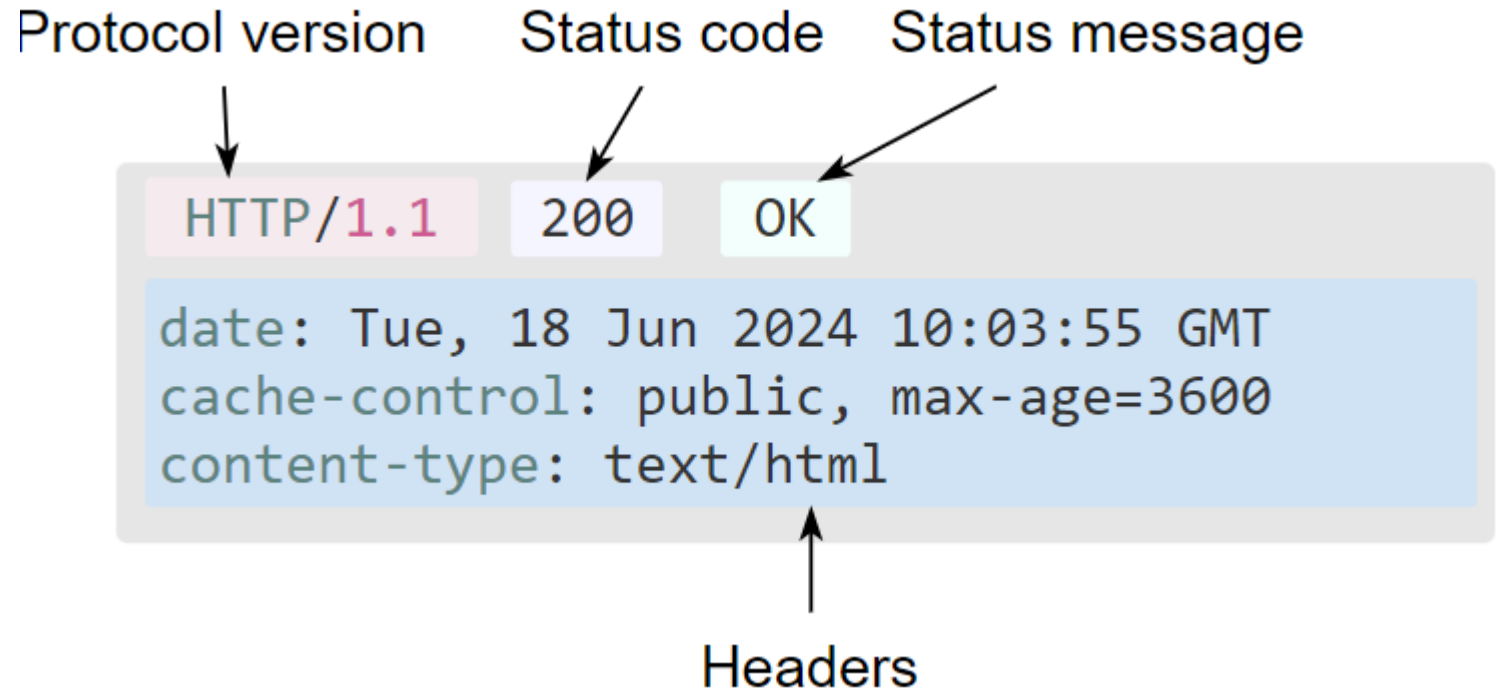
1. Open a TCP connection
2. Send a HTTP message (request)
3. Read the response message sent by the server
4. Close or reuse the connection for further requests

HTTP Request message



Method: specified the operation to be performed. GET, POST, etc.
Path: URL stripped of obvious part
Header: optional. Convey additional information
Body: for some methods like PUT

Response message



Status code: indicates if request was success or not, and why

Status message: short description of status code

Header: optional. Convey additional information

Body: optional. Contains fetched resource for some methods like GET

Successful Response for Get

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 55743
Connection: keep-alive
Cache-Control: s-maxage=300, public, max-age=0
Content-Language: en-US
Date: Thu, 06 Dec 2018 17:37:18 GMT
ETag: "2e77ad1dc6ab0b53a2996dfd4653c1c3"
Server: meinheld/0.6.1
Strict-Transport-Security: max-age=63072000
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Vary: Accept-Encoding,Cookie
Age: 7

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>A simple webpage</title>
</head>
<body>
  <h1>Simple HTML webpage</h1>
  <p>Hello, world!</p>
</body>
</html>
```

Status codes

- 1xx informational response – the request was received, continuing process
- 2xx successful – the request was successfully received, understood, and accepted
- 3xx redirection – further action needs to be taken to complete the request
- 4xx client error – the request contains bad syntax or cannot be fulfilled
- 5xx server error – the server failed to fulfil an apparently valid request
- Common status codes are 200, 404, or 302

PUT request and response messages

```
PUT /new.html HTTP/1.1
Host: example.com
Content-type: text/html
Content-length: 16

<p>New File</p>
```

```
HTTP/1.1 201 Created
Content-Location: /new.html
```

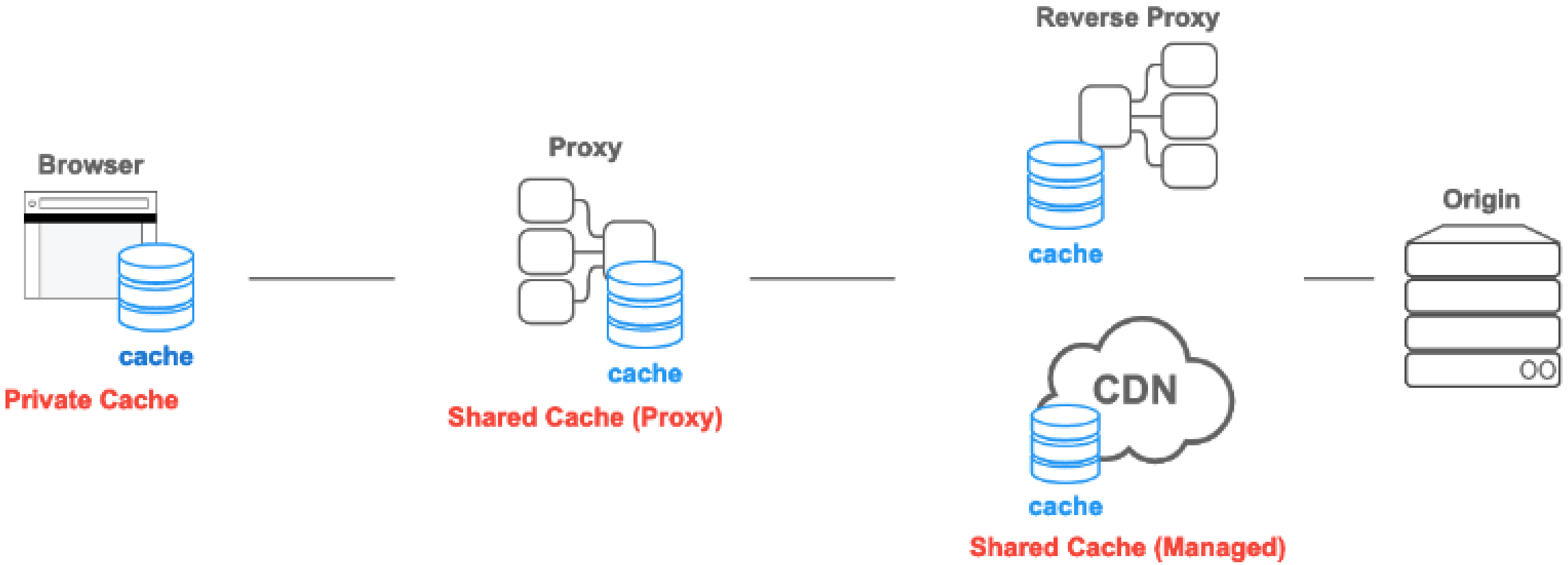
Cache

Stores the response of a request and reuse the stored response

No need to send a request to server, closer the cache to the client, faster the response – browser cache

If no need to generate a response, server cache reduces the load on the serve

Types of cache



Cache control

```
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
```

Max-age tells how long the cache is valid

If cache is invalid, the browser can check with the server, where there any changes using ‘last-Modified’ time

HTTP Cookies

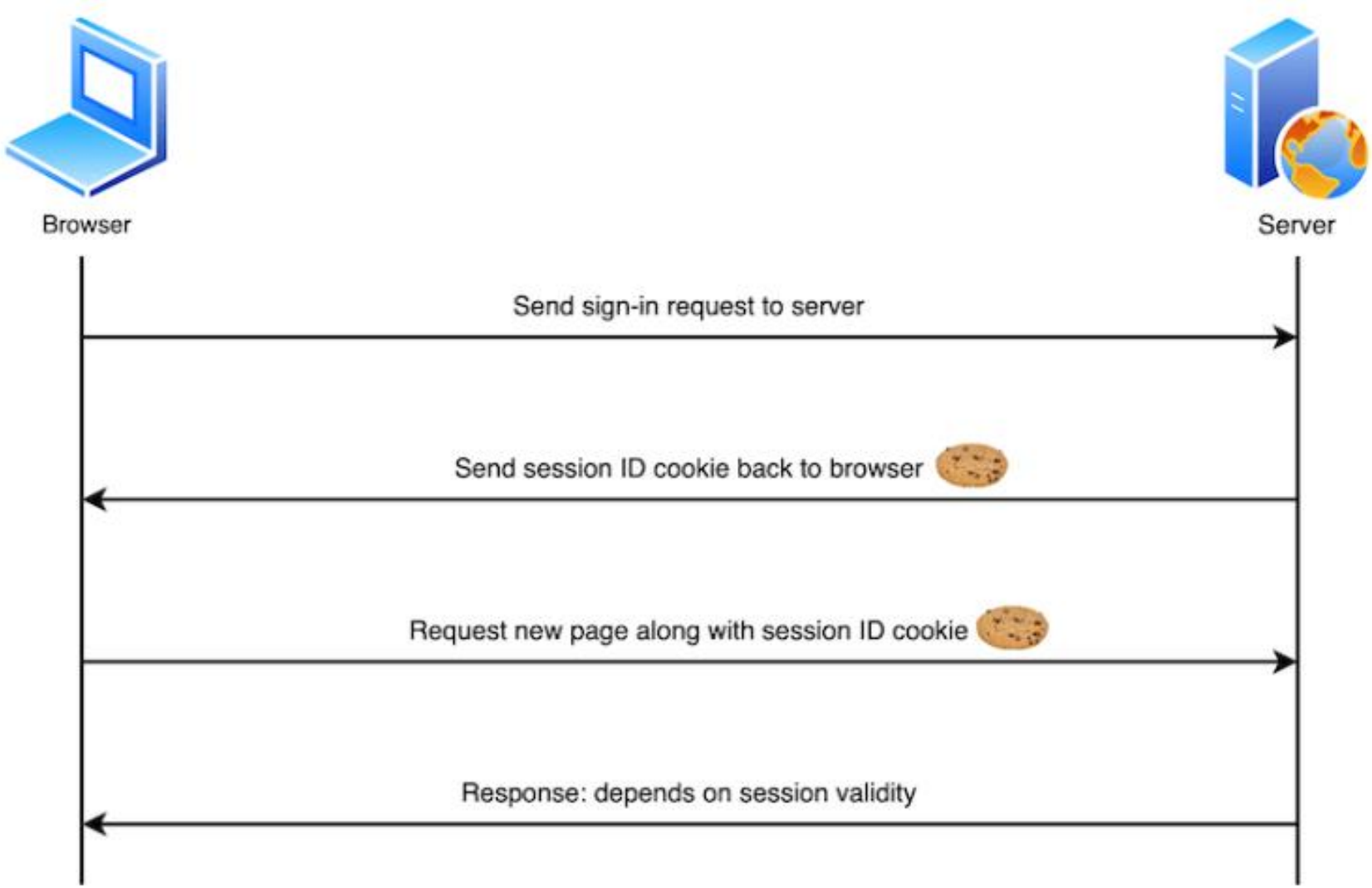
Small piece of information that server sends to a user's browser

Enable WebApps to remember state information (HTTP is stateless)

In early days, used for data storage at client side

Created/deleted/updated using HTTP response

Simple user sign-in system



Sources

1. MDN Web docs - HTTP
2. Stanford CS 142 Web Application – Lectures
3. Caching Tutorial by Mark Nottingham