

# **HTTP**

## **Hypertext Transfer Protocol**

**COMP423 / 2024 Fall / CL13**

**Kris Jordan / The University of North Carolina at Chapel Hill**

**Next Quiz: Wednesday 10/2**

# How does your client-side application interact with the server-side out on the internet?

**Your Machine**

**HTTP Client**

Web Browser  
cURL  
HTTP Client Libraries  
and more...

**????**

**The Internet**

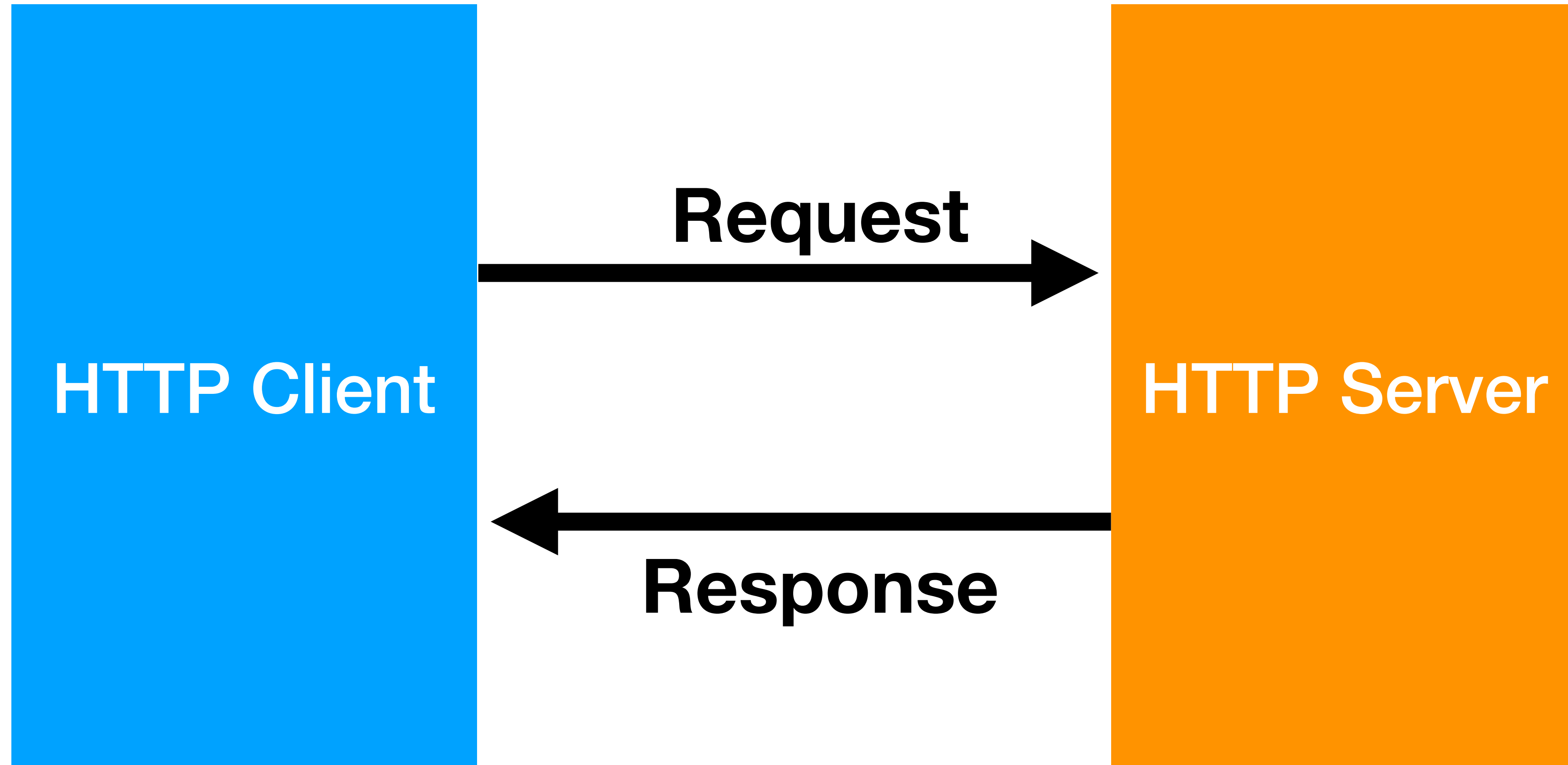
**HTTP Server**

Web Servers  
Application Servers  
Reverse Proxies  
and more...

# HTTP Protocol (Simplified)

**Your Machine**

**The Internet**



# HTTP Protocol

**Your Machine**

Client-side  
Application  
Code

Browser /  
HTTP Library

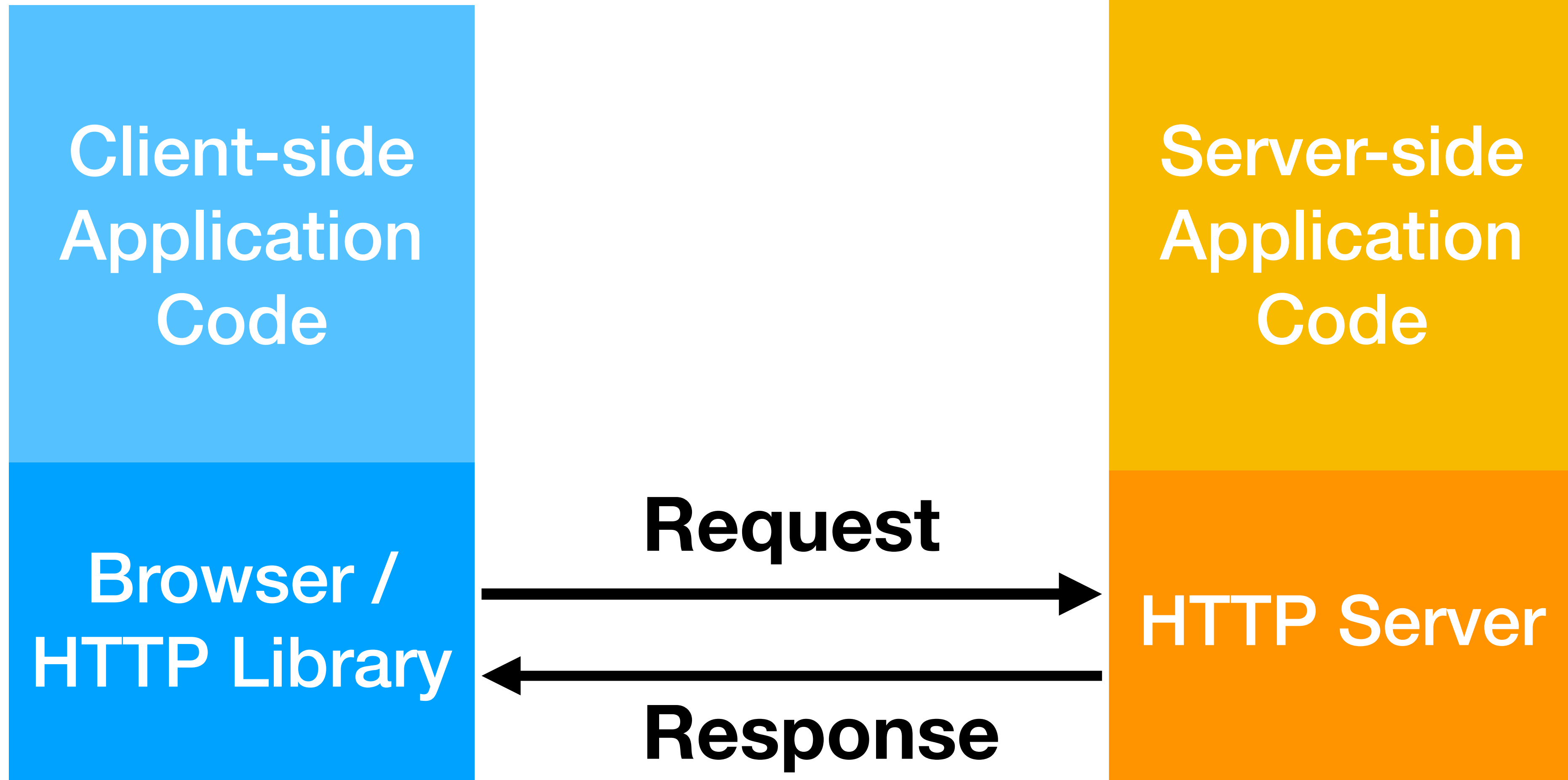
**The Internet**

Server-side  
Application  
Code

HTTP Server

**Request**

**Response**



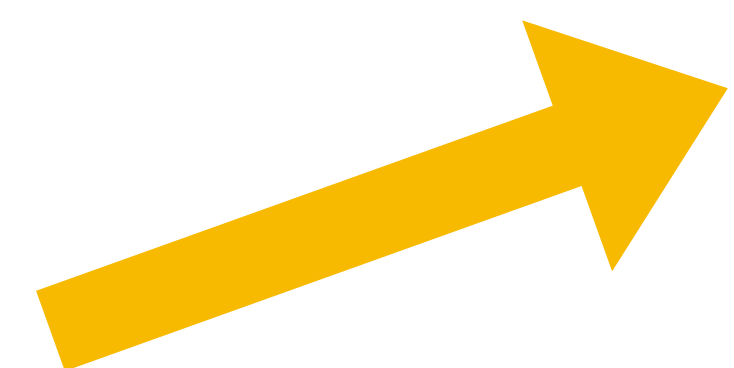
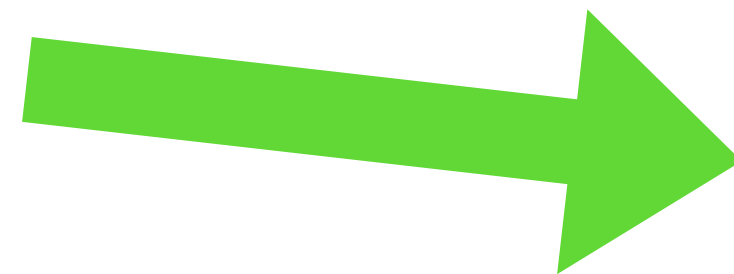
# Scavenger Hunt

## On a team board...

1. What are **4 Common Verbs** or ***METHODS*** used in the HTTP protocol?
2. What about **Content-type**? What is the **Accepts** header?

# An HTTP Request Has

- Request Line
  - Method (GET/POST/PUT/DELETE)
  - Path
  - HTTP Version (e.g. HTTP/1.1)
- Headers
  - Key-Value string pairs delimited by “:”s and separated by new lines
- Body
  - If the request is giving content to the server (such as a form submission, application “post” or “save”)





```
POST /tweet HTTP/1.1

Host: api.twitter.com
Content-Type: application/json
Accept: application/json
Authorization: <JWT_TOKEN>

{“message”:“Hello, World”}
```

# A brief story about accepts headers...

WebKit browsers  
From sources across the web

 Firefox ▼  Safari

## Bug 27267 - HTTP Accept header gives preference of XML over HTML

**Status:** RESOLVED FIXED  
**Reported:** 2009-07-14 10:33 PDT by Kris Jordan  
**Alias:** None  
**Modified:** 2011-03-10 17:05 PST ([History](#))  
**Product:** WebKit  
**CC List:** 10 users ([show](#))  
**Component:** WebCore Misc. ([show other bugs](#))  
**Version:** 528+ (Nightly build)  
**See Also:**  
**Hardware:** PC All

```
application/xml,  
application/xhtml+xml,  
text/html;q=0.9,  
text/plain;q=0.8,  
image/png,  
*/*;q=0.5
```

Parsed and prioritized:

1. application/xml
2. application/xhtml+xml
3. image/png
4. text/html
5. text/plain
6. \*/\*

```
128 #if ENABLE(XHTMLMP)  
129 static const char defaultAcceptHeader[] = "application/vnd.wap.xhtml+xml,application/xhtml+xml;profile='http://www.wapforum.org/xhtml',text/html,application/xml;q=0.9,*/*;q=0.8";  
130 #else  
131 static const char defaultAcceptHeader[] = "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";  
132 #endif
```

 [main](#) ▼ [WebKit / Source / WebCore / loader / FrameLoader.cpp](#) 

```
Copyright (C) 2006-2022 Apple Inc. All rights reserved.  
Copyright (C) 2008 Nokia Corporation and/or its subsidiary(-ies)  
Copyright (C) 2008, 2009 Torch Mobile Inc. All rights reserved. (ht  
Copyright (C) 2008 Alp Toker <alp@atoker.com>  
Copyright (C) Research In Motion Limited 2009. All rights reserved.  
Copyright (C) 2011 Kris Jordan <krisjordan@gmail.com>  
Copyright (C) 2011 Google Inc. All rights reserved.
```



# Scavenger Hunt

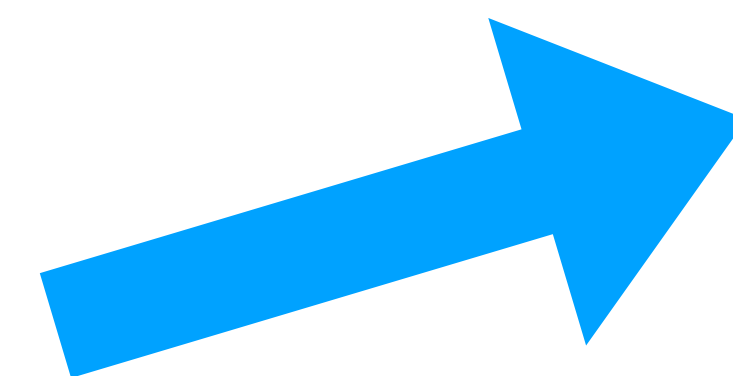
**On a team board, respond**

**Submit to Gradescope as group of up to 4x**

1. What is the meaning of 200-level **HTTP response codes**? Find 2 examples.
2. What is the meaning of 300-level **HTTP response codes**? Find 2 examples.
3. What are the meanings of 400-level **HTTP response codes**?  
500-level? Find 1 example in each range.

# An HTTP Response Has

- Status Line
  - HTTP Version
  - Status Code (e.g. 200, 404, 500)
  - Reason Phrase (e.g. Ok, Not Found, Internal Server Error)
- Headers
  - Just like a request, key-value pairs delimited by ‘:’s and separated by new lines
- Response Body
  - Optional, but more common than in the client. For example, when a web page is requested its HTML comprises the response body.



HTTP/1.1 404 Not Found

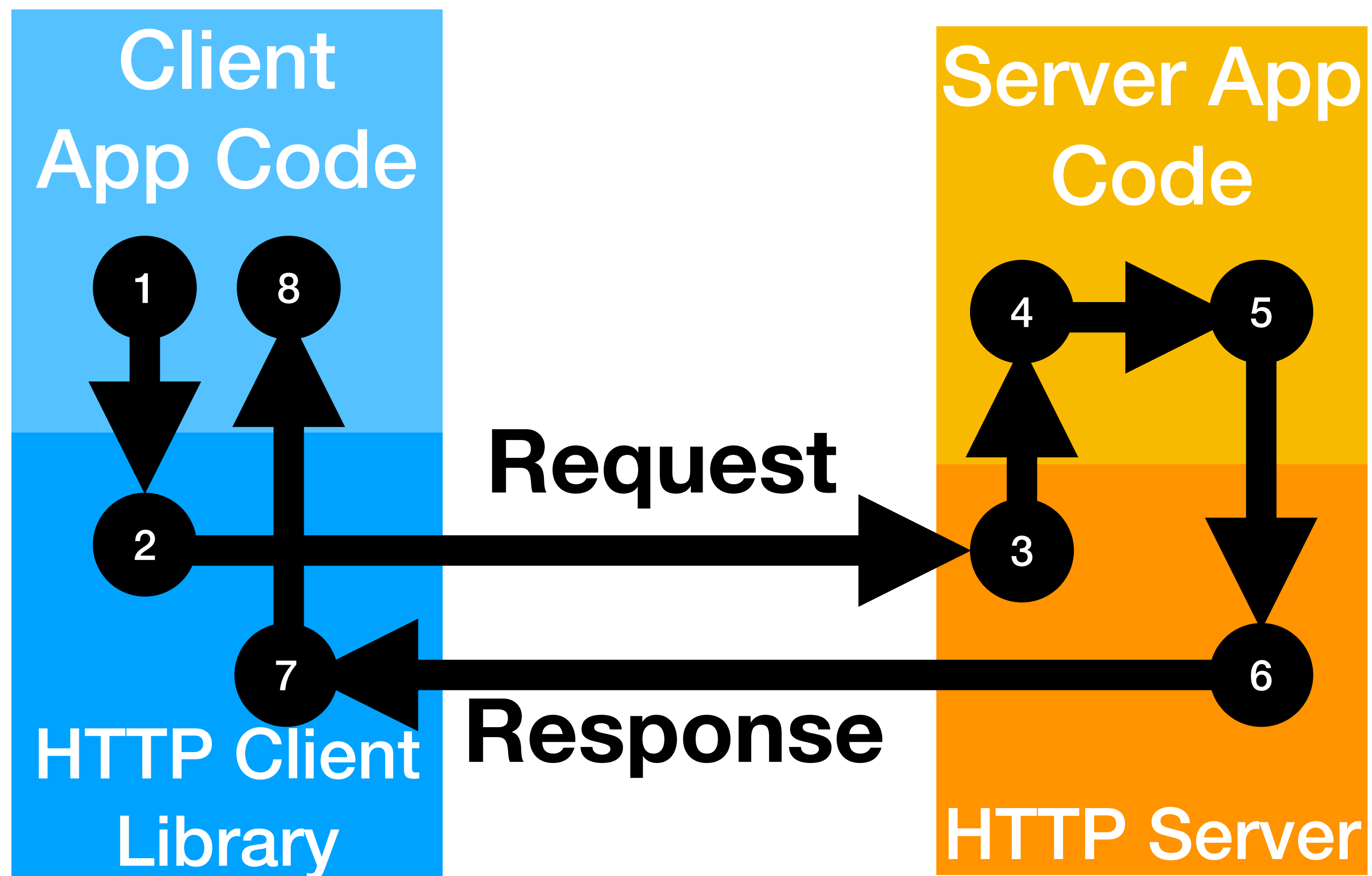
Host: api.twitter.com  
Content-Type: text/html

```
<!doctype html>  
<html>  
  <head>  
    <title>Page Not Found</title>  
    ...
```

# HTTP Protocol for Full-stack Apps

## Your Machine

## The Internet



1. Your app code calls out to HTTP Client Library module. Subscribes for notification of result.
2. HTTP Client Library transforms your request to valid HTTP protocol message, handles connection to server, sends request.
3. HTTP Server receives request, parses it, dispatches out to your server application code.
4. Your server application receives a function/method call with relevant data from request.
5. Your application logic handles request and returns info relevant to response.
6. HTTP server transforms response into valid HTTP response, sends it back to client.
7. HTTP Client Library parses HTTP response and notifies the subscribed client code.
8. Your client can handles the subscription notification of response from the server.

HTTP

Web Apps

This is where we are now focusing in this unit on front-end client development!

This is where we will go in the next unit on backend-end API development!



Notice on the client-side the *request invocation* and *response handling* are asynchronous!

This enables your application to *do other things*, or not block, while waiting on the server to process a request which can take an undetermined amount of time.

# Using Angular's HttpClient

- HttpClient can be used in Services via Dependency Injection:
  - `constructor(private http: HttpClient)`
- It has generic methods for common HTTP Request Verbs, e.g.:
  - `http.get<Profile>('/api/profile')`
  - `http.put<Profile>('/api/profile', updatedProfile)`
- Like axios from EX01, these requests are *asynchronous*. Unlike axios, these methods return *lazy, reactive observables*. Their asynchronous handling is slightly different and cannot be awaited:

```
http.get<Profile>('/api/profile').subscribe({  
  next: (profile) => /* Do something with profile */,  
  error: (err) => /* Handle HTTP error... */  
});
```