#2

Data Platform Development

Web Communication & Protocol

"Hur kommunicerar vi över nätet?"

Table of Contents


Overview

Moduler:

- OSI (bonus)
- TCP / UDP
- HTTP / HTTPS
- Localhost & Ports
- Status codes
- Request & Responses
- HTTP methods
- Content-types
- Inspecting elements

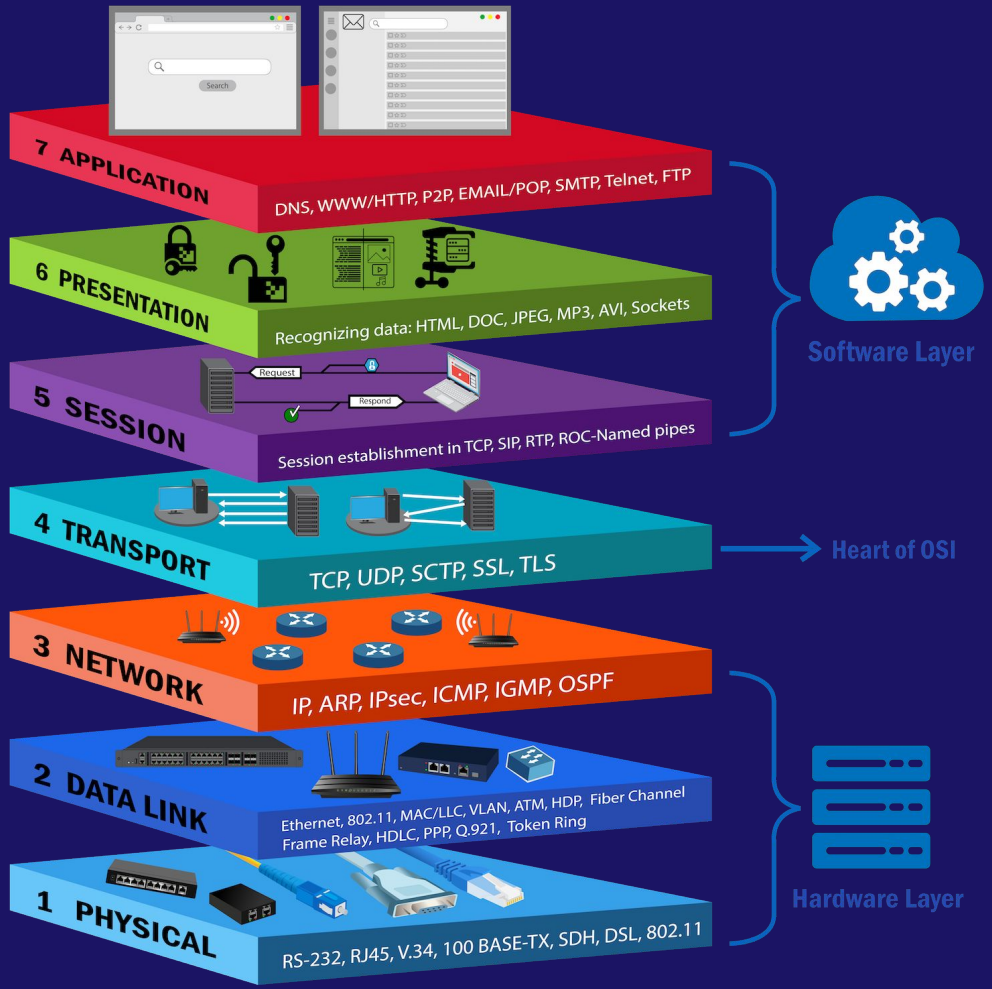
Utbildningsmoment

- Dataplatfformar, bakgrund och syfte
- Git och github i teamkontext
- Komponenter och teknologier i en data platform ✅
- ETL vs ELT
- Utveckling av mjukvara mot databaser ✅
- Använda Python mot relationsdatabaser och andra datakällor såsom csv, http xml/json
- Använda Python mot realtidsdataströmmar såsom message queues och/eller event streaming platforms
- Använda Python och för att rensa, validera och transformera data
- Workflow processer ✅

02

Nätet, Protokoll & Status koder


OSI Model (Bonus)
Open System
Interconnection


7 APPLICATION
DNS, WWW/HTTP, P2P, EMAIL/POP, SMTP, Telnet, FTP
6 PRESENTATION
Recognizing data: HTML, DOC, JPEG, MP3, AVI, Sockets
Request
Respond
5 SESSION
Session establishment in TCP, SIP, RTP, ROC-Named pipes
Software Layer
4 TRANSPORT
TCP, UDP, SCTP, SSL, TLS
Heart of OSI
3 NETWORK
IP, ARP, IPsec, ICMP, IGMP, OSPF
2 DATA LINK
Ethernet, 802.11, MAC/LLC, VLAN, ATM, HDP, Fiber Channel
Frame Relay, HDLC, PPP, Q.921, Token Ring
1 PHYSICAL
RS-232, RJ45, V.34, 100 BASE-TX, SDH, DSL, 802.11
Hardware Layer


Protokoll
TCP & UDP

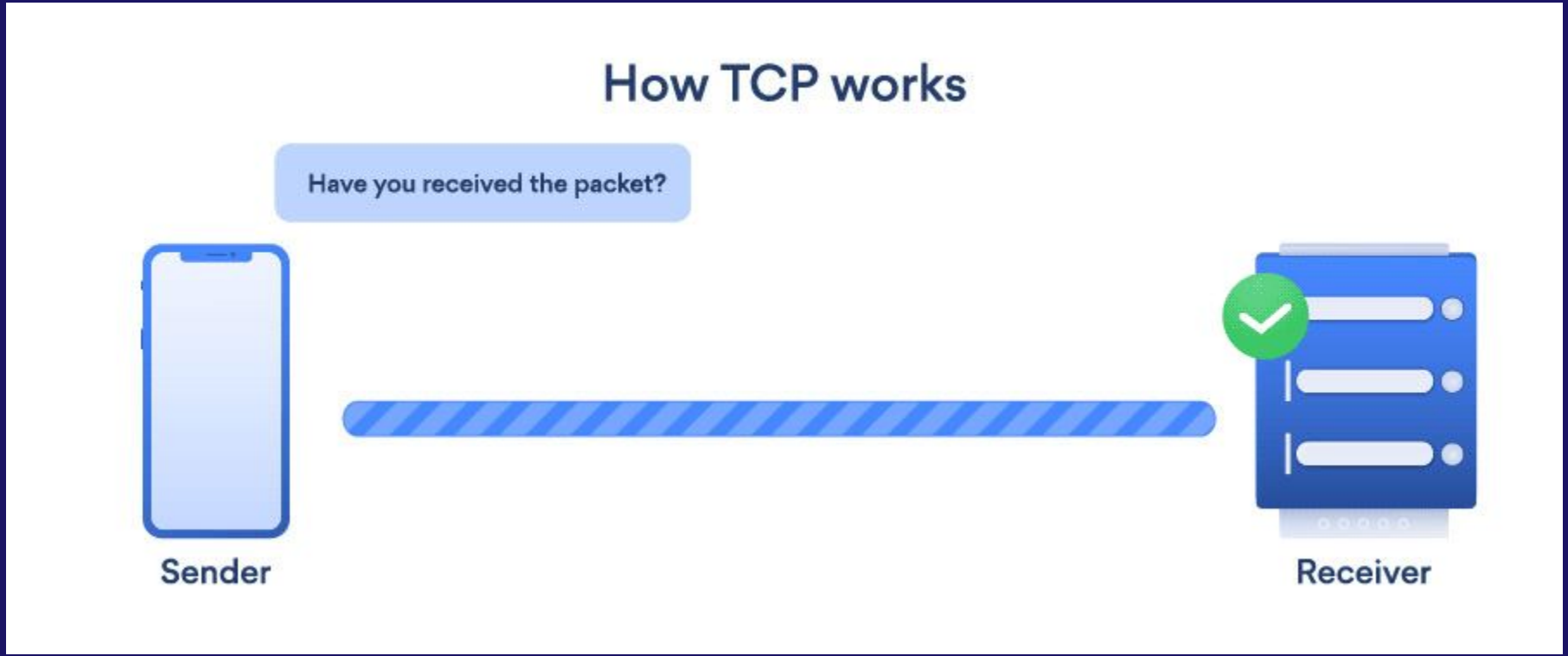
Protocol?

När man diskuterar **‘handshakes’** så är det viktigt att förstå hur vi tillämpar uppkopplingar och kommunicerar.

Det finns två väldigt vanliga protokoll… **TCP & UDP**!

UDP = User Datagram Protocol
TCP = Transmission Control Protocol

Protocol (TCP)



Protocol (TCP)

TCP är till för att skapa en stabil och kontrollerad koppling där vi försäkrar om att uppkopplingen och data ej råkar försvinna!

Data skickas men inte alltid inom rätt ordning... TCP hanterar detta!

Protocol (TCP)

[A, B, C, D]

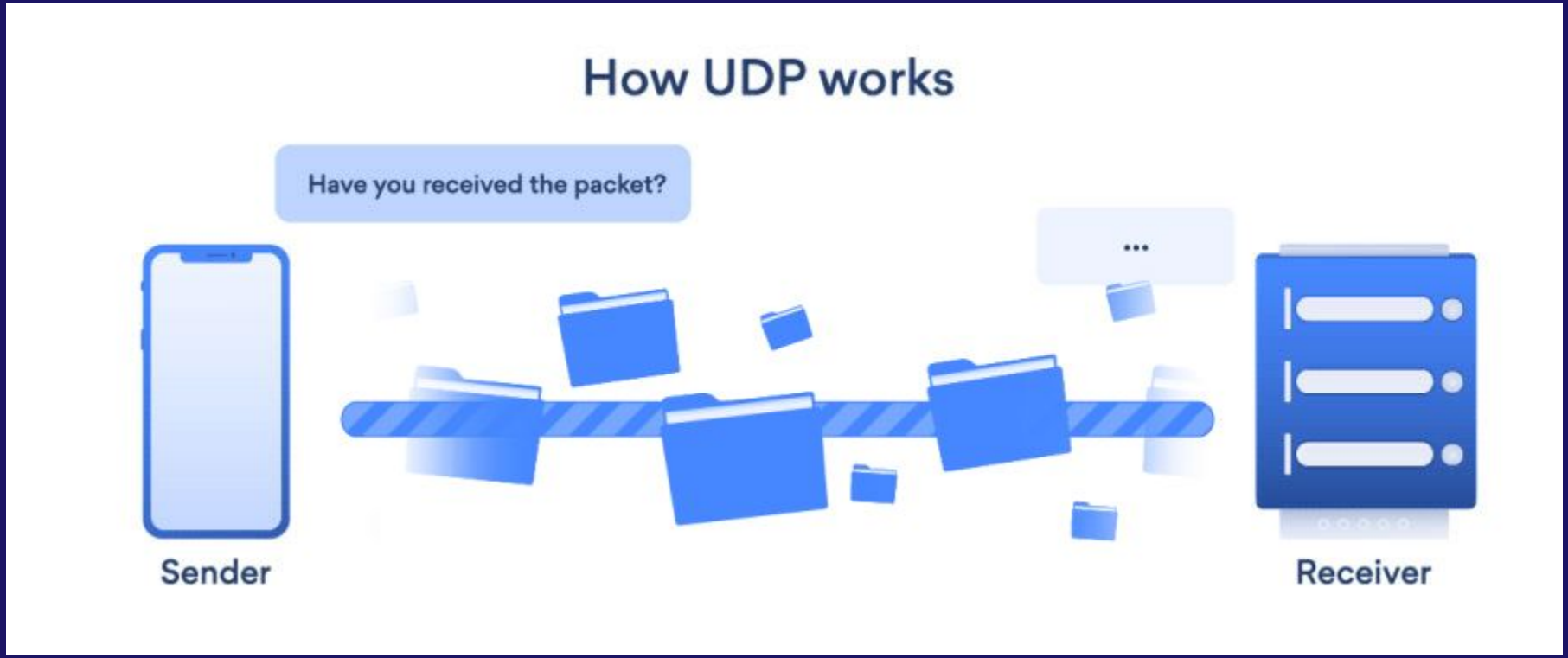
[B, D, C, A]

[A, B, C, D]

Protocol (TCP)

Vi förlorar heller ingen data på vägen, vilket annars hade kunnat göra en fil ofullständig (korrumperad fil).

Protocol
(UDP – No Order)

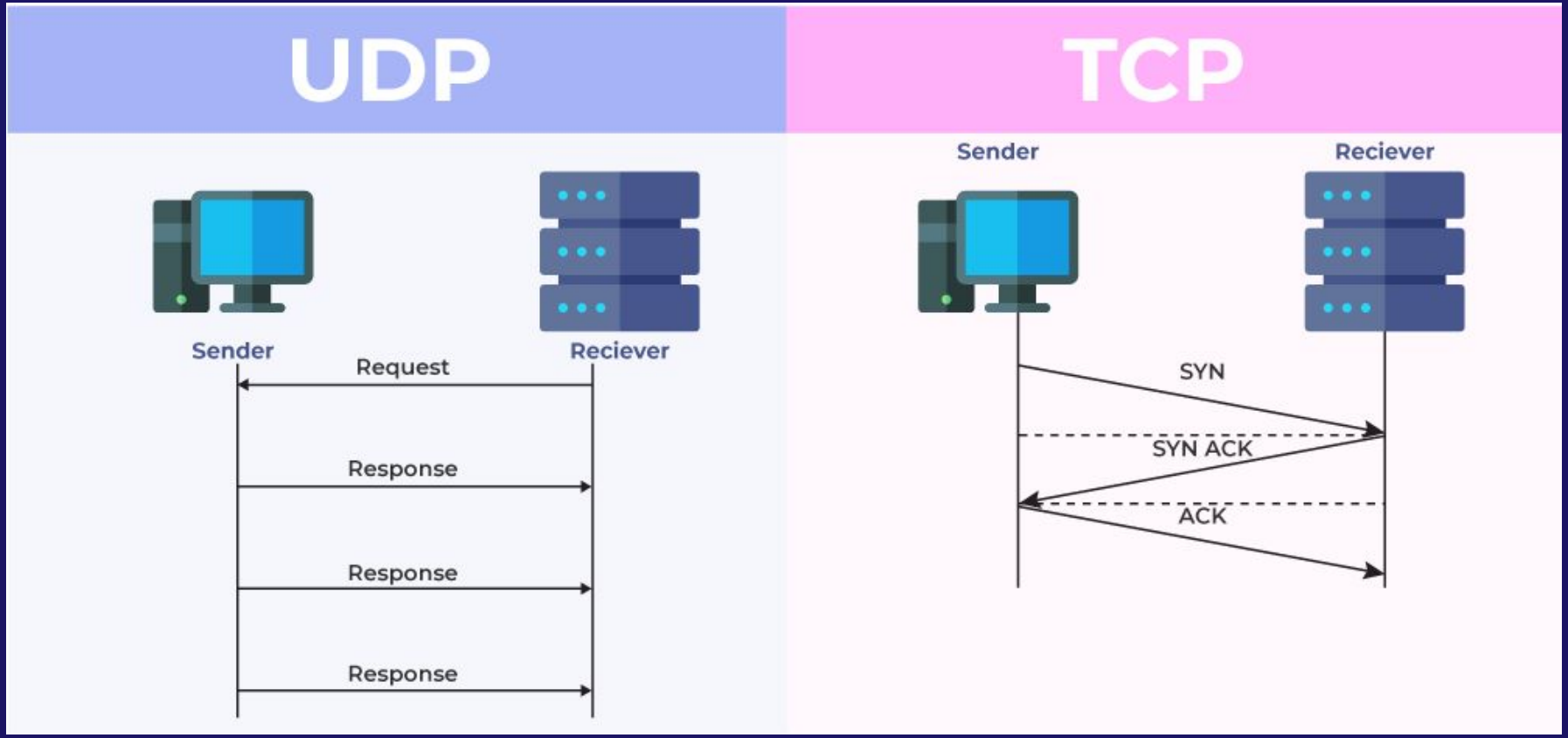


Protocol (UDP)

UDP är mer **sporadisk och kaotisk**,
men effektiv på det sätt att den är snabb.

Ett bra val där det inte gör något om vi förlorar data. T.ex inom streaming eller ‘voice calls’

Protocol?


UDP
TCP
Sender
Reciever
Request
Response
Response
Response
Sender
Reciever
SYN
SYN ACK
ACK

Protocol

TCP

- Slower but more reliable transfers
- Typical Applications:
  - File Transfer Protocol (FTP)
  - Web Browsing
  - Email

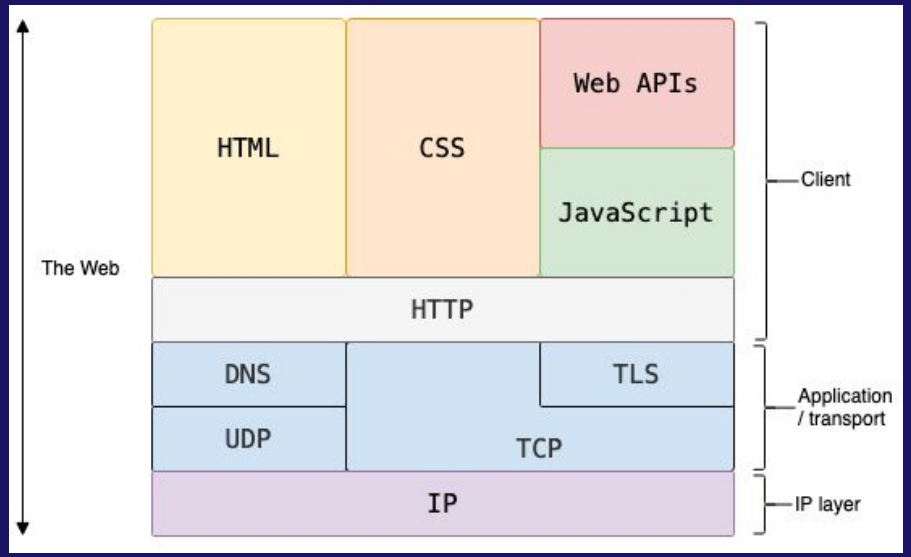
unicast

UDP

- Faster but not guaranteed transfers ("best effort")
- Typical Applications:
  - Live Streaming
  - Online Games
  - VoIP

unicast multicast broadcast

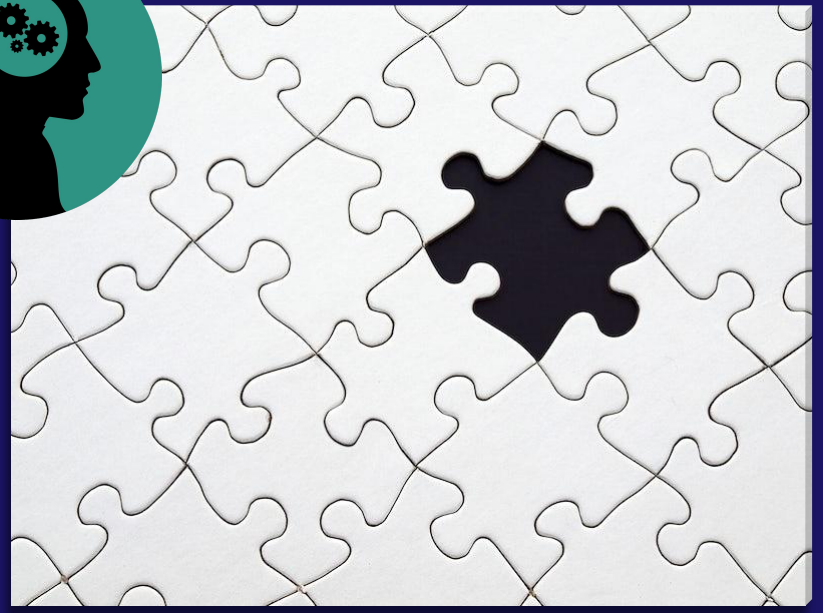
HTTP & HTTPS?



Både HTTP & HTTPS använder sig utav TCP protokoll som standard

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

Frågor?




Kort om:
HTTP & HTTPS

HTTP
(What is it?)

Hypertext Transfer Protocol

"HTTP sker inom det övre lagret inom OSI modellen (#7), och skickar meddelanden mellan web klienter och servrar "

TCP och HTTP är starkt och nära relaterade än idag!

HTTP är det protokoll som används för kommunikation på 'World Wide Web'.

Det är grunden för datakommunikation på internet

HTTP
(TCP + HTTP?)

Car = messages made with HTTP

Bridge = TCP

HTTPS
(What is it?)

Hypertext Transfer Protocol Secure

"Likt HTTP, fast med kryptering.
Säkrar användarnas data genom SSL certifikat!"

Helen

HTTP

http://www.example.com

password: abc123

Without password encryption

Hacker see "abc123"

Carol

HTTPS

https://www.example.com

password: abc123

With password encryption

Hacker see "xyaerXzabc"

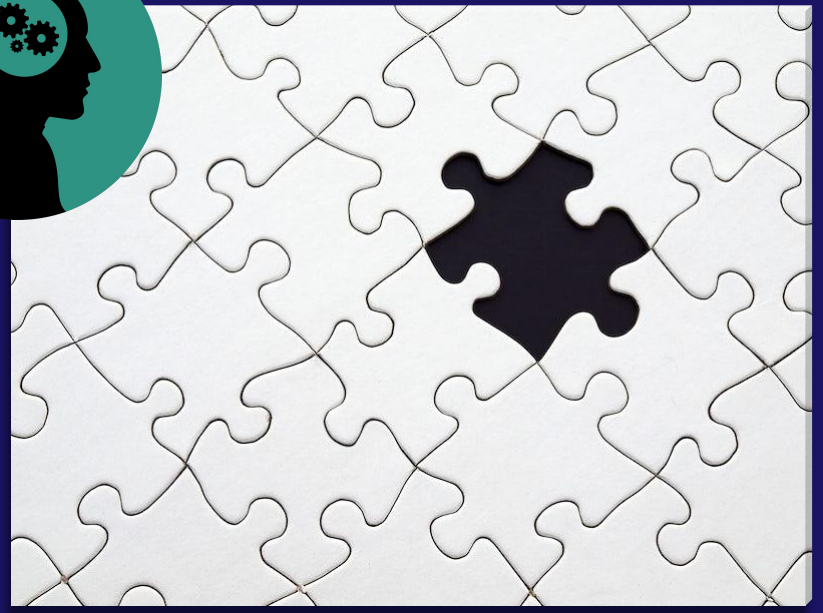
Problem

Vad är ett protokoll till för?

Solution

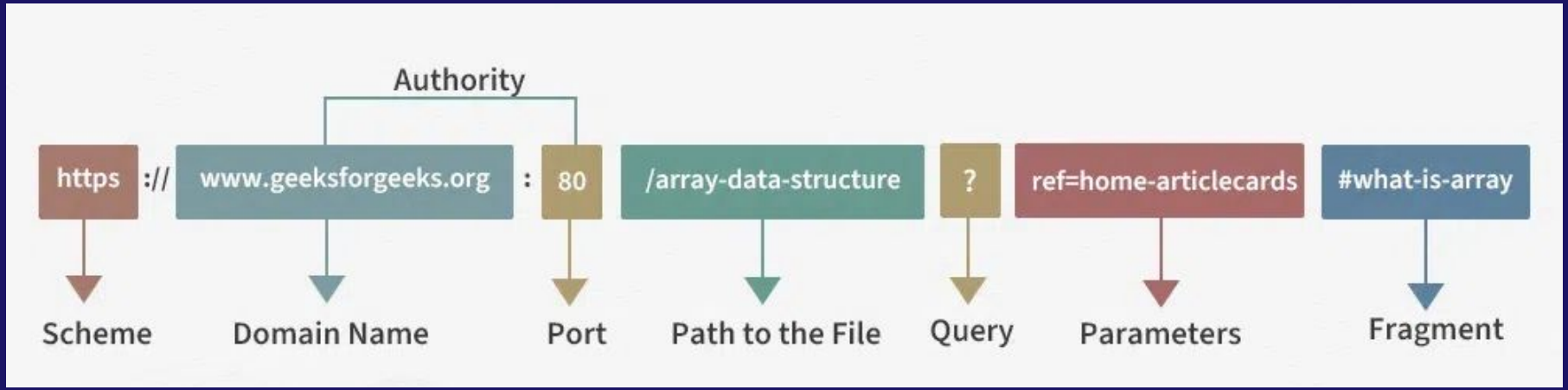
Med protokoll som TCP, kan vi säkra en stadig koppling mellan klient och server och med HTTP kan vi skicka information mellan web klienter och servrar

Frågor?




Localhost
And ports

URL
(Structure)



Source: https://www.geeksforgeeks.org/javascript/what-is-url-uniform-resource-locator/

Localhost (Why?)

When you call an IP address on your computer, you try to contact another computer on the internet, but when you call the IP address 127.0.0.1, you are communicating with the local host. **Localhost** is always your computer. Your computer is talking to itself when you call the local host. Your computer does not always directly identify the local host. Within your network, **localhost** (127.0.0.1) allows your computer to communicate with itself, while **192.168.0.1** is a private IP address used for local network devices. This is usually dynamically assigned by the internet service provider (ISP). Localhost can be seen as a server that is used on your computer.

Source: https://www.geeksforgeeks.org/computer-networks/what-is-local-host/

Localhost (Why?)

Personal note: *Showcase an easy Spring Example*

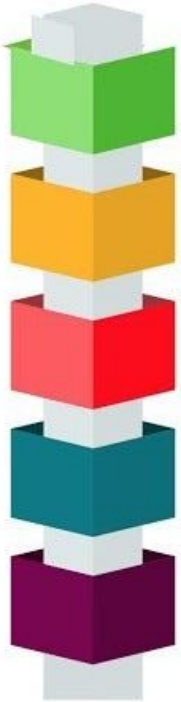

Status Codes

Status Codes
(What are they?)

Status koder är ett utmärkt sätt för oss att meddela om stadiet av ett 'request' och hur det gick.

Tänk er själva - vad händer om vi skriver t.ex google.com/hallådär
Vad kommer hända?

HTTP Status Codes



1XX
INFORMATIONAL

2XX
SUCCESS

3XX
REDIRECTION

4XX
CLIENT ERROR

5XX
SERVER ERROR

Status Codes (bonus) (1xx)

Use to provide informational responses, typically indicating that the request is continuing or the server is waiting for the client to send a request header.

- **100 Continue (bonus)**
  - The server has received the initial part of the request and is indicating that the client can proceed with the rest of the request.
- **101 Switching Protocols (bonus)**
  - The server is indicating that it is changing protocols and the client should switch to the new protocol specified in the response's "Upgrade" header.
- **102 Processing (bonus)**
  - The server has received and is processing the request but has not yet completed the action. This is often used in conjunction with asynchronous processing.

Status Codes (2xx)

Use when the request was successful and the server successfully processed the request.

- **200 OK**
  - Indicates that the request was successful. The server successfully processed the request, and the response body may contain the requested data.
- **201 Created**
  - Indicates that the request has been fulfilled, resulting in the creation of a new resource. Typically used in POST requests to create a new resource.
- **204 No Content**
  - Indicates that the server successfully processed the request but there is no additional content to send in the response body. It's often used in operations where no data needs to be returned.

Status Codes (3xx)

Use when redirecting the client to a different resource or providing multiple choices for the requested resource.

- **300 Multiple Choices (bonus)**
  - Indicates multiple options for the requested resource. The client might choose a different resource or take further action.
- **301 Moved Permanently**
  - Indicates that the requested resource has been permanently moved to a new location. Clients should update their bookmarks or links.
- **304 Not Modified (bonus)**
  - Used in response to a conditional GET request. Indicates that the requested resource has not been modified since the last requested time.

Status Codes (4xx)

Use for client-related errors, such as malformed requests, authentication issues, or insufficient permissions.

- **400 Bad Request**
  - Indicates that the server could not understand the request due to malformed syntax, invalid request message framing, or deceptive request routing.
- **401 Unauthorized (bonus)**
  - Indicates that the request requires user authentication. The client must provide valid credentials to access the resource.
- **403 Forbidden (bonus)**
  - Indicates that the server understood the request but refuses to authorize it. The client does not have permission to access the requested resource.
- **404 Not Found**
  - Indicates that the server cannot find the requested resource. It's a common response for requests to non-existent URLs.

Status Codes (5xx)

Use for server-related errors, indicating that the server encountered an unexpected condition preventing it from fulfilling the request.

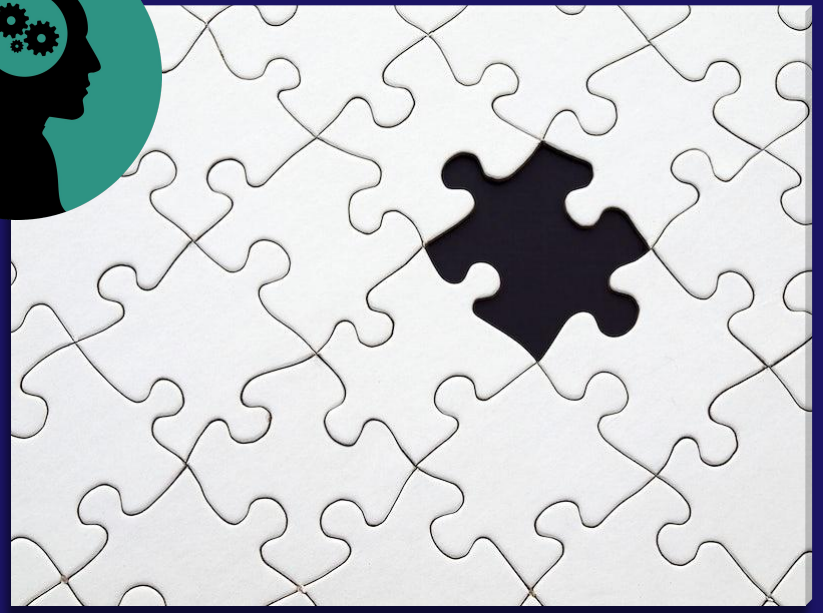
- **500 Internal Server Error**
  - Indicates that the server encountered an unexpected condition that prevented it from fulfilling the request. It's a generic error message when an unexpected condition occurs on the server.
- **502 Bad Gateway (bonus)**
  - Indicates that the server, while acting as a gateway or proxy, received an invalid response from an inbound server it accessed while attempting to fulfill the request.
- **503 Service Unavailable (bonus)**
  - Indicates that the server is not ready to handle the request. Common causes include server overloads or maintenance.

Status Codes (Source)

Read more!

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

Frågor?



03

Web Communication


Web Communication
Overview

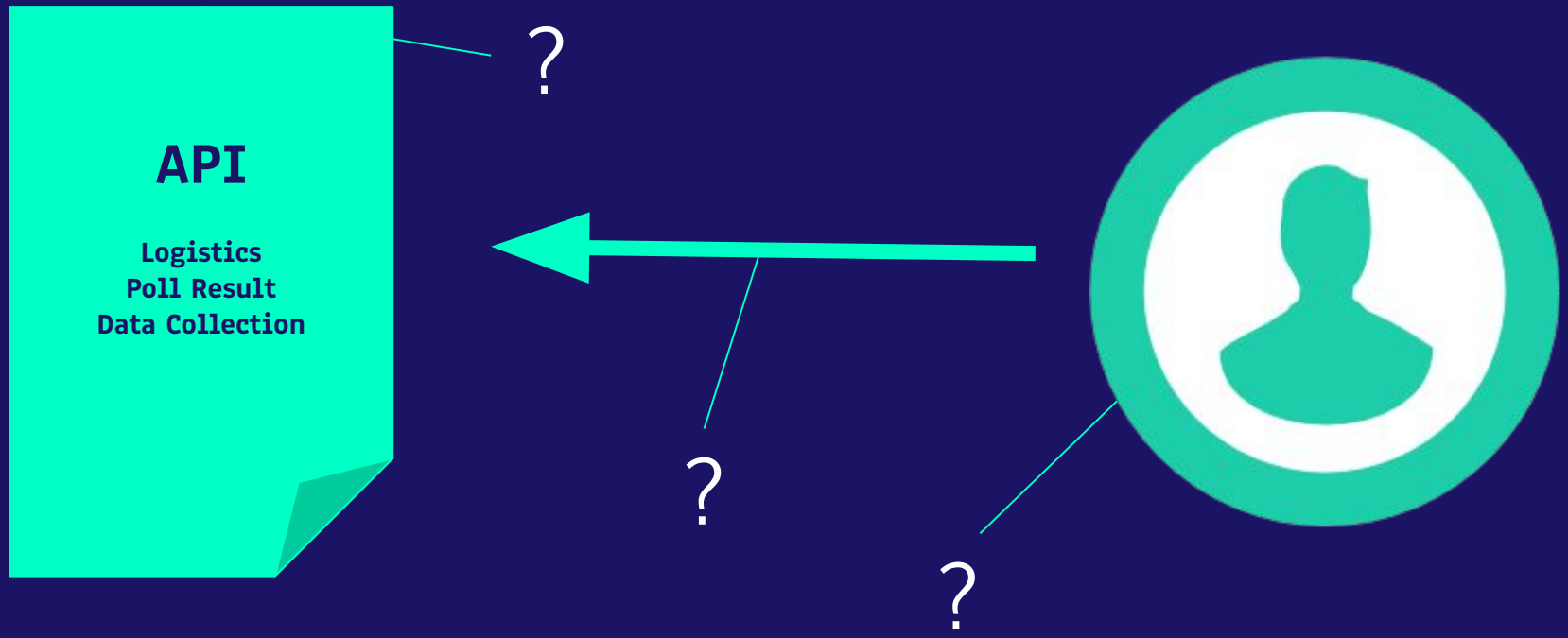
Web Communication
(Data)

API

Logistics
Poll Result
Data Collection

Frågor som uppstår...
Vem gör vad?
Vad händer bakom kulisserna?

Web Communication
(Terminologies…?)



Defining a Server
(What is it for?)

Security

Reliable Access

Data Management

Hosting

Centralized Management

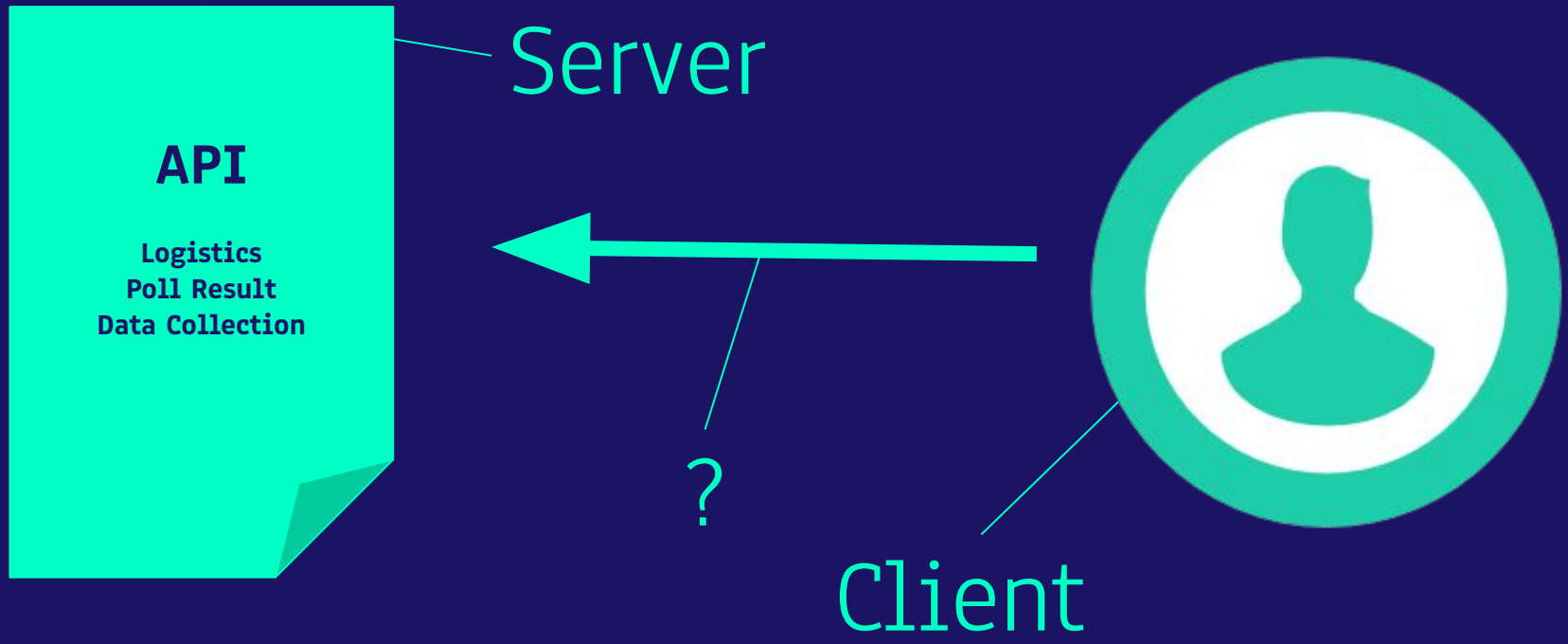
Scalability

Defining a Client
(Who's a client?)

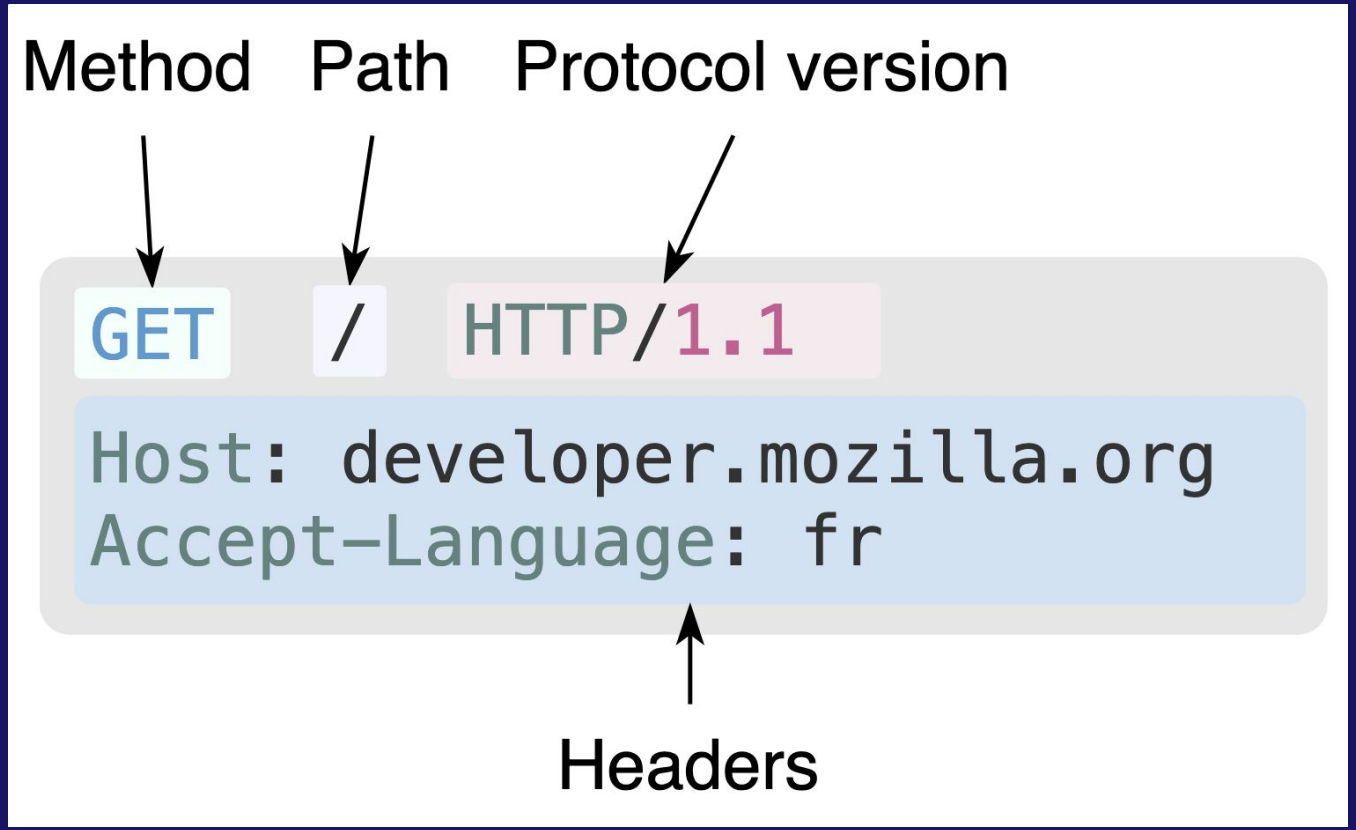
User AKA client

Visits website

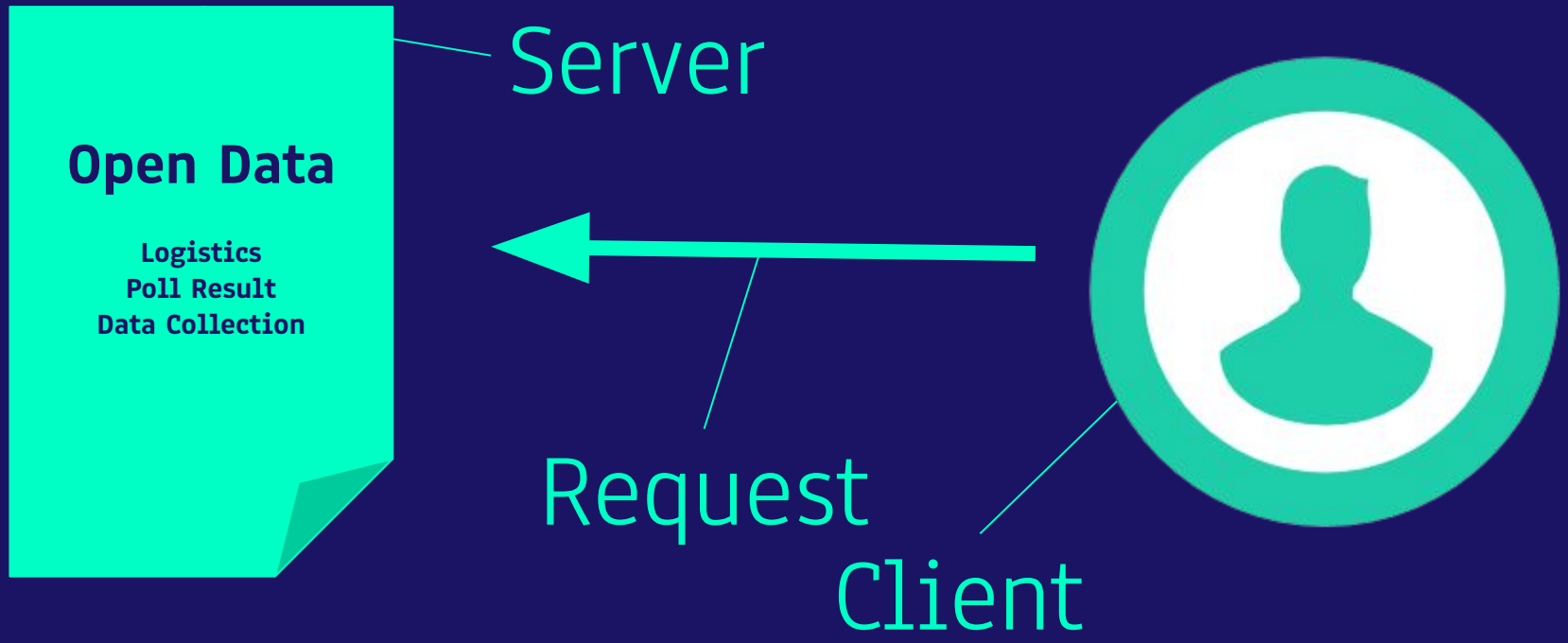
Questions…
(What REALLY happens…?)



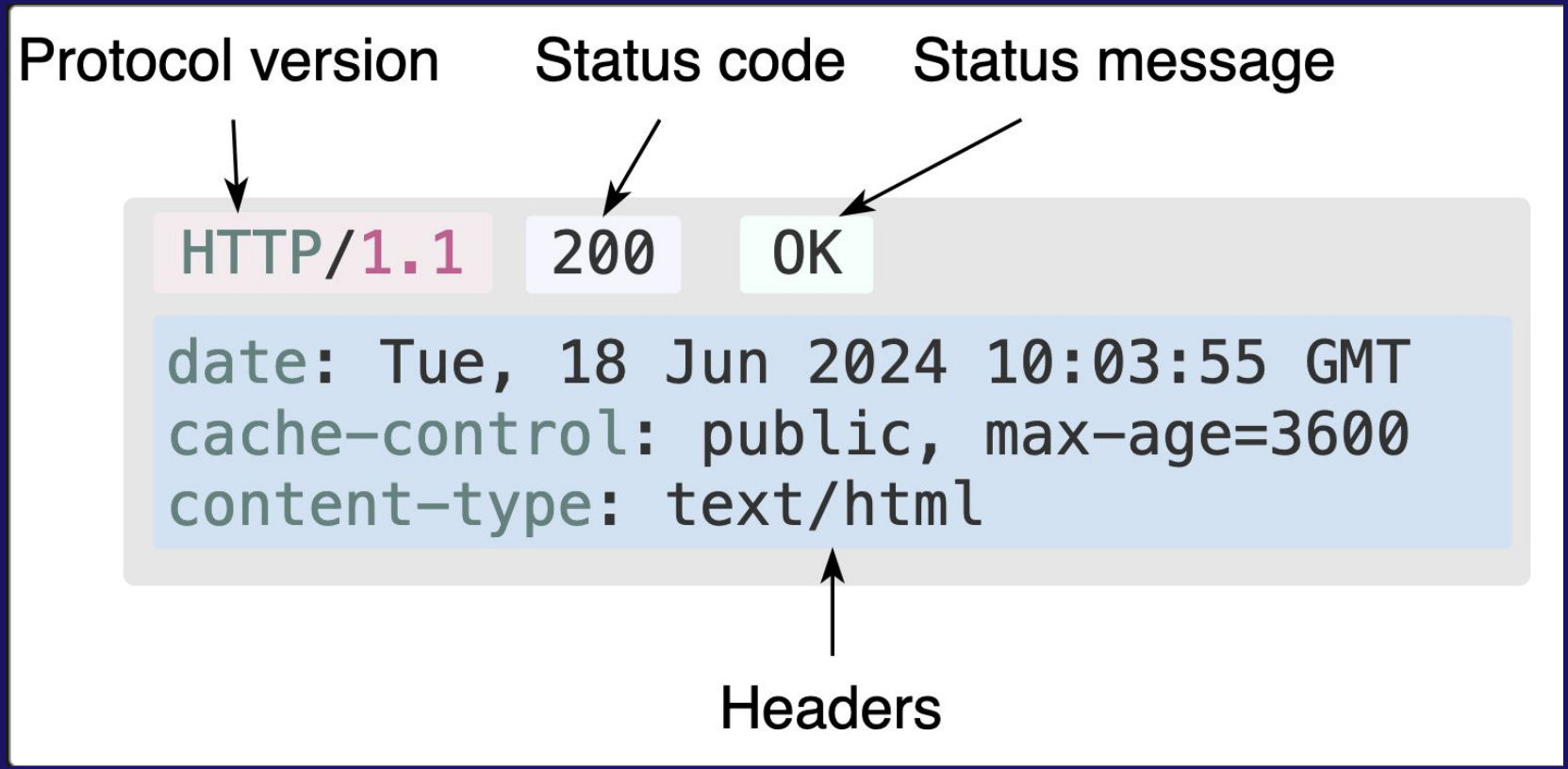
The Request


Method
Path
Protocol version
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
Headers

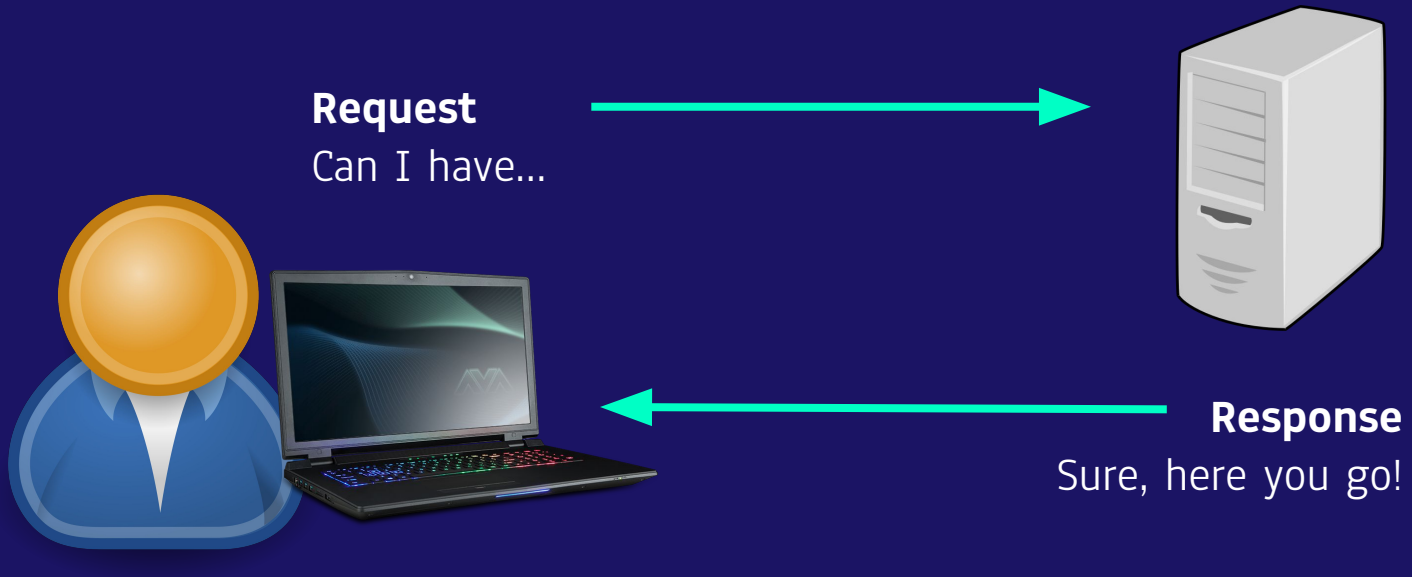
Questions…
(Answered!)



The Response



Communication
(Requests & Responses)



Communication
(In action)

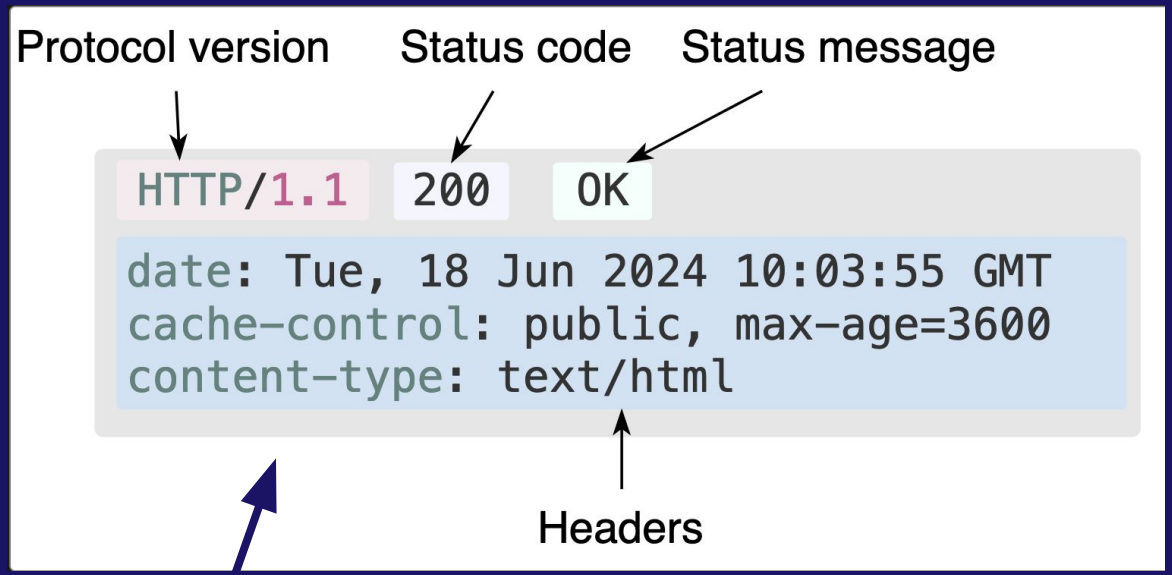

Content type:
Text, XML, JSON…

Vad vill vi beställa?

Vad ska vi förvänta oss för ‘typ av mat’ på restaurangen?

Så tillförs en ‘förfrågan’
Och en ‘respons’

Content-type
(Visualized)



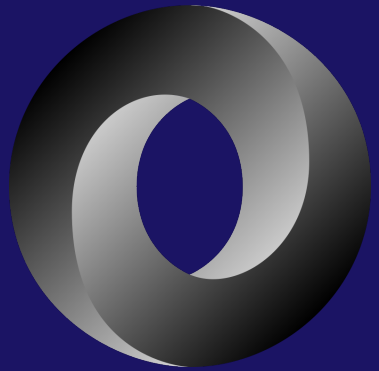
Klientens svarstyp bör överensstämma med vad servern förväntar sig

JSON?
(What is it?)

JSON (JavaScript Object Notation) *is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999*

Source: https://www.json.org/json-en.html

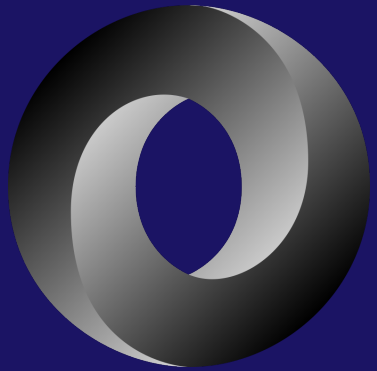
QUESTION
(Is this true?)



Är JSON data som vi skickar mellan klient och server?

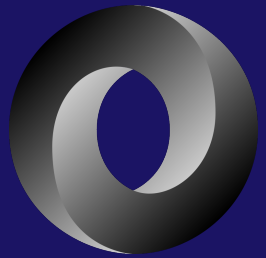
QUESTION
(Is this true?)

Är JSON data som vi skickar mellan klient och server?



YES

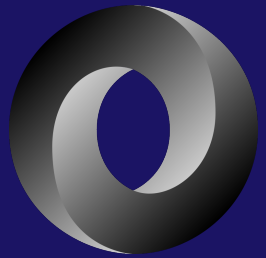
JSON
(Object)



```
{
  "name": "John Doe",
  "age": 30,
  "isStudent": false
}
```

Notera key-pair value, där vänster sida symboliserar nycklar med värden till höger

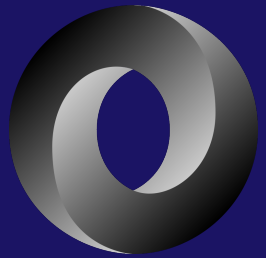
JSON
(Array)



```
{
  "colors": ["red", "green", "blue"]
}
```

Med datastrukturer så kan vi enkelt förse en strukturerad lista via ***[ ]***

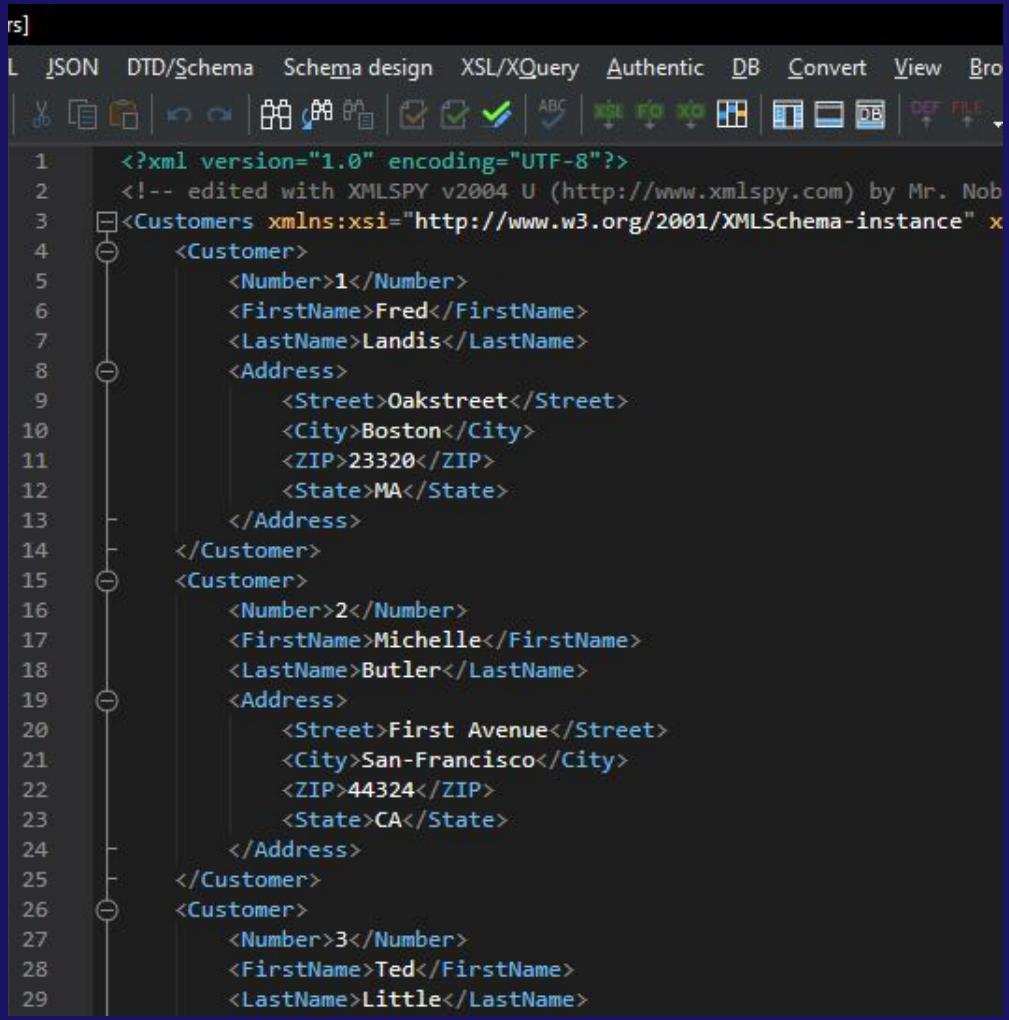
JSON
(Array of Objects)



Kombinera kunskaper som inkluderar objekt med array.

```
{
  "employees": [
    { "name": "Alice", "role": "Developer" },
    { "name": "Bob", "role": "Designer" },
    { "name": "Charlie", "role": "Manager" }
  ]
}
```

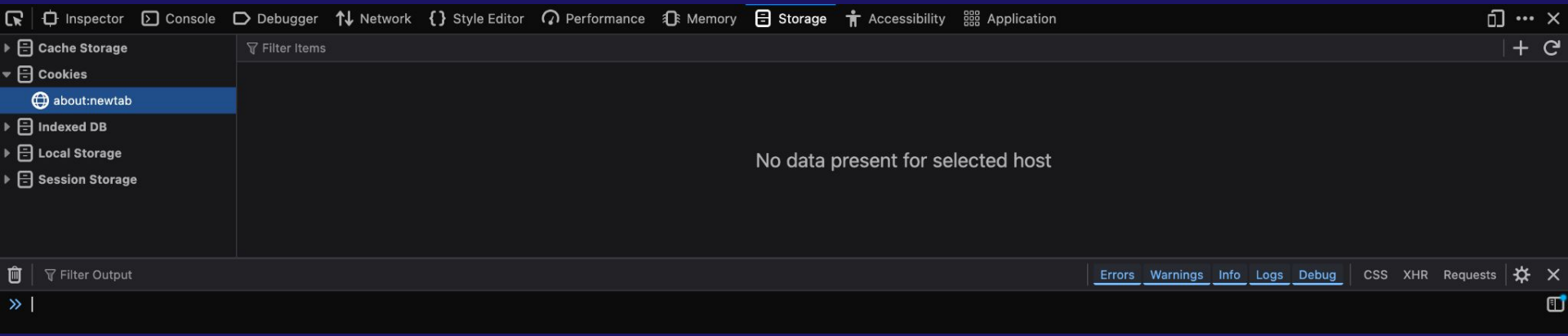
XML
(Array of Objects)




Inspecting
Postman & Browser

Inspector
(Web)

Mac: Cmd + Option + I
Windows: Ctrl + Shift + I



SMHI
(Cookies & Transparency)

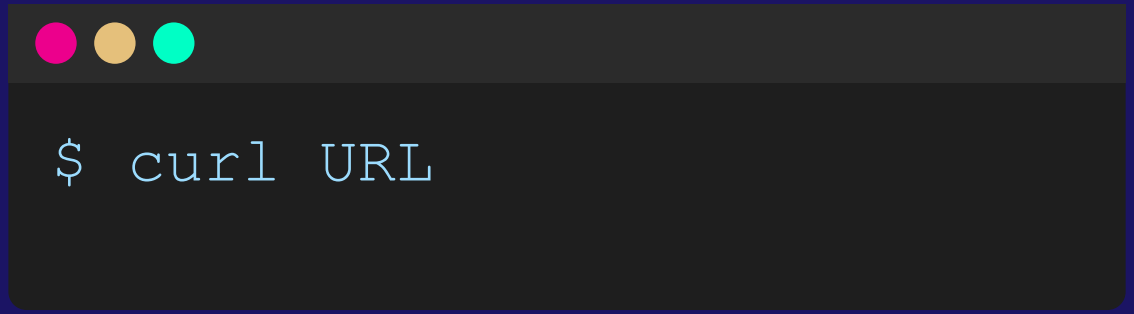
https://www.smhi.se/

SMHI
(Cookies & Transparency)

https://www.postman.com/

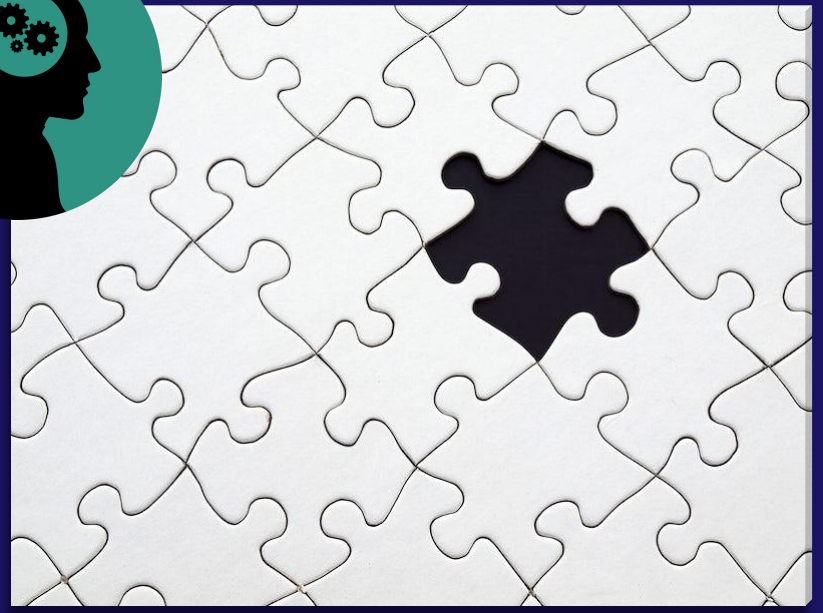
Vscode extension: Thunder Client
Terminal Alternative: curl

Perform Fetch
(Using curl)



Perform a 'get' request towards URL

Frågor?



04

Uppgifter & Eget Arbete

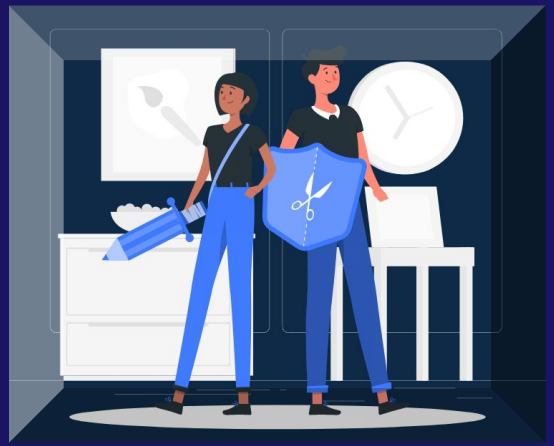
Uppgifter

Välkommen till första uppgiften!

Uppgifterna är till för att testa dina färdigheter och kunskaper för att både öva och repetera på det vi har arbetat med under föreläsningarna.

Dessa är **INTE** obligatoriska.

Men är ämnen ni kommer testas mot.

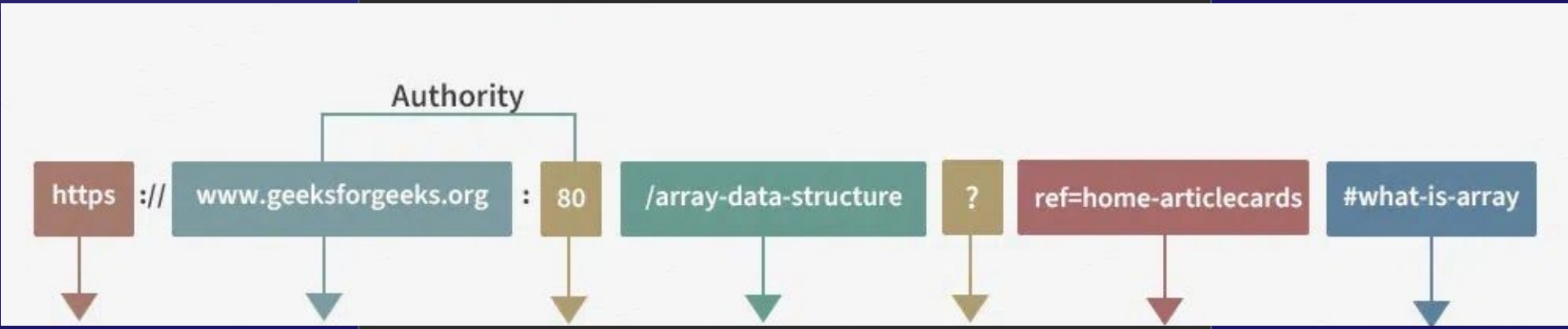


MINNS DU?

```
/*

    Vad händer när du skriver in en
    webbadress i webbläsaren?

    Vad är en HTTP metod och hur hänger
    detta ihop i sammanhanget?

*/
```

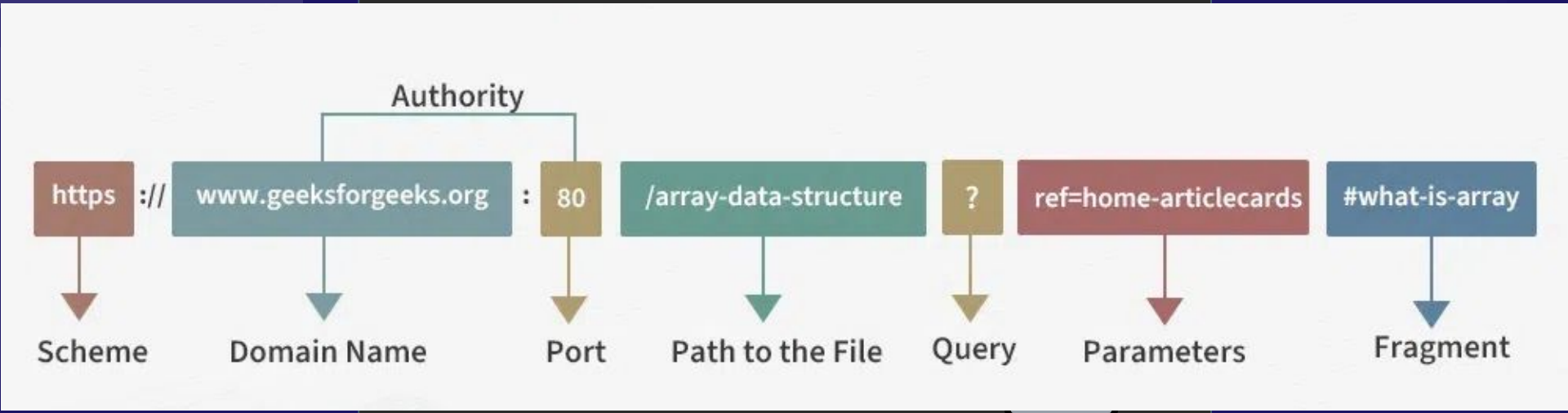
MINNS DU?

```
/*

    Kommer du ihåg vad respektive delar
    Inom URL strukturen, heter?

*/
```



FACIT

```
/*

    / kan också heta 'endpoint'

*/
```



```
1   // -Uppgift #1- //
2
3   /* INSTRUCTIONS
4
5       Öppna Google.com
6       Sök på något valfritt.
7
8       Vad är detta för typ av ‘request’ som
9       gjorts?
10
11  */
12
13  // HINT & Examples
14  hint("Öppna inspektören")
15
16
17
18
19
20
21
22
23
```

Kom igång enkelt med uppgift #1

```
1   // -Uppgift #2- //
2
3   /* INSTRUCTIONS
4
5       Kolla på URL'n
6       Vad tror du händer om ändrar på
7       Värdet som kommer efter = tecknet?
8
9       Vad står ?q= för?
10
11  */
12
13
14
15
16
17
18  // HINT & Examples
19
20  www.google.com/search?q=bananas&sca_esv=fc1
21
22
23
```

```
1   // -Uppgift #3- //
2
3   /* INSTRUCTIONS
4
5       Navigera in till:
6       https://random-d.uk/api
7
8       Läs instruktionerna och försök göra en
9       'request' mot den rätta endpoint.
10
11  */
12
13
14
15
16
17
18  // HINT & Examples
19  hint("Lägg till /random efter URL")
20
21
22
23
```

```
1   // -Uppgift #3- //
2
3   /* INSTRUCTIONS
4
5       Navigera in till:
6       https://random-d.uk/api
7
8       Läs instruktionerna och försök göra en
9       'request' mot den rätta endpoint.
10      Vad händer?
11  */
12
13
14
15
16
17
18  // HINT & Examples
19  hint("Lägg till /random efter URL")
20
21
22
23
```

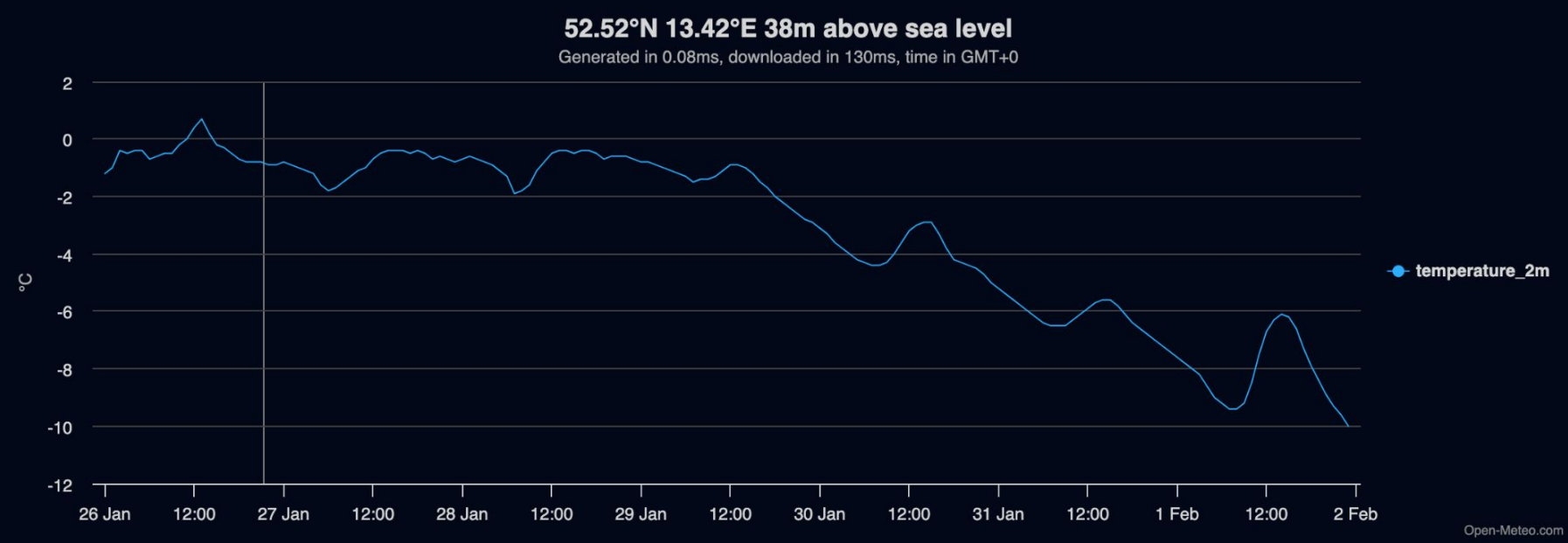
```
1   // -Uppgift #4- //
2
3   /* INSTRUCTIONS
4
5       Navigera in till:
6       https://open-meteo.com/en/docs
7
8       Läs dokumentation och försök få fram ett svar
9       I form av JSON inom Webbläsaren.
10  */
11
12
13
14
15
16
17  // HINT & Examples
18  hint("Pricka in ett par checkboxes. Leta efter
19  API:et på sidan")
20
21  hint("Svar finns på nästa sida")
22
23
```

Uppgift #4

Hourly Weather Variables

- [x] Temperature (2 m)
- [ ] Relative Humidity (2 m)
- [ ] Dewpoint (2 m)
- [ ] Apparent Temperature
- [ ] Precipitation Probability
- [ ] Precipitation (rain + showers + snow)
- [ ] Rain
- [ ] Showers
- [x] Snowfall
- [ ] Snow Depth

Preview: Chart & URL | Python | TypeScript | Swift | Other

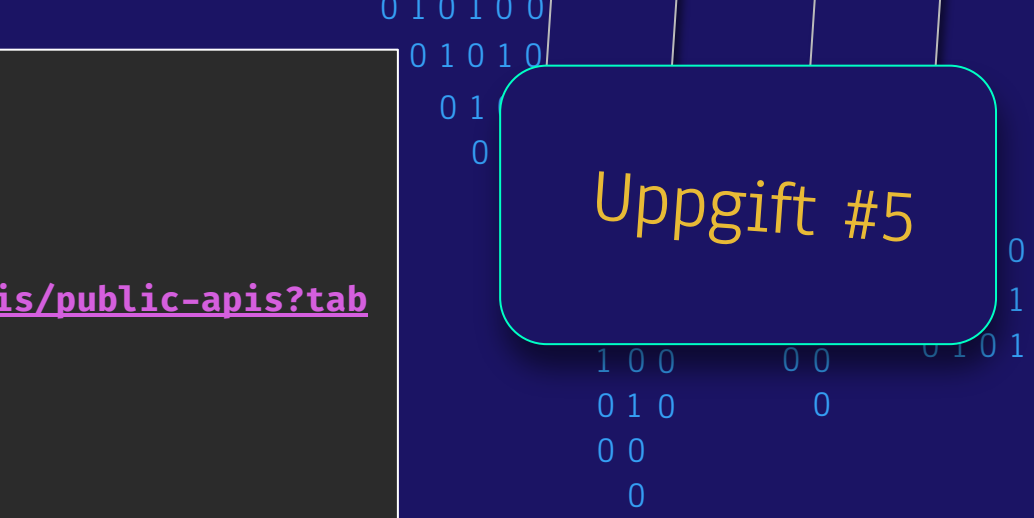


Download XLSX | Download CSV

API URL (Open in new tab or copy this URL into your application)

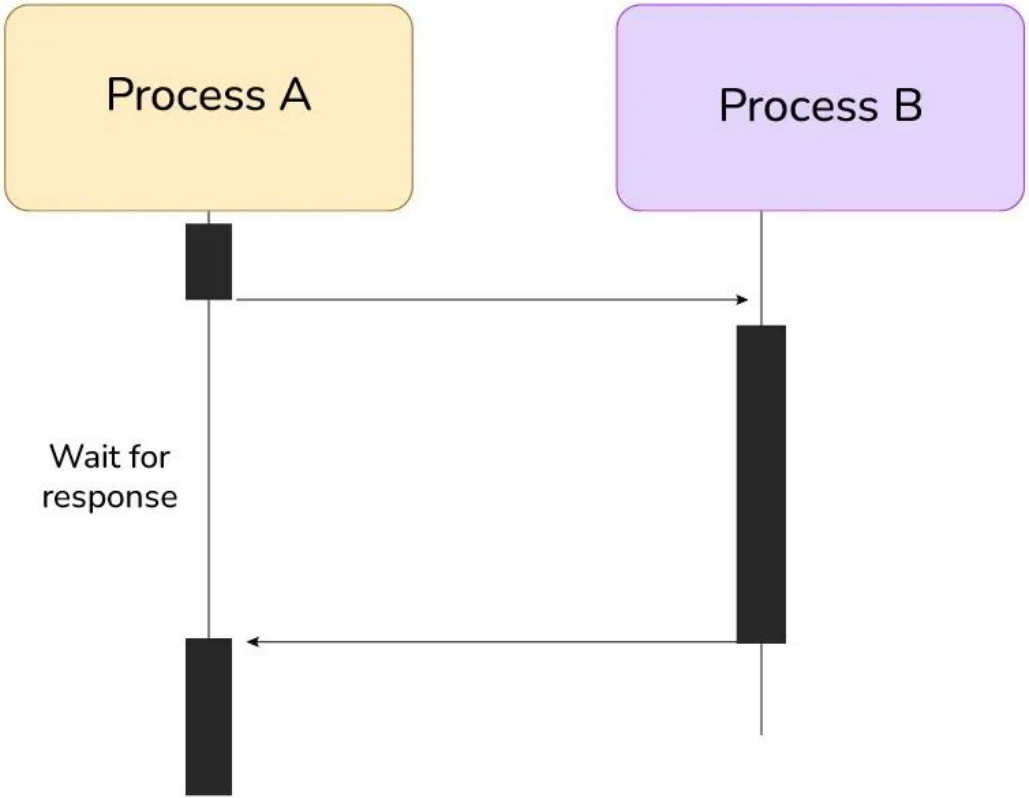
https://api.open-meteo.com/v1/forecast?latitude=52.52&longitude=13.41&hourly=temperature_2m

```
1   // -Uppgift #5- //
2
3   /* INSTRUCTIONS
4
5       Utforska öppna api:er på:
6       https://github.com/public-apis/public-apis?tab
7   =readme-ov-file
8   */
9
10
11
12
13
14
15  // HINT & Examples
16  hint("Den stora databasen är för det mesta gratis.
17  Vill du inte betala, leta efter kolumn 'AUTH', där
18  ska det stå 'NO' om detta ska vara gratis")
19
20
21
22
23
```

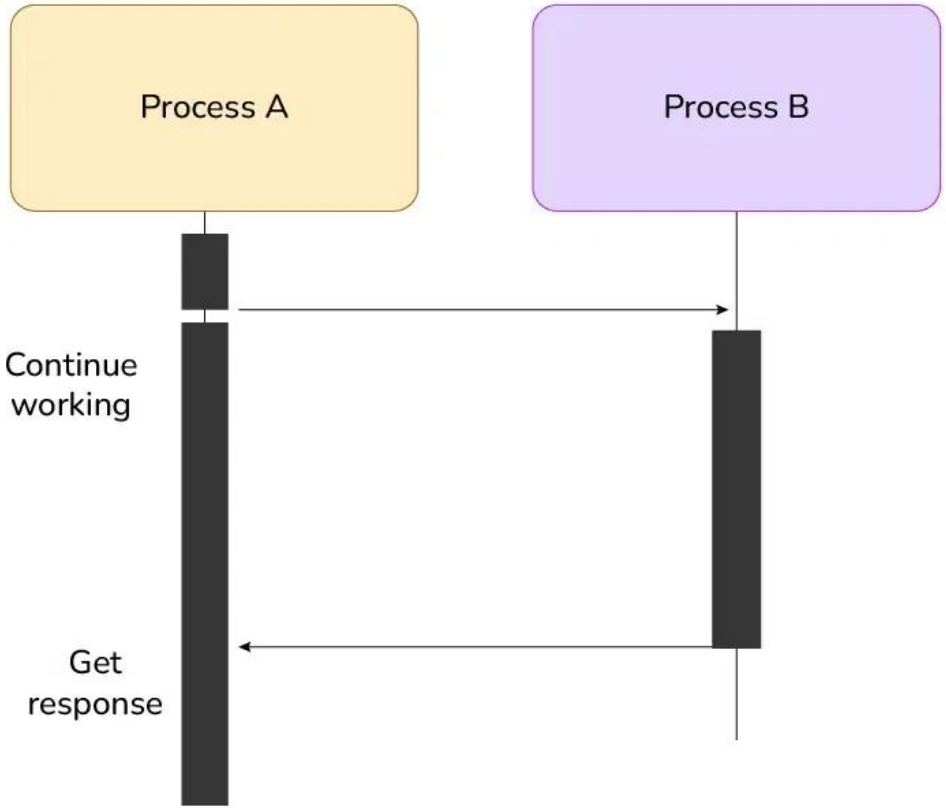



Asynchronous
VS
Synchronous
(bonus)

Synchronous



Asynchronous



When to Use Asynchronous vs Synchronous

Choosing between asynchronous (async) and synchronous (sync) programming depends on the specific needs of your application. One can use sync programming when tasks need to be executed in a strict sequence and when operations are quick, simple, and do not involve extensive waiting periods, such as command-line tools, basic scripts, or tasks requiring strict order. Sync is also preferable when ease of debugging and simplicity in implementation are paramount.

In contrast, async programming is ideal for applications that need to handle multiple operations concurrently, especially when dealing with I/O-bound tasks, high latency operations, or real-time data processing. Async is suitable for web servers, networked applications, user interfaces, and scenarios requiring high responsiveness and scalability, such as chat applications, streaming services, and any application with numerous simultaneous users or tasks. While async programming is more complex and requires careful management of concurrency, its ability to enhance performance and responsiveness makes it indispensable for modern, high-performance applications.

Read more: https://www.geeksforgeeks.org/javascript/synchronous-and-asynchronous-programming/

Problem

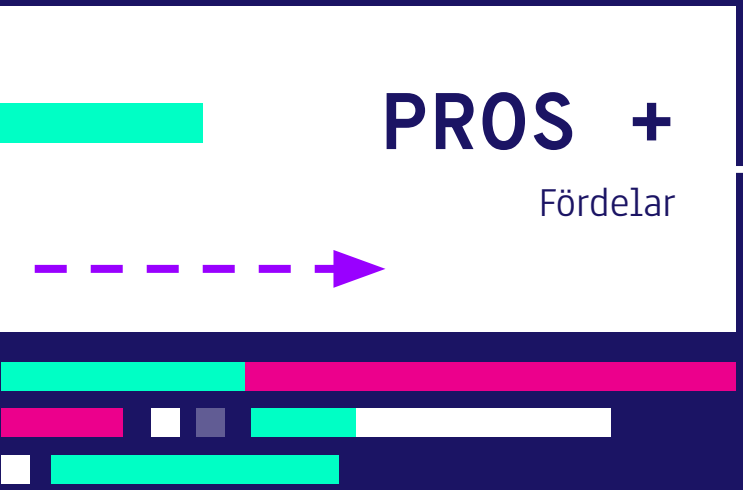
Vad är skillnaderna på Asynk/Synkron programmering?

Solution

Async sker parallelt, medan synkront sker ‘steg för steg’ och inväntar svar tills dessa blir klar.


Sync / Async
Pros & Cons

Synchronous



PROS +

Fördelar

Standard

Enklast att lära sig.

Snabbhet

Snabb överföring av enskild data

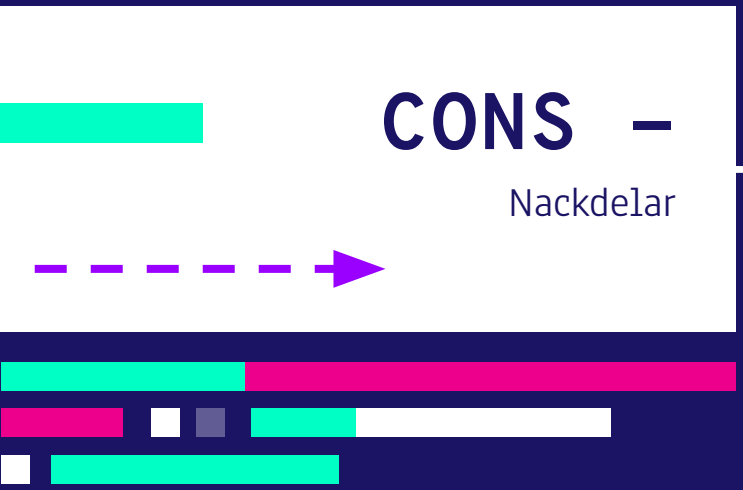
Felbenägenhet

Mindre chans att något går snett då allt sker systematiskt och sekventiellt

Blocking Call

Anses vara ett ‘blocking call’ vilket inväntar slutförandet av processen

Synchronous



CONS -

Nackdelar

Långsamt

Då den inväntar avslutandet av flertal processer, tar det längre tid

‘Unresponsive’

Kan dra ut på processer och känns konsekvent oemottaglig

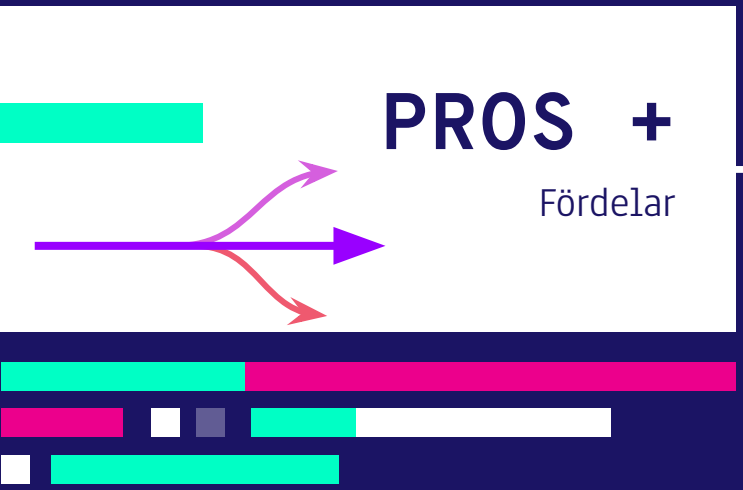
Algoritmer

Processer och algoritmer är extra långsamma som ‘async’

‘Threads’

Använder sig av resurser som heter ‘threads’. Varje request använder sig av en Thread. JVM har totalt 256

Asynchronous



Mer Responsivt

Processerna inväntas ej

Snabbare

Då processer kan köras parallellt
blir dessa avklarade snabbare

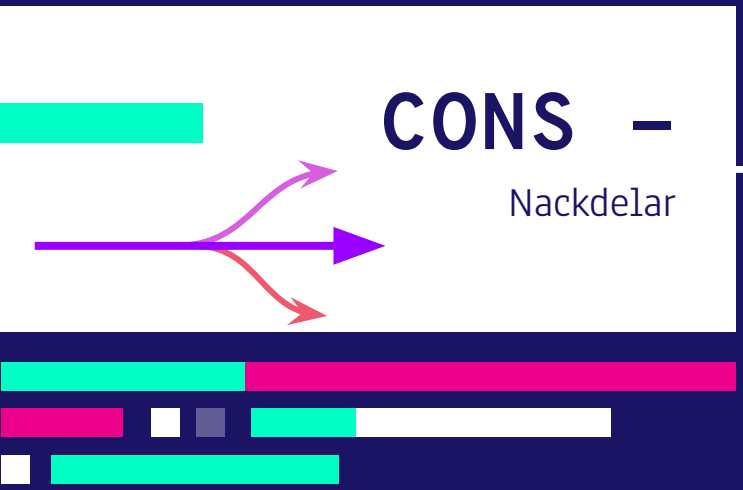
Skalbarhet

Enklare inom felhantering, och att skala upp
som applikation

‘Non-blocking’

Anses vara ett ‘non-blocking call’ vilket
tillåter processer att köras parallellt
med varandra

Asynchronous



CONS -

Nackdelar

Mer felbenäget

T.ex: mm data är beroende av varandra, finns risker att datan är null när det hämtas.

Mer Komplext

Koden blir svårare att hantera och växer drastiskt

Svår debugging

MIMC - More is more complex, kodbasen växer och i helhet blir det svårare att debugga

Överkomplext

Enkelt att överkomplicera en lätt lösning.

THANKS !

Do you have any questions?
kristoffer.johansson@sti.se