API
&
HOW WEB TALKS

TOMÁŠ HODEK

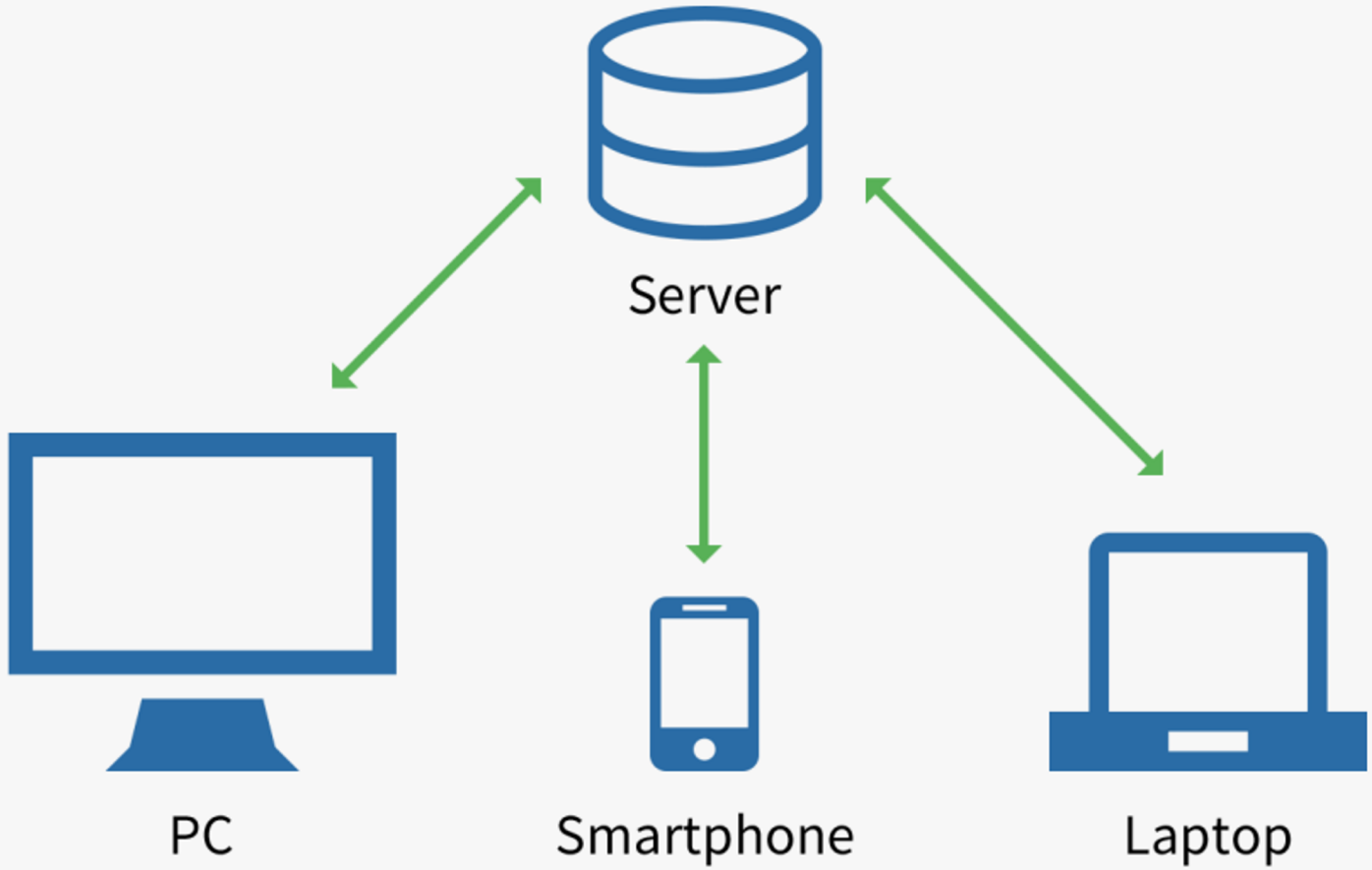
What is API?

- Application programming interface
- Software apps use it to talk to each other
- Consists of
  - Specification (documentation)
  - Interface that represents the specification
- Database APIs, OS APIs, library APIs, browser APIs

Web API

- Most used API
- Uses client <-> server architecture
- Protocols/specification
  - REST
  - GraphQL
  - SOAP, RPC, etc.

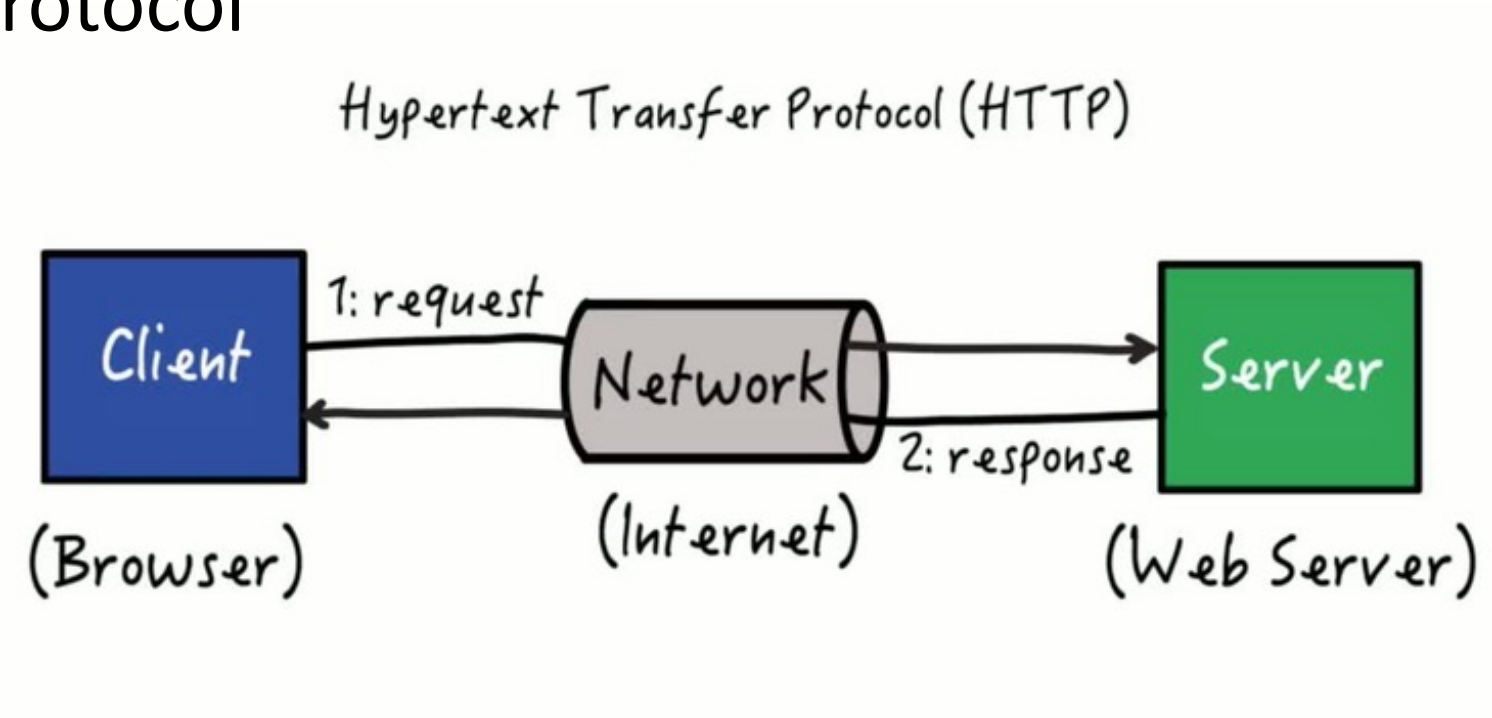
Client-Server Model



- Server
  - remote computer
- Client
  - web browser, mobile, any device connecting to the server
- Usually communication initiated by client

But how they communicate?

In web, via HTTP protocol



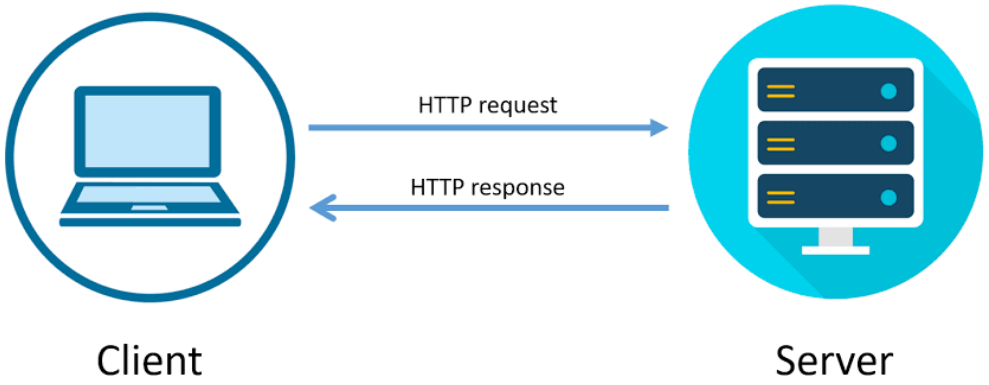
HTTP

- Protocol based on client-server
- Simple, extensible, stateless
- Not sessionless
  - We have cookies!

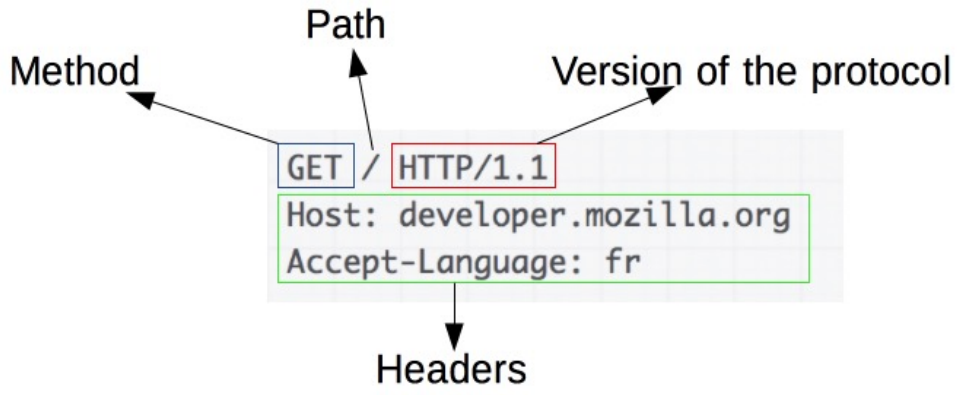
- HTTPS – encrypted HTTP

How HTTP works?

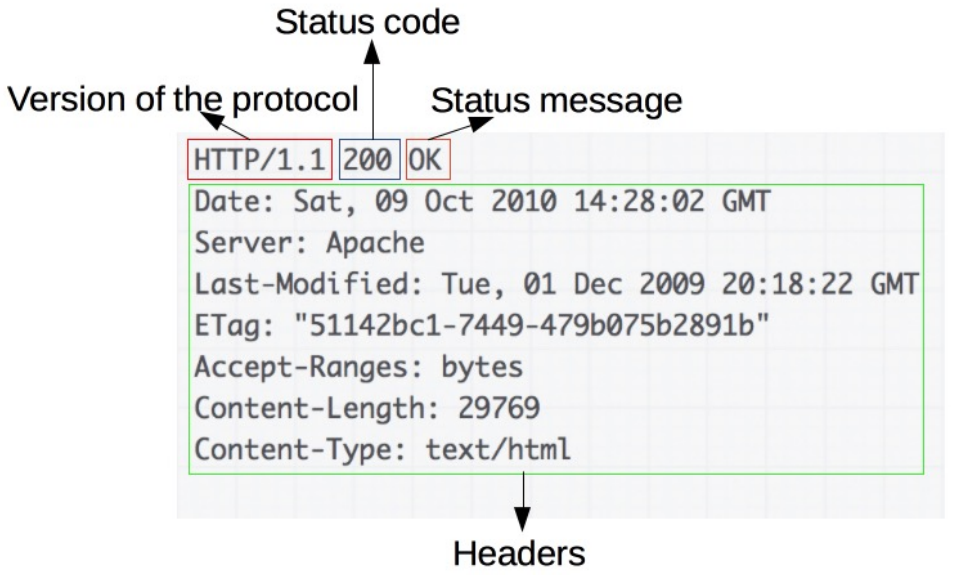
- Client asks, server respond



Request



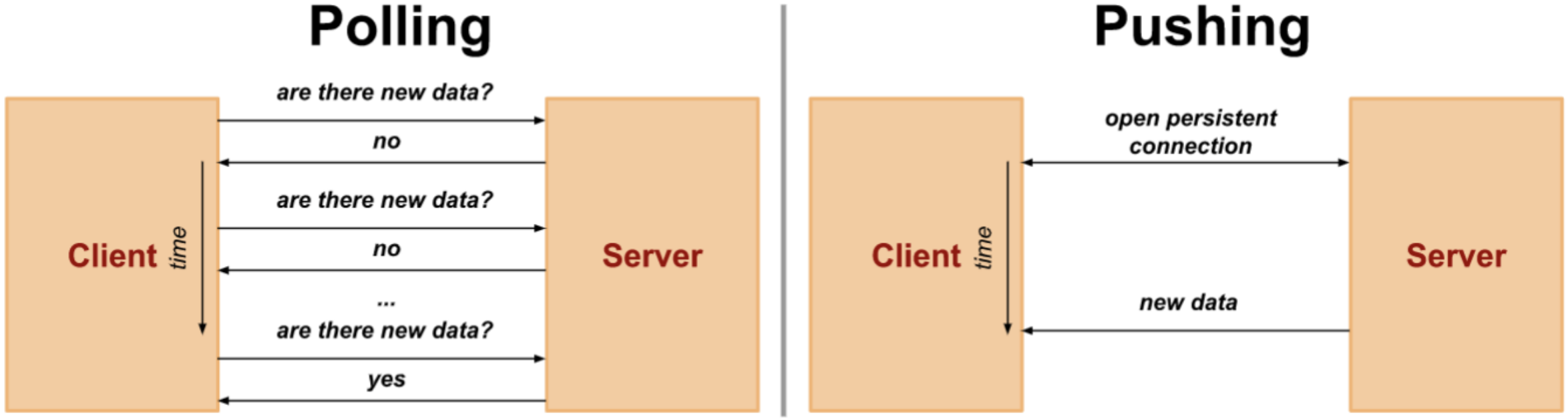
Response



HTTP methods

- Indicates action, but not necessarily
- GET
  - Gets data, addressed in URL, easily cacheable, no body
- POST
  - Has body, transferring data, use with HTTPS for security!
- DELETE, PUT, etc.

Types of comunication



Polling and Pushing

- HTTP Long Polling
  - Server holds connection as long as possible – then refresh
- HTTP Streaming
  - Server does not close connection after first response, send multiple
- Pushing – WebSocket protocol
  - Long-term connection, data both ways, uses HTTP for ”handshake”

Ok, but how do we use it?

\+ fetch() = DATA

Ajax

- Asynchronous JavaScript and XML
- Not blocking the main thread – UI is not frozen
- For data fetching or big computations
- Promise vs async/await

fetch()

- API for using HTTP request
- ”Replace” for older XMLHttpRequest API
- Much simpler
- Asynchronous calls
- Axios

fetch()

Promise

```
fetch('/recipes')
  .then(response => {
    setRecipes(response);
  })
  .catch(error => {
    console.log("Error", error);
  })
```

async/await

```
try {
  const response = await fetch('/recipes');
  setRecipes(response);
} catch (error) {
  console.log("Error", error);
}
```

fetch()

```
fetch(url,  {
  method: "POST",
  headers: { /* ... */ },
  body: ...,
  redirect: "follow",
  signal: ...,
  ...
})
```

Ok, and how should the API looks like?

REST

- Representational state transfer
- “Weak” standard
- Uniformed interface oriented about resources
- Use HTTP methods -> GET, POST, DELETE, PUT,...
- One domain *cngroup.dk/api/v2/people*

REST

- Self-descriptive

  http://api.example.com/song-management/users/{id}/playlists

- Cacheable
- Stateless
- Layered

But, how to REST?

WikiHow won’t help much ...

How to REST

- Use nouns for naming
- Be consistent and use hierarchal structure

  users/{id}/playlists

- Use correct HTTP methods

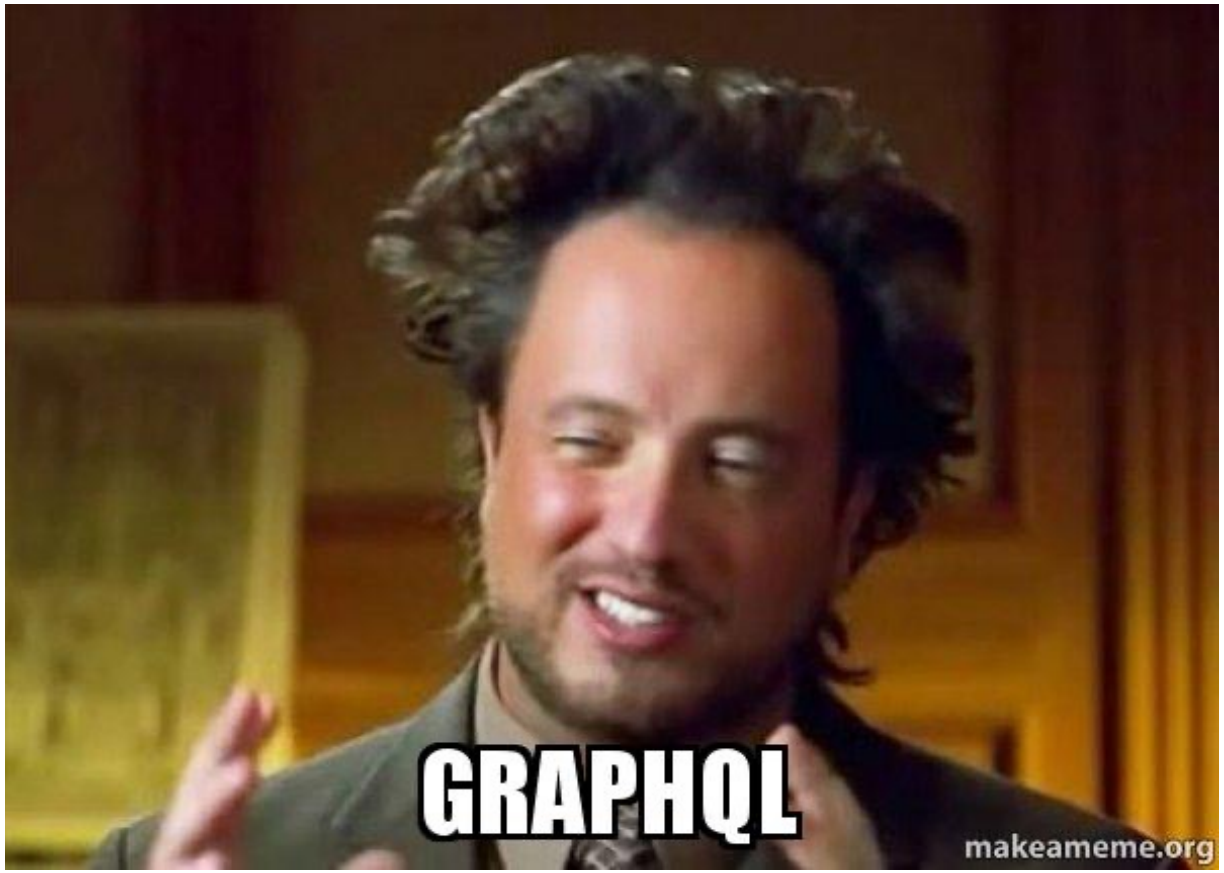
`HTTP GET remove/users/{id}` - DON'T

`HTTP DELETE users/{id}` - DO

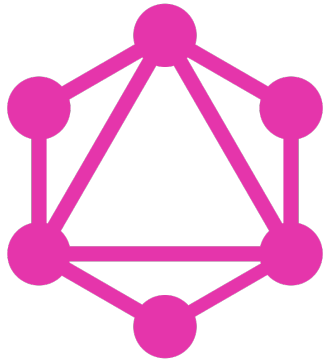
- Use query for filtering


IS THAT
ALL YOU GOT?
memegenerator.net

Almost, but no



GraphQL



- Query language for API
- Data represented by graph
  - Server describes data and actions – queries and mutations
- Self-discovery API
- Single URL - */graphql*
- NOT A REPLACEMENT FOR REST – both can be great

GraphQL

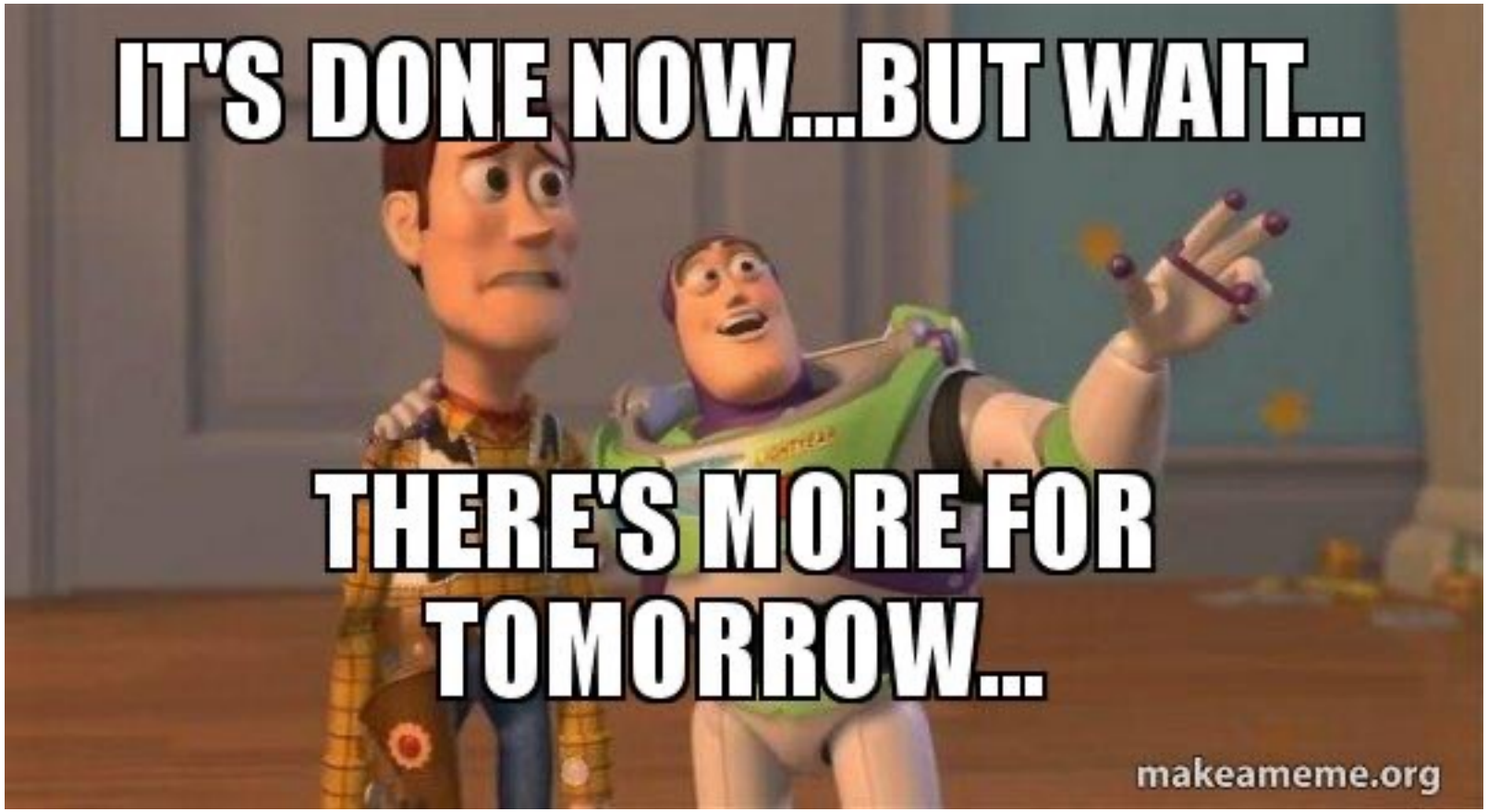
- Client asks only for what he wants

```
type Project {
  name: String
  tagline: String
  contributors: [User]
}

//client asks only for tagline
query {
  project(name: "GraphQL") {
    tagline
  }
}
```

GraphQL

```
fetch('https://www.example.dev/graphql', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    query: `
        query getExampleQuert($now: DateTime!) {
          allEpisode(limit: 10, sort: {date: ASC}}) {
            date
            guest {
              name
              twitter
            }
            description
          }
        }
      `,
    variables: {
      now: new Date().toISOString(),
    },
  }),
})
  .then((res) => res.json())
  .then((result) => console.log(result));
```


IT'S DONE NOW...BUT WAIT...
THERE'S MORE FOR TOMORROW...
makeameme.org

Resources:

- https://developer.mozilla.org/en-US/docs/Web
- https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/
- https://aws.amazon.com/what-is/api/
- https://restfulapi.net/
- https://graphql.org/
- CVUT/VSE Lectures