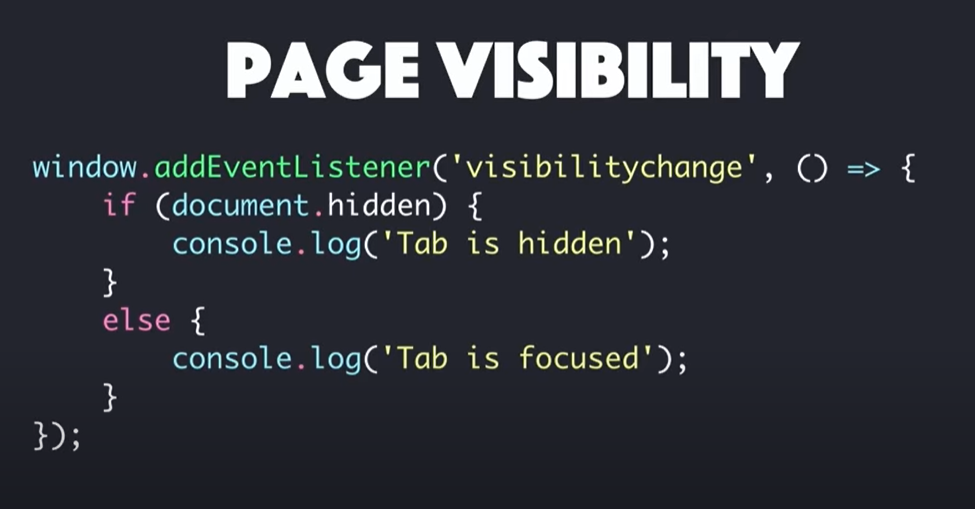
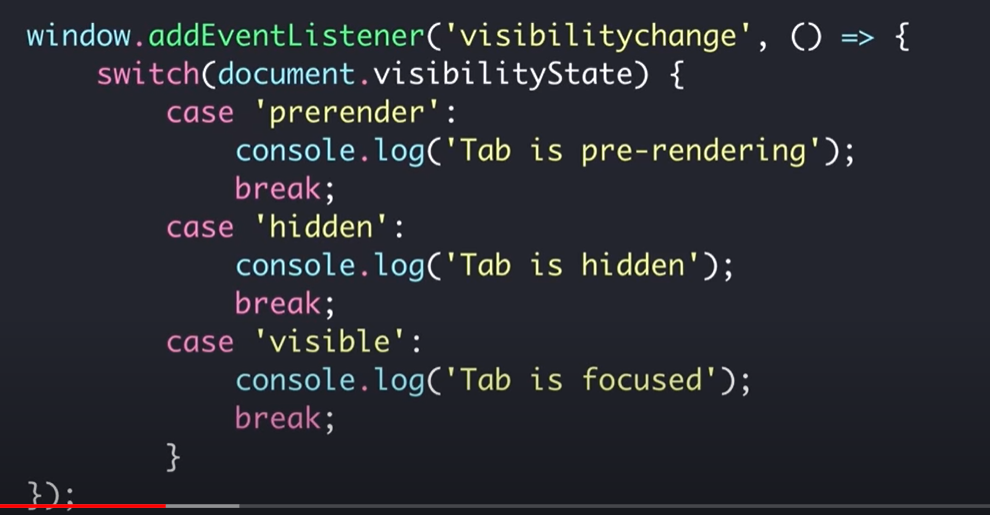
Page Visibility:

Provides an API to ask whether the current page is visible for the user or not.



New property, document.visibilityState

We can identify other kind of status of a page.



Online State: exposes a network connection availability information to the web.

2 different applications can communicate with help of an API.

Example:

Us and kitchen.

We can communicate directly with the kitchen to place our order.

What if we and the chef speak different languages. Then in this scenario, we will need to place the order through the waiter.

Here, the waiter plays an important role in our communication with the kitchen.

Online travel service communicates with different airline applications.

Airlines give access to their information to the 3rd party. They expose their information in form of an API.

The 3rd party application gets information through that API.

It sends a request in form of an API that the airline applications understand and they in turn look at the database and provide information to the 3rd party.

Medium

Format

Medium could be HTTP and format could be JSON or XML.

Phone is the medium and language is the format.

API enables communication between applications.

**REST APIs: The TechCave**

In object oriented programming,

In order to communicate or use the functionalities of an object, we do that through its APIs, i.e, its methods and attributes.

API could be anything in any form. The only thing that it has to be is that it has to be a way to communicate with a software component.

When we communicate with the operating system through the command line, this is like using an API.

How do APIs relate to Web Services ?

We hear REST web services and also hear about REST APIs.

* APIs are created for software components, a way for software to interact with other software.
* The way an API is implemented and what it consists of is not important.

Web services are a set of rules and technologies that enable 2 or more components on the web to talk to each other.

Web services are just APIs in the context of the web.

Not every API is a web service.

REST API = REST web service

On the web, an API and a web service are the same thing.

A REST API is an API that follows the rules of REST specification.

A web service is defined by rules:

* How software components will talk ?
* What kind of messages they will send to each other ?
* How request and responses will be handled ?

We could have an API that does not follow REST rules.

REST: Representational State Transfer

How does HTTP relate to REST ?

HTTP is an application layer protocol for sending and receiving messages over a network.

In HTTP, we can use GET method for all sorts of interactions.

REST is a specification that dictates how distributed systems on the web should communicate.

It restricts the use of http methods.

REST is a way to implement and use the HTTP protocol.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

How does The WEB relate to REST ?

REST is the underlying architecture of the web.

REST is about doing things how they are supposed to be done.

Another specification: GraphQL

Questions every API should answer:

* How can the client tell the service provider which operation it wants to perform ? (Method information)
* How can the client tell the service provider what data to operate on ? (Scoping information)

REST answer to the 2 questions:

1. The method information should be expressed in the HTTP verb.

Not RESTful: GET api/users/delete/:userId HTTP/1.1

RESTful: DELETE api/users/:userId HTTP/1.1

1. Scoping information should go in the URI. For example, userId.

In REST, everything a client can operate on is called a resource.

REST communication is stateless.

Server does not maintain the state of its client.

Express has an intuitive API for building RESTful APIs.

**REST vs GraphQL APIs**:

Web apis

Web service exposes certain entry points where clients can fetch and send data to.

We can build a web service using REST or GraphQL API.

How we build a web service influences how a client uses or queries a web service.

In REST API, multiple endpoints or urls are exposed by the web service.

GraphQL API works with one endpoint only which is reached via a POST request.

(POST /graphql)

In both APIs, JSON data is exchanged.

With rest api, we might also send back binary data. In graphql we can implement a separate endpoint that returns a file.

With both apis, we can use any server side language and any frontend framework.

We can build that api on a server, on the backend using any programming language we like.

GraphQL is created by Facebook.

Its about sending right http request to the web service.

Both of the APIs are stateless. They don’t care about the client. No data about the client is stored on the backend.

We just expose the endpoints and whoever sends the right kind of request combined with authentication gets the data or stores the data depending on what the api exposes to the client.

The difference between these apis is how we send requests to them.

**URL-Driven vs Query-Language**:

REST API:

HTTP verb, Path,

Request body (optional: contains any data)

Not all request require a body. For example, GET requests don’t include a body.

In graphQL, there is one endpoint and this endpoint can have any path we want.

Often the path is /graphql

It always takes a POST request.

In a GraphQL API, we don’t target different endpoints, different urls.

Instead we state what we want in our request body.

Request body contains a query language.

GraphQL query

graphQL api parses the query on the backend.

Similar to how we query a database with sql or with mongodb or with whatever database engine or query language we are using.

We are not talking to a database here.

We are talking to a web service which does the parsing for us.

The backend takes the parsed command and does whatever we define on the backend.

In a REST API, our API endpoints map resources and actions that happen on the server, for example, the action to get all posts or to create a new post, to certain url http verb combinations.

GET: Get a resource from the server

POST: Post a resource to the server (i.e. create or append resource)

PUT: Put a resource onto the server (i.e create or overwrite a resource)

PATCH: Update parts of an existing resource on the server

DELETE: Delete a resource on the server.

OPTIONS: determines whether follow-up request is allowed (sent automatically)

**How does GraphQL work** ?

Same base set up.

Client and server, which want to talk to each other.

Why clients do not talk to database directly ?

<https://www.youtube.com/watch?v=dsAalk1NpVk>

Server is responsible for any server-side logic, database access, etc.

One single endpoint which always takes a POST request.

The one request that we send is not targeting specific actions on the server by the combination of URL and HTTP verb.

POST request contains Query exression (to define the data that should be returned).

By parsing the query expression, the server is able to get the data we need.

A GraphQL query looks like this:

{

query {

user {

name

age

}

}

}

This GraphQL Query is made up of Operation type ( “query” means that we want to get some data)

Other types: “mutation”, “subscription”

“mutation” means that we want to edit some data.

We want to store new data or edit existing data on the server.

With “subscription” we set up a live subscription so that the server actively informs us about the changes that happen there.

This would not be possible over normal HTTP protocol. We will require websockets instead.

After this, we give the operation an identifier or an endpoint.

In this case the operation endpoint is “user”

This is important for the server to understand with which resource we want to work and we cannot send any arbitrary queries.

On the server, we parse the incoming query but then we only support a couple of endpoints or couple of resources that can be targeted.

We don’t want to make everything mutable or we don’t want the client to query for any kind of data.

On the server when we write the GraphQL API, we don’t just parse the incoming query.

We also then define what we do with the parsed result and which kind of queries we want to support.

If we go back to the query, we know what we want to do such as query data or mutate data, which kind of data we would like to get or mutate.

For that specific data, we also specify which fields we want to work on, which fields we want to get back, for example.

Here the requested fields are “name” and “age”.

This is one of the huge advantages of GraphQL compared to REST.

With REST APIs we target a specific resource, for example GET /posts and we retrieve all posts with all fields that belong to a post.

We might use query parameters to implement something like “pagination” to limit the amount of posts which are retrieved or to even limit what is retrieved inside of the posts. But that adds more complexity to the API.

And often we will end up fetching more data than we are interested in.

If our client app just wants to output an overview of all posts and for that it might only need the id, image and the title.

There is no need to also fetch the description, excerpt, creator id, etc.

Fetching too much is a common problem with REST APIs.

REST API is pretty clear, easy to understand.

It is pretty clear that if we send a GET request to

/posts, we get back all data.

With GraphQL, we can be specific about the kind of data we want to retrieve.

Here we only want to get the user name and age. Other information for the user can also be stored in the database such as id, hobbies, etc.

This can reduce the amount of data which is transferred over the wire.

With REST APIs, when writing client side code, we don’t have to use any special packages which we often have to use with GraphQL APIs to write proper queries and have easily manageable code on the client.

Client side code can get cluttered if we are working with a GraphQL API.

Building rest apis tends to be a bit easier than building GraphQL apis.

With Apollo package, it is easy to use GraphQL apis.

If we are building a web application that needs to query different kinds of data on different pages and bandwidth really needs to be conserved and we want to make sure that we can be very specific regarding the data we are retrieving, then a GraphQL API might be much better.

If we a lot of data per page of our app, a REST might be great as fetching too much data may not be a disadvantage here.

Overfetching and

under fetching (have to make additional queries)

Endpoints which do not have additional parameters give all the information for that endpoint.

In GraphQL, we can make selective end points. We don’t have dumb endpoints.

Is GraphQL related to Graph databases ?

No

“graph” in GraphQL means that our queries are now able to crawl into the REST API and pick up selective information.

GraphQL docs.

**Final topic**:

Why Angular, React and Vue should not directly send Database queries:

<https://www.youtube.com/watch?v=dsAalk1NpVk>

