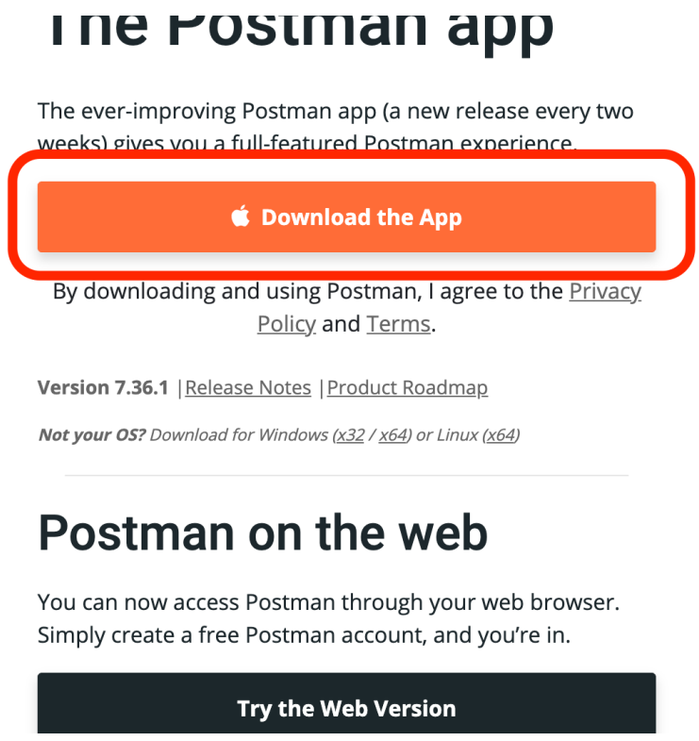
Postman

- Postman är ett verktyg för att arbeta med API:er, vilket vi kommer att göra en del idag.

- När vi tar en rast eller liknande, installera Postman.

  - https://www.postman.com/downloads/
    Jag har inte provat webbversionen, den funkar säkert också om man vet vad man gör.



Klientsidesvalidering

- Vanligtvis skickar vi information till servern (backend) för att kontrollera saker, spara saker i databasen, registrera användare osv.

- Sidan måste då laddas om vilket tar tid, vilket användare ofta är känsliga för.

- Vi kan göra en del kontroller på klientsidan med JS för att skapa bättre UX.

- https://jsfiddle.net/emmio_micke/f287mk4w/

Klientsidesvalidering

- Om en ny användare registrerar sig måste vi med denna approach uppdatera vår kod varje gång en användare registrerar sig.

- Kan vi få tag på en lista av användare utan att ladda om sidan?

Ajax

Asynchronous JavaScript And XML är ett samlingsnamn för flera olika tekniker som kan användas för att bygga applikationer för World Wide Web med bättre interaktivitet än tidigare webbapplikationer.

https://sv.wikipedia.org/wiki/AJAX

Ajax

- Låter oss kommunicera med en server i bakgrunden för att manipulera en del av sidans DOM och styling istället för att ladda om hela sidan.

Ajax: exempel

- **Validering av formulär i realtid** En kontroll kan automatiskt utföras under tiden som man fyller i formulären. Man behöver inte fylla i ett formulär helt och trycka på submit (och vänta på svar från serven) för att se om något fält i formuläret fyllts i fel.

- **Automatisk textkomplettering**

  https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_autocomplete
  (Inte perfekt som exempel eftersom den inte kommunicerar genom Ajax.)

- **Sofistikerade kontroller och effekter i användargränssnittet** träd, menyer, tabeller, kalendrar osv

- **Partiell submit** En HTML-sida kan skicka in en del av den formulärdata som behövs istället för att ladda om hela sidor.

https://sv.wikipedia.org/wiki/AJAX

Ajax: fördelar

- Sidor går fortare att ladda då klienten kör HTML lokalt och endast hämtar den data som behövs från servern. Hela sidan behöver inte laddas om så fort något ska ändras, vilket resulterar i att den upplevs som snabbare och nyttjar mindre bandbredd.
- Känns mer som en applikation.
- Separerar data från layout.

https://sv.wikipedia.org/wiki/AJAX

Ajax: nackdelar

- När klienten laddar dynamiska sidor så registreras inte dessa i webbläsarens historik, så när användaren trycker på bakåtknappen så kan det leda till oönskat resultat.
- Användaren kan få problem med att lägga till ett bokmärke av ett visst tillstånd på sidan om webbsidans URL aldrig förändras. (Mer om detta när ni läser om *routes* i *React* eller liknande.)

https://sv.wikipedia.org/wiki/AJAX

Kommunicera med server

- Hur hämtar vi då data i bakgrunden? Vad är det för typ av data vi vill ha? Hur fungerar kommunikationen?

Application Programming Interface

- När vi gör ett anrop (*Request*) till en webbsida så innehåller svaret (*Response*) oftast HTML.

- HTML är bra för att strukturera upp vad information är för en browser, men det är inte ett bra sätt att hantera informationen programmeringsmässigt, det är svårt att få ut den informationen vi behöver.

- Kan man få andra typer av resultat?
  - https://randomuser.me/api/

- Låter oss hämta det data vi behöver i ett format som vi kan bearbeta, vanligen JSON eller XML.

RestFul API

- Det finns olika typer av API:er, men det vi oftast pratar om är RestFul API:er.

Representational State Transfer (REST) eller RESTful webbtjänst är ett IT-arkitekturbegrepp som beskriver hur tjänster för maskin-till-maskin-kommunikation kan tillhandahållas via webbteknologi.

https://sv.wikipedia.org/wiki/Representational_State_Transfer

JSON

- JSON (JavaScript Object Notation), är ett kompakt, textbaserat format som används för att utbyta data.

- Datatypen nedan är string, men påminner detta format om något?

```
{
  "firstName": "Jason",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    { "type": "home", "number": "212 555-1234" },
    { "type": "fax", "number": "646 555-4567" }
  ],
  "newSubscription": false,
  "companyName": null
}
```

- https://jsfiddle.net/emmio_micke/ahq6pzv9/

Fetch

- Det (just nu) nyaste sättet att hämta data från ett API är att använda `fetch`.

- `Fetch` körs asynkront och returnerar ett `Promise`.

- http://jsfiddle.net/emmio_micke/jhf2borz/
- http://jsfiddle.net/emmio_micke/8do61zy4/
- https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Uppgift

- Forka denna fiddle (eller kopiera bara koden) och ändra så att ni kan skriva ut data från API:et istället för från formuläret.

- *Ajax:* Om ni vill ha en extra utmaning, integrera lösningen med uppgiften vi gjorde i förra veckan där vi la till uppgifterna i en tabell och gav användaren möjlighet att radera rader och uppdatera uppgifter.

- http://jsfiddle.net/emmio_micke/h5j6dzbv/

Klientsidesvalidering

- Tillbaka till klientsidesvalideringen.
- Kan vi få en lista av användare från vår server?
- I vanliga fall har vi en egen webbtjänst där vi kan skriva en endpoint (typ adress, tjänst) som returnerar användare.
- För detta ändamål har jag dock satt upp ett fakeapi här:
- https://5feb168f573752001730a455.mockapi.io/users
- Låt oss kolla i Postman och dokumentationen hur detta kan se ut och fungera.

Ajax och API

- Dessa tekniker är inte beroende av varandra. Man kan göra saker med Ajax som inte använder ett API och man kan använda API:er utan att använda Ajax.

API

- Vad finns det för API:er?
    - Massor av olika. Man kan hitta produktdatabaser, bussavgångar, betalningslösningar, information om kalenderhändelser, statistik, Chuck Norris-skämt…
    - Vissa är öppna för allmänheten, vissa används för internt bruk på företag, vissa kostar…
        - http://apikatalogen.se/
        - http://mashup.se/apikatalog/

Varför API?

- Enkelt sätt att dela ”rätt” data utan att behöva dela hela databasen.
- Att dela data kan ge större nytta för både användare (*consumer*) och delade (*provider*).
    - Om min app som säljer biljetter till fotbollsmatcher kan tipsa om när nästa buss går så gynnar det både min användare och Skånetrafiken.

Behörighet

- Ibland kan man behöva autentisera sig för att få tillgång till API:et av olika orsaker.
- Skydda data
- Skydda servern från överbelastning
- Statistik

Behörighet

- *Autentisering* (authentication) handlar om att bevisa att man är den man är (t ex en inloggningsfunktion).

- *Auktorisering* (authorization) handlar om huruvida användaren har rätt att göra något.

- Även om jag kan bevisa att jag är Micke, så är det inte säkert att Micke har rätt att ändra kunddata, t ex.

Basic authentication

- Metod för att låta en HTTP-användare skicka namn och lösenord i ett request.

- Klienten autentiserar sig genom en header
  - `Authorization: Basic <credentials>`
  - Credentials är en base64-kodad sträng bestående av användare:lösenord.

- https://jsfiddle.net/emmio_micke/w9ftu2d6/

API-key

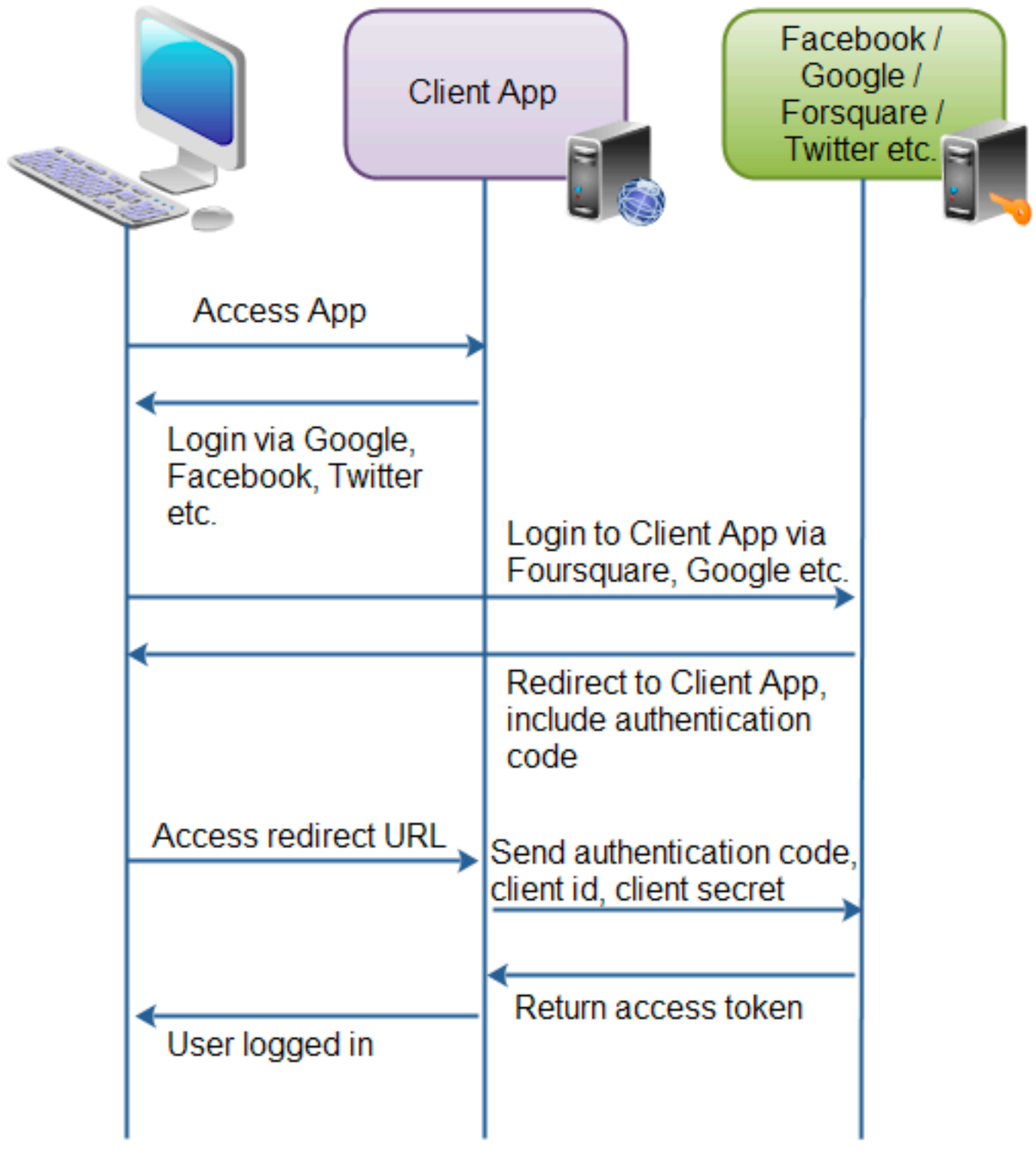
- Ganska simpelt och vanligt sätt att hantera åtkomst.
- Varje användare / konto får en ”nyckel”, en unik sträng, som används i varje request, antingen i URL eller i header.

- http://jsfiddle.net/emmio_micke/p5ntq1ro/

Oauth2

- Ett protokoll för att låta applikationer komma åt varandras data.

- Vi går inte in mer på den just nu, dock bra att ha hört talas om den.



CRUD

- Create, Read, Update, Delete
- Våra API-anrop hittills är Read-anrop där vi använder http-metoden `GET`.
- I vissa API:er kan man även göra de andra händelserna, oftast kopplat till behörighet.

HTTP verb	CRUD
POST	Create
GET	Read
PUT	Update/ Replace
DELETE	Delete

CRUD

- Det finns test-api:er man kan använda för att testa att skapa eller uppdatera data också.
- https://jsonplaceholder.typicode.com/
- Låt oss testa lite i Postman!

Resources

JSONPlaceholder comes with a set of 6 common resources:

/posts	100 posts
/comments	500 comments
/albums	100 albums
/photos	5000 photos
/todos	200 todos
/users	10 users

Note: resources have relations. For example: posts have man
full list.

Routes

All HTTP methods are supported. You can use http or https f

GET	/posts
GET	/posts/1
GET	/posts/1/comments
GET	/comments?postId=1
POST	/posts
PUT	/posts/1
PATCH	/posts/1
DELETE	/posts/1

Note: see guide for usage examples.

Fetch

- Nu när vi vet lite mer vad vi behöver kan vi skapa kod.
- Som vi såg när vi kollade på Basic Authentication så kan man konfigurera fetch genom att ange ett objekt som andra parameter.
  - https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/fetch
- Där kan vi t ex sätta metoden till POST.
  - http://jsfiddle.net/emmio_micke/hgaf2o0r/

Uppgift

- Skapa en liten att göra-applikation.
- Analysera API:et med hjälp av dokumentationen och Postman.
  - https://jsonplaceholder.typicode.com/todos
  - https://jsonplaceholder.typicode.com/guide/
- Skapa ett enkelt gränssnitt och implementera de olika metoderna med fetch.
  - Ni kan säkert använda en del från den andra att göra-listan vi gjorde: https://yh.pingpong.se/courseId/11861/content.do?id=5209549