TDP013 - Webbprogrammering och interaktivitet

Föreläsning 3:
MongoDB, AJAX, CORS, projekt och etikuppgift

Robin Keskisärkkä
Biträdande universitetslektor
Institutionen för Datavetenskap (IDA)

LINKÖPINGS UNIVERSITET

Återblick från föreläsning 2

- JavaScript
    - Repetition av callbacks
- Node.js
    - Serverramverk skrivet i Javascript
    - Stöd för nästan allt i ES6 (om man arbetar med definierar sin kod som module)
- Mocha
    - Ett testramverk för Node.js
- Code coverage med Istanbul och C8

MongoDB

LINKÖPINGS UNIVERSITET

Databas

- Hämta och spara data via Node.js
- Finns många databasalternativ
- Databasen kommer att ligga lokalt
- Vi kommer att använda dokumentdatabasen MongoDB

- **OBS:** Om ni vill använda molntjänsten för MongoDB måste ni ange hur vi kan koppla upp mot den eller hut man istället använder en lokal databas i er README.

SQL
(relationsdatabaser)

ID	NAME	BALANCE
1	Anders	100
2	Erik	20
3	Jalal	200

NoSQL
(grafdatabaser, key-value stores, mixed models …)

Cassandra
CouchDB
MongoDB
DynamoDB
Datastore
….
Sparar information som grafer, key-value, json, xml, etc.

MongoDB

- Information sparas som JSON-objekt
- Lättviktigt, snabbt, skalar bra och är open-source
- Objekten sparas i “collections”
    - Movie, Actor, Director etc.
    - En collection kan jämföras med en tabell i en SQL-databas
- Inget schema
    - … om man inte exempelvis använder Mongoose-modulen
- Ingen garanti för att två objekt i en collection har en viss struktur!

MongoDB: Komma igång

- Installera och starta MongoDB
- Logga in via terminal:

```
mongosh
```

- Visa alla databaser:

```
show dbs
```

- Välja (och skapa) databas:

```
use tdp013
```

- Spara ett nytt objekt i collection movie:

```
db.movie.insert({'a': 1})
```

- Lista alla objekt i collection movie:

```
db.movie.find()
```

- Läs på om olika funktioner i MongoDB-dokumentationen

- **Tips:** MongoDB Compass ger ett GUI mot MongoDB

Node.js: MongoDB driver

- Syntax skiljer sig något åt mellan mongosh och Node.js drivern
- I dokumentationen förespråkas numera **inte callbacks**, använd istället:
    - async/await
    - AsyncIterator (ex. for await (const doc of cursor) { ... })
    - Manuell iteration av cursorn
    - Stream API
    - Event API
- Installera driver för MongoDB:
    - `npm install mongodb`

https://www.mongodb.com/docs/drivers/node/current/

Exempel: MongoDB driver

```
import { MongoClient } from 'mongodb'

async function main(){
    const url = 'mongodb://localhost:27017'
    const client = await new MongoClient(url).connect()
    const db = client.db('tdp013')
    const collection = db.collection('people')

    // add some data
    await collection.insertMany([
        { name: 'Bob', age: 61 },
        { name: 'Dylan', age: 20 },
        { name: 'Jane', age: 14 },
        { name: 'Doe', age: 17 }
    ])

    // find documents
    const cursor = collection.find({ age: { $lt: 19 } })
    let data = await cursor.toArray()
    for(let d of data){
        console.log(JSON.stringify(d, null, 2))
    }
    client.close()
}

main()
```

LINKÖPINGS UNIVERSITET

MongoDB driver: Att tänka på

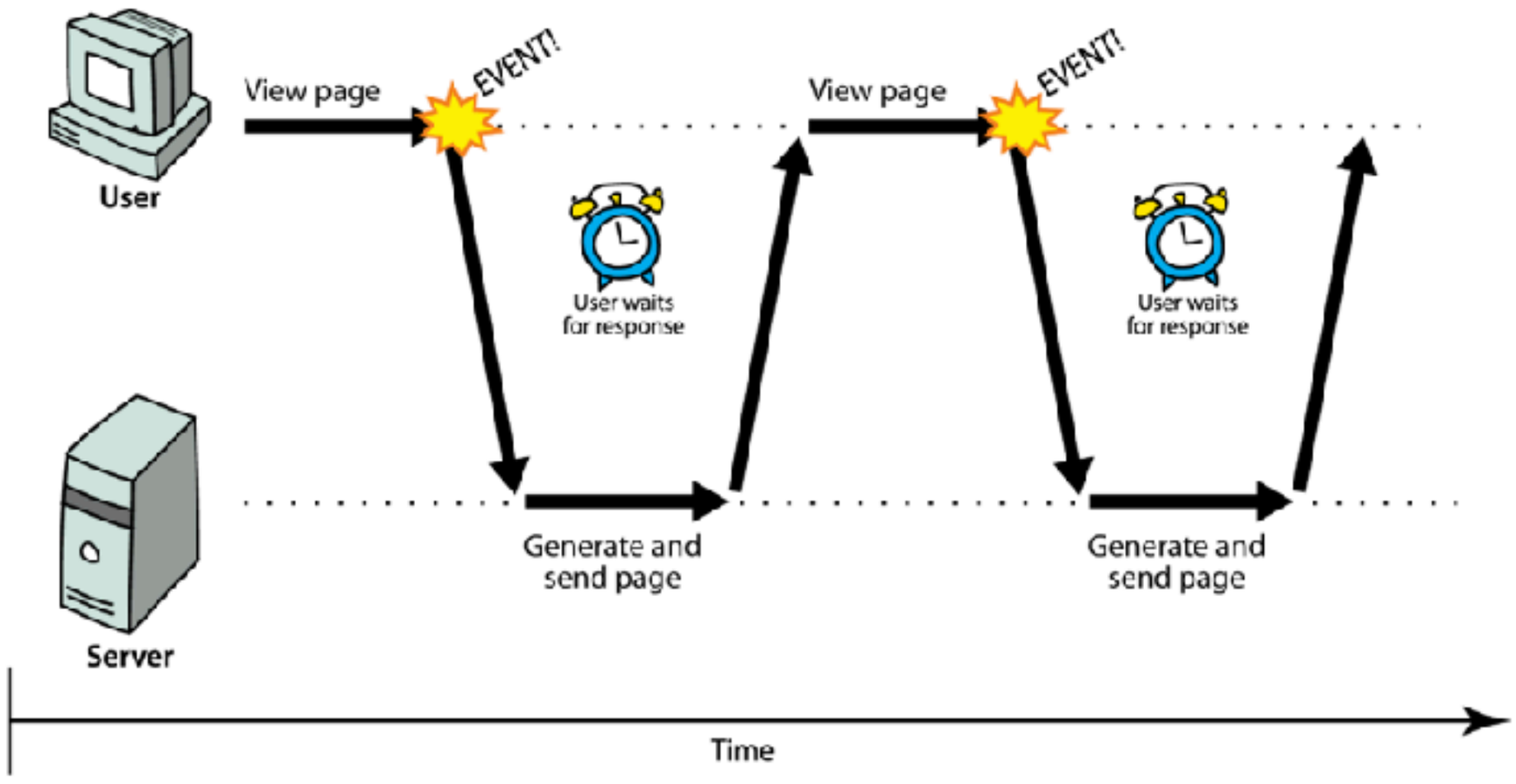
- Även om det inte kommer märkas i labb 2 så är etablering av en ny connection **mycket** tidskrävande (i relativa termer)
- Återanvänd samma klient och “connection pool” så långt som möjligt
- Lita aldrig på klienten utan kom ihåg att validera!
- Unika ID:n är inte triviala
    - Vad händer om servern startas om?
    - Vad händer om en data tas bort?
    - MongoDB har ingen egen funktion för auto-increment

HTTP-anrop

Hämta och skicka data på webben

LINKÖPINGS UNIVERSITET

Synkrona anrop på webben

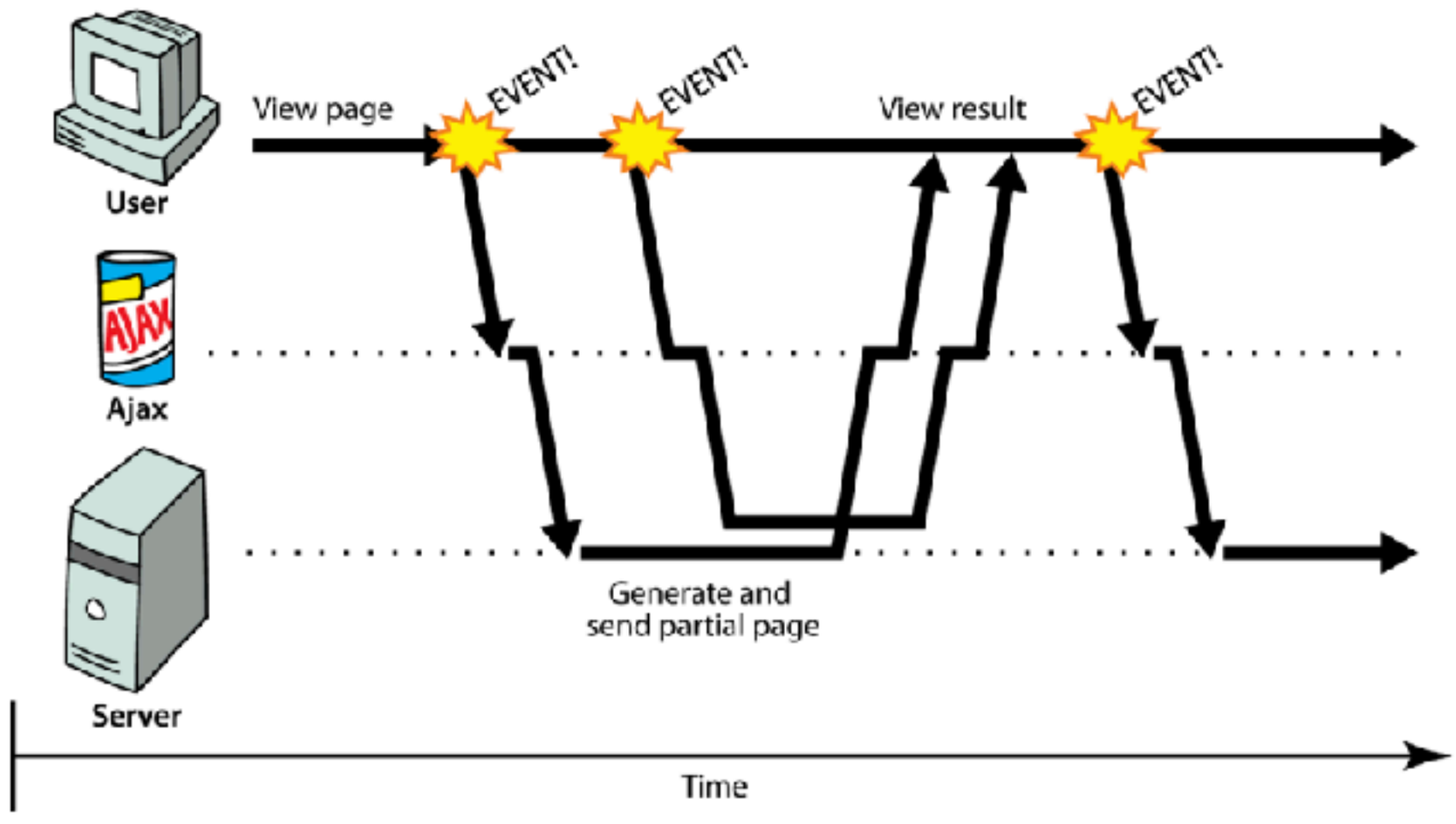


Användaren måste vänta på svar, och kan inte göra något under tiden.

Hela sidan uppdateras.

Bild: https://www.rose-hulman.edu/Class/csse/csse290-WebProgramming/201330/Slides/lecture21-ajax.shtml

Asynkrona anrop på webben



Användaren kan göra annat i väntan på svar från servern.

Endast de påverkade delarna av sidan ändras.

Bild: https://www.rose-hulman.edu/Class/csse/csse290-WebProgramming/201330/Slides/lecture21-ajax.shtml

AJAX

- *Asynchronous Javascript and XML*
- Möjliggör asynkrona anrop på webben via JavaScript
- Görs lite olika beroende på vilken webbläsare som används, men skillnaderna är idag mycket små
- Bibliotek som jQuery kan förenkla i vissa sammanhang men är inte nödvändiga
- Det som kommer tillbaka från servern är (oftast) JSON, XML, binära filer eller text

AJAX: Skicka request

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = () => {
  if (this.readyState == 4 && this.status == 200) {
    let data = JSON.parse(this.responseText);
    console.log(data);
  }
};
xhttp.open('GET', 'https://gorest.co.in/public/v1/users', true);
xhttp.send();
```

```
0 UNSENT
1 OPENED
2 HEADERS_RECEIVED
3 LOADING
4 DONE
```

”true” gör anropet asynkront

AJAX: Skicka request

```
function reqListener() {
  let data = JSON.parse(this.responseText);
  console.log(data);
}

function reqError(err) {
  console.log('Fetch Error :-S', err);
}

let oReq = new XMLHttpRequest();
oReq.onload = reqListener;
oReq.onerror = reqError;
oReq.open('GET', 'https://gorest.co.in/public/v1/users', true);
oReq.send();
```

HTTP-metoder

LINKÖPINGS UNIVERSITET

HTTP-metoder

- Kommunicerar önskad handling
- Vanligaste metoderna
    - **GET** – Be servern att returnera en viss resurs.
    - **HEAD** – Be servern att skicka information om en utpekad resurs (utan att skicka själva innehållet).
    - **POST** – Skicka information till servern som ändrar information på servern ELLER skicka information som är olämpligt att inkludera som en del av URL:en.
    - **PUT** – Lägg till eller uppdatera en resurs.
    - **DELETE** – Radera den utpekade resursen.
    - **OPTIONS** – Be servern att returnerar en lista över de HTTP-kommandon som servern stöder.

HTTP-metoder

- Kommunicerar önskad handling
- Vanligaste metoderna
    - **GET** – Be servern att returnera en viss resurs.
    - **HEAD** – Be servern att skicka information om en utpekad resurs (utan att skicka själva innehållet).
    - **POST** – Skicka information till servern som ändrar information på servern ELLER skicka information som är olämpligt att inkludera som en del av URL:en.
    - **PUT** – Lägg till eller uppdatera en resurs.
    - **DELETE** – Radera den utpekade resurs.
    - **OPTIONS** – Be servern att returnerar en lista över de HTTP-kommandon som servern stöder.

AJAX: Skicka med data med GET

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    let data = JSON.parse(this.responseText);
    console.log(data);
  }
};
xhttp.open('GET','https://gorest.co.in/public/v1/users', true);
xhttp.send();
```

AJAX: Skicka med data POST

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    let data = JSON.parse(this.responseText);
    console.log(data);
  }
};
xhttp.open('POST', 'https://gorest.co.in/public/v1/users', true);
xhttp.setRequestHeader('Content-type', 'application/json',
                       'Authorization', ‘Bearer <access token>');
xhttp.send('{"name": "John Doe", "gender": "male",
             "email": "john.doe@noone.com", "status": "active"}');
```

jQuery

```
let callback = (data) => {
  $(‘#content').text = data;
}
```

```
$.ajax({
  url: 'http://localhost:8888/',
   type: 'POST',
   data: {
     name: 'Marcus',
     filter: 'Employee'
   },
   success: callback
 });
```

jQuery behöver inte användas i någon av labbarna!

fetch(...)

AJAX med promises

LINKÖPINGS UNIVERSITET

fetch: AJAX med promises

```
const url = 'https://gorest.co.in/public/v1/users'
fetch(url, { 'method': 'GET' })
    .then(resp => resp.json()) // parsa och gå vidare till nästa 'then'
    .then(data => {
        console.log(data);
    });
```

Alternativ med await

```
const url = 'https://gorest.co.in/public/v1/users'
let resp = await fetch(url, { 'method': 'GET' })
if (resp.ok) { // HTTP status 200-299
    let data = await resp.json();
    console.log(data);
}
```

CORS

Cross-Origin Resource Sharing

LINKÖPINGS UNIVERSITET

CORS

- Restriktioner på grund av säkerhetsskäl
    - “Cross-site scripting”
    - Risk för injections
        - Kan komma runt autentisering
- AJAX kräver att alla anrop görs till exakt samma domän som klienten kör!
    - Om er sida ligger på domänen http://example.com så får man endast anropa tjänster på http://example.com/...
- CORS används för att servern explicit ska ge rättigheter för vissa domäner

Cross-Site Scripting & CORS

- AJAX kräver att anropen görs till exakt samma domän som klienten kör!
    - Om er sida ligger på domänen http://example.com så får man endast anropa tjänster på den domänen
- Restriktionen finns av säkerhetsskäl
    - “Cross-site scripting”
    - Risk för injections
    - Kan komma runt autentisering
    - Intercepts
- CORS (Cross-Origin Resource Sharing) används för att servern explicit ska ge rättigheter för vissa domäner
- Vi kommer inte fördjupa oss i alla detaljer kring detta, utan fokusera på hur vi implementerar det

CORS

- *Cross-Origin Resource Sharing*
- Webbläsare använder i regel "same-origin policy"
- Innan GET/POST anropet skickas ett **OPTIONS**-anrop till servern
  - Om rätt headers returneras så tillåter webbläsaren att man genomför GET/POST
  - Ett relativt ”snyggt” sätt att göra det på som minimerar för mycket kodändringar i redan existerande system

CORS

- Vid ett cross domain-anrop skickar klienten först ett anrop med metoden OPTIONS.
- Header i svaret från servern beskriver vad som är tillåtet.
- Klienten ansvarar sedan för att bara skicka tillåtna requests
- Sker i regel helt automatiskt

CORS: Response headers

- På **serversidan** lägger man till vad och vilka domäner som ska tillåtas utifrån som skrivs som respons i headers
- Exempel:

```
let headers = {};
headers['Access-Control-Allow-Origin'] = '*';
headers['Access-Control-Allow-Methods'] = 'POST, GET, OPTIONS';
res.writeHead(200, headers);
res.end();
```

- Måste på läggas till i alla utgående "response" som man vill göra tillgängliga
- OBS: Vi väljer här att sätta '*' vilket tillåter alla domäner att anropa servern. I en produktionsmiljö specificerar man i regel domäner som ska få skicka anrop.

CORS: Response headers

- Hur kan man snabba upp och förenkla processen med headers?

```
if(req.method == 'OPTIONS'){
  let headers = {};
    headers['Access-Control-Allow-Origin'] = '*';
    headers['Access-Control-Allow-Methods'] = 'POST, GET, OPTIONS';
    res.writeHead(200, headers);
    res.end();
} else {
    // vid POST, GET, etc.
}
```

CORS: Response headers

- I Express.js kan vi göra det på ett enkelt sätt med .**use(...)**
- .use(...) kallas på varje gång appen tar emot ett request, oavsett vilken route som används

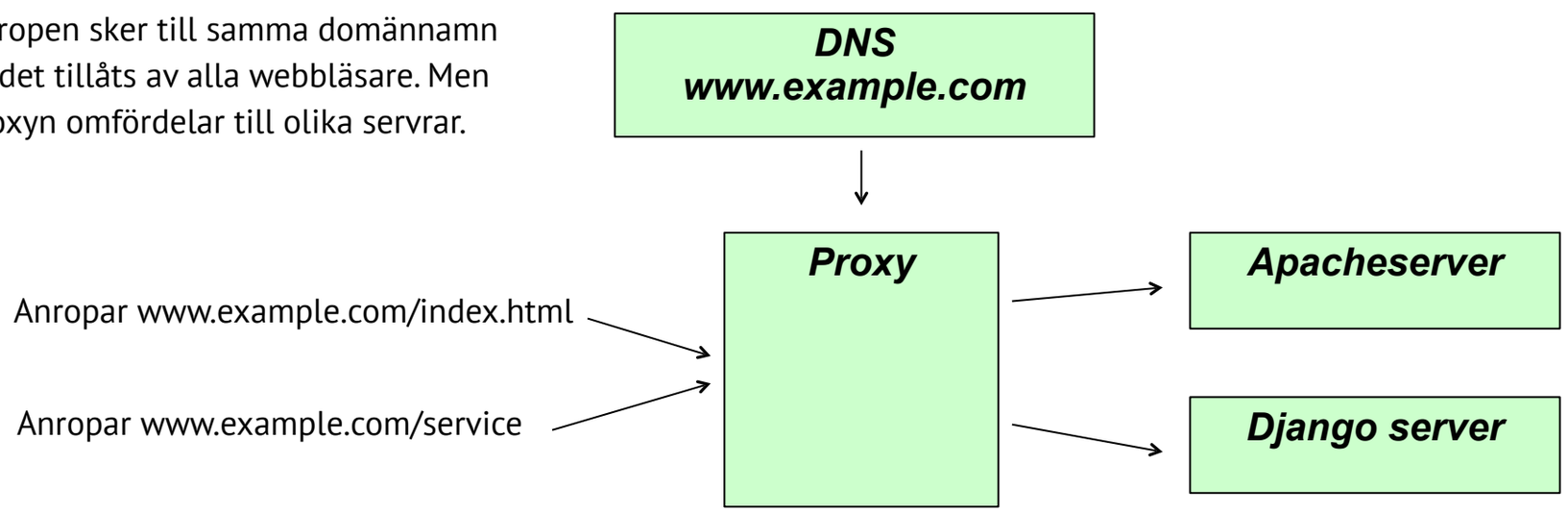
```
app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With,
             Content-Type, Accept');
  next();
});
```

- Finns libs som underlättar arbetet med CORS ytterligare
    - ...men det är en god idé att kontrollera att man inte öppnar upp för mycket!

Kan man klara sig utan CORS?

- Ja, men det blir mer komplicerat
- Man kan använda en “proxy” som hanterar alla anrop till domänen
- En proxy kan även ha andra fördelar så som ”caching” samtidigt som det fungerar med alla webbläsare
- Out-of-scope i denna kurs!

Anropen sker till samma domännamn så det tillåts av alla webbläsare. Men proxyn omfördelar till olika servrar.



Hur kontrollerar vi att CORS fungerar?

- Anropa din server från en extern domän
    - Enkelt om din server ligger online
- Alternativ 1:
    - Skapa en fil som gör ett anrop mot localhost
    - Öppna sedan filen direkt i webbläsaren!
    - **OBS**: Fungerar inte alltid för alla webbläsare
- Alternativ 2:
    - Gör ett anrop med OPTIONS och kontrollera innehåll i headers

Projektet

LINKÖPINGS UNIVERSITET

Projektet

- Projektet genomförs i par (eller enskilt) enligt webreg
- För godkänt ska de grundläggande kraven vara uppfyllda
- För högre betyg finns ytterligare krav
- Deadlines:
    - **Onsdag 11:e oktober 13–17** (projektpresentationer)
    - **Fredag 13:e oktober** (inlämning av kod)
- Se hemsidan för andra redovisningsalternativ och kompletteringar

Projektet: Kortfattat

- En social webbplats
- Användare ska kunna registrera sig, logga in och logga ut på en personlig sida
- Data ska sparas i en databas (MongoDB)
- Lösenord får inte skickas som ren text
- Alla beslut kring interaktion och design ska vara **tydligt genomtänkta**
- Testning av backend ska ske med Mocha och Istanbul/C8

Projektet: Högre betyg

1. Möjlighet för vänner att chatta med varandra i realtid med HTML5 WebSockets och socket.io-plugin till Node.js
2. Använda ett klientramverk för att bygga din applikation (React rekommenderas)

- **Krav för betyg 4:** Grundläggande kraven + 1 **eller** 2
- **Krav för betyg 5:** Grundläggande kraven + 1 **och** 2

Etikuppgift

LINKÖPINGS UNIVERSITET

Etikuppgift

- Leta reda på ett så kallat *värdedrivet* företags etiska kod eller etiska policy
- Besvara frågorna i den anpassade versionen av Gibbs reflektionsmodell och skriv en reflektion på ca 1 sida (~500 ord)
- Inkludera eller länka till etisk kod eller policy
- Lägg upp dokumenten på git och skapa en tag
- Seminarieuppgiften görs enskilt. Synka med er labbpartner och inte samma företag

- Välj företag innan **onsdag 20:e september**
- Deadline för rapport: **tisdag 26:e september**
- Seminarium: **onsdag 27:e september 13:15–17:00**

LINKÖPINGS UNIVERSITET

LINKÖPINGS UNIVERSITET