TDP013 - Webbprogrammering och interaktivitet

Föreläsning 3:
AJAX, CORS, projekt och etikuppgift

Robin Keskisärkkä
Biträdande universitetslektor
Institutionen för Datavetenskap (IDA)

LINKÖPINGS UNIVERSITET

Återblick från föreläsning 2

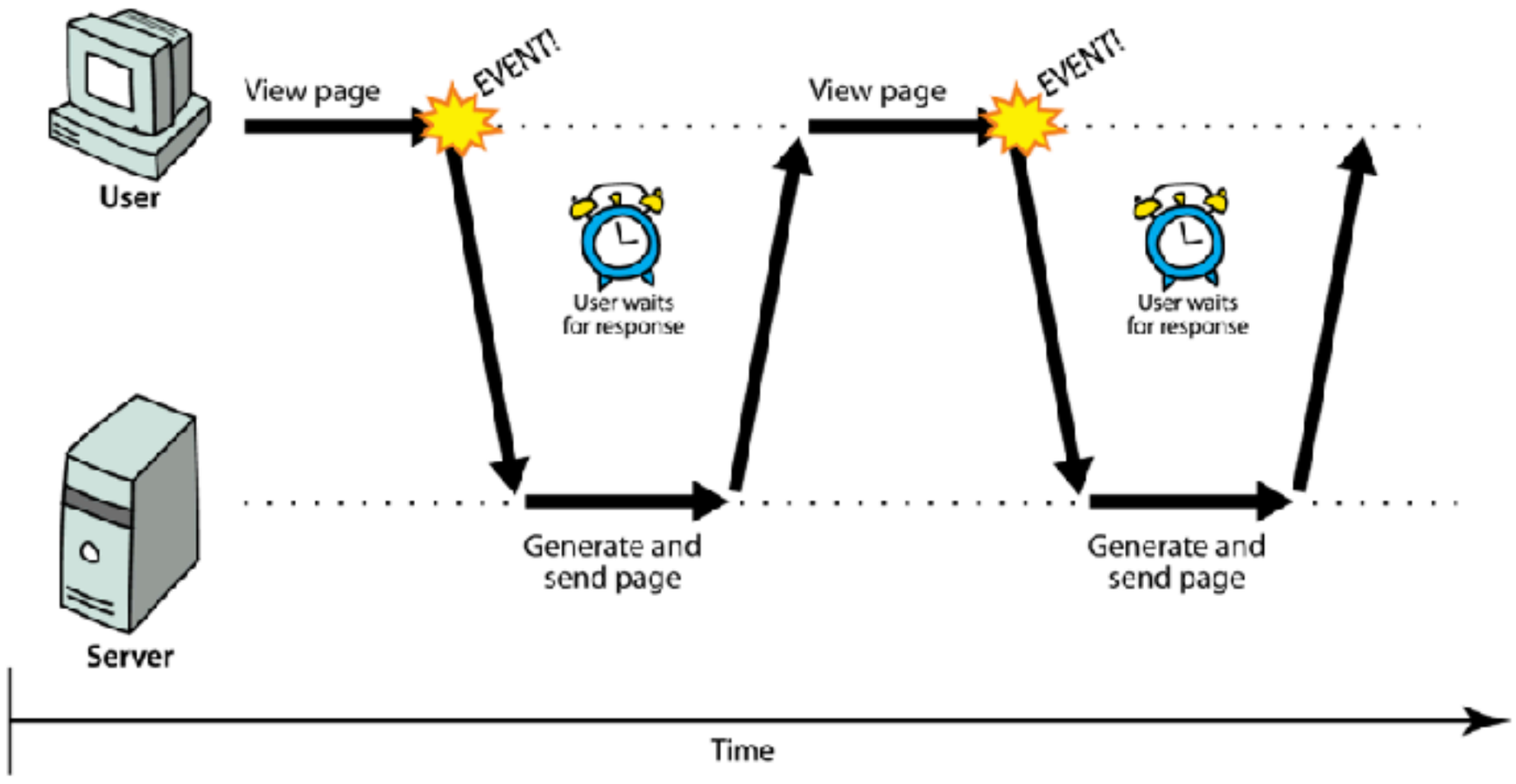
- JavaScript
    - Repetition av callbacks
- Node.js
    - Serverramverk skrivet i Javascript
    - Stöd för nästan allt i ES6 (några få undantag)
    - Fullt stöd för ES6 kräver en ‘code transcoder’ (Babel)
- Mocha
    - Ett testramverk för Node.js
- MongoDB
    - Databas där data sparas som JSON-objekt
    - Inget schema
    - Skalar bra till stora datavolymer

HTTP-anrop

Hämta och skicka data på webben

LINKÖPINGS UNIVERSITET

Synkrona anrop på webben

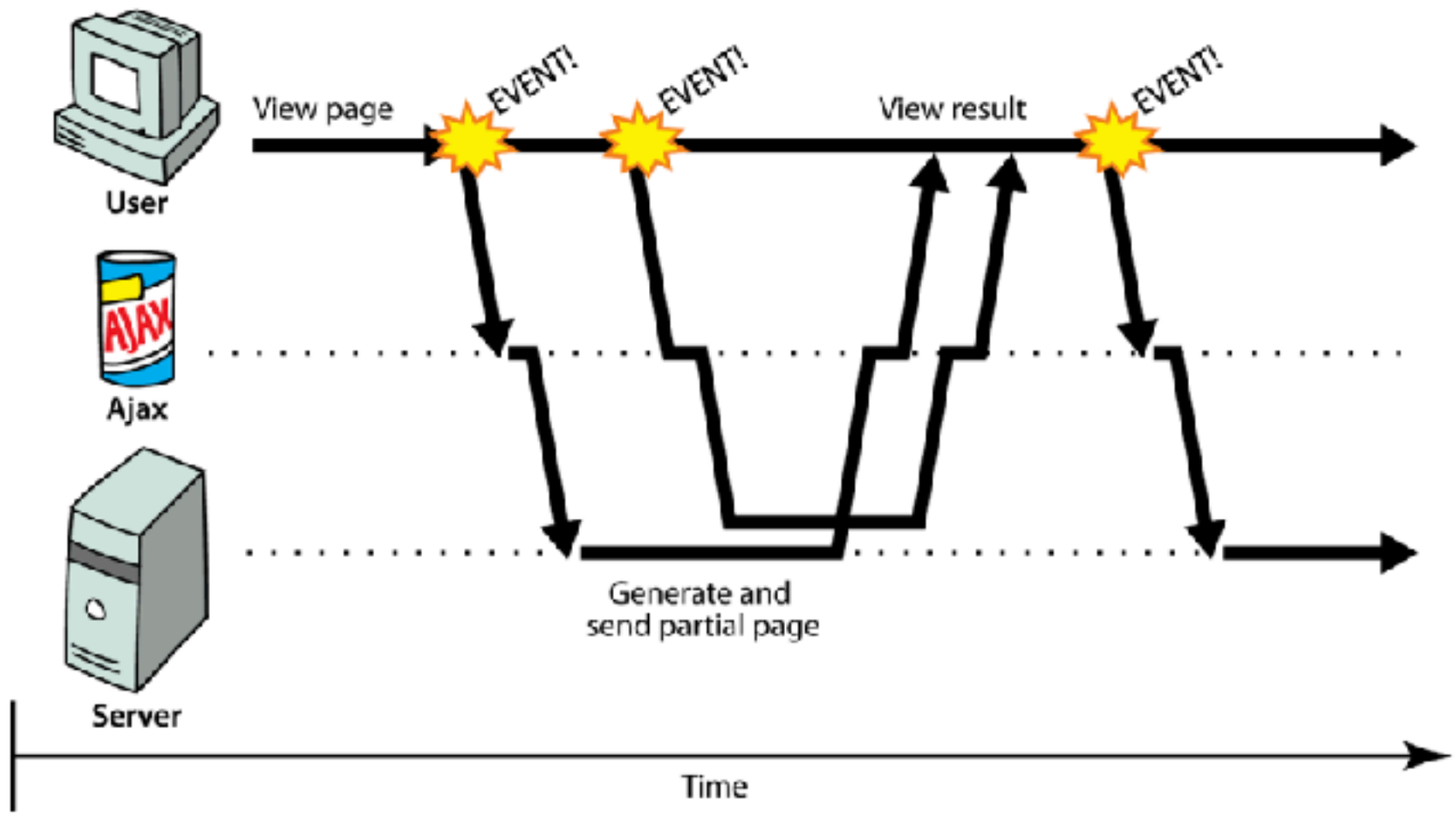


Användaren måste vänta på svar, och kan inte göra något under tiden.

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 ex. jQuery kan förenkla i vissa sammanhang men är inte nödvändiga
- Det som kommer tillbaka från servern är oftast JSON, XML, binära filer eller plain 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

```
fetch('https://gorest.co.in/public/v1/users', {'method': 'GET'})
  .then((resp) => {
    return resp.json(); // Skickas till nästa ".then"
  })
  .then((data) => {
    console.log(data);
  });
```

Alternativ med await

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

LINKÖPINGS UNIVERSITET

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

Kodexempel

LINKÖPINGS UNIVERSITET

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

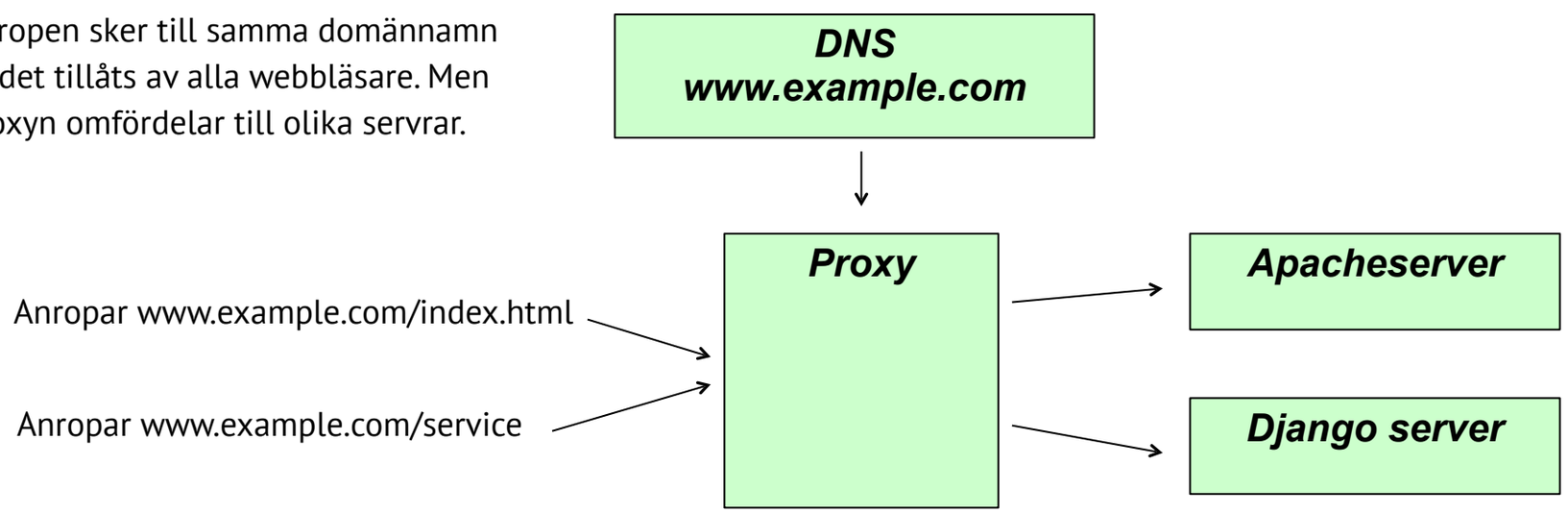
```
app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With,
             Content-Type, Accept');
  next();
});
```

- 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

- Projektet genomförs i par (eller enskilt) enligt webreg
- Kontrollera att er grupp har samma nummer för både labb och projekt för att undvika förvirring
- För godkänt ska de grundläggande kraven vara uppfyllda
- För högre betyg finns ytterligare krav

- Deadlines:
  - **Tisdag 11:e oktober 23:59 CET** (inlämning av kod)
  - **Onsdag 12:e oktober 13:15–17:00** (projektpresentationer)

Projektet: Kortfattat

- En webbplats som är navigerbar
- Användare ska kunna registrera sig och logga in 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/Istanbul

Projektet: Högre betyg

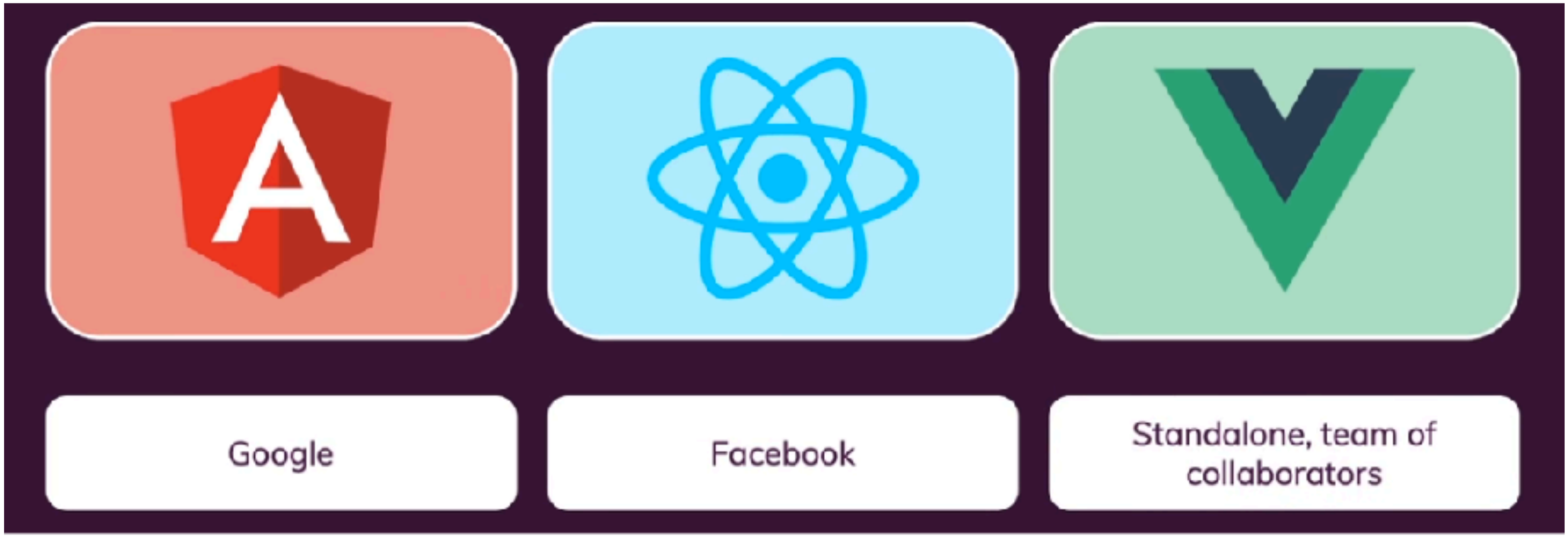
1. Möjlighet för vänner att chatta med varandra i realtid med HTML5 WebSockets och socket.io-plugin till Node.js
2. Använda ett klientramverk för att bygga din applikation (React rekommenderas)

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

Klientramverk

- Möjlighet att organisera hemsidor som komponenter
- Kan hjälpa till att uppdatera delar av en hemsida dynamiskt
- Viss inlärningströskel för samtliga ramverk
- Ramverk kommer vanligtvis med skript för att skapa grundlayout för projekt (rekommenderas)
- Använder generellt ES6

Angular, React, Vue



Backend vs. klient

- Backend
    - Node.js och Express
    - Hanterar kommunikation med databas
    - Tillhandahåller ett API för interaktion
    - **Körs på server och kan inte direkt ses av klienten**
- Client
    - Körs i browsern (hostas via HTTP-server)
    - HTML5, Javascript, CSS
    - Använder ofta ett ramverk för att bygga hemsidan
    - Kommunicerar med backend
    - **Körs av klienten och (nästan) allt kan ses av klienten**

Express för både backend och klient

- En Express-server kan hosta statiska webbsidor (dvs. fungera som en HTTP-server)
- Exempel: Innehållet i mappen ‘./public’ kan göras tillgängligt på servern med:

```
app.use(express.static('public'))
```

  Innehållet blir då tillgängligt under:

```
http://localhost:3000/index.html
```

- Ett annat alternativ är att skicka med filen som svar

```
res.sendFile(__dirname + '/index.html')
```

- Använder man ett klientramverk kör man vanligtvis detta på en separat port

WebSockets

- WebSocket använder HTTP
- ...men den underliggande TCP-kopplingen stängs inte utan hålls vid liv mellan klient och server
- WebSockets gör det möjligt att bygga “real-time” system enkelt (utan exempelvis long-polling)
- Rekommenderat bibliotek i node är Socket.IO

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) utöver eventuellt inkluderad etisk kod eller policy.
- Lägg upp dokumentet på git och dela med labbassarna innan seminariet
- Seminarieuppgiften görs enskilt
- Seminariet
- Deadline för att rapportera in valt företag: **fredag 23:e september 23:59 CET**
- Deadline för rapport: **måndag 26:e september 23:59 CET**
- Seminarium: **torsdag 29:e september 13:15–17:00**

LINKÖPINGS UNIVERSITET