TDP013 - Webbprogrammering och interaktivitet

Föreläsning 4:
Klientramverk, templates och WebSockets

Robin Keskisärkkä
robin.keskisarkka@liu.se

Institutionen för Datavetenskap (IDA)

LINKÖPINGS UNIVERSITET

Återblick från föreläsning 3

- JavaScript
    - Repetition av callbacks och promises
- Testning med Mocha och kodtäckning med Istanbul/C8
- MongoDB
    - Databas där data sparas som JSON-objekt
    - Inget schema
    - Skalar bra till stora datavolymer
- CORS
- AJAX
- Etikuppgift och projekt

Inlämningar med taggar

LINKÖPINGS UNIVERSITET

Projektet: En social webbplats

- En webbplats som liknar en förenklad version av Facebook
- Användare ska kunna registrera sig med användarnamn och lösenord samt logga in på en personlig sida
- Lösenord ska inte skickas eller sparas som läsbar text
- Användare ska bara kunna skriva på sin egen och sina vänners sidor
- Vänner ska endast läggas till efter att en vänförfrågan accepterats. En vänrelation gäller i båda riktningarna.
- Data ska sparas i en databas (MongoDB)
- Testning av backend ska ske med Mocha/Istanbul
- Hemsidan vara tydligt strukturerad och använda lämpliga färger, storlek på element, väl valda avstånd och marginaler

Visa att ni verkligen behärskar CSS!

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

Template engines och klientramverk

LINKÖPINGS UNIVERSITET

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 frontent

- Statiska webbsidor i Express (dvs. fungera som en HTTP-server)
- Exempel:

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

  Innehållet blir då tillgängligt under:

```
http://<my-domain>/index.html
```

- Ett annat alternativ är att skicka med filen som svar

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

- Använder man ett klientramverk kör man vanligtvis klient och backend på olika portar

Livekodning

LINKÖPINGS UNIVERSITET

Template engines

LINKÖPINGS UNIVERSITET

Templates för att generera sidor

- Statiska template-filer där variabler byts ut faktiska värden
- … men mer komplicerade funktioner stöds också!
- Några populära alternativ i Express:
  - Pug (tidigare Jade)
  - Mustache
  - EJS
  - hbs (Handlebars)
  - Se fler på https://expressjs.com/en/resources/template-engines.html

Templates för att generera sidor

- För att använda templates måste man konfigurera sin Express-server:
  - Specificera var template-filerna kommer att ligga
  - Välj template-engine
  - Installera template-engine via npm
  - Rendera för specifika routes

Exempel: Pug

```
app.set('views', './views')
app.set('view engine', 'pug')

app.get('/', (req, res) => {
    let data = { title: 'Hello', message: 'World!' }
    res.render('index', data)
})


// /views/index.pug
html
  head
    title= title
  body
    h1= message
```

Livekodning

LINKÖPINGS UNIVERSITET

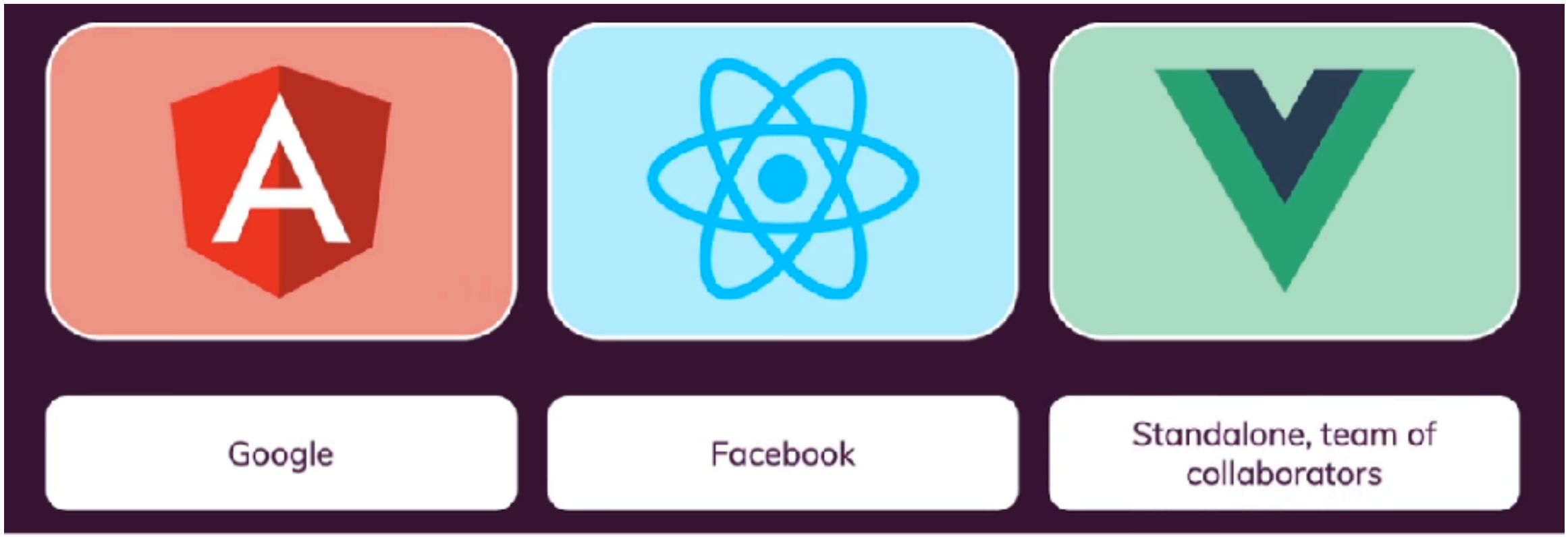
Klientramverk

LINKÖPINGS UNIVERSITET

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



WebSockets

LINKÖPINGS UNIVERSITET

WebSockets

- HTTP fungerar bara i en riktning
- WebSocket bygger på HTTP
- ...men den underliggande TCP-kopplingen stängs inte utan hålls vid liv mellan klient och server
- Kommunikation i båda riktningarna
- WebSockets gör det möjligt att bygga “real-time” system enkelt (utan exempelvis long-polling)
- Rekommenderat bibliotek i node är Socket.IO eller WS

Socket.IO

```
import express from 'express';
import http from 'http';
import { Server } from "socket.io";

const app = express();
const server = http.createServer(app);
const io = new Server(server);

app.use(express.static('client'))

app.get("/test", (req, res) => {
   res.send("Express is still working as expected!")
})

// on connect
io.on('connection', (socket) => {
   console.log('A user connected’);
}

server.listen(3000, () => console.log(‘Server is running’));
```

Socket.IO: Klientsidan

```
socket.on('message', (msg) => {
    console.log(msg)
});
```

Livekodning

LINKÖPINGS UNIVERSITET

LINKÖPINGS UNIVERSITET