**Periode 5 examens spørgsmål**

**Name attributes of HTTP protocol makes it difficult to use for real time systems.**

http fungerer dårligt som et realtime system, da det er bygget op ved hjælp af request,response. Måden den er bygget op på gør at man skal have en masse indhold, som man ikke nødvendigvis behøver. Man skal foreksempel fortælle hvilket format det er, hvor stort det er, hvilken variant osv.. Al denne information har man ikke nødvendigvis brug for hvis man skal foreksempel skal lave et spil, så har man måske bare brug for at vide spiller 1 tager 10 liv fra spiller 2.

**Explain polling and long-polling strategies, their pros and cons**

Polling er en måde hvorpå man kan få systemet til selv at opdatere når der kommet nyt input. Dette fungerer på den måde at clienten går ind og spørg serveren, har du noget nyt til mig, hvor der kommer respons om ja eller nej, hvis der er noget så bliver det sendt som et response til clienten, og hvis ikke bliver der sendt et negativt respons.

Long polling er lidt det samme, men her kommer responset ikke med det samme. I polling kommer responset fra serveren med det samme, hvis der ikke er noget, mens i long polling bliver requestet holdt "åbent" indtil serveren er klar med noget nyt, det vil sige at der ikke kommer respons før der er data klar.

Man kan ikke altid være sikkert på hvornår man vil få noget nyt respons når man arbejder med real time systems, så det kan tage en del kraft hele tiden at lave kald uden nødvendigvis at få noget.

Det kan være smart hvis man ikke behøves data med det samme, da man kan samle det i nogle lidt større kald, i stedet for hele tiden at lave små kald.

**What is WebSocket protocol, how is it different from HTTP communication, what advantages it has over HTTP?**

Websockets er full-duplex, det vil sige de kan have bidirektionel kommunikation.

Med websockets har man mindre headers, man kan lave mere specifikke headers som ikke skal have for meget information.

Websockets kan kommunikere både server til client, men også client til server på samme tid.

**Explain what the WebSocket Protocol brings to the Web-world.**

Websockets skaver en måde hvorpå man kan kommunikere(full-duplex kommunikation) begge veje uden man behøver hjælp fra plugins.

**Explain and demonstrate, using a package like Socket.io (on the Server), the process of WebSocket communication - From connecting client to server, through sending messages, to closing connection:**

**Serveren:** Når der bliver established en connection til serveren laves en socket, associeret med den nye connection

**io.on("connection", (socket) =>**

første parameter er det event som pakken skal lytte efter (her connection).

**io.emit("update", voting.results);**

Emit bruger vi til at sende ud til alle de sockets som er tilknyttet.

**socket.on("disconnect", () =>**

**{**

**console.log("a user disconnected!!");**

**}**

**);**

Ved hjælp af ovenstående kode lukker vi sockets og skriver en besked ud.

**Klienten:**

Først skal vi bruge en socketFactory:

**app.factory("mySocket", function (socketFactory, $location)**

**{**

**return socketFactory();**

**}**

**);**

Vi kan så bruge sockets I eksempelvis en angular ctrl:

**app.controller("VoteCtrl", function ($scope, mySocket)**

Da socketen kommer fra samme pakke som på serversiden, benytter den samme terminologi, hvorfor man kan gøre følgende:

**mySocket.on("update", function (res) mySocket.emit("vote", vote);**

**What's the advantage of using libraries like Socket.IO, Sock.JS, WS, over pure WebSocket libraries in the backend and standard APIs on frontend? Which problems do they solve?**

Fordelen er at disse libraries gør en hel masse arbejde for os, så vi slipper for selv at gøre det. De kan foreksempel tjekke på hvilken browser den som benytter sig af serveren bruger, og derudfra vælge hvilke strategier den skal benytte sig af. Det løser en hel masse og sørger for at vi kan ramme så mange som muligt uden at, bruge al for meget tid på at finde ud af hvad brugeren benytter sig af.