2018

Oppgave 1

Vi skal lage kode som behandler data om ulike geografiske land.

a. Lag en klasse Land som har de fire variablene navn, areal, folketall og toppdomene.

b. Lag en metode som regner ut befolkningstettheten, dvs folketall delt på areal.

c. Lag kode som oppretter objekter for landene Norge, Sverige, Russland og Kina. Disse skal være instanser av klassen Land.

   Norge (.no) har 5320045 innbyggere og har et areal på 323802 km2.
   Sverige (.se) har 9960487 innbyggere og har et areal på 450295 km2.
   Russland (.ru) har 142257519 innbyggere og har et areal på 17098242 km2.
   Kina (.cn) har 1379302771 innbyggere og har et areal på 9596960 km2.

d. Lag kode som skriver ut de fire linjene ovenfor.

Lag en metode som tar inn et domenenavn som argument og returnerer hvilket land det tilhører basert på informasjonen som vi har lagret i objektene våre.

```
class Land {
    //a
    navn: string;
    areal: number;
    folketall: number;
    toppdomene: string;

    component(navn: string, areal: number, folketall: number, toppdomene: string){
        this.navn = navn;
        this.areal = areal;
        this.folketall = folketall;
        this.toppdomene = toppdomene;
    }

    //b
    befolkningstetthet(){
        Land.prototype.befolkningstetthet = () => {
            this.folketall / this.areal;
        }
    }

    //d
```

```
    toString(){
        return(
            `${this.navn} (${this.toppdomene}) har ${this.folketall} innbyggere
             og har et areal på ${this.areal} km2. <br />`
        )
    }
}

//c
const norge = new Land("Norge", 323802, 5320045, ".no");
const sverige = new Land("Sverige", 450295, 9960487, ".se");
const russland = new Land("Russland", 17098242, 142257519, ".ru");
const kina = new Land("Kina", 9596960, 1379302771, ".cn");

//d
console.log(norge.toString());
console.log(sverige.toString());
console.log(russland.toString());
console.log(kina.toString());

//e
function finnLand (toppdomene: string) {
    switch(toppdomene){
        case ".no" : return norge.toString();
        case ".se" : return sverige.toString();
        case ".ru" : return russland.toString();
        case ".cn" : return kina.toString();

        default : return "Domenet tilhører ingen land";
    }
}
```

Oppgave 2

Du skal skrive kildekoden til en applikasjon som gjør det enkelt å holde poengoversikt når familien spiller spill. Poengene skal være lagret i en database. Tabellen med initiell data er allerede opprettet med følgende SQL-setninger:

```
CREATE TABLE Scores (
  id INT NOT NULL AUTO_INCREMENT,
  name TEXT,
  score INT,
  PRIMARY KEY(id)
);

INSERT INTO Scores (name, score) VALUES ('Far', 0);
INSERT INTO Scores (name, score) VALUES ('Lisa', 0);
INSERT INTO Scores (name, score) VALUES ('Knut', 0);
INSERT INTO Scores (name, score) VALUES ('Eli', 0);
INSERT INTO Scores (name, score) VALUES ('Mor', 0);
```

Index

```
import * as React from 'react';
import { createRoot } from 'react-dom/client';
import { Component } from 'react-simplified';
import { NavLink, HashRouter, Route } from 'react-router-dom';
import { Scores, gameService } from './services';
import { Alert, Card, Row, Column, Button, Form, NavBar } from './widgets';

//pengeoversikt familiespill

class Spill extends Component{
  scores: Scores[] = [];

    render(){
        return(
          <Card>
              <Row>
                 <Column width={2}>
                      <h4>Spiller</h4>
                 </Column>
                 <Column width={2}>
                      <h4>Poeng</h4>
                 </Column>
                 <Column width={1}>
                 </Column>
              </Row>

              {this.scores.map((spiller) => {
                 return <Row key={spiller.id}>
                      <Column width={2}>
                          {spiller.name}
                      </Column>
                      <Column width={2}>
                          {spiller.score}
                      </Column>
                      <Column width={2}>
                          <Button.Light onClick={() =>
                              this.addPoint(spiller.score, spiller.id)}>
                              +
                          </Button.Light>
                      </Column>
                  </Row>
              })}

              <Row>
                <Column width={2}>
                <Button.Danger onClick={() => this.resetScore()}>Nullstill</Button.Danger>
                </Column>
              </Row>
          </Card>
        )
    }

  mounted(): void {
      gameService.getPlayers((scores) => {
        this.scores = scores;
```

```
      })
  }

  addPoint(score: number, id: number){
    gameService.addPoint(score +1, id);
    this.mounted();
  }

  resetScore(){
    gameService.resetGame();
    this.mounted();
  }
}


let root = document.getElementById('root');
if (root)
    createRoot(root).render(
        <div>
            <Alert />
            <HashRouter>
              <Spill />
            </HashRouter>
        </div>
    )
```

Services

```
import { pool } from './mysql-pool';
import type { RowDataPacket, ResultSetHeader } from 'mysql2';


export class Scores {
  id!: number;
  name: string | undefined;
  score!: number;
}

class GameService {

  getPlayers(success: (scores: Scores[]) => void){
    pool.query(
      'SELECT * FROM scores', (error:Error, results: any) => {
        if(error) return console.error(error);

        success(results);
      }
    )
  }

  addPoint(score: number, id: number){
    pool.query(
      'UPDATE scores SET score = ? WHERE id=?', [score, id],
```

```
        (error: any, results: RowDataPacket []) => {
        if(error) return console.error(error);

        return results;
      }
    )
  }

  resetGame(){
    pool.query(
      'UPDATE scores SET score = 0', (error: Error, results: any) => {
        if(error) return console.error(error);

        return results;
      }
    )
  }
}

export let gameService = new GameService();
```

OUTPUT