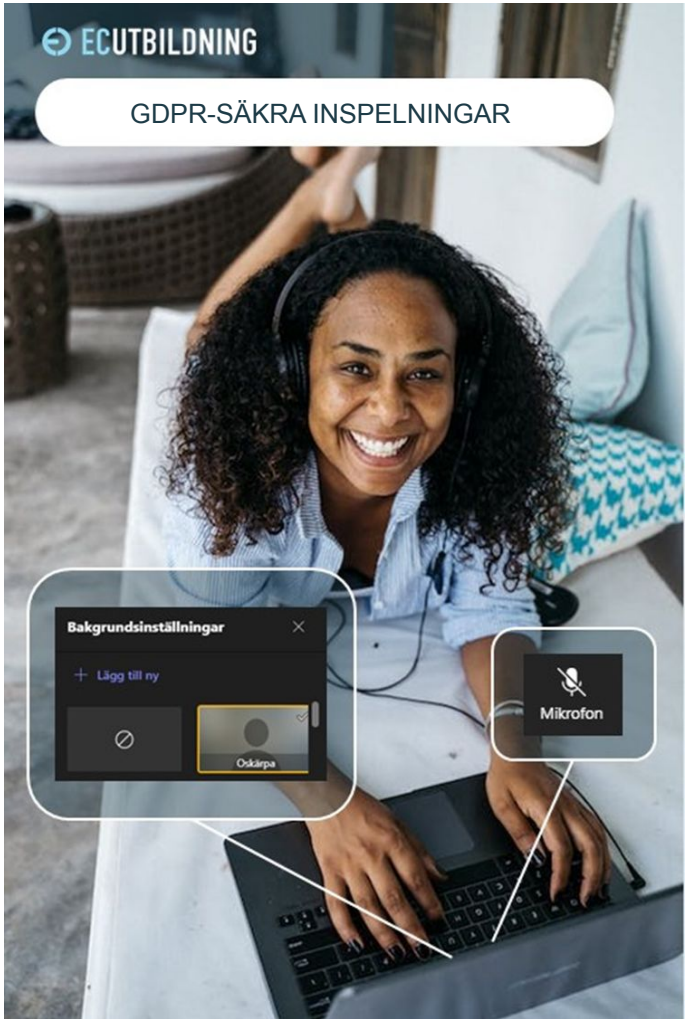
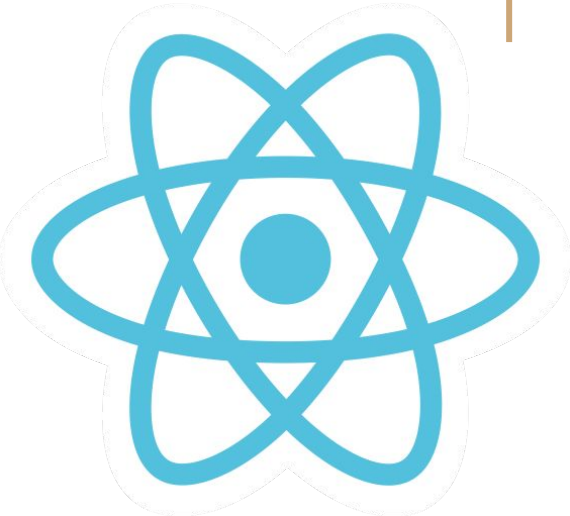
Roligt att just du är här idag!
För allas trevnad och för att jobba med GDPR-säkra inspelningar har EC Utbildning följande rutin kring inspelning av föreläsningar/lektioner.

- Blurra bakgrunden (använd oskärpa) genom att använda funktionen för bakgrundsfilter i Teams om du har kameran på, eller ha kameran avstängd.
- Mikrofonen skall vara avstängd när den inte behövs
- Inga privata samtal eller chatt medan inspelning pågår
- Sitt i en tyst miljö för att undvika bakgrundsljud
- Stäng av din kamera och mikrofon om du lämnar föreläsningen/lektionen tillfälligt och vid paus/rast
- Inspelade föreläsningar/lektioner får inte spridas utanför skolan.

Ha en lärorik dag!

ECUTBILDNING x gaddr





Introduction to React.js

[09/2022]

LECTURE 2

Node server & npm, React.js, Web Server Technology, React Project Architecture, jsx

ECUTBILDNING x gaddr

Course Schedule

v35 - v38

Tuesdays: Lektion @ Göteborg + Halmstad: 09.00 - 16.00

Wednesdays: Lektion @ Halmstad: 09.00 - 16.00

Thursdays: Lektion @ Göteborg: 09.00 - 16.00

On one of the days I will travel and we will have an on-campus lecture in both cities!

Information about submissions

1. Inlämningsuppgift
   - individually graded
   - will be uploaded by the end of this week (probably friday or saturday)
   - deadline will be posted along with that
2. Projekt
   - group work
   - you can receive VG through this
   - will be uploaded by the end of this week (probably friday or saturday)
3. Skriftlig rapport till projekt
   - individually graded
   - information on format etc will be uploaded by the end of this week (possible friday or saturday)

Dagens lektion

- creating a node.js server
- learning about packages and node_modules
- creating our first react app
- react project architecture

node.js and its packages

a small run-through of information

- npm (node package manager)
- packages - Node packages and React packages
- package.json
- node_modules

Creating a node server

1. `npm init`
2. `git init` [optional]
3. write index.js
4. add “start” script in package.json
5. `git add .` [optional]
6. `git commit -m “Initial commit”` [optional]
7. `npm start`
8. Ctrl/cmd + C to stop the server

Basic http server with node

```
const http = require("http"); // es5 syntax

const hostname = "127.0.0.1";
const port = 3000;

const requestListener = (req, res) => {
 res.statusCode = 200;
 res.setHeader("Content-Type", "text/plain");
 res.end("Hello World");
};
```

```
const server =
http.createServer(requestListener);

server.listen(port, hostname, () => {
 console.log(`Server running at
http://${hostname}:${port}/`);
});
```

to run:

`node index.js` OR `npm start`

Basic http server with node: serving an html file

```
const http = require("http"); // es5 syntax
const fs = require("fs");

const hostname = "127.0.0.1";
const port = 3000;
const filePath = "./index.html";

const requestListener = function (req, res) {
 res.writeHead(200, { "content-type":
"text/html" });
 fs.createReadStream(filePath).pipe(res);
};
```

```
const server =
http.createServer(requestListener);

server.listen(port, hostname, () => {
 console.log(`Server running at
http://${hostname}:${port}/`);
});
```

to run:

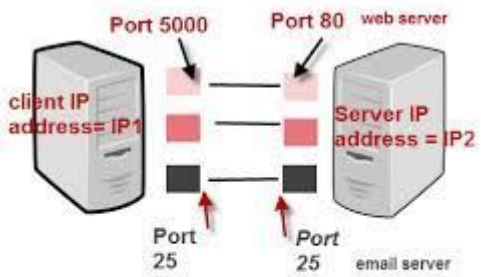
`node index.js` OR `npm start`

ECUTBILDNING x gaddr

Server technology

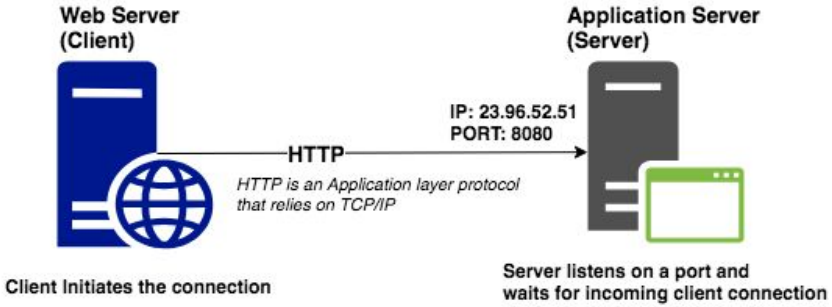
Web server **ports** are the logical endpoints of a network connection that is used to exchange information between a web server and a web client.

For our case, our computer is the server which serves on a port in our local machines, and the browser is the client which requests information from the server.



IP Address + Port number = Socket

TCP/IP Ports And Sockets



(C) Karun Subramanian

Creating a node express server with npm

1. `npm init`
2. `git init`
3. `npm install express`
4. write index.js
5. add “start” script in package.json
6. add `“type”: “module”` to package.json
7. `npm start`
8. Ctrl/cmd + C to stop

Basic express server with node

```
import express from "express" // es6 import style
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port
${port}`)
})
```

to run:

`node index.js` OR `npm start`

package.json

important documentation for your entire project

The heart of your project is package.json

- records important metadata
- stores dependencies - packages required for successfully running the project
- points at the entrypoint file (usually index.js)
- defines important scripts: `start`, `build`, `run`, `test` etc (can create custom ones too)

Can almost call it the most important documentation of your project

package.json

a bit more about scripts and dependencies

- defines important scripts: `start`, `build`, `run`, `test` etc (can create custom ones too) These are basically just the commands executed in the terminal
  to run:
  - npm start
  - npm run build
  - npm run test
  - npm run my_custom_script
- defines important dependencies:
  - packages and modules which are required to run your project

node_modules

important folder containing dependencies

- Important folder which stores all your packages - your project is dependent upon all these packages
- NEVER version controlled = NEVER uploaded to git (even package-lock.json or equivalent)
- contents controlled by package.json, further information about packages in package-lock.json

Finally, react!

Building a basic react app

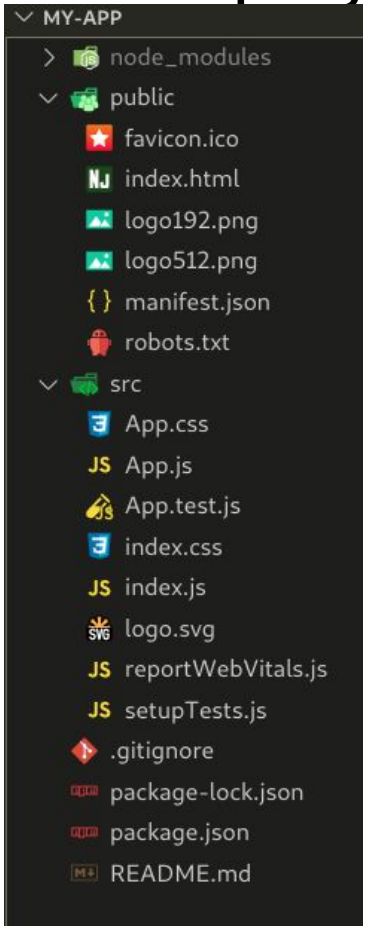
You can start one from scratch (not recommended)

Or, create one using “create-react-app”:

1. `npx create-react-app my-app`
2. `cd my-app`
3. `npm start`

You just built your very first React application!

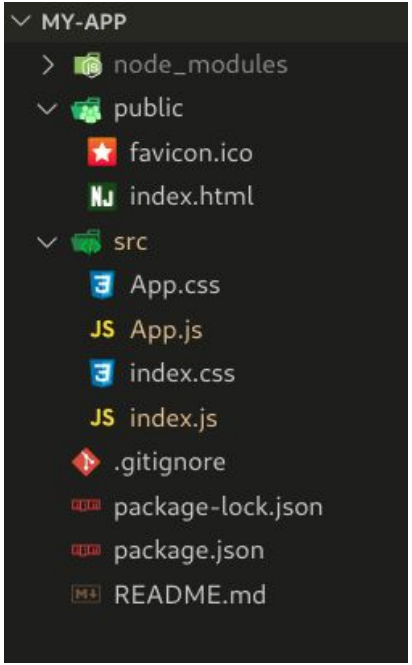
React project structure / architecture



- to keep it simple, we are going to remove some of the extra items which might be confusing at the beginning
- Removing:
  - `reportWebVitals.js` and all its references (inside src)
  - `setupTests.js, App.test.js` and all its references
  - `manifest.json, robots.txt`
  - `logo.svg` and all its references
  - `logo192.png, logo512.png`

We are now left with a very basic react app and can start editing it

React project structure / architecture



- Main components:
  - src folder
  - node_modules folder
  - package. json
  - index.js (or whatever you like to call it)
  - index.html
- Other major components:
  - README.md
  - index.css (and other css files)
  - package-lock.json
- Commonly used:
  - public folder to store static assets (images, etc)

How React looks like and works (`jsx`)

What is jsx?

It is like html tags. But we can create custom ones with react - using components

rules:

- All of the code should be wrapped in a single tag. This could be a `div`, a tag with no content (<>), or any other tag.
- The markup is usually placed after the return statement, either as a single line of code or as a block code.

How React looks like and works (jsx)

jsx basically compiles into the following:

```
<MyButton color="blue" shadowSize={2}>
  Click Me
</MyButton>
```

```
React.createElement(
  MyButton,
  {color: 'blue', shadowSize: 2},
  'Click Me'
```

the final page is served through the React DOM

```
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(<h1>Render me!</h1>,
document.getElementById('app'));
```

DOM is what you see on the webpage: html, div, body tags are all DOM elements

How React looks like and works (jsx)

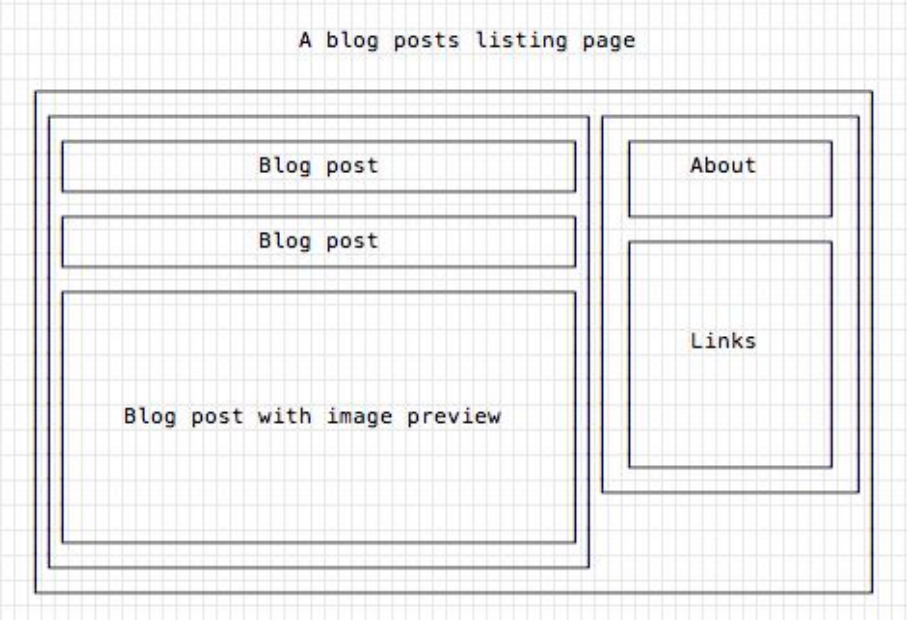
We can also add vanilla javascript right inside our jsx using curly braces {}

```
return (
  <div className="outerDiv">
     <ul>
        { myList.map((item) => <li>{item}</li>) }
     </ul>
  </div>
)
```

React components

They are independent parts which communicate with each other and build up the web page

Can be nested within one another



Experimentet

Minor programming tips :)

Variable naming is important:

- keep all names consistent. If you are using `snake_case`, use it throughout. If you are using `camelCase`, use that throughout your project. Make it look pretty! (Usually in js we always use camel case. Python etc use snake case)
- keep the names as descriptive as possible.
  say no to `const x = [4, 5], y = “important string”`
  say yes to `const idArray = [4,5], tokenSecret = “important string”`
- keep commenting all throughout your code! If you create algorithms and complex (or even simple) functions, add a little comment above it to help you out later on

Keywords for today

1. Node.js
2. NPM, packages and package managing
3. libraries, frameworks and runtime environments
4. Node server
5. Serving static files on a web server
6. jsx

Today's task

Create a basic one-page react webpage from scratch using "create-react-app" command. It should contain the following items:

(All of them except for the last one should be wrapped inside separate components)

1. an image sourced from the public folder

   (the *image* should be placed inside the "public" folder. Then you create a component inside "src" which imports this image)
2. flexbox styling with css
3. a list of items
4. a table styled with css
5. Use the following libraries and add items in your application:
   a. react-back-to-top-button
   b. simple-random-number-generator