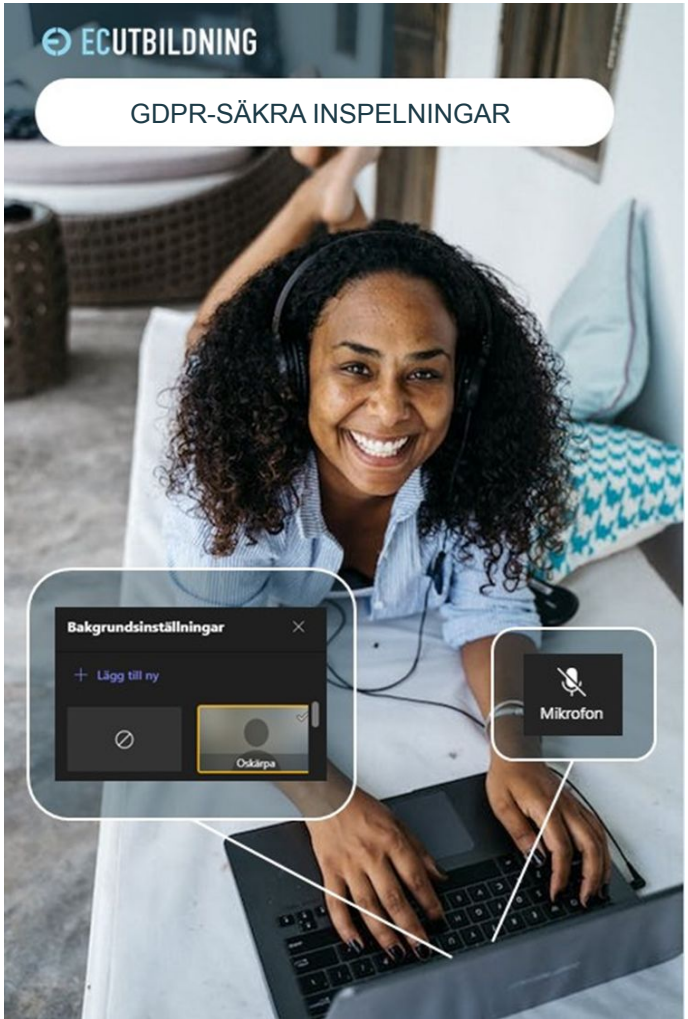
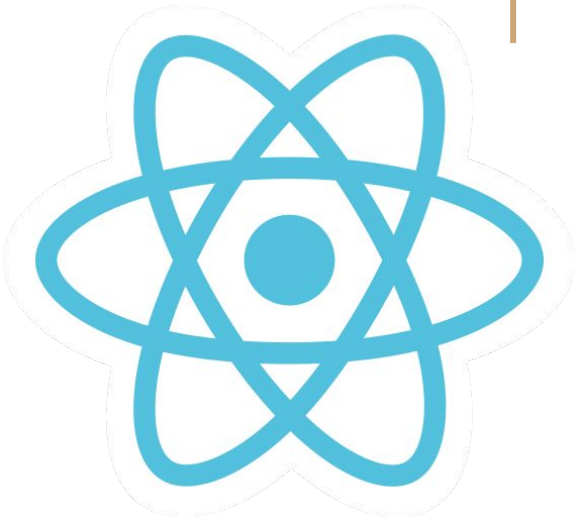
Roligt att just du är här idag!
För allas trevnad och för att jobba med GDPR-säkra inspelningar har EC Utbildning följande rutin kring inspelning av föreläsningar/lektioner.

- 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 5

Fetching from APIs

ECUTBILDNING x gaddr

Rest of the course

- 13 sep: Fetching from APIs, time for finishing your inlämning
- 14 / 15 sep: An interactive web application using TypeScript (Forms, Styling, Styled Components, Props); the different packages useful during development
- 20 sep: overview and questions + time for finishing your project and presentation
- 21 / 22 sep: ***presentations!*** Register your team:
  https://docs.google.com/spreadsheets/u/2/d/1uTF1LXOrtJQXHiTx9_L7IqCcmRZmQXItXtRJ7to5P7k/edit#gid=653812220

Updates to the projekt spec

More information about how to obtain VG is now specified in the document

Updated on Omniway and added as an announcement in the Teams React page.

Dagens lektion

- API’s
- fetching data from API’s
- Display fetched data into our React Application

API

Application Programming Interface

- An application programming interface (API) is a way for two or more computer programs to communicate with each other.
- In our context (for front-end development), it is basically how we can “fetch” data from another server/computer to display on our webpages

What a request looks like

- The data comes from a URL, t. ex.: `http://localhost:8080/getAllTodos`
- Types of (more commonly used) requests:
  a. **GET**
  to request data from a specified resource
  the query string (name/value pairs) is sent in the URL of a GET request after " **?** "
  `/test/demo_form.php?name1=value1&name2=value2`
  b. **POST**
  to send data to a server to create/update a resource
  data sent to the server with POST is stored in the request body of the HTTP request

  more types described in **https://www.w3schools.com/tags/ref_httpmethods.asp**

Fetching data from API

- Usually gotten from a database storage somewhere
  These databases store the data from your website which you can access later
- In our case, it should always return JSON data
  Since JavaScript and JSON work together
- There can also be authentication required to access this data, this authentication token is described in and sent with the API request itself

Sending an API request using “fetch”

1. Choose an API URL from: **https://jsonplaceholder.typicode.com/**
2. use “fetch” to *request* data
3. store the *response* into a variable

Fetch function: (from **https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch**)

```
fetch('http://example.com/movies.json')
  .then((response) => response.json())
  .then((data) => console.log(data));
```

Here we are fetching a JSON file across the network and printing it to the console.

The simplest use of fetch() takes one argument — the path to the resource you want to fetch — and does not directly return the JSON response body but instead returns a promise that resolves with a Response object.

Supplying more request options in “fetch”

```
const url = "http://example.com/answer";
const data = { answer: 42 };

const response = await fetch(url, {
   method: "POST", // *GET, POST, PUT, DELETE, etc.
   mode: "cors", // no-cors, *cors, same-origin
   headers: {
      "Content-Type": "application/json",
   },
   body: JSON.stringify(data), // body data type must match "Content-Type" header
});

return response.json(); // parses JSON response into native JavaScript objects
```

for asynchronous requests

Keywords for today

1. Application Programming Interface (API)
2. fetch function
3. GET, POST requests

Today's task

Create a basic one-page react webpage using command
`npx create-react-app app_name`

It should contain the following items as separate components:

1. Create a graphical display of data using the API:
   https://datausa.io/api/data?drilldowns=Nation&measures=Population
   and npm package: `react-chartjs-2`

   (or any other react package to create charts / any API data of your choice)

(you may also choose to begin/continue work on your inlämningsuppgift/projekt instead of, or after this task)
Note: At this point you should have all the knowledge to address all the (necessary+optional) points in your project too!