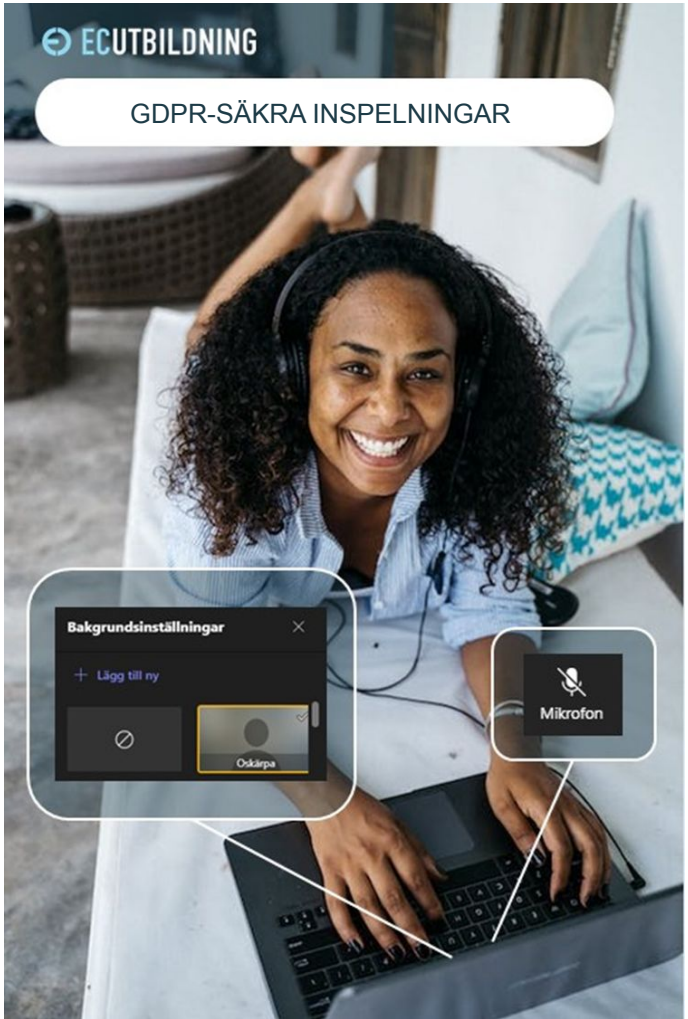
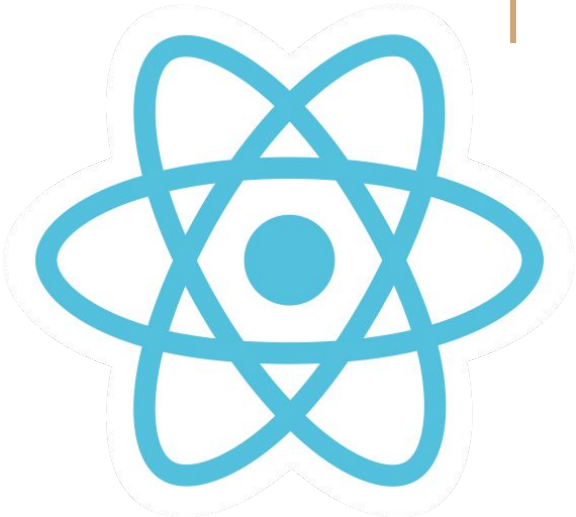
Roligt att just du är här idag!
För allas trevnad och för att jobba med GDPR-säkra inspelningar har EC Utbildning följande rutin kring inspelning av föreläsningar/lektioner.

- 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 4

useEffect hook, React router, responsive styling

ECUTBILDNING x gaddr

Dagens lektion

- the “useEffect” hook
- routing with React
- discussion about responsive styling

The “useEffect” hook

- In React class components, the render method itself shouldn’t cause side effects. It would be too early — we typically want to perform our effects after React has updated the DOM.
- This is why in React classes, we put side effects into `componentDidMount` and `componentDidUpdate`.

The “useEffect” hook

There are three key moments when the Effect Hook can be utilized:

- When the component is first added, or mounted, to the DOM and renders. So KEEP IN MIND that it runs only after the component is rendered/mounted onto the DOM
- When the state or props change, causing the component to re-render
- When the component is removed, or unmounted, from the DOM.

The “useEffect” hook

- **`useEffect`** is a Hook that lets you perform side effects in function components. By using this Hook, you tell React that your component needs to do something after render.
- Similar to **`componentDidMount`** and **`componentDidUpdate`**
- Declaring a **`useEffect`** function syntax (for **`componentDidMount+Update`** all together):

  ```
  useEffect(() => {});
  ```
- Declaring a **`useEffect`** function syntax (for separated **`componentDidMount + Update`**):

  ```
  useEffect(() => {}, [var]);
  ```

useEffect hook: Code along

“useEffect” hook syntax cheatsheet

- ```
  useEffect(() => {
      // your logic here
  });
  ```

  runs after the component is mounted onto the DOM ***and*** every time any state/prop is updated
  basically like **`componentDidMount+Update`** both put together

- ```
  useEffect(() => {
      // your logic here
  }, []);
  ```

  runs ***only once*** after the component is mounted onto the DOM
  basically like **`componentDidMount`**

- ```
  useEffect(() => {
      // your logic here
  }, [myVariable]);
  ```

  runs after the component is mounted onto the DOM ***and*** every time myVariable is update
  basically like **`componentDidUpdate`** but specifically only for when `myVariable` updates
  This can be an array of different variables `[myVar1,myVar2,myVar3]`
  So when ***either*** of those vars are updated, this code logic runs

ECUTBILDNING x gaddr
  ```

“useEffect” clean up

- Some effects require cleanup to reduce memory leaks.
- Timeouts, subscriptions, event listeners, and other effects that are no longer needed should be disposed.
- We do this by including a return function at the end of the `useEffect` Hook.

```
useEffect(() => {
   let timer = setTimeout(() => {
      setCount((count) => count + 1);
   }, 1000);

   return () => clearTimeout(timer);
}, []);
```

Note: To clear the timer, we had to name it.

Routing in React

- What is routing?
  In our context, it is receiving data from a particular endpoint or path
- We can create multiple pages with this in our application
- Your local program would look like this:
  `localhost:3000/home`
  `localhost:3000/about`

React-Router

https://reactrouter.com/

Example- add to main entrypoint file (app.js):

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<App />} />
    <Route path="home" element={<Home />} />
    <Route path="about" element={<About />} />
  </Routes>
</BrowserRouter>
```

React Router: Code along

SPA: Single Page Application

- A single-page application is a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server
- This is done instead of the default method of a web browser loading entire new pages.
- In our case, we have only one `.html` page located inside the public folder. On this page, content keeps getting re-written, instead of the default way of having multiple `.html` pages being called from the server

Responsive styling

- Responsive Web allows us to adjust a website content into the best layout for the device displaying it.
- It is absolutely imperative for modern-day websites
- A Responsive component is a component that can respond to screen size changes, adjusting its contents into the optimal layout for the given parameters.

Responsive styling: ways to do it

With CSS:

1. use flexboxes
2. use grids
3. use @media queries
4. use bootstrap

Otherwise:

1. use pre-existing react packages which help you create responsive components (higher order components)
2. use pre-existing media query modules
3. use a CSS library with pre-created classes as classNames in your components, or create them yourself (example on the next page)

media query example

```
body {
  background-color: pink;
}
@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
```

changes the background-color to lightgreen if the viewport is 480 pixels wide or wider
(if the viewport is less than 480 pixels, the background-color will be pink)

CSS library with pre-created classes

Create a CSS library of your own, or import one

```
.flex-col {
  display: flex
  flex-direction: column
}
```

```
.flex-row {
  display: flex
  flex-direction: row
}
```

```
.align-center {
  align-item: center
}
```

```
.bg-pink {
  background-color: pink
}
```

```
.text-center {
  text-align: center
}
```

```
.text-white {
  color: white
}
```

```
<Component className=".flex-col .align-center .bg-pink" />
```

(use multiple classes for the same component)

Keywords for today

1. “useEffect”
2. side effects
3. routing
4. responsive styling

Today's task

Create a basic one-page react webpage using command

```
npx create-react-app app_name
```

It should contain the following items:

1. a component with 2 or more child components inside it
   a. some of the child components may also contain children of their own
   b. all elements inside the component should be responsively styled
2. props being passed down among components
3. the "useEffect" hook that changes some data based on another change

(you may also choose to begin work on your inlämningsuppgift instead of, or after this task)

Note: At this point you should have all the knowledge to address all the (necessary) points in your project too!