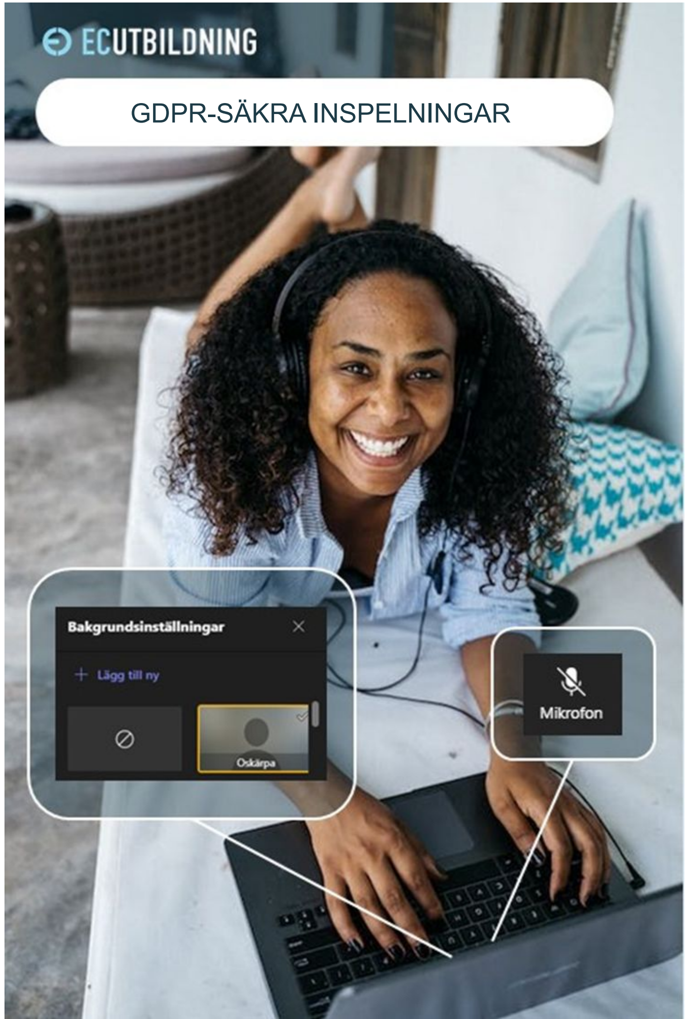
Roligt att just du är här idag!
För allas trevnad och för att jobba med GDPR-säkra inspelningar har EC Utbildning följande rutin kring inspelning av föreläsningar/lektioner.

- 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 3

React Props, React Lifecycle & State, Intro to hooks, useState hook

ECUTBILDNING x gaddr

Dagens lektion

Note: Always be ready with a basic react app before every lecture and open VS Code. You can keep continuing to build upon the very first one

- information about submissions
- React props
- Class components
- React state & lifecycle
- React hooks
- “useState” hook

Information about submissions

1. Inlämningsuppgift
    - individually graded
    - uploaded to Omniway
    - deadline : 14th Sept (if you have problems with this deadline, send me a message)
    - All topics to be covered by Lecture 4. Best if you start immediately and keep adding things along the way as you learn more in the course
2. Projekt
    - group work, **2-4** students per group. GB and HM cannot mix and match. Register yourself as a group here (contact me if you want to do it individually)
    - uploaded to Omniway
    - you can receive VG through this
    - Presentation in the last lecture
3. Skriftlig rapport till projekt
    - individually graded
    - information on format etc uploaded to Omniway along with the project
    - Written in English

React props

- When you want components to interact with each other
- When you want components to communicate with each by passing data

React Props: Code along

React components: Classes vs Functions?

(Experiment)

React State

- The state is a built-in **React object** that is used to contain data or information about the component.
- A component's state can change over time; whenever it changes, the component re-renders.
- This change in state can happen as a response to user action or system-generated events and these changes determine the behavior of the component and how it will render.

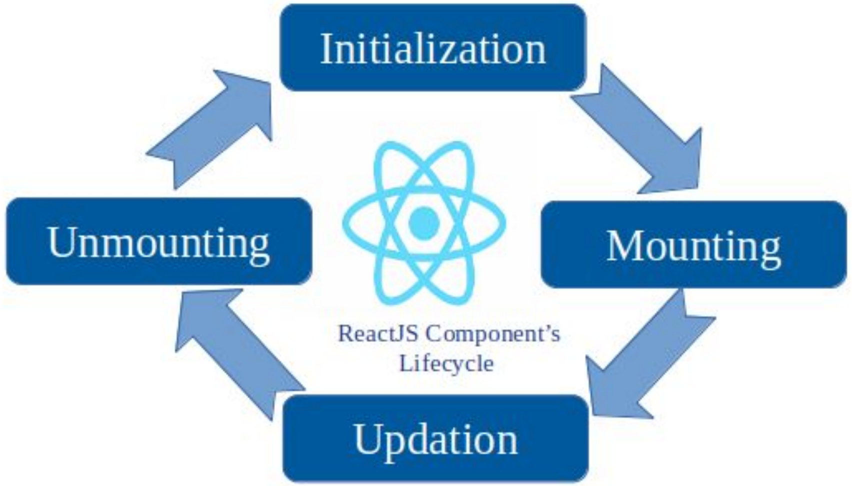
Changing the class component state

Done using specialized methods for this

`setState()`

Inside class components, **`setState()`** is the only legitimate way to update state after the initial state setup.

React lifecycle



Each component in React has a lifecycle which you can monitor and manipulate during its three main phases. The three phases are:

Mounting, **Updating**, and **Unmounting**.

React Lifecycle (methods)

In the case of classes, there are many methods. But the most important ones are:

- Mounting (means putting elements into the DOM)
  `componentDidMount()` - run this code after the component has mounted on the DOM
- Updating (*when* a component is updated ***whenever there is a change in the component's state or props***)
  `componentDidUpdate()` - called after the component is updated in the DOM
- Unmounting (*when* a component is removed from the DOM. NOT NECESSARY TO DO TO EACH COMPONENT.)
  `componentWillUnmount()` - run this code after the component has unmounted from the DOM. Important for clean up of any side effects!!!

Why no more classes

React class components will fade away in the future. If you want to embrace modern React, then you should use function components with hooks.

- classes gradually become unreadable if they store different logic in one place as they grow. For example in componentDidMount() you can place a fetch request, you can set the style for an element, or connect to the WebSocket. All of this is happening in one lifecycle method!
- class components are very confusing for both humans and machines, especially at **binding** and **this** keyword.
- Class components are harder to test compared to functional components.

Why functional components are better

The use of functional components with hooks is a much better way to write the components.

- It’s easier to separate logic from UI, making both reusable.
- You don’t have to waste your time refactoring a functional component into a class component when you need to add a state or lifecycle methods.
- No need to worry about **this** or **bind** anymore.
- It’s easier to share stateful logic between components.

So now we don’t use class components anymore

React hooks

- They let you use state and other React features without writing a class.
- Hooks make React so much better because you have simpler code that implements similar functionalities faster and more effectively.
- You can also implement React state and lifecycle methods without writing classes.

The “useState” hook

- `useState` is a Hook that lets you add React state to function components.
- If you write a function component and realize you need to add some state to it, previously you had to convert it to a class. Now you can use a Hook inside the existing function component.
- Declaring a state variable syntax:

  `const [myVar, setMyVar] = useState(“initial value”);`

  example: `const [carColor, setCarColor] = useState("red");`
- Updating the state variable:

  `setMyVar(“updated value”);`
- This “value” can be anything: strings, numbers, objects, Date, etc

useState hook: Code along

Keywords for today

1. class components vs function components
2. props in React
3. React state
4. React lifecycle
5. React hooks
6. useState

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 3 or more child components inside it
   a. some of the child components should also contain children of their own
2. props being passed down multiple layers
3. at least one component being reused at multiple places with different prop inputs
4. the “useState” hook to maintain and update state for variables
   a. One of them should be a “counter” : every time a button is pressed, it should increment a certain value on the screen