Avancerad JavaScript

Föreläsning 11

NORTH LINK

Vad ska vi göra idag?

- Kort om portals
- Introduktion till hooks
  - useState
  - useEffect

Portals

- Ibland vill vi rendera element utanför applikationens rot-element
- T ex brukar modala dialogrutor renderas i ett element som ligger längst ner i body-taggen för att kunna positionera dialogrutan ovanför alla andra element på sidan
- I React brukar man använda en teknik som kallas portals för att kunna rendera React-komponenter utanför applikationens rot-element

https://reactjs.org/docs/portals.html

React Hooks

- React hooks introducerades i React 16.8
- Med hooks kan vi skriva komponenter med tillstånd utan att skriva klasser
- Läs gärna följande introduktion till hooks som även ger en motivation till varför hooks har lagts till
  https://reactjs.org/docs/hooks-intro.html
- React innehåller ett antal inbyggda hooks men det går även att skapa hooks själv, s.k. “custom hooks”

useState

- I React 16.8 introduceras ett antal “hooks” som alla är funktioner där funktionsnamnet börjar med “use”
- Den första vi ska kolla på är “useState” som används för att ge state till funktionskomponenter
- Tidigare kallades funktionskomponenter för “tillståndslösa funktionskomponenter” men fr o m React 16.8 är de inte längre tillståndslösa

useState

- **useState** används för att skapa en tillståndsvariabel och används genom att anropa funktionen i en funktionskomponent
- useState returnerar en array som både innehåller värdet på tillståndsvariabeln och en funktion som används för att uppdatera den
- Som argument används tillståndsvariabelns initiala värde

```
import { useState }, React from ‘react’;

function MyStatefulComponent(props) {
  const [ value, updateValue ] = useState(0); // Initierar en variabel till
värdet 0
  ...
}
```

NORTH LINK

useState - Exempel

- Här är ett enkelt exempel som använder en tillståndsvariabel för att ändra färg på ett element

https://github.com/argelius/ec-advanced-js-samples/blob/master/lesson11/hooks-use-state/src/App.js

Övning - useState

1. Skapa en ny React-applikation med en funktionskomponent
2. Komponenten ska ha ett formulär med radioknappar där användaren kan välja sitt favoritdjur (t ex hundar, katter, hästar, fiskar, etc.)
3. Formuläret ska kontrolleras genom en tillståndsvariabel som skapas med **useState**
4. Skriv ut något i stil med “Jag gillar också hästar” under formuläret

Flera tillståndsvariabler

- **useState** ger oss enstaka tillståndsvariabler och en funktion som uppdaterar en variabel
- Detta skiljer sig lite från klasskomponenter där tillståndet är ett objekt och vi kan uppdatera delar av objektet med **this.setState**
- För att ge en komponent flera tillståndsvariabler går det att anropa useState flera gånger

```
const [color, updateColor] = useState('red');
const [height, updateHeight] = useState(100);
```

Övning - Beräkna BMI

1. Skapa en ny React-applikation
2. Skapa en funktionskomponent, **BodyMassIndexCalculator**, som har ett formulär där man kan mata in en längd (i cm) och en vikt (i kilogram)
3. Använd längden och vikten för att beräkna ett värde för BMI och skriv ut det på skärmen

Viktigt att tänka på

- Anropa alltid useState (och andra hooks) längst upp i en funktion
- Anropa aldrig hooks inuti if-satser eller loopar
- Detta är viktigt eftersom ordningen som hooks anropas är viktigt

```
const [height, updateHeight] = useState(180);
const [weight, updateWeight] = useState(80);
```

- I koden ovan finns det inget annat sätt för React att veta vilken variabel som är vilken förutom ordningen som useState anropades

useEffect

- För komponenter som har sidoeffekter (t ex att de hämtar data från något API eller uppdaterar DOM:en) har det tidigare inte varit möjligt att använda funktionskomponenter
- Med en hook som heter **useEffect** kan vi göra samma saker som vi gör i livscykelmetoder som **componentDidMount** och **componentDidUpdate**

https://reactjs.org/docs/hooks-effect.html

useEffect - Exempel

- useEffect tar en funktion som argument och den funktionen kommer att exekveras både när komponenten skapas första gången och varje gång den uppdateras
- Därför motsvarar **useEffect** både componentDidMount och componentDidUpdate för klasskomponenter

```
function MyComponent() {
  const [data, updateData] = useState(null);

  useEffect(() => {
    fetchData().then(updateData);
  }, []);

  ...
}
```

NORTH LINK

useEffect - Undvika loopar

- Precis som med **componentDidUpdate** är det möjligt att orsaka en oändlig loop med **useEffect**
- Anledningen är att funktionen körs efter varje uppdatering och att en effekt brukar orsaka en uppdatering
- För att motverka detta kan vi skicka en Array som argument till useEffect med alla värden som vi vill ska orsaka att effekten körs om de ändras
- Detta kan användas på samma vis som när vi jämför **prevProps** med **this.props** i componentDidUpdate
- I exemplet på förra sidan finns det inga props som ska orsaka att vi hämtar ny data så vi anger en tom Array som argument

useEffect - Exempel

Ett exempel som hämtar en slumpmässig aktivitet från Bored API och visar på skärmen

https://github.com/argelius/ec-advanced-js-samples/blob/master/lesson11/hooks-use-effect/src/App.js

Övning - useEffect

1. Skapa en application med en funktionskomponent
2. Använd **useEffect** för att göra ett anrop för att hämta alla länder från det här API:et
   https://restcountries.eu/
3. API:et svarar med en lista av alla världens länder. Rendera en tabell med två kolumner, en för namn och en för huvudstad

Hantera uppdateringar

- I förra övningen hämtade vi bara data när sidan laddades
- Det går att använda **useEffect** för att reagera på uppdateringar
- Till exempel, om vi har en sökruta kan vi göra anrop när innehållet i sökrutan uppdateras
- För att undvika att hamna i en oändlig loop behöver vi använda det andra argumentet till useEffect

```
const [query, updateQuery] = useState('');
const [data, updateData] = useState(null);

useEffect(() => {
  if (query.length > 0) {
    fetchData(query).then(updateData);
  }
}, [query]);
```

Övning - Sökning

1. Utgå från applikationen ni skapade i förra uppgiften
2. Lägg till en textruta och gör det möjligt att söka på länder med namn
   Filtrera inte listan, utan gör ett anrop till API:et
3. Lägg till debounce på textrutan
   Använd gärna https://github.com/xnimorz/use-debounce