React Hooks

Guide för nybörjaren

Hooks är ett fancy namn på funktioner som React gör tillgängliga för utvecklaren. De kallas hooks för att de ”hakar in i” React och dess olika funktioner och finesser. Hooks börjar alltid med ”use”.

useState

Används för att skapa variabler i React.

En sådan här variabel är kopplad till gränssnittet, vilket gör att dess värde alltid reflekteras i där. Dvs om värdet ändras i koden, så syns även förändringen i gränssnittet.

useState tar in en parameter, variabelns initala värde. Den returnerar variabeln samt en funktion för att ändra dess värde. Variabelns värde får endast ändras med hjälp av funktionen som useState returnerar.

```
const [variable, setVariable] = useState(1)
```

Variabler skapade på detta sätt kallas för tillståndsvariabler (state variables).

useEffect

Kör kod varje gång en av givna tillståndsvariabler förändras.

useEffect tar in två parametrar. Den första en funktion, den andra en lista med tillståndsvariabler – även kallad ”the dependency array”. Varje gång någon av de tillståndsvariabler som getts i listan ändras, så kommer den givna funktionen köras.

Här kommer print att köras varje gång variable ändras.

```
function print() {
  console.log('variable uppdaterades!')
}

useEffect(print, [variable])
```

Skulle man skicka med en tom lista så kommer funktionen köras en gång - första gången komponenten renderas.

```
useEffect(print, [])
```

Skulle man utelämna listan av tillståndsvariabler helt, kommer funktionen köras varje gång någon tillståndsvariablel i komponenten ändras.

```
useEffect(print)
```

useEffect är en förkortning av ”side effect”, bieffekt eller biverkning. Alltså att en ändring av en tillståndsvariabel kan ge upphov till någon annan förändring. useEffect används först och främst för att se till att andra system utanför React hålls uppdaterade med det som sker i React.

useContext

För att förstå useContext måste vi först förstå vad Context är.

Context är ett sätt att dela variabler mellan olika komponenter. Man kan likna contexts med scopes i vanlig programmering.

Ett alternativ till att skicka med variabler som props till komponenter . Vill du skapa ex. globala variabler så kan detta göras med context.

Att skapa en Context

createContext är funktionen man använder för att skapa en context. Den returnerar en ny context.

```
const myContext = createContext()
```

I detta objekt så finns det en komponent som heter Provider. Denna komponent visar inget innehåll i gränssnittet, men dess barnkomponenter får tillgång till alla variabler som finns i kontexten. Genom att skicka med en prop 'value' till Provider säger man åt context vilka variabler som ska ingå.

```
<myContext.Provider value={someVal}>
    alla komponenter som renderas här
    får tillgång till myContext.
</myContext.Provider>
```

Att använda en Context

För att få tillgång till variablerna som finns i en kontext behöver vi använda useContext. useContext tar en parameter: din context.

```
const someVal = useContext(myContext)
```

Övriga Hooks

useReducer

useReducer är ett annat sätt att använda tillståndsvariabler. Detta sätt att hantera tillståndsvariabler kallas för "Redux Pattern'. Användbar när man har mer komplexa tillstånd. Används i stället för useState.

useCallback & useMemo

Är båda sätt att begränsa hur ofta en bit kod körs. Används för att optimera prestanda hos ens applikation.

useRef & useImperativeHandle

useRef används för att få en refens till ett speciellt DOM-element. Tänk getElementById. useRef kan användas tillsammans med useImperativeHandle för att skriva mer komplexa och mer modulära komponenter.

useLayoutEffect

Används exakt likadant som useEffect. Skillnaden är att useLayoutEffect är synkron och blockerande. Körs innan alla DOM-element renderas ut. Används när man vill läsa av dimensioner av olika DOM-element.

Egna hooks

En funktion som använder sig av ovanstående hooks med syfte att tillhandahålla någon särskild funktion är en 'Custom Hook'. De brukar läggas i en mapp 'hooks' i mappen 'src' i ditt projekt.

Exempel är hooks som sparar tillståndsvariabler i LocalStorage eller hämtar data från ett api till en tillståndsvariabel.