Björn Edlund UXF22 Chas Academy

TEORIHANDBOK React

1. Ramverket React (en översikt, vad som är unikt för ramverket, fördelar, ev nackdelar, osv)
2. Vad innebär Rendering och Virtual DOM?
3. Vad är JSX? Vad används det till?
4. Vad är ett undantag inom programmering? Eng. Exception
5. Vad innebär autentisering inom webbapplikationer? Vad används det till?

1. Ramverket React är ett populärt JavaScript-bibliotek som används för att bygga användargränssnitt i webbapplikationer. Här är en översikt och några av dess unika egenskaper:

- Komponentbaserat: React fokuserar på att bryta ner användargränssnittet i återanvändbara komponenter. Komponenter är självständiga enheter som kan återanvändas och kombineras för att bygga komplexa gränssnitt. Detta underlättar utveckling, underhåll och testning av applikationer.
- Virtuellt DOM: React använder ett virtuellt DOM-koncept för att effektivt hantera uppdateringar av användargränssnittet. Istället för att direkt manipulera webbläsarens DOM skapar React en virtuell representation av gränssnittet och uppdaterar endast de delar som ändrats. Detta resulterar i snabbare prestanda och bättre användarupplevelse.
- Enkel integration: React kan integreras med andra ramverk och bibliotek, vilket gör det flexibelt och användbart i olika projekt.
- Enkel state-hantering: React erbjuder en smidig hantering av komponenters tillstånd genom konceptet "state". Detta gör det enkelt att hålla reda på och uppdatera data i applikationen.
- Envägsdataflöde: React följer principen om envägsdataflöde, vilket innebär att data flyter nedåt genom komponenterna. Detta gör det lättare att hantera och spåra dataflödet i en applikation.

Fördelar med React är dess effektiva hantering av uppdateringar, dess modularitet och ett stort ekosystem av tillägg och verktyg, liksom en stor community. Nackdelar kan inkludera inlärningskurvan för nybörjare, behovet av att lära sig JSX (JavaScript XML), vissa initiala konfigurationer samt komplexiteten hos större applikationer som kräver ytterligare state-hantering och arkitektur.

2. Rendering är processen att omvandla data och en beskrivning av ett användargränssnitt till en faktisk visning som användaren kan interagera med. I kontexten av React och andra JavaScript-ramverk och bibliotek handlar rendering om att uppdatera webbläsarens DOM baserat på förändringar i applikationens tillstånd eller data.

Virtual DOM (Virtual Document Object Model) är en teknik som används av React för att optimera renderingen. Istället för att direkt manipulera webbläsarens DOM skapar React en lättrörlig kopia av DOM:en som kallas för det virtuella DOM:en. Vid varje uppdatering jämför React den virtuella DOM:en med den faktiska DOM:en och uppdaterar bara de delar av DOM:en som har ändrats. Detta minskar antalet fullständiga ändringar av DOM:en och resulterar i bättre prestanda.

3. JSX står för "JavaScript XML" och är en utvidgning av syntaxen för JavaScript som används i React. Det tillåter utvecklare att skriva HTML-liknande strukturer och komponenter direkt i JavaScript-koden. JSX gör det möjligt att blanda JavaScript-logik med beskrivningar, vilket gör det enklare att skapa och hantera komponenter i React.

Exempel på JSX-kod (med Tailwind css) i React:

```
<button
        type="button"
        onClick={handleThemeSwitch}
        className="fixed z-10 right-5 top-4 bg-sky-700 dark:bg-rose-200
      text-lg p-1 rounded-md">
        {theme === "dark" ? sun : moon}
 </button>
```

4. Inom programmering refererar "undantag" till fel eller oönskade händelser som inträffar under körningen av ett program. När ett undantag uppstår avbryts den

normala programkörningen och kontrollen överförs till en undantagshantering. Genom att använda undantagshantering kan programmet fånga och hantera undantag genom att utföra specifika åtgärder, som att skriva ut felmeddelanden eller vidta nödvändiga åtgärder för att återhämta sig från felet.

5. Autentisering inom webbapplikationer handlar om att verifiera och bekräfta identiteten hos en användare eller en enhet som försöker få åtkomst till skyddade resurser eller funktioner. Autentisering används för att säkerställa att endast auktoriserade användare kan få åtkomst till vissa delar av en applikation.

Vanligtvis involverar autentisering att användare uppger sina identifieringsuppgifter, såsom användarnamn och lösenord. Dessa uppgifter kontrolleras mot en autentiseringskälla, som en databas eller en autentiseringstjänst. Om användarens uppgifter matchar och verifieras anses autentiseringen vara framgångsrik, och användaren tillåts att använda de skyddade funktionerna.

Autentisering kan också implementeras med hjälp av andra metoder, till exempel tokens eller certifikat, beroende på applikationens behov och säkerhetskrav. Syftet med autentisering är att skydda användarens data och applikationens resurser från obehörig åtkomst.

****

I denna del ska du redogöra hur punkterna ovan (främst 1-3) hänger ihop med din portfölj. När du renderar en av dina komponenter, vart hamnar den först? I DOM eller i Virtual DOM? När du skriver en return inuti en komponent, vad kallas det som finns inuti returnen?

Jag har använt mig av React för att bygga min personliga portfolio, eller åtminstone en första enkel version av den. I React skriver jag kod i JSX och placerar den inuti `return`-satsen i komponenterna. Vid första renderingen skapar React en virtuell DOM-struktur. Framöver jämförs denna virtuella DOM med den faktiska DOM:en. Om jag ändrar i koden eller lägger till ny, så kommer React att uppdatera den virtuella DOM:en, för just det kodavsnittet. Den virtuella

DOM:en jämförs med webbläsarens DOM, och endast de förändrade delarna måste renderas.
Exempel på Hooks jag har använt är useEffect. Den används för att utföra sidoeffekter i en komponent, till exempel att prenumerera på händelser, hantera DOM-manipulation eller utföra asynkrona anrop.
I portfolion har jag använt useEffect för att kontrollera om webbläsaren har den önskade färgschemat "dark" eller "light" baserat på användarens systeminställningar. När komponenten läses in körs effektens funktion (callback-funktionen) en gång. Om användarens färgschema är "dark", uppdateras en tillståndsvariabel (theme) till "dark" genom att anropa setTheme("dark"). Annars sätts theme till "light" med setTheme("light").
För att indikera att effekten endast bör köras en gång vid inläsning, har en tom array [] angivits som det andra argumentet till useEffect. Om arrayen innehåller variabler kommer effekten att köras varje gång någon av dessa variabler ändras.

En knapp tillåter användaren att manuellt skifta mellan “dark” och “light” färgsschema. Utssendet styrs genom Tailwind och klasser som exvis `“dark:text-stone-300”`

Utöver React har jag använt mig av React-biblioteket “Styled-components”, och för css har jag använt mig av “Tailwind”.
Med Styled components kan css skrivas direkt in i React-komponenten och behöver inte hänvisa till en extern css-fil, vilket förbättrar överblicken och underlättar återanvändning av en komponent.
Tailwind har blivit enormt populärt på senare tid och var intressant att testa.
Just Tailwind har till viss del känts kontraintuitivt att använda eftersom det påminner om HTML-inline styling och kräver upprepning i flera olika element, lite tvärtemot vad CSS egentligen går ut på. Men fördelarna är ändå många, det går otroligt fort att utveckla, css-stilarna är lättbegripliga och det syns direkt i koden hur elementet kommer att renderas. Ett exempel:

```
<div className="flex justify-center items-center">
```

Det är inte helt lätt att kombinera Tailwind med Styled Components rakt av. Vanligtvis använder sig styled components av vanlig css i stilanvisningarna, men

om styled components ska användas i samma projekt som Tailwind är vore det ju bättre att använda Tailwind rakt igenom. Det har jag inte implementerat ännu, men blir nästa steg i portfolion. Exempel på detta: https://www.freecodecamp.org/news/how-to-style-your-react-apps-with-less-code-using-tailwind-css-and-styled-components/

I React kallas det som finns inuti `return`-satsen för "renderingsinnehållet" eller "renderingskoden" i en komponent. Det är den del av koden som specificerar vad som ska visas och renderas i användargränssnittet när komponenten används. Det kan vara en kombination av HTML, JSX och React-komponenter som beskriver strukturen och utseendet på det som visas för användaren.