TEORIHANDBOK REACT

DEL 1

1. Ramverket React (en översikt, vad som är unikt för ramverket, fördelar, ev nackdelar, osv)

React är ett komponentbaserat Javascript-bibliotek/ramverk för att bygga gränssnitt och är det nu mest använda ramverket. Det utvecklades från början av Facebook, och har ett stort community som man kan kommunicera med och få hjälp av.

Det finns många fördelar med att använda just React.

Då det är komponentbaserat går det enkelt att översätta design till kod, och skapa fristående komponenter som går att återanvändas i byggandet. React har dessutom redan existerande komponenter att importera och använda vilket ger stor tidsbesparing och underlättar själva programmerandet.

Programmeringsflödet går i en riktning från de överordnade komponenterna till de underordnade och detta ger kod som är mer förutsägbar, och man kan därför lättare undvika buggar som annars uppstår i en mer komplex kod. Ändringar som görs i de underordnade komponenterna kommer inte ändra den överordnade.

React har en modulär struktur som gör det väldigt flexibelt, och lättare att underhålla koden. Man kan ändra i enskilda delar utan att påverka hela logiken. Detta gör också att man lättare kan testa komponenter och felsöka, och vid uppdatering av en komponent sker samma ändring på alla platser där den är placerad.

En annan fördel är hur den samarbetar med Virtual DOM, vilket kommer beskrivas närmare nedan.

Självklart finns det också nackdelar med React, som med alla bibliotek och ramverk.

Det finns exempelvis inget stöd för back-end i React, utan det går bara att jobba med det som sker direkt i webbläsaren, d.v.s. front-end. Detta gör att man måste lägga till fler verktyg för att täcka upp för back-end och datalagring.

2. Vad innebär Rendering och Virtual DOM?

En av de unika funktionerna i React är dess virtuella DOM (Document Object Model) och renderingen. React bygger en virtuell representation av den faktiska DOM och publicerar sedan till webbläsaren. Detta gör att utvecklare effektivt kan uppdatera och skapa komponenter utan att direkt manipulera den verkliga DOM, och man kan testa ändringar först för att se hur bra de fungerar. I renderingen uppdateras bara den ändrade delen istället för hela projektet. Det här ger bättre prestanda och möjliggör snabbare uppdateringar av användargränssnittet.

3. Vad är JSX? Vad används det till?

JSX (JavaScript XML) och det gör det möjligt att skriva HTML-syntax direkt i JavaScript-koden, och konverterar detta sedan till JS-element. Detta förenklar

skapandet av UI-komponenterna och visualiseringen av själva användargränssnittet.

4. Vad är ett undantag inom programmering?

Komponenter i React kan utlösa s.k. undantag, även kallade fel, som kan fångas upp och hanteras för att förhindra att programmet kraschar. Det kan vara ett fel när gränssnittet ska visas som t.ex. ogiltiga data, eller något som stör flödet eller när ett data från ett API ska hämtas. Då kan program byggda i React smidigt återhämta sig och visa informativa felmeddelanden och fortsätta fungera trots felet.

5. Vad innebär autentisering inom webbapplikationer? Vad används det till?

Autentisering handlar om att verifiera identiteten för en användare eller en enhet. Det är en säkerhetsåtgärd som förhindrar obehörig åtkomst, skyddar data och skapar ett förtroende mellan användaren och webbapplikationen.

Det vanligaste sättet att autentisera är med ett användarnamn och lösenord. Det kan även utföras med inloggning genom sociala medie-konton, tvåfaktorautentisering eller smartcard, fingeravtryck osv.

DEL 2

Sidan jag byggt består av många komponenter, med modulerna som i en trädstruktur.

Till exempel är korten i portfolion byggd utifrån ett ursprungs-kort, med dess olika delar i en map-funktion. Där ligger en Json-lista med en rubrik, en text och en bild som hämtas in från assets och distribueras i en array. Dessutom finns en knapp som är byggd separat och länken kopplad till respektive kort för knappen är inhämtad via Router/switch till ytterligare sidor.

Dessa delar läggs in löpande för varje kort som props och bygger på det sättet upp innehållet på sidan.

```
import Card from '../Card'


const Skills = () => {
    const skillsData = [
        {
            id: 1,
            cardPic: amecThumb,
            headline: "AMEC Awards material",
            description: "All the graphic material for the intern
            buttonLink: "/amec"
        },
        {
            id: 2,
            cardPic: posterThumb,
            headline: "Library posters",
            description: "Mediearkivets posters to hang in public
            buttonLink: "/posters"
        },
        {
            id: 3,
            cardPic: officeThumb,
            headline: "Custom Office icons",
```

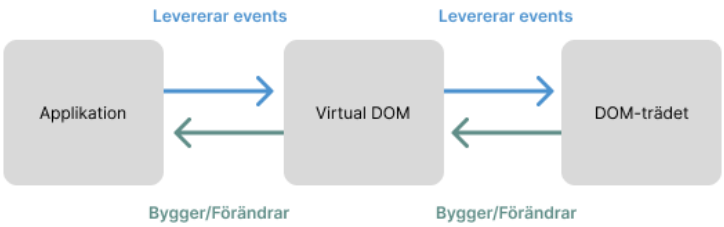
Json-listan över innehållet på respektive kort

```
function Card(props) {

    return (
        <CardContainer>

            <StyledCardPic src={props.cardPic} alt="Project thumbnail" />
            <CardHeader>{props.headline}</CardHeader>
            <CardBody>{props.description}</CardBody>

            <Cardbutton link={props.buttonLink}></Cardbutton>

        </CardContainer>
    );
}
export default Card;
```

Kortet som är en funktion och itererar innehållet på sidan

Footern, headern och menyn är andra exempel på komponenter, oberoende och återanvändbara.

När komponenterna renderas hamnar de först i Virtual DOM för att sedan skickas vidare till DOM.



Det som ligger inuti returnen kallas React-element som sedan konverteras till HTML-element.

I projektet finns många exempel på hur man kan skriva HTML-syntax såsom div, footer, header, p-taggar och H-taggar osv. Detta renderas sedan till JavaScript-element och placeras i DOM.