# Teorihandboken

Joanna Olofsson, UXF

## 1. Ramverket React

React är ett bibliotek för user interfaces. I React, kan vi bygga komponenter, som vi sedan kan återanvända och sätta ihop till hela webbplatser. Till vår hjälp har vi olika ramverk och bibliotek som vi kan installera. Efter installationen, så importerar vi det som vi behöver i projektet.

En av fördelarna med React är att vi kan göra förändringar i komponenterna och direkt se att webbplatsen uppdateras. En annan att koden blir väldigt tydlig, då datan går från “parent” till "child"-komponenter. Det gör det enkelt för flera personer att arbeta med samma projekt eller att leta efter fel i koden.

Jag tycker att nackdelarna med React är få. En är dock att det kan vara tidskrävande att påbörja ett nytt React projekt. Detta på grund av installationer, behov av att ibland radera filer eller kod och att sätta upp en router för att kunna navigera på sidan. En annan är att det tar lite tid att lära sig React och att förstå idéen med att importera och exportera komponenter inom projektet. Jag har även läst att det kan vara en nackdel att DOMen inte uppdateras direkt, när man skall hämta vissa API’er.

### 2. Vad innebär Rendering och Virtual DOM?

När vi går in på en webbplats och begär att få se webbplatsen, skickas ett html-sida från servern.

Genom att exempelvis använda React och style-components kan vi sedan förändra hur utseendet ser ut i browsern. Det vill säga hur browsern uppfattar ditt aktuella “state” av din applikation.

Det är genom rendering, som React beskriver ett UI utifrån applikationens “state” och prop. React renderas när programmet startar och när det sker förändringar av ett “state”.

När vi kodar i React, förändras utseendet i browsern. Förändringar i en komponents “state” innebär att alla komponenter som påverkas av denna komponent behöver renderas igen. När detta sker så skapas en ny virtuell DOM, som visar vilka komponenter som behöver uppdateras i “Real DOM”.

**3. Vad är JSX? Vad används det till?**

JSX står för Javascript XML. Det är en extension av Javascript. I React kan vi använda den för att skriva kod som liknar html. Den kan också användas blandat med javascript eller till och med insprängt i själva javascript koden. Ibland även tvärtom, när vi skapar dynamiska delar i koden.

För den som använt html och även css kan det vara enklare att förstå JSX än javascript, vilket jag 100% kan hålla med om.

Det är under renderingen som JSX konverteras till DOM elements och blir till en representation av hur html strukturen skall se ut. Detta gör det enklare att hitta fel, genom att titta på inspektorn och med dess hjälp få indikationer om hur man skall fortsätta koda.

**Vad är ett undantag inom programmering?**

Det innebär att programmet inte körs som programmeraren förväntat sig. Dessa fel är sådana som kan uppstå på grund av att vi försöker få programmet att hantera sådant som det inte kan lösa. Ett exempel är exempelvis att dela ett tal med 0. Det kan vara semantiska fel, i koden, som såsom stavfel. Det kan också vara logiska fel, som kan uppstå när utvecklaren tänkt fel.

Du kan själv skapa undantag för att pröva om koden, som du skrivit fungerar och därmed få en indikation om vad som gått fel. Det är olika mellan olika programmeringsspråk hur man gör men man kan exempelvis använda någon

specifik funktion eller vissa ord, såsom try and catch. Där berättar vi för programmet vilket exception som du vill “catch” och om en viss typ dyker upp i try blocket, så blir koden verkställd. Genom detta kan vi få en indikation om vilket fel du gjort eller få annan information som hjälper dig att komma vidare i din kodning.

Genom exceptions så kan man lättare hantera fel av olika slag som påverkar hur programmet flyter och blir mer tillförlitligt.

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

Det innebär att exempelvis jag behöver kunna visa på att jag är jag, när jag använder en app. Det kan exempelvis vara när jag för första gången besöker en sida. Oftast behöver man direkt logga in, även om jag skulle vilja ha möjligheten att kolla runt lite först.

Kanske är det på grund av detta, som det oftast finns ett erbjudande om att få prova appen i x antal dagar, innan man köper. På så sätt behöver man fylla i sina personuppgifter.

Ett annat exempel på autentisering kan vara att visa upp något som vi har. Kanske en biljett av något slag. Ytterligare ett exempel är att visa upp något du är. Det kan vara något som har med ansiktsigenkänning eller fingeravtryck att göra. Ibland använder man två av dessa typer, exempelvis lösenord och ansiktsigenkänning, och då kallas det två-faktorlösning.

Ett vanligt authentication innebär ofta att användaren får identifiera sig med exempelvis username och password på någon form av login-sida. Informationen skickas när användaren loggar in. Där valideras informationen ofta mot information i en databas. Om informationen stämmer så blir användaren autentiserad.

Om autentiseringen fungerar, så skapar applikationen en session för användaren. Detta innebär att ge denna ett session ID. Applikationen sparar detta i minnet eller i en databas. Syftet är att identifiera och tracka användarens interaktion med applikationen.

När authentication gått igenom kan utvecklaren lägga in olika regler för vad just denna specifika användare får göra eller inte. Det kan handla om att en användare

får access till en viss sida men inte till en annan, efter behörighetsgrad eller vilken avdelning personen arbetar på

# Sammanfattning

Det var verkligen så mycket enklare att använda React än det var med Javascript. Detta särskilt med användandet av style-components istället för css. Jag upplevde att allt hängde ihop på ett enklare och mer logiskt sätt.

En av fördelarna med React tycker jag att det går att se en tydlig struktur. Det är också betydligt roligare än javascript, då man precis som med css ser snabba resultat.

En nackdel var att jag upplevde att det tog lite tid att förstå att det inte är en katastrof om fel uppstår. Ibland är det bara så att jag glömt att importera en komponent i app eller att jag glömt att lägga upp sidan på router. Även om mitt projekt inte är så stort så tycker jag att det lätt kan bli lite rörigt med att hitta olika delar av koden, när jag behöver fixa till något. Jag inser att det är viktigt att vara noggrann och konsekvent i koden när man arbetar tillsammans med andra och projekten blir riktigt stora.

Det jag skriver i return delen är JSX och hamnar först inuti virtual dom. Det jag skriver inuti returen, kallas för return method och är data från komponenterna, som liknar html men som är Javascripts motsvarighet av den struktur som finns i komponenten. Denna metod används bara när du vill returnera ett värde från en komponent, som jag återanvänder.

På några ställen har jag använt javascript för att alternativa de olika staten. Vid dessa exempel är det enklare att förstå hur virtuell dom och real dom fungerar. Jag förstår också att det är en viss skillnad mellan render method och return method. Att det senare är det jag ser i min main.js och som sker direkt när jag startar mitt applikation, samt när jag skapar nya “children”. Medan “return” sker konstant när jag gör justeringar i komponenterna.

Jag valde att använda Vite och nästan alla mina filer är JSX filer. För att se till att vissa delar av min kod inte är statisk så har jag blandat JSX med javascript på några

få ställen. Jag gjorde några tutorials med bara JS- filer men tyckte att startsträckan var längre på dessa, då jag var tvungen att radera fler filer och strukturera om. Jag tycker också att det är betydligt enklare att få en överblick över koden i de JSX filer, som jag har sett.

Jag har använt mig lite av Javascript i koden men kämpade lite med att få ett grepp om hur jag skulle få det att fungera med jsx, till exempel när jag skulle göra min hamburgermeny. Jag tror nog att jag hade kunnat använda mig av ternary operator för att öppna och stänga menyn men fick inte riktigt till det med att kunna få menyn att stänga/collapse när jag klickade på länkarna, som tog mig till en annan sida. Så jag gjorde om den till en “toggle grej”.  Så utmaningen för mig är att bli bättre på Javascript, så att jag kan använda React lite mer och framför allt att inte glömma bort det jag lärt mig under detta första år.