Josef Forkman
SUT24

Vad är HTML och vad används det till? Beskriv också kort historiken för HTML.

HTML står för Hypertext Markup Language. HTML är ett Märkspråk som innebär att HTML inte står för någon logik eller design. Utan den tar hand om strukturen och innehållet på en hemsida. Innehållet är olika element som kan vara allt emellan en rubrik (h1-h6) till ett formulär med alla sina inputs, textarea, select och label. Varje element har en start tag och en stängnings tagg men vissa är själv stängande så som img taggen. Många element har en betydelse inte bara för att få en bra struktur på sidan utan att har en förklarande mening vad det är får något. Detta är extra viktigt för människor som har det svårt att ta till sig innehållet på sidan till exempel om man har en nedsatt syn med mera. När elementen förklarar innehållet brukar man säga att elementen är semantiskt korrekt. Men när detta inte räcker till så kan man lägga till extra attributet på elementen. Det kan vara olika ARIA attributs som hjälper att beskriva vad som finns på sidan. Varje element kan man lägga till class och id attribut. Syftet med det är för andra språk ska kunna komma åt det och modifiera olika saker. Dessa språk är antingen JavaScript eller CSS.

Men HTML kan inte bara användas till hemsidor utan också till spelutveckling, olika mobil och desktopapplikationer med mera. För att kunna göra mobil och desktopapplikationer behöver man oftast en typ en motor för att kunna köra HTML i deras miljö. Denna motor kan antingen vara Electron eller Tauri för att nämna någon.

Historian om hur HTML startade var att fysikern Tim Berners-Lee tyckte att det var svårt att kunna dela forsknings papper mellan kollegor. Tim Berners-Lee var fysikern på CERN. Detta gjorde att han ville komma på en standard för att kunna kommunicera med varandra. Han gjorde ett försök med hjälp av en annan forskare Robert Cailliau för att komma på denna kommutation verktyget. År 1993 fick vi reda på att han hålade på att jobba på HTML. Året efter kom HTML+ som la till tabeller och formulär. HTML 2.0 satte dom sig ner och skrev HTML-specifikationen som skulle vara en riktlinje för kommande versioner. När HTML 4.0 och HTML 4.01 år 1998 så hände det inte så mycket mer än att man började ta bort några gamla ”Netscape's visual markup”. Efter några år med HTML 4.01 så började man fundera på ny version som vi fortfarande är på nu. De började jobba på HTML 5 och hur den skulle utformas redan år 2004 men lanserades först år 2014. Varför det hade tagit så långt var att man ville få webben mer tillgänglig för allmänheten och följa W3C rekommendation. Man gjorde några nya html element som faktiskt betyder något mer än vad div gör. Några saker man introducerade var article, header och footer. Men sen så gjorde man så man kunde lägga till video och ljud på sidan utan att använda Silverlight och flash. I HTML 3 så tog W3C över standardiseringen efter IETF hade stängt arbetsgruppen. W3C gjorde det till 2019 då WHATWG tog över standardiseringen och är de som gör det än idag.

Vad är CSS och vad används det till? Beskriv också kort historiken för CSS.

CSS står för Cascading Style Sheets och används primärt för att ändra designen på hemsidor. Man använder CSS för att kunna presentera en snygg hemsida på olika enheter och utskrifter. Något som är bra med att man har separerat CSS och HTML är att man inte nödvändigt viss behöver ha all CSS på varje sida om inte allt används. Detta leder till att man förbättrar laddningstiden för sidan ska bli färdigt laddad.

Med CSS kan man med en eller flera regler bestämma hur HTML element ska se ut och hur det ska vara placerade relaterade till varandra. CSS reglerna läser uppifrån och ner. Om det är något annat längre ner som ändrar utseendet på samma sak så kommer det att skriva över. Detta sker bara om det som är längre ner är antingen lika specifik eller mer specifik. Vad som avgör om något är mer specifikt än något annat avgörs med några färdig bestämda regler på hur webbläsaren ska kunna hantera det.

Det var Håkon Wium Lie och Bert Bos som jobbade på CERN som forskare. De jobbade tajt med Tim Berners-Lee som var den som skapade HTML. Efter som tanken med HTML i början var bara att kunna visa olika forsknings papper med minimalt med design. Så var det inga problem med att göra all design direkt i HTML med specifika HTML element som till exempel <font> och <u>. Håkan och Bert började redan två år efter HTML fundera på CSS Level 1 år1994. De fundera på hur vi skulle kunna förbättra upplevelsen och ta bort röran som blir i de mer avancerade layouterna. Första versionen av CSS kom ut 1996 med textfärger, typsnittets, bakgrund, margin, padding och boxmodellen. Med CSS Level 2 kom bland annat olika set att kunna positionera HTML element och göra HTML fungera för olika skärmstorlekar. Vi fick position och flote för att enklare kunna placera de olika elementen. Sen fick vi media queries för att kunna jobba med olika skärmstorlekar. Sen 2011 kom CSS Level 2.1 som var mer korrigera de fel och oklarheter som fanns i de tidigare versionerna. Men med CSS Level 2.1 fick de officiell W3C-rekommendation. Sen kommer vi till CSS Level 3 som vi är på nu. Det största som hände med denna version är att det blev ett nytt sätt som de släpper nya funktionalitet. Skillnaden blev att de delade upp sig i mindre grupper som fokuserar bara på vissa områden i CSS som till exempel selektorer, Boxmodellen och textfärger. CSS Level 3 har funnits från 1998. Nu håller de på att fundera på vad som ska ingå i CSS Level 4. Något som de har kommit fram till än så länge är nya färgscheman som är P3, Oklab och CIELAB. Dessa nya färgscheman ska kunna ge både en större färg rymd och mer livfulla färger. Dessa färger kräver att man har en skärm som klarar av dessa nya färgscheman.

Förklara vad responsiv design innebär.

Responsiv design innebär att oavsett vilken skärm / webbfönster storlek är så ska innehållet anpassas sig efter utrymmet det får. Detta kan man göra på lite olika sätt. Det kan antingen vara att man använder relativa värden, låta CSS komma fram hur saker sak placeras / ändras eller använda media queries. Exempel på saker man kan ändra på är typsnittets storlek eller att man ändra hur HTML element är placerade.

Relativa värden är värden som ändras beroende på ett vist villkor som uppnås. Ett exempel på det är ”viewport” enheten som har en för bredden och en för höjden. För att använda det så skriver man vw för ”viewport” brädd och vh för ”viewport” höjd. Dessa är en procent på bredden och höjden på webbläsarens fönster. Detta är extra bra om man vill att något måste vara relativt till webbläsarens fönster men det är något man ska vara försiktig med. Ett annat exempel är ch som får bredden på typsnittets 0 (noll). Ch är riktigt bra om man vill begränsa texten på sidan att alltid vara 65ch. Detta gör så att läsbarheten på sidan ökar. Denna brädd kommer att ändras baserat på vilken typsnittets storlek man använder sig av. Ett exempel på att göra så att CSS komma fram hur saker sak placeras / ändras är att använda flex. Flex har en bra sak man kan para ihop som är flex-wrap: wrap;. Detta gör så att så fort html elementen går utanför boxen den lever i går till nästa rad/kolumn för att kunna få plats. Ett annat sätt att få det responsiv är att använda media queries. Media queries funkar så att man bestämmer en brytpunkt då skärmens brädd så ska man ändra på vissa CSS regler. Detta för att kunna göra så att det ser bra igen. Ett tips då är att man antingen jobbar sig upp eller ner ifrån en stor skärmbrädd. Det brukar inte funka så bra att om man hoppar fram och tillbaka.

Vad är JavaScript och vad används det till?

JavaScript eller js för korthetens skull är ett ”scripting language”. JavaScript är Single Threadeded språk som bara kan göra en sak åt gången. Då kan man undra hur funkar promes i JavaScript. Det funkar så att JavaScript skickar arbetas uppgiften till en webb API som tar hand om det i bakgrunden. JavaScript är JIT (Just-In-Time Compilation) som innebär att man inte behöver kompilera till maskinkod innan man kan använda det. Detta gör så att man kan få fel meddelandet direkt till exempel i webbläsaren. Något som behövs tas upp är att JavaScript och Java inte är samma sak. Detta är något som många som inte har holt på med programmering tror att det är samma sak bara för att JavaScript har Java i sig. JavaScript är först och främst ett functional programming språk men kan skrivas i OOP (Object-oriented programming).

JavaScript är det vanligaste och mest använda språket som programmerare använder enligt stack overflow 2024 survey. Detta är kanske bara för att JavaScript är ett språk som man kan göra det mesta. Några få saker man kan göra utöver att göra hemsidor är Node.js, Apache CoushDB och Adobe Acrobat. Bara med Node.js kan man använda det till många olika saker så som att göra API och använda olika webbramverk.

Vad är ECMA-script och hur hänger det ihop med JavaScript?

ECMA-script är en organisation som sätter en standard på olika programmeringsspråk. Där den mest populäraste som de har satt standard på är JavaScript. Detta är för att tillexempel olika webbläsare ska kunna veta hur man ska kunna tolka JavaScript. JavaScript följer allt som ECMA-script har att erbjuda för dom. Men de har också lagt till massa ut över detta. ECMA-script har från början varit för front-end men har blivit mer och mer en full-stack standardisering när man införde Node, Bun och Dino.js.

Förklara översiktligt vilket ansvar HTML, CSS och JavaScript har i teknikstacken inom frontend på webben.

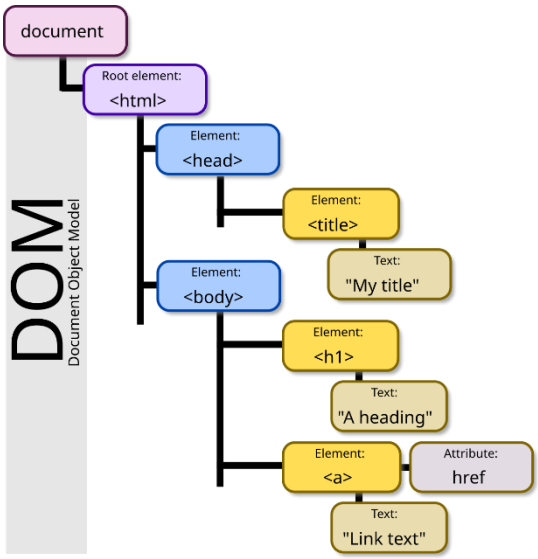
HTML har uppgiften att ta hand om allt innehåll på sidan så som text, bilder och videos. Men också ta hand om tillgängligheten på sidan.

CSS är det som ändrar utseendet på html så att det blir lite roligare att kunna ta in informationen som html har att ge. Men den har också uppgiften att göra hemsidan mer tillgängligt. Detta gör CSS med till exempel att använda en läs bar font och sätta fonten i rätt storlek med mera.

JavaScript har uppgiften att göra hemsidan mer interaktiv. Det jag menar med det är att när man trycker på en knapp så kommer det hända något på sidan. CSS och HTML är begränsade på hur mycket de kan göra för att göra det mer interaktiv. Något annat som man oftast gör med JavaScript är att kunna hämta dynamiska data som sedan ska läggas in på sidan.

Beskriv vad DOM är och hur vi använder det när vi skapar en hemsida.

DOM står för Dokument Object Module. DOM är ett object som är en representation av hemsidan. DOM är strukturerad i ett träd liknande format som man kan se på bilden nedan för. Ett ställe man kan se detta är i att inspektera/F12 på webbläsaren för att kunna se trädstrukturen.



DOM har alla egenskaper och webb API:er som krävs för att rendera html på sidan. Den ger också oss en bra interface / ett sett att prata med hemsidan. Detta kan ske antingen med JavaScript eller HTML för att kunna modifiera hur olika element pratar med varandra.

Hur vi använder DOM är så fort vi skriver något i HTML där vi lägger saker inuti varandra som till exempel lägga en h1 inuti en main element. När man vill göra något mer dynamiskt så kan man ta hjälp av JavaScript. Med JavaScript kan man välja ett element med dockument.getElementById(”myElement”) som gör att du kan modifiera innehållet för detta specifika element. Om vi inte hade DOM så skulle vi inte ha denna hjälp av denna API för att modifiera eller lägga till HTML element.

Vad menas med ett JavaScript-ramverk och vad tillför det till ett projekt jämfört med att bara använda ren JavaScript?

Vad ett JavaScript-ramverk är en samling av metoder och andra funktionaliteter som man ofta använder i ett projekt. Det kan var att göra det enklare att uppdatera en specifik information på sidan. Det skulle också kunna vara att få hjälpa med att hämta data från externa källor. Dessa samlingar av metoder och andra funktionaliteter brukar samlas i olika npm paket som man kan använda. Detta kan kännas irriterande att ladda ner samma paket

varje gång och sätta upp en viss mappstruktur varje gång man ska starta ett nytt projekt. Då kommer det oftast ett hjälpkommando från de olika JavaScript-ramverken som gör allt detta. Ett populärt verktyg är att använda Vite för att kunna sätta upp olika populära JavaScript-ramverk. Detta gör man med att skriva npm create vite@latest och följer instruktionerna som den ger. Detta kommer att sätta upp mapp strukturen för det valda JavaScript-ramverket med alla npm paketen för att köra det. Sen är det bara för oss att antingen sätta i gång att programmera eller ladda ner andra npm paket. Något som JavaScript-ramverk kan hjälpa till med är att göra en komplex html kod med tillhörande JavaScript till små hanterbara komponenter. Detta för när hemsidan växer och kanske har jätte många under sidor så blir det svårare att hålla kål på hur saker och ting fungerar till slut. När man använder sig av ett JavaScript-ramverk så brukar det uppmuntras till att hålla en viss kodstruktur på hur man skriver sin HTML, CSS och JavaScript. Detta hjälper att andra som är ska kolla på projektet att snabbare kunna förstå projektets innehåll.

Många anledningar varför det är bra att välja ett JavaScript-ramverk kommer till att man vill spara tid och ibland lite frustation. Många anledningar till man ska välja JavaScript-ramverk över ren JavaScript har jag redan nämnt i vad ett JavaScript-ramverk är för något. Många JavaScript-ramverk kommer med två alltnativ en ren odlad front-end och en full-stack lösning. Oftast vill man använda full-stack lösningen även om man har en redan befintlig back-end. Detta för att man då kan göra en mer säkrare koppling med SSR (Server-side Rendering) eller SSG (Static Site Generation). Den icke full-stack lösningen brukar rikta sig mer till personer som gör dessa full-stack lösningen och inte vanliga konsumenter. Nu när vi vet att man ska använda ett full-stack lösning så öppnar det lite dörrar. Den första är att göra CRUD på en databas med sin favorit ORM. De två mest populäraste ORM är Drizzle och Prisma. Detta gör det enkelt att hantera back-end saker i ett projekt utan att hoppa mellan olika front-end och back-end och se till att bägge är synkade med varandra. Förmodligen när man använder ett JavaScript-ramverk så kommer du att använda npm. Som ger en oändligt med möjligheter för antagligen finns det ett npm packet som löser ett problem man har. Det ger att man inte behöver skriva samma sak flera gången. Som i sin tur leder till min sista anledning varför man ska välja JavaScript-ramverk i stället för ren JavaScript. Det är att då kan man hålla sin kod mer DRY.

Lista tre vanligt förekommande JavaScript-ramverk. För varje ramverk ska du sedan kort beskriva det. Efter din lista med de tre ramverken ska du översiktligt förklara vad som skiljer dessa åt.

- **React:** använder Vite för att skapa projektet. Alla komponenter är skriven i en JSX fil eller TSX. Skillnaden mellan JSX och TSX är om man vill använda TypeScript eller inte. För att skapa en komponent så skapar man funktion som returnerar HTML liknande kod. I funktionen är där man gör all logik för att rendera med data som är tillhörande. En CSS fil kan bli importerade i varje komponent eller ha en mer global CSS i APP.jsx / APP.tsx filen. Sen kan man använda sin komponent i src/App.JSX eller src/App.TSX med att lägga till en HTML liknande element.
- **Vue:** använder antingen Vite eller sin egen npm kommando för att skapa projektet. Alla komponenter är skriven i en vue fil. Man kan skriva både JavaScript och TypeScript beroende på vilka inställningar man gör i vue komponenten. I en vue komponent består av en Script tag, en template tag för HTML och style tag. Sen kan man använda sin komponent i src/APP.vue med att lägga till en HTML liknande element.
- **Angular:** använder antingen Vite eller sin egen npm kommando för att skapa projektet. För att skapa en komponent använder man Angulars egna CLI kommando. Kommandot är ng generate component <name>. Kommandot kommer att skapa en mapp med myComponent.component.ts, myComponent.component.html, myComponent.component.css och myComponent.component.spec.ts. De viktigaste filerna är de tre första för att de presenterar TypeScript, HTML och CSS. I TS filen är knutpunkten för att lägg till CSS och HTML. Angular använder sig av class för all TS logik och en @Component för att sätta upp CSS selector, html och CSS. När man är färdig med komponenten kan man använda vad man har skrivit i CSS selector som en HTML tag där man vill använda den komponenten.

Skillnaden mellan de tre mest vanligaste JavaScript-ramverken är först och främst hur en komponent ser ut. React som vill ha allt på samma ställe, Vue som vill ha lite mer separation i en fil och Angular som vill separera ut det i flera små filer. Sen är det en skillnad på Angular och de andra, som är att Angular använder sig av class. Angular kan man bara skriva i TypeScript medan React och Vue kan man välja om man vill göra det. Så slutsatsen är att Angular är mer strikt på hur saker och ting ska bli gjort medan React och Vue är mer förlåtande.

Vad är tillgänglighetsanpassning av webbplatser och varför är det viktigt?

Att göra en hemsida tillgänglighetsanpassad innebär att oavsett eventuella funktionsvariationer så ska man kunna ta till sig innehållet på sidan. För varje person som besöker sidan funkar lite annorlunda hur de vill ta till sig informationen på sidan. Vissa läser sidan uppifrån och ner medan andra förlitar sig på att kunna tabba sig igenom med mera. Det en uppsjö av saker man ska göra för att få sin hemsida tillgänglighetsanpassad men här är några få av de. Det första är att text ska ha tillräckligt med kontrast till bakgrunden så att man

kan se vad som står. Ett sätt att kunna kolla det är med WCAGs standard där man antingen ska följa AA eller AAA. Vilken riktlinje beror på hur stor taxen är och fetstil det är. Text kan inte var får liten eller får stor så att man inte kan se texten. Sen ska användaren ha möjlighet att kunna ändra textstorlek. Bilder som tillför något på sidan behöver ha en allt text eller lämnas tom. Något annat som utmärker en bra tillgänglighetsanpassad hemsida är om innehållet följer semantisk elements och har en bra ordning på h1-h6. Detta för att om man ska få till exempel sidan uppläst så ska man kunna veta vilka saker som hör ihop. Om man har animationer på sidan så ska det vara möjligt att stänga av det. Detta kan man göra med att i sitt operativsystem ställa in ”minskad rörelse” för att ta bort onödiga rörelser.

Men varför ska man göra hemsidan tillgänglighetsanpassad? Först och främst är för att man ska kunna nå ut till så många som möjligt. Det andra är att det kan förbättra sin SEO (sökmotoroptimering). Det gör så man kommer längre upp på listan när någon söker på sidan. Det är extra viktigt för offentliga aktörers webbplatser eftersom både WCAG 2.1 och att det ska bli en europeisk standard 28 juni 2025. Exempel på ställen som det gäller för är bland annat Skolor, sjukvård och kommuner. Man började redan förbereda detta år 2019.

Vad är ett webb-API och vad används det till?

Webb-API:er är att kunna prata med enkla kod till något komplicerat i webbläsaren. Ett exempel skulle vara om vi inte hade Canvas web-API då skulle det innebära att man skulle behöva prata direkt till GPU med mera för att göra det man vill. Webb-API:er används till allt man gör i webbläsaren för att antingen ta input från användaren eller om man vill ändra / läsa något från webbläsaren.

Förklara REST och redogör för dess huvudprinciper.

Varför vi har REST är för att man ska kunna göra så olika program kan prata med varandra. Kommunikationer sker oftast fram och tillbaka mellan Back-end och front-end för att kunna komma år olika data. För att komma åt REST api gör man med http med färdiga bestämda url:er. Men för att man ska kunna veta när man ska skapa data eller om man vill ha data använder man en http metod. Dessa metoder matchar bra ihop med create, read, update och delite (CRUD). Med informationen man får av CRUD så kan man avgöra vad man vill göra i back-end. På back-end som REST lever på brukar vara ett ställe som man knyter samman en eller fler tjänster. Dessa tjänster kan var en databas, auth, betalnings lösning med mera. Vad gör man när man vill skicka med data som back-end är beroende av för att göra den uppgift man vill att den ska göra. Man skickar med data i antingen i url:en (query parameters) eller i förfrågan body. Både alternativen har ett syfte. Syftet med query parameters är antingen när man vill göra något med en specifik sak i DB eller om man vill skicka med ett egenskaper för ett filter. Body är om man vill skicka med mycket data. Denna data är oftast fermenterat i JSON. Om man vill koppla en händelse med en användare brukar man skicka med en nyckel i

hedder på varje reqest. Detta gör man för att i back-end kunna lättare kolla upp vem som gjorde förfrågan.

(VG) Lista fem (5) vanligt förekommande missar inom tillgänglighet på webbplatser. Förklara dessa missar övergripande, vad de innebär och vad de beror på. Resonera sedan nyanserat kring tekniska utmaningar kring dessa missar. För varje miss ska du alltså resonera kring varför detta kan vara svårt att uppnå, vad som är utmaningen rent tekniskt och vad man bör tänka på och förhålla sig till för att inte göra dessa missar.

Alt-text för bilder: Om det inte finns någon alt-attribut på en bild så kommer inte programvaror som ska hjälpa ta in informationen om sidan veta vad det är får någon bild. Även om det är en bild som inte tillför något till sidan så ska det finns en alt som har en tom text. Sen en annan vanlig miss är att alt texten inte beskriver bilden på ett bra sätt så man vet vad som händer på bilden. En anledning för att det kanske saknas är för att man laddar in bilderna dynamiskt och att där följer det inte med någon beskrivning av bilden. Något man kan göra för att undvika denna miss är om man till exempel vill beskriva bilder på en produkt så kan man använda en alt-beskrivning som innehåller produktnamnet. En annan miss är att man i alt texten skriver ”bild på …”. Detta är också en miss för att oftast så brukar programvaror som ska hjälpa ta in informationen säga att det är en bild. Då behöver man inte säga det två gånger för användaren.

Tangentbordsnavigering: Många element i HTML går att fokusera på för att sedan få det uppläst. Detta är extra bra för användare som förlitar sig på att tabba sig vidare mellan de olika fokusera bara elementen. Men när detta blir dåligt är när det finns för många att gå igenom innan man kommer till huvudinformationen som man var ute efter när man gick in på sidan. Detta kan förekomma vid olika navelement där det finns minst 20st att tabba sig igenom. Första gången kanske det är okey men när detta blir flera gånger blir detta ett problem. Ett sätt för att undvika detta är att innan navelement sätta en osynlig länk för att hoppa över den långa navelement med alla länkar. Utmaningen är nog att man glömmer att testa hur det är att tabba sig igenom sidan och blunda. Då kan man både se var man hamnar någonstans och vad uppläsningen säger. För det är så lätt för de som kan se innehållet att navigera på sidan som det inte är för andra personer. En annan utmaning är att man försöker själv mixtra med tab index. Som ledder till att man hoppar överallt på sidan och tappar kontexten av sidan vad den vill säga.

Dålig fokusindikator: Liksom bristen på tangentbordsnavigeringen så kan det vara svårt att veta var någonstans man har tabbat till. Detta för att många utvecklare tycker det är fult eller att det bidrar med något dåligt när det kommer till design. Detta gör så att man inte har någon skans att veta var man är någonstans när man tabbar. För att kunna undvika detta är att antingen behåller man standard som kommer eller att man ändrar på färg och rammen som kommer när man tabbar.

Felaktiga rubriker: När man skriver ett HTML så måste man ha en bra ordning på sina rubriker och att man bara har en h1 per sida. Varför man måste ha en bra ordning på sina rubriker är för att man lättare ska kunna förstå innehållet på sidan. Problem är då att manga tänker att bara för att i designen så är det en mindre rubrik än den som kommer föra så behöver man använda en fel rubrik typ. Till exempel att man går från en h2 till en h4 som inte är korrekt. När man egentligen bara ändra utseendet och behålla rätt rubrikelement. Detta för att varje rubrik tjänar ett syfte. En annan utmaning är att det inte alltid man ska följa designen för att i HTML inte alltid stämmer överens. Ett exempel kan vara i en produkt kort så finns det en bild och sen en tillhörande rubrik. Om man skulle ha samma ordning på elementen som på bilden så kommer man aldrig att veta ett produktbilden hör ihop.

Formulär: Det finns många missar man kan göra i formulär. Det jag ska fokusera på är felmeddelanden. Ett felmeddelande är riktigt bra om det finns eftersom man vill veta vad man har gjort fel. Det kan vara att det måste minst vara 8 tecken långt eller att man missar en @ i e-posten. Om man inte får reda på felet kan det leda till irritation varför det inte händer något när jag trycker på skicka/bekräfta knappen. Anledningen kan vara en man missar att göra fel hanteringen rent av eller att man glömmer att visa det i formuläret och bara gör en console log. Om man använder react så finns det många bra npm paket som React hook form som använder zod. Detta gör det enkelt att hantera fel i formulär. Sen en annan utmaning är att man inte vet hur man ska koppla samman felmeddelanden och inputfält för att det ska bli upplästa rätt.