# Reflektion från kodgranskning av “Inloggninssida”

När jag började läsa de andras koder fick man en bra inblick i hur dem tänker. För att förstå hur de har tänkt så lägger jag upp min kodgranskning på liknande sätt som jag tror de flesta skriver den.

Först granskar jag HTML för att se strukturen av vad som finns här och nu. Samtidigt kollar jag de olika ”containers” man har skapat för att lättare navigera mig i CSS och senare JavaScript filen. I CSS letar jag främst över någon större ändring, till exempel tillkommande klasser och Id’s som bara används i DOM/JavaScript eller om man gömmer något. Det som inte syns i HTML vill jag se i koden och på så sätt kan jag lättare dela upp vad som händer och vart någonstans. Blev ganska smidigt att komma i gång efter den uppdelningen.

## Vilken är den viktigaste lärdom tar du med dig från kodgranskningen?

Man blir ganska ödmjuk inför att det faktiskt går att lösa problem på många sätt. Nu hade flera av oss i gruppen liknande lösningar. Och det bästa som kodgranskare var att få se ett väldigt simpelt strukturerat dokument som var lätt att följa från början till slut; där allt kommer i ordning, användning av bra och generella variabelnamn som man kan återanvända. Blir lätt att man snöar in sig på att använda kod och namn just för en specifik situation, att man inte ser ”skogen för alla träden”. Fördelen med en god välstrukturerad kod är att det är så mycket lättare att hitta felen och buggarna.

Det var väldigt lärorikt att få feedback om sin egen kod och vice versa. Lärde mig faktiskt lika mycket om kodningen genom att förstå och försöka lösa andras problem som det var att göra på sin egen sida. Man blir lätt trött på sin egen kod och gärna hemmablind. Förenklar arbetet att komma framåt med projektet när man har någon att bolla idéer och frågeställningar med. Det blir också mycket roligare.

En viktig läxa är att kodgranskningen och framför allt feedbacken behövs. Mer tid på oss att diskutera är bara positivt. En del kanske inte vågar säga vad dem tycker och vice versa. Själv trivs jag som bäst att få feedback samt leka ”djävulens advokat”, så jag verkligen lär mig vad som händer i koden; att få testa andras och sina egna teser. Men som sagt i presentationsform är det alltid svårt. För man vet inte vart man har den andre personen riktigt. Man kanske inte känner varandra sedan innan, och ju mindre tid man har på sig desto svårare är det. Våga visa sig sårbar, mer feedback och fler diskussioner förespråkar jag så samtliga känner sig mer bekväma i kritik.

## Vad tillförde du under kodgranskningen?

Jag hoppas jag hann tillföra några lösningar så att det inte bara blev adjektiv på den tiden. Till exempel var det en person i gruppen som skapade många funktioner och deklarationer för samma syften och metoder. Man hade kunnat banta ner koden genom att antingen:

* Skapa sina sidor i JavaScript snarare än i HTML.
* Skapat ”containers” som täcker en större del av tillhörande element som man hade kunnat gömma, snarare än var för sig för varje enskild element.

En annan tips för en kollega var att inte glömma validera sin kod. Även om det mesta gick köra ändå så var det några funktionella saker som hade kunnat tillföra buggar i vissa situationer. Likaså med att deklarera saker som inte används; och det är lätt hänt när man döper olika värden till liknande variabler och glömmer sedan bort vilka som är i bruk.

En tredje var CSS. Även om den gärna bortses vid MVP så kan det bli snurrigt för användaren att veta vad som är korrekt eller inte, och märkte även det som kodgranskare. Olika element är positionerade i ”absolute” och får situationer där beroende på skärmstorlek så täcks vissa element över av andra, eller att hela formuläret inte syns och så vidare. Ett problem som jag såg samtliga i gruppen hade var förståelsen av hur ”width” och ”flex” samverkar. Mycket kod som var överflödig då man missar ”flexboxens” uppbyggnad (något jag själv haft återkommande problem med vid testning).

## Ge exempel på när du anser att man bör använda DOM-manipulering med hjälp av javaScript, och varför?

Allmänt vill jag gärna se DOM användning vid tillfällen där något som inte finns statiskt på sidan ska hända temporärt eller dynamiskt. Nu hade vi till exempel denna uppgift där vi använde oss av key/value pair i localStorage för att kolla om vi var inloggade. Jag skapade element som tidigare inte fanns i HTML filen, till exempel en välkomst ruta för lyckad inloggning. Den vill man gärna skapa med hjälp av DOM då den är tillfällig och meddelar användaren just då. På samma sätt vill man ta bort den välkomstrutan med DOM när man tar sig till en ny ”fysisk” sida eller om du loggat ut. Dessa element behöver ingen egen sida och tillför ingen nytta för den bredare HTML strukturen.

På liknande sätt kunde vi skicka vidare värden (namn och lösenord) som användaren skrev in i formuläret för inloggning. Detta kunde man göra genom att använda en ”submit” knapp som man via ”addEventListener()” lyssnar av den funktion man tilldelar elementet. I mitt fall hämtas värden av vad användaren skrivit in i fälten för att sedan kunna testas mot de värden som redan existerar för att se om de är lika. Det är tack vare DOM som vi kan hämta elementen (och skapa m.m.), för att sedan kunna manipulera det genom att sätta en metod och funktion som exekveras när någon användare interagerar med elementet i fråga.

Ett annat exempel utanför vår uppgift där man skulle kunna använda sig av DOM till sin fördel är till exempel en kundvagn vid e-handel. Tänk dig att du är inne på sidan och hittar varor du vill köpa; du klickar på en knapp för att lägga till den i din varukorg (varukorgen kan sparas som en JSON string i localStorage). Vid tryckning så hämtas informationen och den visuella delen med hjälp av DOM. Den uppdaterar då varukorgen (varorna läggs till i en Array) och sparar den i localStorage. Vi bör dock inte spara prisinformation eller annan känslig information däri men varorna i sig går bra.

Du kanske stänger ner datorn ett tag för att sedan vilja återuppta tidigare session och då finns din varukorg sparad med det som hade lagts till tidigare. Genom att spara icke känslig information i localStorage så belastas inte servern så mycket heller. Innebär också att man skulle kunna bygga en e-handel med en varukorg som funkar utan internet så länge man shoppar runt. När man är redo att betala så kopplar man sig upp mot internet och varukorgen uppdateras med nuvarande prisinformation och tillgänglighet.