Modul 4 - Projekt

Namn: Filip Bernhard Arrdal
Kurs: Grundläggande javascriptprogrammering GJP100
Datum: 10/01/2024

Innehållsförteckning

1. Inledning

Jag har i detta modul fått som uppgift att skapa en applikation som visar mina förmågor inom Javascript programmering. Jag har jobbat med ett projekt som inom den tidsramen jag har givits visar upp min förmåga att på bästa sätt. Tidsramen jag har fått att jobba efter är ca 40 timmar totalt för hela projektet. Jag har valt att jobba en vecka med projektet för att få en rimlig representation av detta.

Eftersom den begärda komplexiteten av applikationen inte är definierad så har jag valt ut ett par tekniker från kursen att använda mig av. För att förhindra att projektet drar över på tiden, så väljer jag bort några funktioner som hade varit intressanta att jobba med.

2. Presentation av applikationen

Jag har skapat en applikation som jag har valt att kalla “Flashcards”. Det är ett verktyg för att förbättra sig på ett språk genom så kallade “flash cards”, alltså kort som visar ett ord på både ens modersmål samt det språk man vill öva på.

Applikationen fungerar sådant att man har en *kortlek* med ett antal *kort* i, man får sedan upp ett kort i taget och får då gissa för sig själv vad den korrekta översättningen är, sedan visar man det riktiga svaret och ser om man hade rätt eller inte. Detta är enligt (Lama, Torres, 2024) en väldigt användbar teknik för att förbättra sig på ett språk.

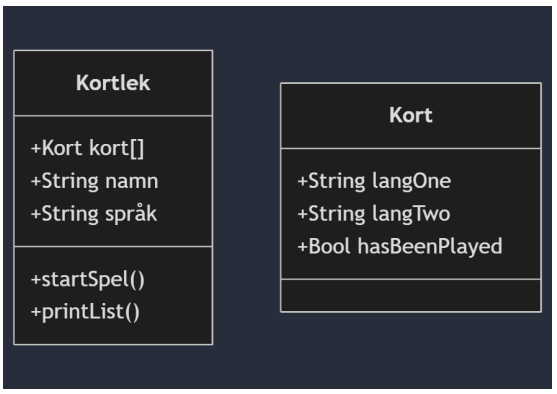
Man får i applikationen lägga till egna *kortlekar* med egenskapade *kort* som man sedan kan klicka på i en meny för att använda i själva spelet. Korten kommer fram i slumpad ordning för att inte göra inlärningen för förutsägbar och man får själv klicka om man hade rätt eller fel svar. När alla kort har visats så kommer resultatet upp på skärmen för att se hur många rätt man hade.

Alla kortlekar sparas i webbläsarens minne, så man kan återkomma senare och alla ens kortlekar ska finnas kvar för användning.

3. Planering och Förberedelse

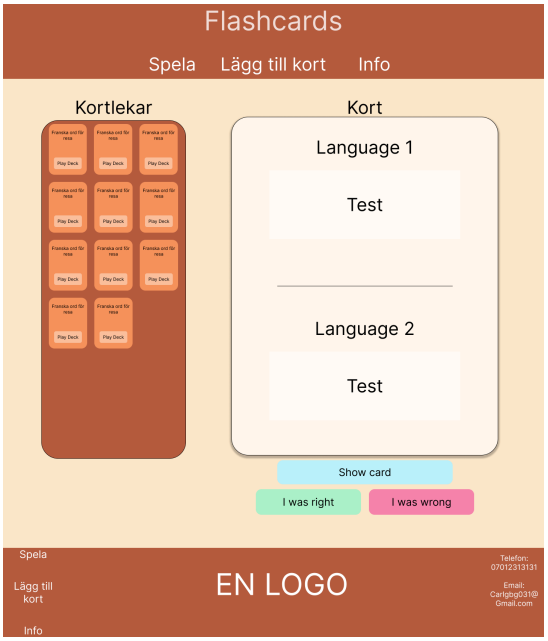
Det första jag gjorde i min planering var att bestämma vad syftet med applikationen var. Jag ville skapa ett verktyg som var enkelt och lättillgängligt. Idén är att man ska kunna komma in och köra igång direkt utan komplikationer.

Jag ville att sidan skulle ha ett relativt enkelt koncept. Det skall finnas kortlekar som innehåller kort. För att komma igång med detta började jag med att göra mina klassdiagram med hjälp av *Mermaid*s verktyg på webben.



Dessa klasser kommer att ändras något senare, men detta ger mig en bra startpunkt för att komma igång med kodandet.

Efter detta skapade jag en snabb skiss i Figma också, för att få en idé om hur jag vill att applikationen ska se ut rent visuellt.



Denna skiss är inte heller riktigt som det senare kom att bli i slutändan, men man kan se några likheter. Igen så ger detta ett bra startläge när man väl ska börja designa hemsidan i CSS.

4. Utformning av Koncept och Design

Som tidigare nämnt var konceptet i mitt huvud en visuellt enkel och lätt användbar applikation. I min figma design kan vi se hur jag tänkte att uppbyggnaden skulle vara. Header, footer samt en *main* del uppdelad i två sidor. En sida för alla kortlekar man har att använda sig av samt en ruta med själva "spelet". Jag ville att detta skulle vara nära inpå varandra för att snabbt kunna hoppa in i ett spel utan att behöva ladda in nya sidor.

Den enkla designen skall göra så att man efter några sekunders inspektering förstår sig på hur det fungerar.

(Chapman, n.d.) menar att en enkel design gör hemsidan mer "scan-bar" vilket gör det enklare att hitta själva innehållet och komma igång.

5. Arbetsprocessen

Min plan när jag började arbetet var att jag skulle bygga upp ett helt skal som fungerar med alla funktioner och sedan när allt detta fungerar börja med design. Detta visade sig vara svårare än planerat dock.

Jag hade väldigt svårt att få en idé hur koden skulle fungera utan någon visuell hjälp. Efter att jag hade skrivit klart grunden för mina klasser och skapade några objekt av klassen *kortlekar* så vill jag se till att dessa kunde listas ut på hemsidan. Men eftersom jag ville att dessa skulle skrivas ut på ett visst sätt som text på sidan så behövde jag genast börja arbetet med att designa elementet där kortlekarna skulle placeras.

Det blev på detta vis jag senare kom att göra alla mina funktionaliteter. Först skriva en grundkod för att senare skapa en grunddesign och sedan jobba upp båda delarna pö om pö.

När jag kände att en funktion, till exempel hur själva kortspelet skulle gå till, fungerade som det skulle, först då gick jag vidare till nästa del.

5.1. Implementering av Funktionalitet

Jag har jobbat efter främst en sida, “index.html” där alla funktioner ska få plats. Jag började med att separera till två filer, men detta visade sig senare skapa en väldigt massa problem.

Nämligen att om jag ändrade värdet på en variabel genom att klicka på en knapp i ena filen, så ändrades inte variabeln i den andra. Detta var ett problem eftersom jag i början hade lagt upp det sådant att man la till nya kortlekar på en sida som jag sedan stoppade in i en array, men array kom ej att uppdateras väl på index sidan. Efter mycket om och men beslutade jag att ha allt på en index-sida där jag bytte ut vilka html element som faktiskt visas på sidan.

Något som också skapade lite av ett problem var det att jag verkligen ville jobba med typescript. Eftersom typescript som standard kompilerade till javascript av typen “commonJs” blev mina *imports* av typen *require* vilket min webbläsare inte ville använda sig av. Till slut hittade jag en forumtråd på (Stackoverflow, 2015) som förklarade hur jag kunde kompilera till es6. Detta är enligt (Geeksforgeeks, 2024) en sorts javascript som mer liknar på typescript då typescript enkelt sagt är ett superset av es6.

5.2. Tekniker

Mitt projekt är uppbyggt främst av *eventListeners* och funktioner, utöver detta har jag jobbat väldigt mycket med DOM-noder. I grunden har jag skapat två klasser varpå jag byggt mina huvudsakliga variabler. Med mina klasser har jag skapat tre “lager” av arrayer och variabler som mitt program är byggt på. Dessa tre går i led *kortlekar* som innehåller element av typen *Kortlek* varpå denna array innehåller element av typen *Kort*, det går alltså Kortlekar till Kortlek till kort.

Vad det gäller loopar har jag använt mig av både **for** och **foreach** loopar. Jag har använt mig av for-loopar när innehållet i loopen är något längre. Detta för att enligt (Code, 2023) en for-loop ger något mer kontroll, samt att variabler deklarerade i en for-loop stanna inom scopet av just den loopen och inte följer med ut så att säga. Foreach loopar har jag använt mig av när jag bara behöver göra en eller max två operationer inuti loopen.

En teknik som jag visste direkt att jag vill inkludera är *localStorage*. Det innebar att jag kunde be webbläsaren lagra informationen om vilka kortlekar och kort som finns. På så sätt finns alltid de kortlekar man har skapat kvar när man återkommer. Att använda localStorage visade

sig dock vara klurigare än jag hade tänkt. Eftersom jag sparade information som ren text, och sedan ladda tillbaka det som JSON så innebar det att det inte lästes in korrekt som typ av mina klasser “Kortlek” och “Kort”. Jag hittade igen på (Stackoverflow, 2021) ett svar från en användare som introducerade begreppet *reviver* för mig. Jag använde hans kod som grund för att jobba in detta så som jag behövde det i mitt projekt och till slut så kunde jag får JSON texten att lagras korrekt.

5.3. Testning och Felsökning

Jag har främst använt mig av två typer av testning i mitt arbete: Unit testing och End-to-end testing.

Unit testing innebär att man testar enskilda “enheter” när man skapat dem för att se till att de funkar som man tänkt, detta enligt (Birnir, n.d.). Något som jag har gjort varje gång jag har skrivit klart en funktion eller bara en färdig bit kod är att jag direkt ser till först: att koden faktiskt körs, detta ofta genom console.log() meddelanden. men jag ser också till att allt som som funktionen skall åstadkomma, uppfylls direkt.

Utöver detta har jag som nämnt använt mig av end-to-end testing, vilket enligt (Birnir, n.d.) innebär att man testar sin webbapplikation från start till slut. Tanken är då att man testar alla komponenter så som en användare skulle göra, innan man ger över projektet till klienten till exempel. Jag har använt detta som en metod för att säkerställa att hemsidan fortfarande fungerar efter att jag var färdig med en större bit av sidan. På detta sätt kan jag göra en stabil backup på med hjälp av git och vet då när jag börjar igen att det vid denna punkt fungerade det som det skulle.

5.4. Kompatibilitet

Jag har använt mig av caniuse.com för att identifiera några saker på min applikation som kanske inte stöds till fullo av äldre webbläsare. Det första jag har testat är om min hemsidas element faktiskt funkar i olika webbläsare och det första jag slogs av är att stödet för *grid layout* inte finns i många äldre versioner av bland annat Mozilla Firefox och Google Chrome, detta gäller också för flexbox.

Utöver detta ville jag också testa stödet för ES6 vilket är versionen av Javascript jag har använt mig av. Stödet här är väldigt brett, men igen saknar stödet i äldre versioner av nästan alla stora webbläsare.

6. Reflektion och Utvärdering

Efter att ha slutfört mitt projekt finns det många saker jag hade kunnat göra bättre. Trots att jag är nöjd med slutresultatet så blev det inte alltid så som jag hade tänkt mig i huvudet innan jag började.

Jag fick göra ganska många kompromisser under arbetets gång som möjligtvis hade kunnat undvikas med bättre planering. Bland dessa var faktumet att jag ville ha tre separata filer, men då min kompetens och hur mycket tid jag hade begränsade detta fick jag jobba om ganska mycket. Resultatet av detta är att hemsidan i min mening inte känns lika robust som

den hade kunnat göra, utan man måste vara ganska försiktig när man ändrar i kod som det är nu.
På en ljusare not så är all funktionalitet som jag ville ha där. Applikationen går att använda som tänkt, så från användarperspektiv så är det för tillfället inga större problem vad jag känner till.

I framtida projekt så kommer jag lägga oerhört mycket mer tid på förarbetet. Fastän jag föredrar att faktiskt sitta och knacka kod, så är det inte lika roligt när man sitter frustrerad över att ens vision inte är möjlig som man tänkt.

Så om det är något jag har lärt mig av allt detta är det vikten av god planering.

7. Referenser

Lama, X. Torres, G (27 november 2024). Do flashcards really work? Yes, - Here's How! . Lingopie.
https://lingopie.com/blog/do-flashcards-really-work-yes-heres-how/?form=MG0AV3

Chapman, C (n.d/) Simplicity in Good Web Design : Advantages & How -to. Jotform.
https://www.jotform.com/blog/simplicity-in-good-web-design-advantages-how-to/?form=MG0AV3

@zmii (25 maj 2015) How do I transpile TypeScript to ES6?. Stackoverflow.
https://stackoverflow.com/questions/30439869/how-do-i-transpile-typescript-to-es6

@StuartM (27 april 2021) Converting a JSON object into specific type. Stackoverflow.
https://stackoverflow.com/questions/67287257/converting-a-json-object-into-specific-type

Geeksforgeeks. (10 juli 2024) Difference Between ES6 and TypeScript.
https://www.geeksforgeeks.org/difference-between-es6-and-typescript/

Code, L (30 oktober 2023) Difference Between forEach and for Loop in JavaScript: A Deep Dive. https://thelinuxcode.com/difference-between-foreach-and-loop-javascript/

Birnir, A (n.d.). A Beginner's Guide to Testing Your Code. Skillcrush.
https://skillcrush.com/blog/a-beginners-guide-to-testing-your-code/#4