Inledning

I denna rapport förklarar jag hur jag har gått tillväga för att lösa uppgiften U3 i kursen 1ME322 på Linnéuniversitetet. Uppgiftens slutmål bestod av att få ett fullt fungerande memoryspel. Detta skulle uppnås genom att komplettera de redan implementerade HTML och CSS filerna med funktionalitet från en javascript fil. Följande berättar jag om hur jag har valt att strukturera min kod och vilka funtioner jag har skapat för att nå slutmålet.

Kodstruktur

Jag valde att samla alla funktioner i en och samma fil. Att dela upp funktionerna i olika javascript filer är ofta en bra struktur för att hålla ordning och reda på koden. Jag ansåg dock att uppgiften inte var så pass komplex samt att den inte krävde allt för mycket kod för att detta skulle vara fördelaktigt. Min javascript fil , som är döpt till script.js, laddas in som vanligt via en script tag i HTML filen index.htm.

Jag har 7st huvudfunktioner som är direkt nödvändiga för att spelet ska fungera, dessa kommer jag att berätta mer om nedan. Jag har också 3st funktioner som används av huvudfunktionerna och finns till för att det är återkommande funktionalitet som används av flera funktioner och hjälper då till att minska duplicering av kod.
Dessa tre funktioner är:

Funktionen **addOnClick()** - Itererar igenom alla brickor, är brickan inte tom så läggs en onclickfunktion på brickan som leder till funktionen turnBrick. Skulle brickan vara tom så får den ingen onclick funktion, anledningen till detta är att turnBrick inte ska triggas när man klickar på en tom bricka.

Funktionen **removeOnClick()** - Itererar också igenom alla brickor men tar istället bort onclick. Denna funktion används för att undvika att onclick funktioner kan triggas då möjligheten att vända på brickor inte ska finnas.

Funktionen **disableButton(button, boolean)** - en funktion som aktiverar eller inaktiverar en knapp beroende på vilket värde man skickar med som parameter. Används för att kontrollera flödet av spelet, så att man t.ex inte kan starta ett nytt spel mitt i ett pågående spel, eller att man inte kan klicka på nästa knappen innan man har vänt på två stycken brickor.

Funktionen init

Detta är funktionen som körs när sidan laddats in. Här kopplas de variabler som behövs för att starta spelet till respektive HTML element samt lägger till nödvändiga onClick funktioner. Spelarens tidigare poäng hämtas också in här från localstorage som sedan visas i informationsrutan. Jag valde också att lägga till ett anrop till funktionen setBricks, detta för att sätta ett initialt val av antal brickor, så att man kan starta spelet direkt utan att välja antal brickor.

Funktionen startGame

Denna funktion exekveras när spelaren trycker på knappen “Starta spelet”. Då inaktiveras alla knappar och det enda spelaren kan klicka på är brickorna. Jag sätter också värdet 0 på variabler som används under spelets gång.
Avslutningsvis i denna funktion kallar jag på funktionen addOnClick för att göra det möjligt för spelaren att vända på brickorna.

Funktionen turnBrick

Funktionen exekveras enbart via ett klick på en bricka. När första brickan vänds sparas ett värde för just den brickan. Detsamma sker vid andra vändningen men här tillkommer viss

funktionalitet. Värdet på antal gånger ett par brickor har vänts uppdateras, next knappen aktiveras samt möjligheten att klicka på andra brickor inaktiveras via removeOnClick funtionen. Värdena för brickorna som vänst spara som globala variabler och jämförs i next funktionen.

Funktionen next

Funktionen exekveras när spelaren klickar på nästa knappen. Då aktiveras brickorna och blir vändbara igen. Vidare kontrolleras värdena av de två tidigare vända brickorna genom en if else sats. Har dom samma värde på attributet src, mao samma namn på filnamnet, är det ett par och tilldelas ett nytt attribut för att göra dom osynliga. Här kallar jag också på funktionerna removeOnClick samt addOnClick för att de osynliga brickorna inte ska ha en onClick funktion. Skulle de valda brickorna inte vara ett par vänds dom tillbaka.
Efter denna sats återställs värdet för hur många brickor som visas samt att “nästa” knappen inaktiveras igen.
Slutligen kontrolleras det om alla brickor har vänts och spelet därmed är slut. Detta görs och genom en if else sats och om den visar sig vara sann kallas funktionen endGame.

Funktionen setBricks

Denna funktion exekveras i init funktionen samt när spelaren väljer att ändra antalet brickor. Funktionen bygger upp och skapar nya bild element i indexfilen beroende på hur många brickor spelaren valt att ha. Spelbrickan tilldelas en stil utfirån hur många brickor spelaren har valt att ha på bredden. Denna stil specifierar då hur bred spelbrickan ska vara och detta görs för att brickorna ska blir uppradade på ett symmetriskt sätt.
I funktionen, innan spelbrickan tilldelas brickor, tas alla brickor bort för att undvika att det blir fler brickor än vad man har valt.
Efter att brickorna tagits bort ska det fyllas på med nya brickor. Detta sker genom en for loop som baseras på antalet brickor spelaren har valt. Spelaren väljer alltså storlek på spelbrickan från fem stycken fördefinierade val, där antalet på höjden multipliceras med antalet på bredden för att få fram hur många brickor som ska finnas på spelbrickan.
Inuti for loopen skapas det sedan en ny bricka med alla attribut som krävs för en bricka som visar baksidan och denna puttas då in i en array med referenser till alla brickelement.

Funktionen assignImages

Här fylls en array med referenser till olika bilder. En iteration görs baserat på längden av arrayen med img elementen som sätts i funktionen setBricks. Denna längd divideras sedan med två och i varje iteration läggs en bild till två gånger i en bild array, detta görs för att skapa rätt antal par till spelbrickan. Den nya arrayen med referenser till bilderna blandas sedan, detta görs också via en for loop, där man i stora drag, tar fram ett slumpmässigt tal för att sedan byta plats på brickan kopplat till loop variabeln i mot brickan som finns på det slumpmässiga talets plats i arrayen. Arrayen som skapats med bilder finns genom hela omgången som en global variabel och används av andra funktioner.

Funktionen endGame

Denna funktion kontrolleras i funktionen next och exekveras endast när man har hittat sista paret. Här görs en uträkning av poängen via formeln som tillhandahölls via uppgiftsbeskrivningen. Poängen adderas till den tidigare poängen som sparats i localstorage och uppdateras sedan i informationsrutan. Knapparna för att välja antal brickor och starta spelet aktiveras igen så att spelaren kan fortsätta spela utan att ladda om sidan.

Slutsats

Valet av den struktur och de funktioner som finns i min applikation tycker jag föll sig ganska naturligt, då mycket baseras på den information och krav som uppgiften gav. Jag har hela tiden försökt hålla det ganska simpelt och därav skrivit funktionerna utifrån de händelser som sker under ett memory spel, samt de knappar som fanns till användning för spelet i den fördefinierade HTML filen.