Jag började med att läsa igenom koden, titta på de förklarande filmerna kopplade till uppgiften samt på beskrivningen och tipsen. Efter det kände jag att jag fått en okej förståelse för vad som skall göras och vad resultatet förväntades bli.
Eftersom uppgiften var ganska stor så var jag fundersam över hur jag skulle börja men jag beslutade mig för att försöka slutföra punkterna listade i krav-delen i uppgiften en i taget i den ordning de var skrivna.
Jag kommer följa samma struktur här i rapporten, på så sätt kan man följa hur jag tänkte kring de olika delarna och då även se hur min kod utvecklades i kronologis ordning.

”Då man klickar på knappen för nytt spel, ska spelet initieras samt knappen för nytt spel inaktiveras och knappen för nya brickor aktiveras.”
- Likt tidigare uppgifter vi fått göra i u1&u2 började jag med att skapa en init funktion och en event listner som skall trigga denna funktion då vårt webb-läsarfönster laddas.
Jag vet även att vi kommer arbeta med alla knappar så jag deklarerar globala variabler för att spara referenser till dessa element i init funktionen. Jag använder sedan event-listners, ”.disabled” samt skapar grunden för newGame funktionen för att kunna känna mig klar med denna del.

”Då man klickar på knappen för nya brickor, får man fyra nya brickor som väljs slumpmässigt bland brickorna 1-40. Samma bricka ska inte kunna väljas mer än en gång. De nya brickorna placeras i rutorna ovanför knappen.”
- Nu börjar det bli lite mer komplicerat och efter lite funderande tar jag den enklaste vägen jag kan tänka mig. I new game skapar jag en global array, ”initImgList”, som i newGame funktionen fylls med strings som kommer funka som referenser till de 40 olika bilderna vi har i vår img-mapp.
Nu gör jag även en ny funktion ”newBricks” som kopplas till nya brickor knappen.
I denna funktion använder jag initImgList och likt den stil jag använt tidigare i uppgifter väljer jag ut slumpmässiga tal mellan 0 & 40, lägger in i elementen via ”.src” och sedan tar ”klipper” bort den relevanta delen av listan en bit i taget med ”.splice” via en for loop.

”Man kan sedan dra de nya brickorna till valfri plats i spelplanen. Man drar en bricka i taget i valfri ordning.”
- Detta löser jag genom en for loop som gör dessa element dragbara samt kopplar listners för ”dragstart” & ”dragend” till dem som även leder till nya funktioner med dessa namn; dragStartBrick & dragEndBrick. Här gick jag faktiskt tillbaka till föreläsningarna ang. drag & drop för att få en bättre förståelse av hur de fungerar samt för hur en fungerande struktur av dessa kan ser ut. Samt hur datatransfer funkar med objekt. Denna struktur använde jag för att uppfylla kravet nämnt ovan samt följande krav; ” Man ska kunna släppa brickan över en tom plats i spelplanen. Medan man drar, ska den tomma plats man drar över markeras med en annan bakgrundsfärg.”.
-Jag hade några problem med att få in rätt bit av den text som låg i elementet som flyttas ”src” del då den innehåller ”…localhost…”. Detta löste jag initialt med textmanipulation som var ganska enkel då jag alltid var intresserad av den bit som kom efter bokstav ”i” i elementets src del. När jag i slutskedet sedan använde mitt program via netlify fick jag

problem då länken till bilderna kunde se olika ut. Till skillnad från localhost länken hade netlify´s länkad ofta flera i sig vilket gjorde att jag tog fram fel siffror. Jag felsökte länge algoritmen för beräkning och undrade hur resultaten kunde skilja sig åt. Med hjälp av chromes debugger hittade jag dock att de berodde på att fel siffror lades in i id elementen då de extraherades från länkarna. Jag skrev då om koden så den sista biten av img länken togs fram på ett sådan sätt att den alltid endast skulle innehålla de siffror jag var intresserad av och därmed var problemet löst. Detta problem var alltså när jag ansåg att jag var klar med koden och skulle testa det på netlify.

Jag hade även börjat oroa mig för hur jag skulle räkna poäng och tänkte att det kunde vara smart i samband med den delen jag jobbade med nu försöka lösa detta. Jag valde därför att lägga in siffran på bilden i elementets id. Detta gjorde jag med hjälp av ett regEx uttryck och sparade sedan numret i imgNr för att sedan lägga in det i elementets id attribut. För att man endast skulle kunna släppa en bricka på en tom plats la jag även in ” this.id === "" ” i rad 97. Här hade jag ännu inte tänkt på css klasserna empty & brick, då hade jag kanske använt dem här i stället.

”Då alla nya brickor dragits till spelplanen, ska knappen för nya brickor aktiveras igen, så att man kan klicka fram fyra nya brickor.”
- Även kod för detta skrevs i ”drop” delen av ”dragEndBrick”.
Jag använde ” initImgList.length > 24” för att man inte skulle kunna be om nya brickor om alla 16 platser var fyllda och jag skapade även en ny global variabel för att räkna hur många brickor som lagts med namnet ”brickCounter”. Jag visste att det var efter var fjärde dropp som man skulle kunna be om nya brickor och därför passade modulo räknaren perfekt till detta.

”Både i spelplanen och i rutorna för de fyra brickor man fått slumpmässigt ska två olika klasser användas. Empty ska användas då rutan är tom. Brick ska användas då rutan innehåller en bricka, dvs då den innehåller ett nummer.”
Detta var enkelt då jag hade en fin struktur för detta i drop delen av ”dragEndBrick”. Jag la därför bara till en rad för att interagera med elementets klass namn i rad 117.
Jag behövde även skapa något som ”nollställde” spelplanen då man började ett nytt spel så inte gammal data låg kvar i spelplanens element. I funktionen newGame la jag därför in kod för att återställa alla spelplans elements id´s samt klasser.

” Då spelplanen är fylld ska alla rader och kolumner kontrolleras. Om det är en stigande serie, dvs numret i varje bricka är högre än numret i föregående bricka, ska det markeras med en grön bock (✓) till höger om raden eller under kolumnen. I annat fall ska det markeras med ett rött kryss (✗).”
-Detta var det mest komplexa att få ordning på. Jag ville egentligen använda något som en 2d lista men insåg att det inte skulle gå med javascript. Jag ville sedan få fram en lösning som gav mig resultat horisontellt och vertikalt i en och samma struktur. Jag ville även ha något som behandlade siffrorna steg för steg och då även avbröt beräkningen av en ”rad” när man insåg att man inte skulle få några poäng och i stället då gick vidare till nästa rad.
Många timmar senare gav jag upp dessa försök. Jag gjorde det enkelt för mig och delade upp

den enklaste fungerande koden jag hade så att jag beräknade en del i taget. Ett problem som tog mig förvånansvärt lång tid att lösa var at jag inte kunde får korrekta resultat när jag skulle jämföra siffrornas värde med hjälp av dess id. Långt senare insåg jag att jag försökt använda logiska operatorer på text och inte på siffror. När jag insåg detta och la till ”parseInt” i rad 161 löste jag äntligen detta otroligt frustrerande och förvirrande problem.

”Antal korrekta rader räknas och detta antal blir poäng, som skrivs ut under spelplanen.”
-Detta var enkelt efter att ha löst föregående problem.
Jag bara använde .innerHTML och la in relevanta värden genom den.

”Poängen för en spelomgång ska adderas till totalpoängen. En räknare för antal spel ska också räknas upp. Totalpoängen och räknaren ska visas i rutan i det övre högra hörnet.”
- Samma sak här, detta var enkelt efter att ha löst föregående problem. Jag skapade relevanta variabler och använde dem likt tidigare. Jag hade dock två stycken för poäng, en lokal i scoreCalc till poäng för nuvarande omgång samt en global till totala poäng över flera omgångar.

”Både totalpoäng och räknaren för antal spel ska sparas i en variabel i localStorage eller i en cookie. Båda värdena ska sparas i samma variabel eller cookie. Dessa läses in då man öppnar sidan, så att de poäng och antal spel som man haft tidigare finns kvar, då man kommer tillbaks till sidan. De ska då också skrivas ut på sidan.
-Detta var ganska enkelt med då vi fått så tydliga instruktioner på detta i föreläsningarna. Jag skapade funktionerna ”getCoockieData” & ” saveStats” för att kunna föra över data från session till session.

” Då spelet är klart, ska knappen för nytt spel aktiveras igen, så att man kan spela en gång till.”
- Detta var enkelt då jag endast behövde lägga in en rad i slutet av scoreCalc för att aktivera nytt spel knappen.

Jag la under tidens gång även in saker som det i rad 112 för att man inte skulle kunna fuska och flytta samma bricka flera gånger.
Nu när jag är klar tror jag även att jag hade kunnat förbättra koden för poängberäkning. Jag tror inte jag tog hänsyn till att jag kunde använd listan ” boardBrickElems” för att iterera genom alla element på ett mindre komplext sätt. Nu har jag dock en lösning jag är rätt nöjd med då jag fått använda ganska många olika sätt för att ta mig genom elementen vilket varit väldigt lärorikt så jag vill inte ändra på den lösningen. Att det är den mest effektiva vill jag dock inte påstå.