**Opdracht 33312FA4**

1. Je hebt als het goed is jouw spel al herschreven en worden de kaartjes via de *fetch()* API binnengehaald. Je gaat het spel nu voorzien van nieuwe functies en logica waardoor het echt een spel wordt. De gekozen kaartjes moeten getoond worden en als er een match is, moeten deze uit het spel genomen worden. Is dit niet het geval, dan moeten ze weer omgedraaid worden en moet de speler de volgende twee kaartjes kunnen omdraaien.
2. In een tekstveld moet bijgehouden worden hoe vaak er een poging is ondernomen, hoe vaak de poging succesvol was en optioneel – voor de dare devils – de verstreken tijd in minuten en seconden.
3. Je moet nu heel logisch en helder gaan denken, misschien dingen in een flow chart op papier gaan zetten omdat je stap voor stap moet gaan vertellen hoe het spel werkt en reageert. Je moet alle functies van het spel verdelen in kleine logische functies die uitgevoerd worden op het juiste moment. Denk aan *onClickCard(), evaluateMatch(), keepScore(), nextMove(), resetGame()* etc. Ook moet je naar eigen inzicht const en let gebruiken om waarden in op te slaan. Je hebt een let nodig om de aangeklikte kaartjes in op te slaan, om ze vervolgens te kunnen evalueren, je moet het aantal weggespeelde sets in een let bijhouden, en het aantal pogingen.
4. Wat je het beste kunt doen, is het werken met addEventListener() op het Field (het hele speelveld met de kaartjes), die je met removeEventListener() verwijdert als iemand op twee kaartjes heeft geklikt, zodat tijdens het bekijken van de omgedraaide kaartjes de gebruiker niet op nog meer kaartjes kan klikken. Er mogen er op enig moment maar twee omgedraaid zijn. Pas als de kaartjes teruggedraaid zijn, mag de gebruiker weer klikken en moet je dus de event listener weer toepassen. Dit is echter op verschillende manieren op te lossen.
5. Je moet pauzes inlassen omdat de speler even de tijd moet krijgen de omgedraaide kaartjes te bekijken vóór ze teruggedraaid óf weggenomen worden. Bestudeer de JavaScript method *setTimeout()* om de pauzes in te kunnen lassen, bijvoorbeeld om de kaartjes weer terug te draaien en verder te kunnen. Zoek naar het onderwerp js\_timing en js settimeout op w3schools. Gebruik [deze website](https://link.loi.nl/r/loi-101276) en [deze](https://link.loi.nl/r/loi-101277).
6. Als extra uitdaging kun je een timer bouwen met behulp van JavaScript. Hier zijn online voorbeelden van te vinden: Zie bijvoorbeeld [w3schools 'timing clock'](https://link.loi.nl/r/loi-101278).
7. Bestudeer de JavaScript methods *setInterval() en clearInterval()* om daarmee een timer te kunnen bouwen. Bekijk o.a. [w3schools 'setInterval'](https://link.loi.nl/r/loi-101279" \t "_blank).
8. Bestudeer vooral de voorbeelden met de named functions en clearInterval() om de timer te pauzeren.
9. Als de speler op een kaartje klikt, wordt ook het bijbehorende geluid afgespeeld. Kinderen vinden het geweldig! Je vindt de geluiden in de map snd.
10. De speler moet zijn naam invoeren via een prompt. De naam wordt opgeslagen in de localStorage. Als het spel opnieuw wordt geopend, moet de speler met een alert() welkom geheten worden. Deze data komt uit de local storage. Extra uitdaging: sla de high scores op in de localStorage en toon die na elk spel. Echt een lastige omdat er drie levels zijn. Kun je die tonen met een pop-upvenster?

Ook hier weer de nodige uitdagingen en uren puzzelen. Voor de echte programmeur verslavend! Maar het helpt je enorm jouw programmeervaardigheden te scherpen en leren denken in logica, stappen, oorzaak en gevolg en consequenties van gekozen oplossingen. Als je er niet uit komt, stel vragen aan de docent via 'Stel een vraag'. Onthoud, dit gaat niet om het cijfer voor de module maar om jouw leerproces, als je dit soort puzzels kunt 'kraken', word je een goede programmeur.

**Beoordeelcriteria:**

* Het spel werkt.
* De data komt uit een JSON-bestand en wordt via de fetch() API geladen.
* De naam van de speler wordt gevraagd via een prompt en opgeslagen in de localStorage. Bij het starten wordt een welkomstscherm getoond waarin de naam van de speler wordt getoond, eventueel mét high scores.
* Met addEventListener en removeListener is ervoor gezorgd dat de speler alleen op bepaalde momenten op een kaartje kan klikken.
* Het geluid wordt afgespeeld, en er is een specifieke functie voor geschreven.
* Met setTimeout() wordt er voor gezorgd dat de kaartjes een aantal seconden blijven liggen.
* De kaartjes worden weggenomen als ze matchen, zo niet dan worden ze na een paar seconden omgedraaid. Pas dan kan de gebruiker weer klikken.
* Het aantal pogingen én het aantal succesvolle pogingen worden in een tekstveld weergegeven.
* Extra uitdaging: Er loopt een timer met minuten en seconden!
* Extra uitdaging: Sla de high scores op in de local storage en toon die na elk spel!