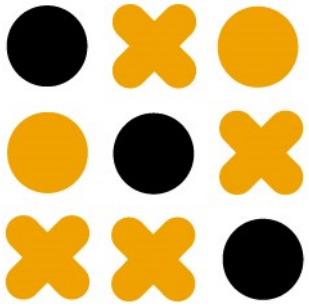
VERSIE 1.0

MULTIPLAYER BOTER-KAAS-EIEREN



AUTEUR: J.J. STROOTMAN

ALFA-COLLEGE

WEBSOCKETS EN JAVASCRIPT

INHOUDSOPGAVE

VOORAF

De web socket server die ter beschikking staat is een basic server gericht op het spelen van het spel Boter-Kaas-Eieren (BKE) in multiplayer mode. Met deze server kan het spel met maximaal twee spelers gespeeld worden. Het spel speelt men in meerdere rondes. Zolang de twee spelers verbonden zijn met de server zullen er steeds rondes gespeeld worden. De server slaat geen scores op. Bij een volgende verbinding zullen de scores weer op 0 staan.

Deze handleiding is bedoeld om met JavaScript een spel client te programmeren die volgens richtlijnen, die in deze handleiding worden behandeld, met de server communiceert.

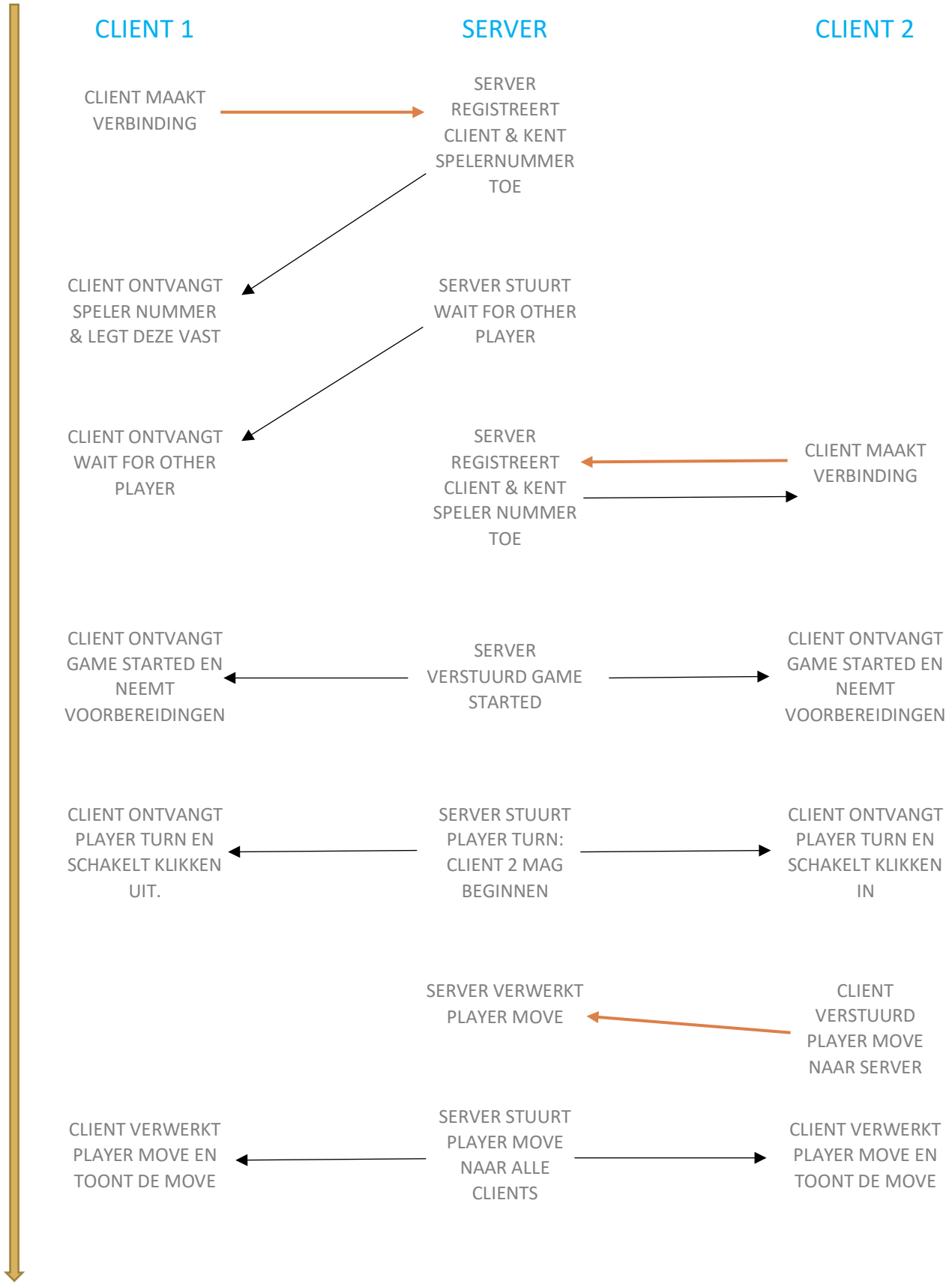
Een spel client krijgt data van de server binnen en interpreteert deze, maar een spel client stuurt ook data naar de server. De data die naar de server gestuurd wordt zal dan ook weer doorgestuurd worden naar de andere verbonden spel client. Dus alles wat er in de game gebeurd loopt via de server.

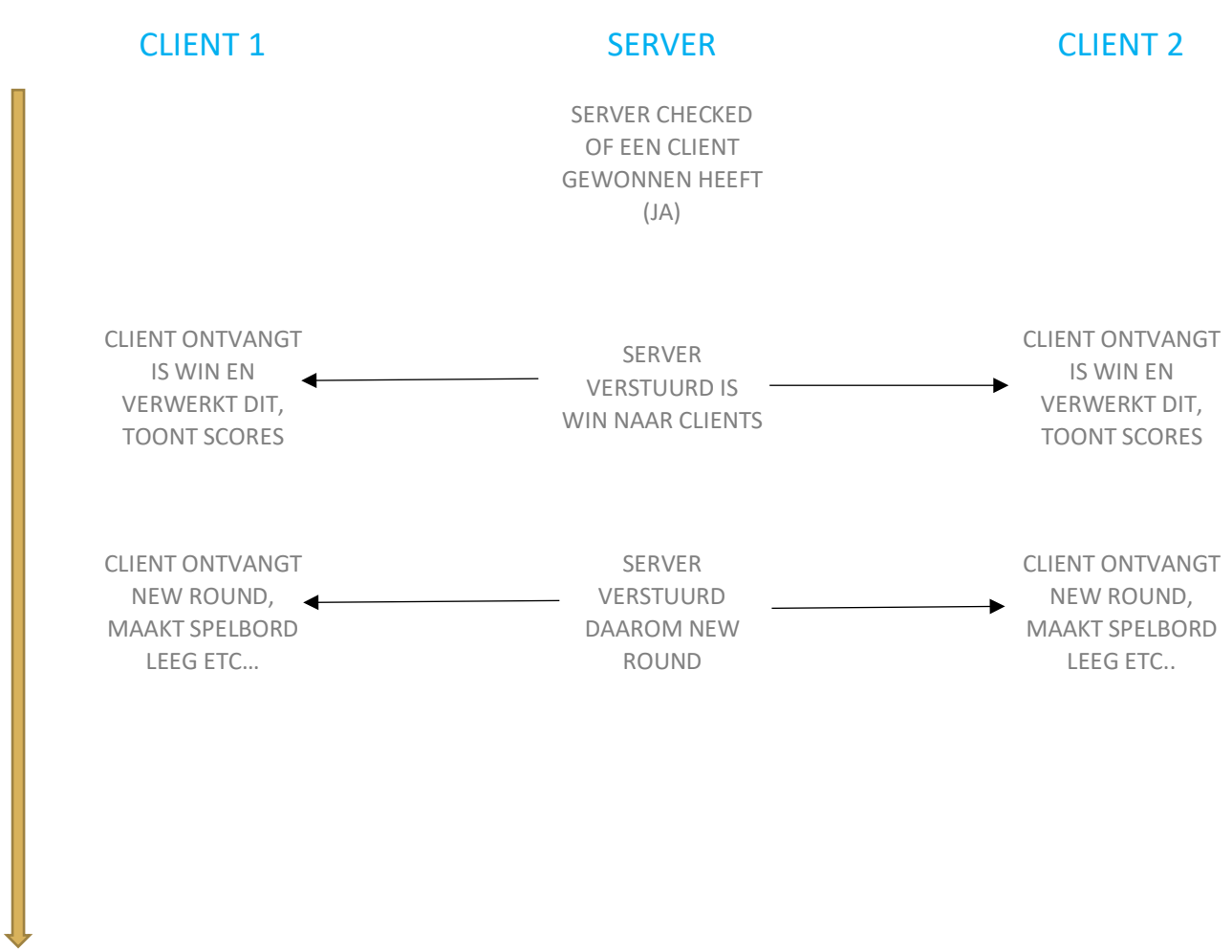
LAY-OUT VAN DE GAME

De server is zo gebouwd dat het zich niet bezig houd met de lay-out van de clients. De clients zijn dus volledig vrij in de vormgeving. In het voorbeeld, wat bij de repository geleverd is, is gebruik gemaakt van images en een opbouw van het speelveld d.m.v. tabellen.

Enige voorwaarde is dat een speelveld altijd uit 9 cellen bestaat.

VERLOOP VAN DE COMMUNICATIE TUSSEN SERVER EN CLIENT




CLIENT 1
SERVER
CLIENT 2
SERVER CHECKED OF EEN CLIENT GEWONNEN HEEFT (JA)
CLIENT ONTVANGT IS WIN EN VERWERKT DIT, TOONT SCORES
SERVER VERSTUURD IS WIN NAAR CLIENTS
CLIENT ONTVANGT IS WIN EN VERWERKT DIT, TOONT SCORES
CLIENT ONTVANGT NEW ROUND, MAAKT SPELBORD LEEG ETC…
SERVER VERSTUURD DAAROM NEW ROUND
CLIENT ONTVANGT NEW ROUND, MAAKT SPELBORD LEEG ETC..

CONSTANTEN/COMMANDS VOOR DE COMMUNICATIE

Plaats de onderstaande constanten, waarden voor COMMANDS, in je eigen JavaScript code

```
const CMD_ROOM_FULL = 0;

const CMD_START_GAME = 1;
const CMD_YOUR_PLAYER_NUM = 2;
const CMD_WAITING_FOR_PLAYER = 3;
const CMD_RESET_GAME = 4;

const CMD_PLAYER_MOVE = 5;

const CMD_NEW_ROUND = 6;
const CMD_IS_WIN = 7;
const CMD_IS_DRAW = 8;
const CMD_PLAYER_TURN = 9;
```

De constanten zijn **COMMANDS** voor de server en de client.

Receive betekent dat een client de COMMAND ontvangt, dus de server verstuurd deze.

Send betekent dat de client deze stuurt naar de server

In de volgende tabel vind je de toelichting op deze COMMANDS.

COMMANDS	SEND of RECEIVE	UITLEG
CMD_ROOM_FULL	*Receive*	Deze ontvang je van de server als er al twee spelers verbonden zijn. Er dus geen ruimte voor nog een speler.
CMD_START_GAME	*Receive*	Deze ontvang je van de server als er twee spelers verbonden zijn. Het geeft de spel client aan dat het spel kan beginnen. De spel client dient dan de interactie met de speler mogelijk te maken.
CMD_YOUR_PLAYER_NUM	*Receive*	De server bepaalt welk nummer een speler krijgt. Wanneer de spel client deze ontvangt wordt de spel client verteld welk nummer de speler heeft.
CMD_WAITING_FOR_PLAYER	*Receive*	Deze ontvangt de spel client van de server om aan te geven dat een tweede speler nog een verbinding moet maken.
CMD_RESET_GAME	*Nvt*	Deze wordt nog niet gebruikt.

COMMANDS	SEND of RECEIVE	UITLEG
CMD_PLAYER_MOVE	*Send*	Hiermee geeft een spel client aan dat de speler een cel in het speelveld heeft aangeklikt. In de body van de te versturen data wordt dan aangegeven welke speler (nummer) het is en op welke cel geklikt is.
CMD_PLAYER_MOVE	*Receive*	Alle spel clients krijgen de ontvangen CMD_PLAYER_MOVE doorgestuurd. Pas na ontvangst van deze constante zullen de spel clients de MOVE van een speler zichtbaar maken in het spelbord.
CMD_NEW_ROUND	*Receive*	Deze wordt verstuurd door de server als een nieuwe ronde is gestart. De spel client moet hier dan op reageren door b.v. het spelbord voor de nieuwe ronde leeg te maken en de juiste informatie te laten zien m.b.t. scores en round.
CMD_IS_WIN	*Receive*	Op de achtergrond controleert de server na elke MOVE of een van de twee spelers gewonnen heeft. Als dat zo is wordt deze COMMAND gestuurd naar de spel clients. De winnaar krijgt van de server 2 punten.
CMD_IS_DRAW	*Receive*	De server controleert na elke MOVE of er een gelijk spel is. Zolang er nog een cel in het spelbord leeg is geldt geen gelijk spel. Deze COMMAND wordt dus verstuurd wanneer alle cellen bezet zijn en er geen winnaar is. Beide spelers krijgen van de server 1 punt.
CMD_PLAYER_TURN	*Receive*	Op de server wordt na elke MOVE en bij begin van een ronde bepaald welke speler mag beginnen. Met deze COMMAND wordt de spel clients verteld welke speler aan de beurt is.

HET SPEELVELD

CEL 0 *Standaard waarde voor leeg*	CEL 1 *Standaard waarde voor leeg*	CEL 2 *Standaard waarde voor leeg*
CEL 3 *Standaard waarde voor leeg*	CEL 4 *Standaard waarde voor leeg*	CEL 5 *Standaard waarde voor leeg*
CEL 6 *Standaard waarde voor leeg*	CEL 7 *Standaard waarde voor leeg*	CEL 8 *Standaard waarde voor leeg*

De standaard waarde voor leeg op de server is 0. Bij begin van het spel of het begin van een ronde zijn alle cellen op de server gevuld met 0.

De server vult de cellen met de nummers van de spelers.

COMMUNICATIE TUSSEN SERVER EN CLIENT

Alle data die verstuurd wordt vanuit een spel client dient in JSON-format als een string gestuurd te worden.

VAN EEN JSON-OBJECT EEN JSON-STRING MAKEN IN JAVASCRIPT

VOORBEELD

Let op, dit is slechts een voorbeeld en niet de juiste manier om met de echte server te communiceren.

JavaScript

Stel we hebben de volgende JSON-object in JavaScript:

```
let data_structure = {
    command: 1,
    title: 'Dit is een titel',
    scores: [
        0,
        0
    ]
}
```

Middels de volgende JavaScript opdracht maken we hier een juiste string van (een zogenaamde JSON-string):

```
let json_string = JSON.stringify(data_structure);
```

De variabele, welke nu de string versie van het JSON-object bevat, kan nu worden verstuurd naar de server.

VAN EEN ONTVANGEN JSON-STRING EEN JSON-OBJECT MAKEN

```
let json_object = JSON.parse(ontvangen_data);
```

SPEL CLIENT COMMUNICATIE MET DE SERVER

SPELER DOET EEN ZET

De client stuurt bij elke click op een lege cel in het speelveld de CMD_PLAYER_MOVE command door naar de server. De client doet dit door de onderstaande JSON-object te voorzien van de juiste informatie:

JSON-Object:

```
{
        command: CMD_PLAYER_MOVE,
        cell: <nummer van de cel waarop geklikt is>,
        player_num: <nummer van de speler in deze client>
}
```

Van dit object moet dus eerst een JSON-string gemaakt worden met de JavaScript opdracht JSON.stringify(…)

De **CMD_PLAYER_MOVE** is de enige COMMAND die een client naar de server stuurt.

SERVER COMMUNICATIE MET DE SPEL CLIENT

Structuur van de data die de server verstuurd:

JSON-Object:

```
{
        state:   {
                 scores:  [ <score speler 1>, scrore_speler_2  ],
                 round: <ronde nummer>,
                 current_player: <nummer speler met beurt>
        },
        data:    {
                 command:        <CMD_.....>,
                 :                       Hier kan per command verschillende
                 :                       elementen staan, zie hieronder
                 message:        “Tekstbericht naar de clients”
        }
}
```

STATE

Hierin vinden we steeds de nieuwste staat van het spel, zoals de scores van de spelers, de ronde nummer en welke speler aan de beurt is.

DATA

In dit deel van de structuur zijn twee elementen altijd aanwezig, namelijk ***command*** en ***message***.

Daarnaast kan er per command extra informatie meegegeven worden, zie hieronder.

CMD_YOUR_PLAYER_NUM

Deelt aan de client die net een verbinding heeft gemaakt het nummer voor de speler mee.

JSON-Object:

```
{
        state:  {
                scores:  [ <score speler 1>, scrore_speler_2  ],
                round: <ronde nummer>,
                current_player: <nummer speler met beurt>
        },
        data:   {
                command:        <CMD_.....>,
                player_num:     <toegekende speler nummer>,
                message:        “Tekstbericht naar de clients”
        }
}
```

WAT DOET DE CLIENT NA ONTVANGST CMD_YOUR_PLAYER_NUM?

De client past lokaal de gegevens aan en laat de message zien aan de speler.

CMD_WAITING_FOR_PLAYER

Deelt de verbonden client mee dat een tweede speler nog niet verbonden is en dat hier op gewacht wordt.

JSON-Object:

```
{
        state:  {
                scores:  [ <score speler 1>, scrore_speler_2  ],
                round: <ronde nummer>,
                current_player: <nummer speler met beurt>
        },
        data:   {
                command:        <CMD_.....>,
                message:        “Tekstbericht naar de clients”
        }
}
```

WAT DOET DE CLIENT NA ONTVANGST CMD_WAITING_FOR_PLAYER?

De client laat de message zien aan de speler.

CMD_ROOM_FULL

Deelt aan een client mee, die net een verbinding heeft gemaakt, dat verbinden niet meer mogelijk is. Het maximaal aantal spelers en dus ook verbindingen is bereikt. De connectie wordt automatisch verbroken door de server. Dit is de enige COMMAND die geen state meestuurt.

JSON-Object:

```
{
        data:   {
                command:        <CMD_.....>,
                message:        “Tekstbericht naar de clients”
        }
}
```

WAT DOET DE CLIENT NA ONTVANGST VAN CMD_ROOM_FULL?

De client kan niets doen, behalve een nieuwe poging doen om een verbinding te maken.

CMD_START_GAME

Verteld beide clients dat het spel gestart is. Info over de ronde en de speler die aan de beurt is staat in de state.

JSON-Object:

```
{
        state:   {
                 scores:  [ <score speler 1>, scrore_speler_2  ],
                 round: <ronde nummer>,
                 current_player: <nummer speler met beurt>
        },
        data:    {
                 command:        <CMD_.....>,
                 message:        “Tekstbericht naar de clients”
        }
}
```

WAT DOET DE CLIENT NA ONTVANGST VAN CMD_START_GAME?

De client volgt de volgende stappen:

1. Leg de startende speler vast op basis van de info in de game state.
2. Laat de message zien aan de speler
3. Past de game info aan op basis van de info in de game state en laat dit zien aan de speler
4. Initialiseert het spel lokaal:
    a. Maakt het speelveld leeg
    b. Activeert de click events op elke cel

CMD_IS_WIN

Verteld de clients welke speler gewonnen heeft en dat de ronde is beëindigd.

JSON-Object:

```
{
        state:  {
                scores:  [ <score speler 1>, scrore_speler_2  ],
                round: <ronde nummer>,
                current_player: <nummer speler met beurt>
        },
        data:   {
                command:        <CMD_.....>,
                player_num:     <nummer winnende speler>,
                message:        “Tekstbericht naar de clients”
        }
}
```

WAT DOET DE CLIENT NA ONTVANGST VAN CMD_IS_WIN?

1. Reset de game
    a. Maakt het speelveld leeg
    b. Deactiveert de click events op de cellen van het speelveld

2. Laat de message zien aan de speler

CMD_IS_DRAW

Verteld de clients dat de ronde is beëindigd en dat het om een gelijk spel gaat.

JSON-Object:

```
{
        state:  {
                scores:  [ <score speler 1>, scrore_speler_2  ],
                round: <ronde nummer>,
                current_player: <nummer speler met beurt>
        },
        data:   {
                command:        <CMD_.....>,
                message:        “Tekstbericht naar de clients”
        }
}
```

WAT DOET DE CLIENT NA ONTVANGST CMD_IS_DRAW?

Zie CMD_IS_WIN

CMD_PLAYER_TURN

Verteld de clients welke speler aan de beurt is. Het nummer van de speler is te vinden in de state.

JSON-Object:

```
{
        state:  {
                scores:  [ <score speler 1>, scrore_speler_2  ],
                round: <ronde nummer>,
                current_player: <nummer speler met beurt>
        },
        data:   {
                command:        <CMD_.....>,
                message:        “Tekstbericht naar de clients”
        }
}
```

WAT DOET DE CLIENT NA ONTVANGST VAN CMD_PLAYER_TURN?

1. Legt lokaal vast welke speler aan de beurt is en laat dit zien aan de speler

2. Controleert of de speler van de client de speler is die aan de beurt is:
    a. Zo ja, activeer de click events op de beschikbare cellen in het speelveld.
    b. Zo nee, deactiveer de click events op de cellen in het speelveld.

3. Toont de actuele info, te vinden in de game state, aan de speler

CMD_NEW_ROUND

Verteld dat clients dat een nieuwe ronde is begonnen. Het nummer van de nieuwe ronde evenals het nummer van de speler die mag beginnen staat in de state.

JSON-Object:

```
{
        state:   {
                 scores:  [ <score speler 1>, scrore_speler_2  ],
                 round: <ronde nummer>,
                 current_player: <nummer speler met beurt>
        },
        data:    {
                 command:        <CMD_.....>,
                 message:        “Tekstbericht naar de clients”
        }
}
```

WAT DOET DE CLIENT NA ONTVANGST VAN CMD_NEW_ROUND?

1. Reset de game
    a. Maakt speelveld leeg
    b. Deactiveert click events op de cellen van het speelveld

2. Toont de actuele info, te vinden in de game state, aan de speler

3. Als de speler in de client de speler is die aan de beurt is dan activeren we de click events op de cellen in het speelveld.

CMD_PLAYER_MOVE

Verteld alle clients dat een speler een zet heeft gedaan op het spelbord.

JSON-Object:

```
{
        state:  {
                scores:  [ <score speler 1>, scrore_speler_2  ],
                round: <ronde nummer>,
                current_player: <nummer speler met beurt>
        },
        data:   {
                command:        <CMD_.....>,
                cell:           <nummer van de cel in het spelbord>,
                player_num:     <toegekende speler nummer>,
                message:        “Tekstbericht naar de clients”
        }
}
```

WAT DOET DE CLIENT NA ONTVANGST VAN CMD_PLAYER_MOVE?

Na een klik op een spel laat de client niet gelijk het symbool van de speler in het speelveld zien. De client wacht eerst op een confirmatie van de server. De confirmatie komt middels deze COMMAND.

1. Toont het symbool van de speler die een move heeft gedaan in de juiste cel
   De benodigde informatie staat in de game data.

2. Laat een message zien aan de speler (OPTIONEEL)

LINKS

Hieronder vind je een link naar en website met uitleg over hoe je in JavaScript web sockets kunt gebruiken.

https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

https://javascript.info/websocket