Othello Text-Based Game Documentation

คำนำ :

- Code นี้สร้างมาเพื่อทำให้สามารถเล่น othello ได้ใน terminal(text based) เขียนด้วย Javascript (othello คือ boardgame 8*8 ที่ให้ผู้เล่นทั้งสองฝ่ายผลัดกันวางหมากบนกระดานและหากหมากของฝ่ายตรงข้ามถูกล้อม หมากตัวนั้นจะถูกเปลี่ยนเป็นของฝั่งล้อม)

requirements :

- ต้องมี git และ nodejs ในเครื่อง
- Git
- Node.js

Download :

- เปิด terminal/cmd แล้วรันโค้ดต่อไปนี้

```
cd path ของ folder ที่ต้องการ clone งานนี้
git clone https://github.com/lemuruz/othello_text.git
cd othello_text
npm install
```

Run the game:

- เปิด terminal/cmd แล้วรันโค้ดต่อไปนี้

```
cd 'path ของ folder ที่ clone งานนี้ไว้' (เช่น /media/user/HDD)
node gamev2.js
```

Improvement from v1:

- เพิ่ม bot เข้ามาในเกม
- ตกแต่งกระดานกับหมากให้สวยและมองง่ายมากขึ้น
- จัดระเบียบ code ทำให้เข้าใจง่ายขึ้น

Code structure

- *Class GAME*: จัดการเกี่ยวกับการเริ่มเกม,การเลือกโหมด
  - selectMode()
  - play()
- *Class BOARD*: จัดการ ตัวเกมั้งหมด logic
  - printBoard()
  - switchPlayer()
  - isValidMove(coordinates)
  - placePiece(x, y)
  - hasValidMoves()
  - calculateScore()
  - gameOver()
- *Class PLAYER*: เก็บtoken ของผู้เล่น รับ input
  - playerInput()
- *Class BOT*: จัดการlogic ของ bot return ตำแหน่ง
  - bestMove(board)

Detailed

- Class GAME

```
class GAME{
    constructor(){
        this.board = new BOARD()
        this.selectMode()
    }
```

- module selectMode() ใช้หลังจากสร้าง class ทำการรับค่าจากผู้เล่นเพื่อเลือกว่าจะสู้กับคนหรือบ็อทถ้าเป็นบ็อทให้เลือกระดับของบ็อทด้วย จากนั้นเก็บค่าการเลือก(บ็อทหรือคน)ไว้ที่ attribute ของ player

```
selectMode(){
    while(true){
        let input = prompt('select game mode player vs player(pvp),player
vs bot(pvb) : ')
        if (input == 'pvp'){
            this.board.player2 = new PLAYER('🟢')
            this.play()
            break
        }
        else if(input == 'pvb'){

            while(true){

                input = prompt('select difficulty (lv1)(lv2) : ')
                if(input == 'lv1'){
                    this.board.player2 = new BOT('🔵','lv1')
                    break
                }else if (input == 'lv2'){
                    this.board.player2 = new BOT('🟢','lv2')
                    break
                }
                console.clear()
                console.log('invalid input')
            }
            this.play()
            break
        }else{
            console.clear()
            console.log('invalid input')
        }
    }

}
```

- module play() ใช้เพื่อเริมเล่นเกมทำการsetboardเริ่มต้นจากนั้นรันเกมในwhile loop ทำการprintboard จากนั้นใช้playerinput ในการขอinput จากplayer และทำการสลับturn และเช็คว่าหากplayerปัจจุบันไม่สามารถวางได้ให้โยนturnให้อีกฝั่งจนทั้งสองฝั่งไม่สามารถลงได้(ใช้ async ในการ delay การทำงานของบ็อท)

```
async play(){
    let playerone_hasmove = true;
    let playertwo_hasmove = true;
    this.board.board[3][3] = this.board.player1.playerPiece;
    this.board.board[3][4] = this.board.player2.playerPiece;
    this.board.board[4][3] = this.board.player2.playerPiece;
    this.board.board[4][4] = this.board.player1.playerPiece;
    let game_is_running = true
    while (game_is_running) {
        //swich player if currentplayer cant place
        let checkThisPlayerValidMove = this.board.hasValidMoves();
        if (checkThisPlayerValidMove == false){
            if (this.board.currentPlayer == this.board.player1)
{playerone_hasmove = false}
            else if(this.board.currentPlayer == this.board.player2)
{playertwo_hasmove = false}
            this.board.switchPlayer()
            if (this.board.hasValidMoves() == false){
                if (this.board.currentPlayer == this.board.player1)
{playerone_hasmove = false}
                else if(this.board.currentPlayer == this.board.player2)
{playertwo_hasmove = false}
            }
        }else if(checkThisPlayerValidMove == true){
            if (this.board.currentPlayer == this.board.player1)
{playerone_hasmove = true}
            else if(this.board.currentPlayer == this.board.player2)
{playertwo_hasmove = true}
        }
        this.board.printBoard();
        this.board.calculateScore();
        if (playerone_hasmove == false && playertwo_hasmove == false){
            this.board.gameOver(this.board.board)
            break
        }
        let input = null;
        if (this.board.currentPlayer instanceof PLAYER) {
            input = this.board.currentPlayer.playerInput();
        } else {
            console.log('Bot turn');
            await delay(1500);
            input = this.board.currentPlayer.bestMove(this.board.board);
        }
        if (this.board.isValidMove(input)) {
            this.board.switchPlayer();
        }}}}
```

- Class BOARD

```
class BOARD{
    constructor(){
        this.board = Array.from({ length: 8 }, () => Array.from({ length: 8
}, () => "▧"));
        this.player1 = new PLAYER('◯');
        this.player2 = null;
        this.currentPlayer = this.player1;}
```

- module printBoard() แสดงกระดานทุกครั้งที่เริ่มเกม และ มีผู้เล่นวางหมาก โดยการใช้ for loop เพื่อทำให้สามารถบรับแต่งกระดานได้ง่ายขึ้น

```
printBoard(){
    console.clear()
    console.log("  a b c d e f g h");
    console.log("  ================");
    for (let i = 0; i < 8; i++) {
        let row = "";
        for (let j = 0; j < 8; j++) {
            row += this.board[i][j] + "";
        }
        console.log(i + '|' + row.trim());}}
```

- module switchPlayer() : ทำการเปลี่ยน currentPlayer เป็นอีกฝั่งนึง

```
switchPlayer() {
    this.currentPlayer = this.currentPlayer === this.player1 ? this.player2
: this.player1;}
```

- module isValidMove(coordinates) : รับค่าตำแหน่งจากผู้เล่น แล้วทำการเช็คว่า ตำแหน่งนั้นๆสามารถวางได้หรือไม่

```
isValidMove(coordinates) {
    if (!coordinates) {
        return false;}
    let col = coordinates[1].charCodeAt(0) - 'a'.charCodeAt(0);
    if (this.board[coordinates[0]][col] === '◯') {
        this.placePiece(parseInt(coordinates[0]), col);
        return true;
    } else {
        console.log('This coordinate is invalid, please select "◯"');
        return false;}}
```

- module placePiece(x, y) : ถูกเรียกใช้โดย module isValidMove(coordinates) ,รับค่าเป็นตำแหน่ง แล้วทำการวางหมากในตำแหน่งที่รับมา จากนั้นทำการเช็ครอบตำแหน่งที่จะทำการวาง(8ทิศ)(เช็คจนสุดกระดานของแต่ละทิศ)หากเจอหมากฝั่งตัวเองอยู่ในทิศนั้นๆจะทำการกลับด้านหมากฝั่งตรงข้ามที่อยู่ระหว่างหมากฝ่ายเรา

```
placePiece(x, y) {
    let board = this.board;
    let opposite = this.currentPlayer === this.player1 ?
this.player2.playerPiece : this.player1.playerPiece;
    let currentPlayer = this.currentPlayer.playerPiece;
    let checkDirection = [[-1, 1], [0, 1], [1, 1], [1, 0], [1, -1], [0,
-1], [-1, -1], [-1, 0]];

    for (let direction of checkDirection) {
        let [dix, diy] = direction;
        flipDirection(x, y, dix, diy, opposite, currentPlayer);
    }
    this.board[x][y] = currentPlayer;


    function flipDirection(current_x, current_y, direction_x, direction_y,
opposite, currentplayerToken) {
        let positions_to_flip = [];
        let checking_x = current_x + direction_x;
        let checking_y = current_y + direction_y;

        while (checking_x >= 0 && checking_x < 8 && checking_y >= 0 &&
checking_y < 8) {
            if (board[checking_x][checking_y] === '▧' || board[checking_x]
[checking_y] === '◯') {
                break;
            } else if (board[checking_x][checking_y] === opposite) {
                positions_to_flip.push([checking_x, checking_y]);
            } else if (board[checking_x][checking_y] ===
currentplayerToken) {
                for (let pos of positions_to_flip) {
                    board[pos[0]][pos[1]] = currentplayerToken;
                }
                return;
            }
            checking_x += direction_x;
            checking_y += direction_y;
        }
    }
}
```

- module hasValidMoves() : ทำการเช็คว่าตำแหน่งไหนเป็นตำแหน่งที่สามารถวางได้บ้าง โดยการรันloopเช็คทุกช่องบนกระดานและทำการเช็ครอบตำแหน่งนั้นๆ(8ทิศ)(เช็คจนสุดกระดานของแต่ละทิศ)หากมีหมากฝ่ายเราอยู่ และ มีแต่หมากฝั่งตรงข้ามที่อยู่ระหว่างหมากฝ่ายเรา(ไม่มีช่องว่าง) กำหนดให้ตำแหน่งที่เช็คอยู่สามารถวางได้โดยการแทนที่ตำแหน่งบนกระดานด้วยวงกลมสีแดง และreturn true ถ้ามีตำปหน่งที่วางได้ และ false ถ้าไม่มี

```
    hasValidMoves(){
        let hasValidMoves_ = false
        let board = this.board;
        for (let i = 0; i < 8; i++) {
            for (let j = 0; j < 8; j++) {
                if (this.board[i][j] === '🟢') {
                    this.board[i][j] = '🟩';
                }
            }
        }
        let hasEmptySpace = false;
        let checkDirection = [[-1, 1], [0, 1], [1, 1], [1, 0], [1, -1], [0,
-1], [-1, -1], [-1, 0]];
        let opposite = this.currentPlayer === this.player1 ?
this.player2.playerPiece : this.player1.playerPiece;
        let currentPlayer = this.currentPlayer.playerPiece;
        for (let i = 0; i < 8; i++) {
            for (let j = 0; j < 8; j++) {
                if (this.board[i][j] === '🟩') {
                    let valid = false;
                    for (let direction of checkDirection) {
                        let [dix, diy] = direction;
                        if (checkIfThatDirectionValid(i, j, dix, diy,
opposite, currentPlayer)) {
                            valid = true;
                            hasValidMoves_ = true
                            break;
                        }
                    }
                    if (valid) {
                        hasEmptySpace = true;
                        this.board[i][j] = '🟢';

                    }
                }
            }
        }
```

- module calculateScore() นับจำนวนตัวหมากของผู้เล่นและแสดงค่า ใช้ระหว่างเกม

```
calculateScore(){
    let player1_score = 0
    let player2_score = 0
    for (let i = 0; i < 8; i++) {
        for (let j = 0; j < 8; j++) {
            if (this.board[i][j] == this.player1.playerPiece) {
                player1_score += 1;
            } else if (this.board[i][j] == this.player2.playerPiece) {
                player2_score += 1;
            }
        }}

console.log(this.player1.playerPiece,player1_score,this.player2.playerPiece
,player2_score)
}
```

- module WgameOver() : รับกระดานจากboardทำการเช็คคะแนนของแต่ละฝั่งและแสดงผลแพ้ชนะ จากนั้นจบการทำงานของเกม

```
gameOver(board){
        let player1_score = 0
        let player2_score = 0
        for (let i = 0; i < 8; i++) {
            for (let j = 0; j < 8; j++) {
                if (board[i][j] == this.player1.playerPiece) {
                    player1_score += 1;
                } else if (board[i][j] == this.player2.playerPiece) {
                    player2_score += 1;
                }
            }}
        this.printBoard()
        console.log('Game over');
        console.log('Player 1', this.player1.playerPiece, 'score:',
player1_score);
        console.log('Player 2', this.player2.playerPiece, 'score:',
player2_score);
        if (player1_score > player2_score) {
            console.log('Player 1 WINS!');
        } else if (player1_score < player2_score) {
            console.log('Player 2 WINS!');
        } else {
            console.log('TIE!');}
        return false
    }}
```

- Class PLAYER

```
class PLAYER {
    constructor(token) {
        this.playerPiece = token;
    }
```

- module playerInput() : รับค่าจากผู้เล่น,เช็คความถูกต้องของค่าที่รับมา หากไม่ถูกให้เรียกใช้ module นี้ซ้ำ จากนั้น return ค่าไปให้ BOARD

```
playerInput() {
        console.log('Player', this.playerPiece, 'turn');
        let input = prompt('Enter your move (e.g., 0a): ');
        if (input.length === 2 && !isNaN(input[0]) && input[0] >= '0' &&
input[0] <= '7' && input[1] >= 'a' && input[1] <= 'h') {
            return input;
        } else if (input === 'exit') {
            gameIsRunning = false;
            return false;
        } else {
            console.log('Invalid input');
            return this.playerInput();
        }
    }}
```

- Class BOT

```
class BOT{
    constructor(piece, difficulty) {
        this.botPiece = piece;
        this.difficult = difficulty;
        this.scoreBoard = [
            [100, -10, 10, 5, 5, 10, -10, 100],
            [-10, -20, 1, 1, 1, 1, -20, -10],
            [10, 1, 3, 3, 3, 3, 1, 10],
            [5, 1, 3, 1, 1, 3, 1, 5],
            [5, 1, 3, 1, 1, 3, 1, 5],
            [10, 1, 3, 3, 3, 3, 1, 10],
            [-10, -20, 1, 1, 1, 1, -20, -10],
            [100, -10, 10, 5, 5, 10, -10, 100]
        ];}
    //เพื่อให้ง่ายต่อการใช้ตัวหมากของบอท สามารถ bot.playerPiece ได้เหมือน player จึงไม่ต้อง
เขียนโค๊ดแยกสำหรับหมากของบอทโดยเฉพาะ
    get playerPiece(){
        return this.botPiece
    }
```

- module bestMove(board) : รับค่าจาก BOARD จากนั้นทำการคำนวนตาม logic ของ bot ที่ตั้งไว้จากนั้น return ตำแหน่งที่ได้ให้ BOARD

- bot lv1 จะเป็น bot แบบสุ่ม
- bot lv2 จะทำการเทียบตำแหน่งที่มีคะแนนมากกว่าในการวางและวางในตำแหน่งที่มีคะแนนสูงกว่า(ใน othello การวางหมากที่ขอบหรือมุดจะทำให้มีโมโอกาศถูกอีกฝ่ายกินหมากน้อยกว่า)

```
bestMove(board) {
    let placeableCoordinates = [];
    for (let i = 0; i < 8; i++) {
        for (let j = 0; j < 8; j++) {
            if (board[i][j] === '〇') {
                placeableCoordinates.push(i.toString() +
String.fromCharCode('a'.charCodeAt(0) + j));
            }
        }
    }

    switch (this.difficult) {
        case 'lv1':
            return placeableCoordinates.random();
        case 'lv2':
            let highestScoreCoordinates = placeableCoordinates[0];
            for (let i of placeableCoordinates) {
                if (this.scoreBoard[highestScoreCoordinates[0]]
[highestScoreCoordinates[1].charCodeAt(0) - 'a'.charCodeAt(0)] <
                    this.scoreBoard[i[0]][i[1].charCodeAt(0) -
'a'.charCodeAt(0)]) {
                    highestScoreCoordinates = i;
                }
            }
            return highestScoreCoordinates;
    }}}
```