```
// Necessary modules
const rl = require('readline-sync')   // Allows accepting user input and actually waiting for it.

// Colors for POSIX-compliant terminals
let green = '\033[0;32m'
let cyan = '\033[0;36m'
let red = '\033[0;31m'
let boldGreen = '\033[1;32m'
let boldCyan = '\033[1;36m'
let boldGray = '\033[1;37m'
let boldRed = '\033[1;31m'
let yellow = '\033[0;33m'
let reset = '\033[0m'
let clear = '\033[H\033[2J'

// Initializing required variables
let turn = `${cyan}X` // Switches around to control what gets placed.
let logTurn           // What to log for the "Turn: " label next to the board
let logTaken          // Whether or not to log the "Already taken" error
let logOOR            // Whether or not to log the "Not 1-9" error
let winner = 'none'   // Will change when there is a winner
let boardState = []   // Initializing array for the state of each chunk
let boardChoices = [1,2,3,4,5,6,7,8,9]  // Choices possible; with the "Not 1-9" error
let winConditions = []     // Initalizing conditions for how the letters must match in order for the game to be won


// Win Checks
let checkWins = () => {
  winner = 'none'   // Double checks that winner variable is set to right thing at first
  let drawCheck = 0
  for (let i = 0; i < winConditions.length; i++) {
    if (winConditions[i][0] !== 0) {
      if (winConditions[i][0] === winConditions[i][1] && winConditions[i][0] === winConditions[i][2] && winConditions[i][0] === 1) {
        winner = 'x'
      } else if (winConditions[i][0] === winConditions[i][1] && winConditions[i][0] === winConditions[i][2] && winConditions[i][0] === 2) {
        winner = 'o'
      } else if (winConditions[i].indexOf(0) === -1) {
        drawCheck += 1
      }
    }
  }
  if (drawCheck === 8) {
    winner = 'draw'
  }
}

let board = `
   ┌─────┐  ###          ###
   |     |  ###          ###
   |     |  ###          ###
   |     |  ###          ###
   └─────┘  ###          ###
##################################
##################################          1 |2 |3
           ###    \\    /  ###               ──┼──┼──
           ###     \\ /   ###               4 |5 |6
           ###      X    ###               ──┼──┼──
           ###     / \\   ###               7 |8 |9
           ###    /    \\  ###
##################################
##################################
           ###          ###
           ###          ###
           ###          ###
           ###          ###
           ###          ###
`
// Template board for numbers
let numBoard = `
1 |2 |3
──┼──┼──
4 |5 |6
──┼──┼──
7 |8 |9
`
```

readline-sync is not my own:
https://www.npmjs.com/package/readline-sync

```
 77
 78
 79 let topLeft = [
 80   `${boldGray}          #\n          #\n          #\n          #\n          #\n##########`,
 81   `${cyan}   \\   /  ${boldGray}#\n    ${cyan}\\ /   ${boldGray}#\n     ${cyan}X    ${boldGray}#\n    ${cyan}/
\\   ${boldGray}#\n   ${cyan}/   \\  ${boldGray}#\n${boldGray}##########`,
 82   `${green}  ┌─────┐  ${boldGray}#\n  ${green}|     |  ${boldGray}#\n  ${green}|     |  ${boldGray}#\n  ${green}|
|  ${boldGray}#\n  ${green}└─────┘  ${boldGray}#\n${boldGray}##########`]
 83 let topMid = [
 84   `${boldGray}##          ##\n##          ##\n##          ##\n##          ##\n##          ##\n#############`,
 85   `${boldGray}##   ${cyan}\\   /  ${boldGray}##\n##    ${cyan}\\ /   ${boldGray}##\n##     ${cyan}X
${boldGray}##\n##    ${cyan}/ \\   ${boldGray}##\n##   ${cyan}/   \\  ${boldGray}##\n#############`,
 86   `${boldGray}##  ${green}┌─────┐  ${boldGray}##\n##  ${green}|     |  ${boldGray}##\n##  ${green}|     |
${boldGray}##\n##  ${green}|     |  ${boldGray}##\n##  ${green}└─────┘  ${boldGray}##\n#############`]
 87 let topRight = [
 88   `${boldGray}#          \n#          \n#          \n#          \n#          \n##########`,
 89   `${boldGray}#   ${cyan}\\   /  ${boldGray}\n${boldGray}#    ${cyan}\\ /   ${boldGray}\n${boldGray}#
${cyan}X    ${boldGray}\n${boldGray}#    ${cyan}/ \\   ${boldGray}\n${boldGray}#   ${cyan}/   \\
${boldGray}\n${boldGray}##########`,
 90   `${boldGray}#  ${green}┌─────┐  ${boldGray}\n${boldGray}#  ${green}|     |  ${boldGray}\n${boldGray}#  ${green}|
|  ${boldGray}\n${boldGray}#  ${green}|     |  ${boldGray}\n${boldGray}#  ${green}└─────┘
${boldGray}\n${boldGray}##########`]
 91 let midLeft = [
 92   `${boldGray}##########\n          ${boldGray}#\n          ${boldGray}#\n          ${boldGray}#\n
${boldGray}#\n          ${boldGray}#\n${boldGray}##########`,
 93   `${boldGray}##########\n   ${cyan}\\   /  ${boldGray}#\n    ${cyan}\\ /   ${boldGray}#\n     ${cyan}X
${boldGray}#\n    ${cyan}/ \\   ${boldGray}#\n   ${cyan}/   \\  ${boldGray}#\n${boldGray}##########`,
 94   `${boldGray}##########\n  ${green}┌─────┐  ${boldGray}#\n  ${green}|     |  ${boldGray}#\n  ${green}|     |
${boldGray}#\n  ${green}|     |  ${boldGray}#\n  ${green}└─────┘  ${boldGray}#\n${boldGray}##########`]
 95 let midMid = [
 96   `${boldGray}#############\n##          ##\n##          ##\n##          ##\n##          ##\n
##\n#############`,
 97   `${boldGray}#############\n##   ${cyan}\\   /  ${boldGray}##\n##    ${cyan}\\ /   ${boldGray}##\n##
${cyan}X    ${boldGray}##\n##    ${cyan}/ \\   ${boldGray}##\n##   ${cyan}/   \\  ${boldGray}##\n#############`,
 98   `${boldGray}#############\n##  ${green}┌─────┐  ${boldGray}##\n##  ${green}|     |  ${boldGray}##\n##  ${green}
|     |  ${boldGray}##\n##  ${green}|     |  ${boldGray}##\n##  ${green}└─────┘  ${boldGray}##\n#############`]
 99 let midRight = [
100   `${boldGray}##########          ${yellow}1 |2 |3\n#                   ${yellow}─┼─┼─\n#
${yellow}4 |5 |6\n#                   ${yellow}─┼─┼─\n#                    ${yellow}7 |8 |9\n#
\n##########`,
101   `${boldGray}##########          ${yellow}1 |2 |3\n#   ${cyan}\\   /            ${yellow}
─┼─┼─\n${boldGray}#    ${cyan}\\ /             ${yellow}4 |5 |6\n${boldGray}#     ${cyan}X              ${yellow}
─┼─┼─\n${boldGray}#    ${cyan}/ \\             ${yellow}7 |8 |9\n${boldGray}#   ${cyan}/   \\
\n${boldGray}##########`,
102   `${boldGray}##########          ${yellow}1 |2 |3\n#  ${green}┌─────┐            ${yellow}
─┼─┼─\n${boldGray}#  ${green}|     |            ${yellow}4 |5 |6\n${boldGray}#  ${green}|     |
${yellow}─┼─┼─\n${boldGray}#  ${green}|     |            ${yellow}7 |8 |9\n${boldGray}#  ${green}└─────┘
\n${boldGray}##########`]
103 let botLeft = [
104   `${boldGray}##########\n          #\n          #\n          #\n          #\n          #`,
105   `${boldGray}##########\n   ${cyan}\\   /  ${boldGray}#\n    ${cyan}\\ /   ${boldGray}#\n     ${cyan}X
${boldGray}#\n    ${cyan}/ \\   ${boldGray}#\n   ${cyan}/   \\  ${boldGray}#`,
106   `${boldGray}##########\n  ${green}┌─────┐  ${boldGray}#\n  ${green}|     |  ${boldGray}#\n  ${green}|     |
${boldGray}#\n  ${green}|     |  ${boldGray}#\n  ${green}└─────┘  ${boldGray}#`]
107 let botMid = [
108   `${boldGray}#############\n##          ##\n##          ##\n##          ##\n##          ##\n##          ##`,
109   `${boldGray}#############\n##   ${cyan}\\   /  ${boldGray}##\n##    ${cyan}\\ /   ${boldGray}##\n##
${cyan}X    ${boldGray}##\n##    ${cyan}/ \\   ${boldGray}##\n##   ${cyan}/   \\  ${boldGray}##`,
110   `${boldGray}#############\n##  ${green}┌─────┐  ${boldGray}##\n##  ${green}|     |  ${boldGray}##\n##  ${green}
|     |  ${boldGray}##\n##  ${green}|     |  ${boldGray}##\n##  ${green}└─────┘  ${boldGray}##\n`]
111 let botRight = [
112   `${boldGray}##########\n#          \n#          \n#          \n#          \n#          `,
113   `${boldGray}##########\n#   ${cyan}\\   /  ${boldGray}\n${boldGray}#    ${cyan}\\ /
${boldGray}\n${boldGray}#     ${cyan}X    ${boldGray}\n${boldGray}#    ${cyan}/ \\   ${boldGray}\n${boldGray}#
${cyan}/   \\  `,
114   `${boldGray}##########\n#  ${green}┌─────┐  ${boldGray}\n${boldGray}#  ${green}|     |
${boldGray}\n${boldGray}#  ${green}|     |  ${boldGray}\n${boldGray}#  ${green}|     |  ${boldGray}\n${boldGray}#
${green}└─────┘  `]
115
116
117 // 0: blank, 1: 'x', 2: 'o'
118 let drawBoard = (boardArray) => {
119   tl = topLeft[boardArray[0]].split('\n')
120   tm = topMid[boardArray[1]].split('\n')
121   tr = topRight[boardArray[2]].split('\n')
122   ml = midLeft[boardArray[3]].split('\n')
123   mm = midMid[boardArray[4]].split('\n')
124   mr = midRight[boardArray[5]].split('\n')
```

```
  bl = botLeft[boardArray[6]].split('\n')
  bm = botMid[boardArray[7]].split('\n')
  br = botRight[boardArray[8]].split('\n')
  for (let i = 0; i < tl.length; i++) {
    console.log(tl[i] + tm[i] + tr[i])
  }
  for (let i = 0; i < ml.length; i++) {
    console.log(ml[i] + mm[i] + mr[i])
  }
  for (let i = 0; i < bl.length; i++) {
    if (i === 0) {
      logTurn = `          Turn: ${turn}${boldGray}`
    } else {
      logTurn = ''
    }
    console.log(bl[i] + bm[i] + br[i] + logTurn)
  }
}

// Main procedure
mode = 'local'  // Old thing for when I build in global functionality
if (mode === 'local') {
  boardState = [0,0,0,0,0,0,0,0,0]
  let alreadyTaken = []

  console.log(`${clear}${boldGray}`)
  drawBoard(boardState)

  while (true) {
    if (winner === 'none') {
      console.log(`${boldGray}\nWhat is your move?`)
      if (logTaken) {
        console.log(red + '\033[1BAlready taken! Try Again!\033[2A' + boldGray)
        logTaken = false
      }
      if (logOOR) {
        console.log(red + '\033[1BNot 1-9! Try Again!\033[2A' + boldGray)
        logOOR = false
      }
      let move = rl.question('> ')
      if (alreadyTaken.indexOf(Number(move)) > -1) {
        logTaken = true
      } else if ((boardChoices.indexOf(Number(move)) > -1) === false) {
        logOOR = true
      } else if (turn === `${cyan}X`) {
        boardState[Number(move)-1] = 1
        alreadyTaken.push(Number(move))
        turn = `${green}O`
        console.log(turn)
      } else if (turn === `${green}O`) {
        boardState[Number(move)-1] = 2
        alreadyTaken.push(Number(move))
        turn = `${cyan}X`
      }
      console.log(`${clear}${boldGray}`)
      drawBoard(boardState)
      winConditions = [
    [boardState[0],boardState[1],boardState[2]],
    [boardState[3],boardState[4],boardState[5]],
    [boardState[6],boardState[7],boardState[8]],
    [boardState[0],boardState[3],boardState[6]],
    [boardState[1],boardState[4],boardState[7]],
    [boardState[2],boardState[5],boardState[8]],
    [boardState[0],boardState[4],boardState[8]],
    [boardState[2],boardState[4],boardState[6]]
    ]
      checkWins()
    } else {
      console.log(`\n${yellow}GAME OVER!`)
      if (winner === 'draw') {
        console.log(`${boldRed}DRAW! ${boldGray}Nobody Wins!`)
      } else if (winner === 'x') {
        console.log(`${boldCyan}X ${boldGray}Wins!`)
      } else if (winner === `o`) {
        console.log(`${boldGreen}O ${boldGray}Wins!`)
      }
      break
    }
  }
}
```