Checkers

```
var SCREEN_WIDTH = 320;
var SCREEN_HEIGHT = 450;
var BOARD_SIZE = 280;

// Piece object. Positions are relative to the board.
function Piece(x, y, color)
{
  this.x = x;
  this.y = y;
  this.color = color;
  this.drawColor = color;
  this.king = false;
}
Piece.prototype.draw = function() {
  setStrokeColor("red");
  setFillColor(this.drawColor);
  circle(this.x * BOARD_SIZE / 8 + BOARD_SIZE / 16, this.y * BOARD_SIZE / 8 + BOARD_SIZE / 16, BOA
};
Piece.prototype.canMoveTo = function(x, y) {
  // Moves.
  if(Math.abs(x - this.x) == 1 &&
    ((y == (this.color == "red" ? this.y - 1 : this.y + 1)) || (Math.abs(y - this.y) == 1 && this.
    board[this.x + this.y * 8] && !board[x + y * 8])
    return true;
  // Captures.
  if(Math.abs(x - this.x) == 2 &&
    ((y == (this.color == "red" ? this.y - 2 : this.y + 2)) || (Math.abs(y - this.y) == 2 && this.
    !board[x + y * 8] && this.color != board[(x + this.x) / 2 + (y + this.y) / 2 * 8].color)
    return true;
  return false;
};
Piece.prototype.setKing = function() {
  this.king = true;
  this.drawColor = this.color == "red" ? "rgb(255, 128, 128)" : "white";
};
// Move object. Only one needs to be created because only one move happens at a time.
var move = {
  set: function(x, y) {
    if(!this.p) {
      if(board[x + y * 8] && board[x + y * 8].color == turn) {
        this.p = new Piece(x, y, board[x + y * 8].color);
        this.p.king = board[x + y * 8].king;
        this.p.drawColor = this.p.color == "red" ? "darkred" : "rgb(38, 38, 38)";
        this.p.xHistory = [x];
        this.p.yHistory = [y];

        setProperty("cancel", "hidden", false);
        setProperty("finalize", "hidden", false);
      }
      return;
    }
```

```
    if(this.p.canMoveTo(x, y))
    {
      this.p.x = x;
      this.p.y = y;
      this.p.xHistory.push(x);
      this.p.yHistory.push(y);
      clearCanvas();
      drawAll();
      this.p.draw();
    }
  },
  cancel: function() {
    this.p = null;
    clearCanvas();
    drawAll();
    setProperty("cancel", "hidden", true);
    setProperty("finalize", "hidden", true);
  },
  finalize: function() {
    for(var i = this.p.xHistory.length - 1; i >= 1; i--)
      if(Math.abs(this.p.xHistory[i] - this.p.xHistory[i - 1]) == 2)
      {
        var piecePosition = (this.p.xHistory[i] + this.p.xHistory[i - 1]) / 2 + (this.p.yHistory[i
        if(board[piecePosition].color == "red") red--;
        else black--;
        board[piecePosition] = null;
      }
    var pieceToMove = board[this.p.xHistory[0] + this.p.yHistory[0] * 8];
    var x = this.p.xHistory.pop();
    var y = this.p.yHistory.pop();
    board[x + y * 8] = pieceToMove;
    board[pieceToMove.x + pieceToMove.y * 8] = null;
    pieceToMove.x = x;
    pieceToMove.y = y;
    // Check for king.
    if((pieceToMove.color == "red" && y == 0) || (pieceToMove.color == "black" && y == 7))
      pieceToMove.setKing();
    changeTurn();
    this.cancel();
  }
};
setProperty("cancel", "hidden", true);
setProperty("finalize", "hidden", true);
onEvent("cancel", "click", function() {
  move.cancel();
});
onEvent("finalize", "click", function() {
  move.finalize();
});

// Create the board.
createCanvas("board", BOARD_SIZE, BOARD_SIZE);
setPosition("board", SCREEN_WIDTH / 2 - BOARD_SIZE / 2, SCREEN_HEIGHT / 8);
var board = [];

// Create the objects for the pieces.
```

```
var B_START = 1;
var B_STOP = 7;
for(var a = 0; a <= 2; a++)
{
  for(var b = B_START; b <= B_STOP; b += 2)
    board[b + a * 8] = new Piece(b, a, "black");
  B_START = B_START == 0 ? 1 : 0;
  B_STOP = B_STOP == 6 ? 7 : 6;
}
B_START = 0;
B_STOP = 6;
for(var a = 7; a >= 5; a--)
{
  for(var b = B_START; b <= B_STOP; b += 2)
    board[b + a * 8] = new Piece(b, a, "red");
  B_START = B_START == 0 ? 1 : 0;
  B_STOP = B_STOP == 6 ? 7 : 6;
}

drawAll();

var turn = "red";
var red = 12;
var black = 12;
//console.log(board[40].move(2, 4));
//board[19].move(2, 3);

onEvent("board", "click", function(event) {
  var x = Math.floor((event.x - getXPosition("board")) / BOARD_SIZE * 8);
  var y = Math.floor((event.y - getYPosition("board")) / BOARD_SIZE * 8);
  move.set(x, y);
});

function changeTurn()
{
  turn = turn == "red" ? "black" : "red";
  setText("turn", turn == "red" ? "Turn: Red" : "Turn: Black");
  // Check for winner.
  if(black == 0) setText("title", "Red wins!");
  if(red == 0) setText("title", "Black wins!");
}

function drawAll()
{
  setStrokeColor("black");
  var color = "red";
  setFillColor(color);
  for(var a = 0; a < 8; a++)
  {
    for(var b = 0; b < 8; b++)
    {
      rect(a * BOARD_SIZE / 8, b * BOARD_SIZE / 8, BOARD_SIZE / 8, BOARD_SIZE / 8);
      color = color == "black" ? "red" : "black";
      setFillColor(color);
    }
    color = color == "black" ? "red" : "black";
```

```
164       setFillColor(color);
165     }
166
167     for(var i = 0; i < board.length; i++)
168     {
169       if(!board[i]) continue;
170       board[i].draw();
171     }
172   }
```

PDF document made with CodePrint using Prism