Web Tech Assignment II
CSE1 94

Axel Geist (5315972), Severin Bratus (5483026)

January 2022

2 Client-side JavaScript

2.1 Plan of action

The only interactive element of the game screen is the board, broadly speaking. At a finer level of detail, a player can move their pieces when it is their turn to move (a player will be notified of this through the prompt above the board). A player makes a move by first clicking on the tile, where a piece of his color stands, and then clicking on another tile (these tile may be highlighted for more usability), indicating where the chosen piece should move. If a player first selects a piece, and then clicks on a tile not available for movement, the selection is cancelled (at the same time, another piece can be selected). If a piece is captured, the counter of pieces lost / won is updated on both sides. Two tiles of the last move are highlighted, as are checks (and hence checkmates)

On pawn promotion (after a move results in a pawn reaching the 8th rank), a pop-up window asks the user whether they want to replace the pawn with a knight, bishop, rook, or queen.

2.2 Object design patterns

The assignment manual permits using additional JavaScript libraries to check for valid moves. We chose kokopu, because it supports Chess960, and the Universal Chess Interface. To access these features of kokopu we use the `kokopu.Position` class (source), which represents a valid chess game state, i.e. the state of the 8x8 chessboard. Thus we do not implement the game state object and game board object ourselves.

We do not need a chess piece class, because we can move pieces (and check if a move for validity) via the `Position#IsMoveLegal` function (source), which returns `false` in case the move is illegal, and a function which returns a `MoveDescriptor`. That function is called without any arguments, except in the case of pawn promotion, when the chosen pawn replacement must be specified by the user. In kokopu, the internal representation of a chess piece type is simply a one-character string (e.g. `'b'` for a bishop)

Note that the `Position` class is implemented implemented with the prototype-based constructor design pattern, as all of its functions are assigned to `Position.proto`.

In `Session.js` we do use an object for representing a game session, with a prototype-based constructor. This object is used to coordinate messages between the client instances.

3.3 List of messages

- `GAME_STARTED` from server to clients, contains the initial configuration of the chess board (`schernaglCode`).
- `GAME_MOVE` from client-A to server or from server to client-B. Here client-A is the one who made the move (`to`, `from` coordinates).
- `PLAYER_TYPE` from server to client, set client color to black or white.
- `GAME_ABORTED` from server to client, notify that the other client has abandoned the session. This message may be sent even after the game is completed.

- `GAME_OVER` from client to server, includes game statistics (minutesPlayed, piecesCaptured). A client sends out this message immediately after having received the `GAME_ABORTED` message. The message results in the session (and hence the two accompanying websocket connections) being deleted.