P08 JavaScript 3

1
Finally, we want to implement a multiplayer mode for our game. This is done by providing a server api that returns JSON objects. All calls are done using HTTP-GET, some of them require url parameters. The main advantage of GET is that you can test them in a web browser. The following methods are provided:

/addGame: creates a new game on the server and waits for a player to join. Accepts the parameter *player1* to show the name of the person who added the game. Returns a unique *id* for this game.
/games: Returns an array of all games that are not joined. Each game has a number of properties, like *player1* and *id* (see *addGame*), as well as a timestamp *dateTime* that shows when the game has been added.
/joinGame: Joins an existing game. This method expects the parameters *id* (for the game) and *player2* (for the name of player2).
/game: Returns the state for the game with the provided id. The properties for the game are the same as in /games when the game is not joined. If the game is joined, there is the additional property from /joinGame (player2) as well as the property *moves* that contains a string with the column numbers of each move (see below).
/nextMove: Each move from a player must call this function. The provided parameters are *id* (for the game) and *move* (containing the column of the user has chosen).

With this information, try to think how the setup for the multiplayer mode should work. Draw a small sketch.

2
Try to implement the multiplayer mode in your game. If the multiplayer mode should work, we need to extend our popup from P07/2. Add new popups that ask if the user would like to play a single or multiplayer game. In case of a multiplayer game, the user should be able to create a new one (master) or join an existing one (slave).