[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 0 - Getting Started](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Getting Started](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Blockchain 101](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Blockchain History](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Public and Managed Blockchains](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Consensus in Distributed Networks](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Cryptography](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Self-Assessment Quiz](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Go Introduction - First Steps](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Go Basics](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Go Interfaces](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Control Structures in Go](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Arrays and Slices in Go](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Standard Packages in Go](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Concurrency in Go](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Docker Introduction](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Introduction to the Interchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[The Interchain Ecosystem](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Getting ATOM and Staking It](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[A Blockchain App Architecture](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Accounts](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Transactions](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Messages](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Modules](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Protobuf](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Multistore and Keepers](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[BaseApp](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Queries](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Events](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Context](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Testing](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Relaying with IBC](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Interchain Security](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Bridges](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Migrations](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 1 Quiz](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 2 - First Steps](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[First Steps](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Setup Your Work Environment](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Run a Node, API, and CLI](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Ignite CLI](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Store Object](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Create Custom Messages](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Create and Save a Game Properly](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Add a Way to Make a Move](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Emit Game Information](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Record the Game Winner](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 2 Exercise](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[What is IBC?](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[IBC Token Transfer](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[IBC Tooling](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[What is CosmJS?](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Your First CosmJS Actions](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Compose Complex Transactions](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Learn to Integrate Keplr](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Put Your Games in Order](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Auto-Expiring Games](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Let Players Set a Wager](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Handle wager payments](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Integration tests](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Incentivize Players](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Help Find a Correct Move](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Understand IBC Denoms](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Go Relayer](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Hermes Relayer](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[CosmJS Advanced](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Create Custom Objects](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Create Custom Messages](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Get an External GUI](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Backend Script for Game Indexing](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[IBC Deep Dive](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[IBC Application Developer Introduction](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Create a Leaderboard Chain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Run in Production](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Prepare the Software to Run](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Prepare a Validator and Keys](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Prepare Where the Node Starts](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Prepare and Do Migrations](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Simulate Production in Docker](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Tally Player Info After Production](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Simulate a Migration in Docker](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Final Exam](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[What's Next?](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

[Continue Your Interchain Journey](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

Docs Version Switcher

On this page

[Prepare the integration with the checkers blockchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#prepare-the-integration-with-the-checkers-blockchain)

[Prepare Webpack](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#prepare-webpack)

[GUI data structures](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#gui-data-structures)

[Board converter](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#board-converter)

[Game converter](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#game-converter)

[Obtain a client](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#obtain-a-client)

[Pass RPC parameters](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#pass-rpc-parameters)

[Create the StargateClient](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#create-the-stargateclient)

[Show all games](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#show-all-games)

[All games container](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#all-games-container)

[A specific GUI method](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#a-specific-gui-method)

[All games call](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#all-games-call)

[Individual game menu container](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#individual-game-menu-container)

[Show one game](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#show-one-game)

[Integrate with Keplr](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#integrate-with-keplr)

[Identify the checkers blockchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#identify-the-checkers-blockchain)

[Prepare a signing client](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#prepare-a-signing-client)

[Create a new game](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#create-a-new-game)

[Play a move](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#play-a-move)

[Further exercise ideas](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#further-exercise-ideas)

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#integrate-cosmjs-and-keplr) **Integrate CosmJS and Keplr**



Make sure you have all you need before proceeding:

* You understand the concepts of [CosmJS](https://ida.interchain.io/tutorials/7-cosmjs/1-cosmjs-intro.html).
* You have the checkers CosmJS codebase up to the external GUI. If not, follow the [previous steps](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/3-external-gui.html) or go ahead and clone and checkout [this branch (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/tree/unwired-gui) to get the version needed for this tutorial.

In the previous sections:

1. You created the [objects](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/1-cosmjs-objects.html), and the [messages and clients](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/2-cosmjs-messages.html) that allow you to **interface** any GUI with your checkers blockchain.
2. You learned how to [start a running checkers blockchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/2-cosmjs-messages.html#prepare-your-checkers-chain), in Docker or on your local computer.
3. You imported an external checkers **GUI** to use.

Now, you must **integrate the two** together:

* Adjust the React app to be able to package CosmJS.
* Work on the CosmJS integration.

For the CosmJS integration, you will:

* Work with the GUI's data structures.
* Fetch all games from the blockchain and display them (without pagination).
* Integrate with Keplr for browser-based players.
* Create a new game.
* Fetch a single game to be played.
* Play on the game, making single moves and double moves.

This exercise assumes that:

1. You are running your [checkers blockchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/2-cosmjs-messages.html#prepare-your-checkers-chain) with:



Copy

$ docker network create checkers-net

$ docker run --rm -it \

-p 26657:26657 \

-p 1317:1317 \

--name checkers \

--network checkers-net \

--detach \

checkersd\_i:standalone start

$ sleep 10

$ docker run --rm -it \

-p 4500:4500 \

--name cosmos-faucet \

--network checkers-net \

--detach \

cosmos-faucet\_i:0.28.11 start http://checkers:26657

$ sleep 20

1. And have the following in .env:



Copy

RPC\_URL="http://localhost:26657"

FAUCET\_URL="http://localhost:4500"

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#prepare-the-integration-with-the-checkers-blockchain) Prepare the integration with the checkers blockchain

As always, when moving code to a browser, adjustments are needed.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#prepare-webpack) Prepare Webpack

Your GUI uses React v18, which uses Webpack v5. Therefore you need to [adjust Webpack's configuration (opens new window)↗](https://github.com/cosmos/cosmjs/blob/v0.28.11/README.md#webpack-configs) to handle some elements (see also the CosmJS documentation). To modify the Webpack configuration in a non-ejected React app, use [react-app-rewired (opens new window)↗](https://www.npmjs.com/package/react-app-rewired) as [explained here (opens new window)↗](https://stackoverflow.com/questions/63280109/how-to-update-webpack-config-for-a-react-project-created-using-create-react-app):

1. Install the new package:

**Local**

**Docker**



Copy

$ npm install react-app-rewired@2.2.1 --save-dev --save-exact

Copy

$ docker run --rm -it \

-v $(pwd):/client \

-w /client \

node:18.7-slim \

npm install react-app-rewired@2.2.1 --save-dev --save-exact

1. Add a new config-overrides.js:



Copy

require("dotenv").config()

const webpack = require("webpack")

module.exports = function override(config, env) {

config.plugins.push(

new webpack.ProvidePlugin({

Buffer: ["buffer", "Buffer"],

}),

new webpack.EnvironmentPlugin(["RPC\_URL"]),

)

config.resolve.fallback = {

buffer: false,

crypto: false,

events: false,

path: false,

stream: false,

string\_decoder: false,

}

return config

}

config-overrides.js

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/config-overrides.js" \l "L1-L21" \t "_blank)



You can also pass along the RPC\_URL as an environment variable, as seen in the code block above.

1. Change the run targets to use react-app-rewired in package.json:



Copy

{

...

"scripts": {

...

- "start": "react-scripts start",

- "build": "react-scripts build",

- "test-react": "react-scripts test --env=jsdom",

+ "start": "react-app-rewired start",

+ "build": "react-app-rewired build",

+ "test-react": "react-app-rewired test",

+ "eject": "react-scripts eject"

},

...

}

package.json

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/package.json" \l "L8-L11" \t "_blank)

Be careful **not** to put react-app-rewired in the "eject" command.

See a [previous section](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/1-cosmjs-objects.html) for how to set RPC\_URL in process.env.RPC\_URL. It also assumes that you have an RPC endpoint that runs the checkers blockchain, as explained in the previous section.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#gui-data-structures) GUI data structures

The checkers GUI uses different data structures, which you must understand to convert them correctly and with less effort.

1. The IPlayerInfo has a [name: string (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/sharedTypes.ts#L27) field which can be used as the player's address.
2. The IGameInfo has a [board: number[][] | null (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/sharedTypes.ts#L11) field. You must do a conversion from b\*b\*... to this type. And at some point, you must ensure that the alignments are correct.
3. The IGameInfo has a [turn: number (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/sharedTypes.ts#L17), which can be mapped to "b" or "r".
4. The IGameInfo lacks an index or id field. Instead the GUI developer identified games by their index in an array, which is not adequate for this case. You must add an index field to IGameInfo. This is saved as a string in the blockchain but in practice is a **number**, which the GUI code expects as game index. Add:



Copy

export interface IGameInfo {

...

+ index: number

}

src /

sharedTypes.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/sharedTypes.ts" \l "L18" \t "_blank)

To avoid compilation errors on index: elsewhere, temporarily add index?: number. Remember to come back to it and remove the ?.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#board-converter) Board converter

In order to correctly convert a *blockchain* board into a *GUI* board, you want to see how a GUI board is coded. In components/Game/Board/Board.tsx, add a [console.log(props.squares) (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/Board/Board.tsx#L18). Let React recompile, open the browser console, and create a game. You should see printed:



Copy

[

[0, 1, 0, 1, 0, 1, 0, 1],

[1, 0, 1, ... ],

...,

[... , 2, 0, 2],

[2, 0, 2, 0, 2, 0, 2, 0]

]

Where 1 represents player 1, i.e. black pieces. Compare that to how a game is saved in the blockchain:



Copy

"\*b\*b\*b\*b|b\*b\*b\*b\*|\*b\*b\*b\*b|\*\*\*\*\*\*\*\*|\*\*\*\*\*\*\*\*|r\*r\*r\*r\*|\*r\*r\*r\*r|r\*r\*r\*r\*"

To convert the board, create a new file src/types/checkers/board.ts and code as follows:



Copy

const rowSeparator = "|"

export const pieceTranslator = {

"\*": 0,

b: 1,

r: 2,

}

export const playerReverseTranslator: Player[] = ["b", "r"];

export const pieceReverseTranslator = ["\*", ...playerReverseTranslator];

export function serializedToBoard(serialized: string): number[][] {

return serialized

.split(rowSeparator)

.map((row: string) => row.split("").map((char: string) => (pieceTranslator as any)[char]))

}

src /

types /

checkers /

board.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/board.ts" \l "L5-L18" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#game-converter) Game converter

Next you convert a StoredGame into the IGameInfo of the GUI:



Copy

export function storedToGameInfo(game: StoredGame): IGameInfo {

return {

board: serializedToBoard(game.board),

created: new Date(Date.now()),

isNewGame: game.moveCount.equals(0),

last: new Date(Date.parse(game.deadline) - 86400 \* 1000),

p1: {

name: game.black, // Addresses are used instead of names.

is\_ai: false, // To make it simple.

score: 0,

},

p2: {

name: game.red,

is\_ai: false,

score: 0,

},

turn: (pieceTranslator as any)[game.turn],

index: parseInt(game.index),

};

}

export function storedsToGameInfos(games: StoredGame[]): IGameInfo[] {

return games.map(storedToGameInfo);

}

src /

types /

checkers /

board.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/board.ts" \l "L20-L43" \t "_blank)



Here:

* You use Cosmos addresses instead of names.
* You put today in the creation date because it is not stored in StoredGame.
* You set the last played date to deadline minus expiry duration (an adequate solution).
* The possibility of "AI" is not important.



Could an AI player and the blockchain mix together? If *the AI* has access to a private key that lets it send transactions, it would be a bona fide player. This could be implemented with backend scripts running on a server. See [checkers backend scripts](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html) for an example of backend scripts for a different use-case.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#obtain-a-client) Obtain a client

For your React components to communicate with the blockchain they need to have access to a StargateClient, and at some point to a SigningStargateClient too. Among the information they need is the RPC URL.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#pass-rpc-parameters) Pass RPC parameters

The RPC URL is already saved into process.env.RPC\_URL thanks to react-app-rewired. However, it is better to reduce your components' reliance on process.env. A simple way to give the RPC URL to the components is for them to:

* Receive an rpcUrl: string in the properties.
* Keep a StargateClient in the component's state that would be instantiated lazily.

With this setup, only index.tsx would use process.env.RPC\_URL.

For instance, prepare access to rpcUrl for MenuContainer.tsx by adding it to the following:

1. The properties of src/components/Menu/MenuContainer.tsx:



Copy

export interface IMenuContainerProps {

...

+ rpcUrl: string

}

src /

components /

Menu /

MenuContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/MenuContainer.tsx" \l "L14" \t "_blank)

1. The properties of the component call stack, first in src/components/App.tsx:



Copy

+ export interface AppProps {

+ rpcUrl: string;

+ }

- const App = () => {

+ const App = ({ rpcUrl }: AppProps) => {

...

<Routes>

- <Route path="menu" element={<MenuContainer location={""} />} />

+ <Route path="menu" element={<MenuContainer location={""} rpcUrl={rpcUrl} />} />

...

</Routes>

...

}

src /

components /

App.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/App.tsx" \l "L38-L50" \t "_blank)



You have to create the *missing* AppProps entirely.

1. Finally, to src/index.tsx:



Copy

+ import \_ from "../environment"

root.render(

<React.StrictMode>

<BrowserRouter>

- <App />

+ <App rpcUrl={process.env.RPC\_URL} />

</BrowserRouter>

</React.StrictMode>,

)

src /

index.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/index.tsx" \l "L21" \t "_blank)



If your compiler still believes that .RPC\_URL is string | undefined, you may append an ! to force it to string.

Whenever another component needs the rpcUrl in this tutorial, adapt and reproduce these steps as necessary.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#create-the-stargateclient) Create the StargateClient

Whenever you need a StargateClient in a component, you can repeat the following chain of actions for the component in question. For instance, to prepare the client in the state of src/components/Menu/MenuContainer.tsx:

1. Add the field in the state:



Copy

interface IMenuContainerState {

...

+ client: CheckersStargateClient | undefined

}

src /

components /

Menu /

MenuContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/MenuContainer.tsx" \l "L21" \t "_blank)

1. Initialize it to undefined in the constructor:



Copy

constructor(props: IMenuContainerProps) {

...

this.state = {

...

+ client: undefined,

};

...

}

src /

components /

Menu /

MenuContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/MenuContainer.tsx" \l "L31" \t "_blank)

1. Add a method that implements the conditional instantiation:



Copy

protected async getStargateClient(): Promise<CheckersStargateClient> {

const client: CheckersStargateClient =

this.state.client ?? (await CheckersStargateClient.connect(this.props.rpcUrl))

if (!this.state.client) this.setState({ client: client })

return client

}

src /

components /

Menu /

MenuContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/MenuContainer.tsx" \l "L47-L52" \t "_blank)



This creates it only once by saving the client in the state.

Instantiating a CheckersSigningStargateClient is more involved, as you will see later on.

First, make use of what you already prepared. Why choose to start with MenuContainer? Because that is where all the games are listed.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#show-all-games) Show all games

To show all games, you only need the read-only StargateClient. There is no need to worry about private keys for now. Where do you query for the games?

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#all-games-container) All games container

Look into src/components/Menu/MenuContainer.tsx for the componentDidMount method, which accesses the browser storage for saved games. You can replace the storage logic with the following steps:

* Obtain a StargateClient with the getStargateClient method.
* Obtain the blockchain games (without pagination).
* Convert the blockchain games to the format the component expects.
* Save them in saved: IGameInfo[] of the component's state.

Before adding lines of code in the component, a good next step is to add a new function to CheckersStargateClient to handle the call and the conversion, or to add a helper function that takes a client as a parameter.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#a-specific-gui-method) A specific GUI method

In the future you may want to reuse the CheckersStargateClient code in another GUI or for backend scripts. To avoid polluting the code, and to avoid a less-elegant helper where you need to pass a Stargate client as a parameter, add [an extension method (opens new window)↗](https://www.c-sharpcorner.com/article/learn-about-extension-methods-in-typescript/#:~:text=Extension%2Dmethod%20gives%20you%20the,any%20data%2Dtype%20you%20want) to CheckersStargateClient.

In a new src/types/checkers/extensions-gui.ts:

1. Declare the new extension method that extends your CheckersStargateClient *module*:



Copy

declare module "../../checkers\_stargateclient" {

interface CheckersStargateClient {

getGuiGames(): Promise<IGameInfo[]>

}

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L14-L16" \t "_blank)

1. Add its implementation:



Copy

CheckersStargateClient.prototype.getGuiGames = async function (): Promise<IGameInfo[]> {

return (

await this.checkersQueryClient!.checkers.getAllStoredGames(

Uint8Array.from([]),

Long.ZERO,

Long.fromNumber(20),

true,

)

).storedGames.map(storedToGameInfo)

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L33-L42" \t "_blank)



Note this does not care about pagination or games beyond the first 20. This purely-GUI detail is left as an exercise.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#all-games-call) All games call

Next, in MenuContainer.tsx:

1. Add an empty import for the extension method file, otherwise the method is not known nor compiled:



Copy

import {} from "../../types/checkers/extensions-gui"

src /

components /

Menu /

MenuContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/MenuContainer.tsx" \l "L4" \t "_blank)

1. Change componentDidMount to async and replace the storage code with a call to the new method:



Copy

- public componentDidMount(): void {

+ public async componentDidMount(): Promise<void> {

const queries = QueryString.parse(this.props.location.search)

if (queries.newGame) {

this.setState({ showModal: true })

}

- if (typeof Storage === "undefined") {

- ...

- }

- // gameToLoad = null;

this.setState({

- saved: Lockr.get("saved\_games") || [],

+ saved: await (await this.getStargateClient()).getGuiGames(),

})

}

src /

components /

Menu /

MenuContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/MenuContainer.tsx" \l "L38-L46" \t "_blank)

Restart npm start and you should see the list of games in your running chain.



Remember that this exercise assumes that you are running a checkers chain as described at the beginning of this section.

If your list is empty, you can quickly create a game with:



Copy

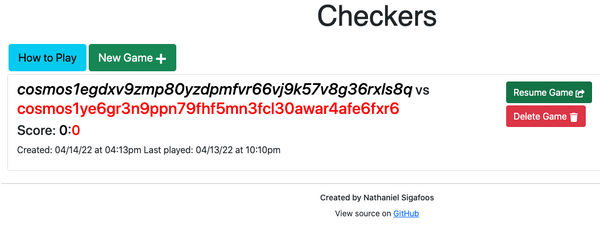
$ docker exec -it checkers sh -c \

"checkersd tx checkers \

create-game \$ALICE \$ALICE 0 stake \

--from alice --keyring-backend test \

--yes"



[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#individual-game-menu-container) Individual game menu container

Each game is now listed as a [src/components/Menu/MenuItem.tsx (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/MenuItem.tsx). However, in src/components/Menu/Menu.tsx the index of the game is taken from its *index* in the games array, which is not what you want here. The list needs to use the proper index of each game. In Menu.tsx make this change:



Copy

const Menu = (props: IMenuProps) => {

const menuItems = props.games.map((game, index) => (

- <MenuItem ... index={index} key={"game" + index} />

+ <MenuItem ... index={game.index} key={"game" + game.index} />

))

...

}

src /

components /

Menu /

Menu.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/Menu.tsx" \l "L21-L23" \t "_blank)

If the compiler complains, temporarily add ! (as in game.index!) if you previously added the IGameInfo.index as index?: number.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#show-one-game) Show one game

What happens when you click on a game's Resume Game? It opens a page of the form http://localhost:3000/play/1. The component is src/components/Game/GameContainer.tsx, from which you have to pick the game information from the blockchain. This process begins as with MenuContainer:

* Pass a [rpcUrl (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx#L19) along the chain to [GameContainerWrapper (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/App.tsx#L51) then the [GameContainer (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/App.tsx#L21-L27).
* Add a [client: CheckersStargateClient | undefined (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx#L39) state field, [initialize it (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx#L62), and add a [lazy instantiation method (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx#L116-L121).

Looking into GameContainer, you see that componentDidMount gets all the games from storage then sets the state with the information. Instead, as you did for all games, you have to take the game from the blockchain.

1. In extensions-gui.tsx declare another extension method to CheckersStargateClient dedicated to getting the game as expected by the GUI:



Copy

declare module "../../checkers\_stargateclient" {

interface CheckersStargateClient {

...

+ getGuiGame(index: string): Promise<IGameInfo | undefined>

}

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L17" \t "_blank)

Now define getGuiGame:



Copy

CheckersStargateClient.prototype.getGuiGame = async function (index: string): Promise<IGameInfo | undefined> {

const storedGame: StoredGame | undefined = await this.checkersQueryClient!.checkers.getStoredGame(index)

if (!storedGame) return undefined

return storedToGameInfo(storedGame)

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L44-L48" \t "_blank)

1. In GameContainer.tsx add an empty import to the extension methods:



Copy

import {} from "../../types/checkers/extensions-gui"

src /

components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx" \l "L9" \t "_blank)

1. Make the componentDidMount method async:



Copy

public async componentDidMount(): Promise<void>

src /

components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx" \l "L73" \t "_blank)

1. Split the componentDidMount method, and move the game loading lines to their own loadGame function:



Copy

- public componentDidMount(): void {

+ public async componentDidMount(): Promise<void> {

// Allow a player to make a move by double-clicking the screen.

// This is mainly for touchscreen users.

window.addEventListener("dblclick", this.makeMove)

+ await this.loadGame()

+ }

+ public async loadGame(): Promise<void> {

const savedGames: IGameInfo[] = Lockr.get("saved\_games") || []

...

}

src /

components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx" \l "L76-L79" \t "_blank)

1. In loadGame replace the storage actions with a fetch from the blockchain:



Copy

public async loadGame(): Promise<void> {

- const savedGames: IGameInfo[] = Lockr.get("saved\_games") || []

- ...

- if (index === 0 && savedGames.length === 0) {

+ const game: IGameInfo | undefined = await (

+ await this.getStargateClient()

+ ).getGuiGame(this.props.index)

+ if (!game) {

+ alert("Game does not exist")

return

}

- const game: IGameInfo = savedGames[index]

this.setState({

...

})

...

}

src /

components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx" \l "L80-L86" \t "_blank)

1. Adjust the setState call so that isSaved is always true:



Copy

public async loadGame(): Promise<void> {

...

this.setState({

...

- isSaved: !querys.newGame,

+ isSaved: true,

...

});

...

}

src /

components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx" \l "L91" \t "_blank)



Here:

* + You force isSaved to true since it is always saved in the blockchain.
  + Saving game.index to state is unimportant because it is actually passed in IGameContainerProps.index.
  + By having a separate loadGame, you can call it again if you know the game has changed.
  + It is important that this component fetches the game from the blockchain on its own, because the game page http://.../play/1 could be opened out of nowhere and may not have access to the list of games. Moreover, it may be entirely missing from the list.

Restart npm start and you should now see your game.

To quickly make a move on your created game with an id of 1, you can run:



Copy

$ docker exec -it checkers \

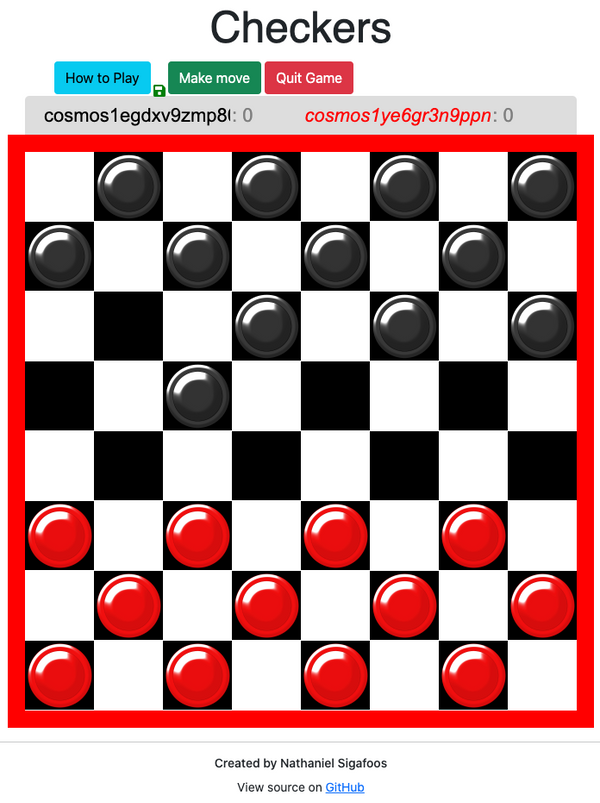
checkersd tx checkers \

play-move 1 1 2 2 3 \

--from alice --keyring-backend test \

--yes

Refresh the page to confirm the change.



[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#integrate-with-keplr) Integrate with Keplr

So far you have *only* made it possible to show the state of games and of the blockchain. This allows your users to poke around without unnecessarily asking them to connect their wallet and thereby disclose their address. However, to create a game or play in one, users need to make transactions. This is where you need to make integration with the Keplr wallet possible.

Install the necessary packages:

**Local**

**Docker**



Copy

$ npm install @keplr-wallet/types@0.10.17 --save-exact --save-dev

Copy

$ docker run --rm -it \

-v $(pwd):/client \

-w /client \

node:18.7-slim \

npm install @keplr-wallet/types@0.10.17 --save-exact --save-dev

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#identify-the-checkers-blockchain) Identify the checkers blockchain

Keplr will need information to differentiate your checkers blockchain from the other chains. You are also going to inform it about your REST API.

1. Adjust your .env file:



Copy

RPC\_URL="http://localhost:26657"

+ REST\_URL="http://localhost:1317"

...

.env

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/.env" \l "L2" \t "_blank)

1. Adjust the associated type declaration:



Copy

RPC\_URL: string

+ REST\_URL: string

...

environment.d.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/environment.d.ts" \l "L5" \t "_blank)

1. Adjust its passing through Webpack:



Copy

new webpack.EnvironmentPlugin(["RPC\_URL"]),

+ new webpack.EnvironmentPlugin(["REST\_URL"]),

...

config-overrides.js

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/config-overrides.js" \l "L10" \t "_blank)

With this, prepare your checkers blockchain info in a new src/types/checkers/chain.ts file:



Copy

import { ChainInfo } from "@keplr-wallet/types"

import \_ from "../../../environment"

export const checkersChainId = "checkers-1"

export const getCheckersChainInfo = (): ChainInfo => ({

chainId: checkersChainId,

chainName: "checkers",

rpc: process.env.RPC\_URL!,

rest: process.env.REST\_URL!,

bip44: {

coinType: 118,

},

bech32Config: {

bech32PrefixAccAddr: "cosmos",

bech32PrefixAccPub: "cosmos" + "pub",

bech32PrefixValAddr: "cosmos" + "valoper",

bech32PrefixValPub: "cosmos" + "valoperpub",

bech32PrefixConsAddr: "cosmos" + "valcons",

bech32PrefixConsPub: "cosmos" + "valconspub",

},

currencies: [

{

coinDenom: "STAKE",

coinMinimalDenom: "stake",

coinDecimals: 0,

coinGeckoId: "stake",

},

{

coinDenom: "TOKEN",

coinMinimalDenom: "token",

coinDecimals: 0,

},

],

feeCurrencies: [

{

coinDenom: "STAKE",

coinMinimalDenom: "stake",

coinDecimals: 0,

coinGeckoId: "stake",

gasPriceStep: {

low: 1,

average: 1,

high: 1,

},

},

],

stakeCurrency: {

coinDenom: "STAKE",

coinMinimalDenom: "stake",

coinDecimals: 0,

coinGeckoId: "stake",

},

coinType: 118,

features: [],

})

Expand



src /

types /

checkers /

chain.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/chain.ts" \t "_blank)

Note the use of process.env.RPC\_URL again.



The chainId value has to **match exactly** that returned by client.getChainId(), or the transaction signer will balk. The ChainInfo object is copied from the one you used for Theta in the [first steps with Keplr](https://ida.interchain.io/tutorials/7-cosmjs/4-with-keplr.html) section.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#prepare-a-signing-client) Prepare a signing client

Just as components that need a CheckersStargateClient keep one in their state, components that need a SigningCheckersStargateClient keep one in their state too. Some components may have both.



A component may have both CheckersStargateClient and SigningCheckersStargateClient. That is not a problem. In fact it may benefit your project, because with a simple CheckersStargateClient you can let your users poke around first, *and build their trust*, before asking them to connect their Keplr account.

The steps to take are the same for each such component.

For instance, in src/components/Menu/NewGameModal/NewGameModal.tsx:

1. Add an [rpcUrl: string (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx#L25) to the props, passed along [Menu.tsx (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/Menu.tsx#L47) which also needs it [as a new prop (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/Menu.tsx#L17), in turn passed from [MenuContainer.tsx (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/MenuContainer.tsx#L82).
2. Add a signing client to the component's state, as well as the potential address of the Keplr user:



Copy

interface INewGameModalState {

...

+ creator: string;

+ signingClient: CheckersSigningStargateClient | undefined;

}

src /

components /

... /

NewGameModal /

NewGameModal.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L30-L31" \t "_blank)

Do not forget to initialize them to nothing in the constructor:



Copy

this.state = {

...

+ creator: "",

+ signingClient: undefined,

}

src /

components /

... /

NewGameModal /

NewGameModal.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L50-L51" \t "_blank)

The address obtained from Keplr is saved in creator, because it is accessible from the OfflineSigner but not from the SigningStargateClient.

1. Add a tuple type to return both of them:



Copy

interface CreatorInfo {

creator: string;

signingClient: CheckersSigningStargateClient;

}

src /

components /

... /

NewGameModal /

NewGameModal.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L17-L20" \t "_blank)

This is done because the setState function does not ensure the state is updated immediately after it has been called, so the lazy instantiation method has to return both.

1. Inform TypeScript of the *Keplr window* object:



Copy

import { Window as KeplrWindow } from "@keplr-wallet/types"

...

declare global {

interface Window extends KeplrWindow {}

}

src /

components /

... /

NewGameModal /

NewGameModal.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L3-L15" \t "_blank)

1. Add a function that obtains the signing client and signer's address (a.k.a. future transaction creator) by setting up Keplr and connecting to it:



Copy

protected async getSigningStargateClient(): Promise<CreatorInfo> {

if (this.state.creator && this.state.signingClient)

return {

creator: this.state.creator,

signingClient: this.state.signingClient,

}

const { keplr } = window

if (!keplr) {

alert("You need to install Keplr")

throw new Error("You need to install Keplr")

}

await keplr.experimentalSuggestChain(getCheckersChainInfo())

await keplr.enable(checkersChainId)

const offlineSigner: OfflineSigner = keplr.getOfflineSigner!(checkersChainId)

const creator = (await offlineSigner.getAccounts())[0].address

const client: CheckersSigningStargateClient = await CheckersSigningStargateClient.connectWithSigner(

this.props.rpcUrl,

offlineSigner,

{

gasPrice: GasPrice.fromString("1stake"),

},

)

this.setState({ creator: creator, signingClient: client })

return { creator: creator, signingClient: client }

}

src /

components /

... /

NewGameModal /

NewGameModal.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L61-L84" \t "_blank)

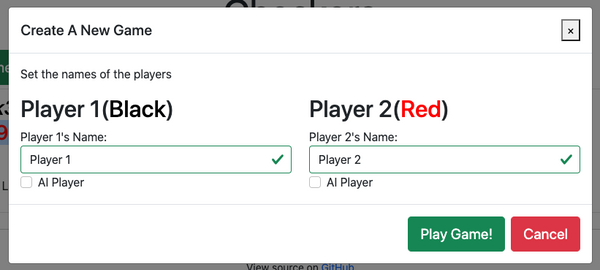


Setting up Keplr is [idempotent (opens new window)↗](https://www.merriam-webster.com/dictionary/idempotent#:~:text=Definition%20of%20idempotent,plural%20idempotents), so repeating these operations more than once is harmless. You may want to separate these actions into more defined methods at a later optimization stage.   
  
Note too that a default gas price is passed in, so that you can use "auto" when sending a transaction.

Your component is now ready to send transactions to the blockchain. Why choose NewGameModal.tsx as the example? Because this is where a new game is created.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#create-a-new-game) Create a new game

This modal window pops up when you click on New Game:



The player names look ready-made to take the player's blockchain addresses.

Examine the code, and focus on src/components/Menu/NewGameModal/NewGameModal.tsx. Thanks to the previous preparation with CheckersSigningStargateClient it is ready to send transactions:

1. In extensions-gui.ts declare an extension method to your CheckersSigningStargateClient that encapsulates knowledge about how to get the newly created game index out of the events:



Copy

declare module "../../checkers\_signingstargateclient" {

interface CheckersSigningStargateClient {

createGuiGame(creator: string, black: string, red: string): Promise<string>

}

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L26-L28" \t "_blank)

Now define it:



Copy

CheckersSigningStargateClient.prototype.createGuiGame = async function (

creator: string,

black: string,

red: string,

): Promise<string> {

const result: DeliverTxResponse = await this.createGame(creator, black, red, "stake", Long.ZERO, "auto")

const logs: Log[] = JSON.parse(result.rawLog!)

return getCreatedGameId(getCreateGameEvent(logs[0])!)

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L64-L72" \t "_blank)



Keep in mind:

* + For the sake of simplicity, a possible wager is completely omitted.
  + The getCreatedGameId is defined in a [previous section](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/2-cosmjs-messages.html).

1. In NewGameModal.tsx add an empty import of the extension method so that it is compiled in:



Copy

import {} from "src/types/checkers/extensions-gui"

src /

components /

... /

NewGameModal /

NewGameModal.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L8" \t "_blank)

1. Since you are going to paste addresses into the name field, make sure that the GUI does not truncate them. In src/components/Menu/NewGameModal/PlayerNameInput.tsx:



Copy

...

- maxLength={20}

+ maxLength={45}

src /

components /

... /

NewGameModal /

PlayerNameInput.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/PlayerNameInput.tsx" \l "L36" \t "_blank)

1. Back in NewGameModal.tsx, change the declaration of handleSubmit and make it async:



Copy

private async handleSubmit(event: any): Promise<void>

src /

components /

... /

NewGameModal /

NewGameModal.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L128" \t "_blank)

1. In handleSubmit, avert the need to save to local storage and call your create game extension method:



Copy

if (p1Valid && p2Valid) {

- const info: IGameInfo = {

- ...

- }

- const saved: IGameInfo[] = Lockr.get("saved\_games") || []

- Lockr.set("saved\_games", [info, ...saved])

+ const { creator, signingClient } = await this.getSigningStargateClient()

+ const index: string = await signingClient.createGuiGame(creator, p1Name, p2Name)

this.props.close()

}

src /

components /

... /

NewGameModal /

NewGameModal.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L156-L157" \t "_blank)

1. Next send the player directly to the newly created game:



Copy

...

this.props.close()

+ window.location.replace(`/play/${index}`)

src /

components /

... /

NewGameModal /

NewGameModal.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L159" \t "_blank)

1. In render() change the React link around the Button to a regular div so that window redirection appears smooth:



Copy

public render() {

return (

...

- <Link

- ...

- >

+ <div style={this.linkStyles} onClick={this.handleSubmit}>

<Button color="success" size="lg">

Play Game!

</Button>

- </Link>

+ </div>

...

)

}

src /

components /

... /

NewGameModal /

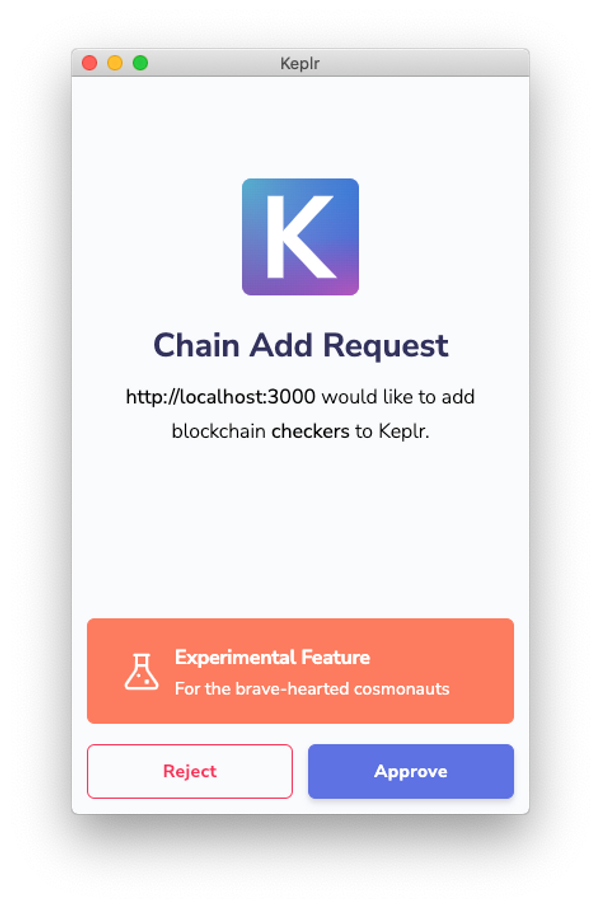
NewGameModal.tsx

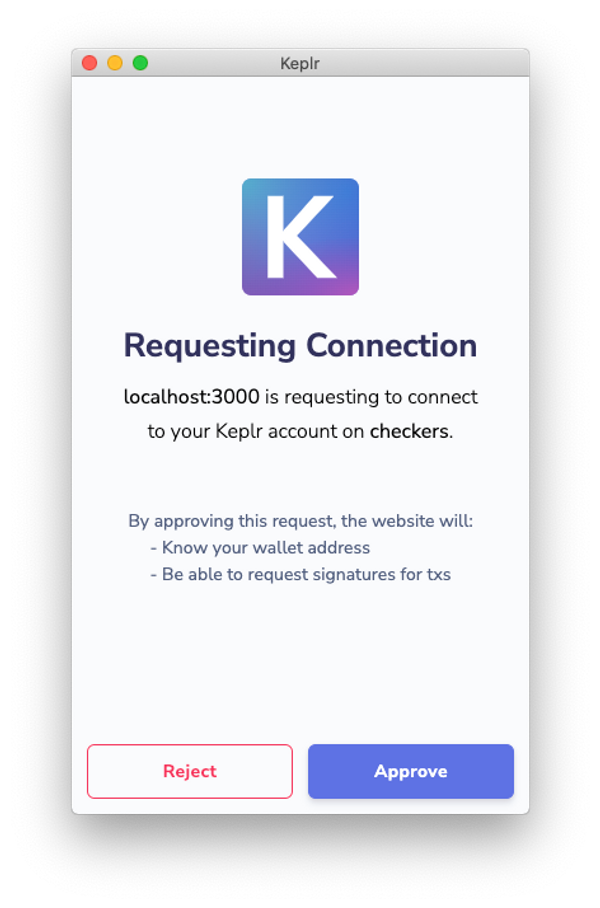
[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Menu/NewGameModal/NewGameModal.tsx" \l "L115-L119" \t "_blank)

You have now added game creation to your GUI.

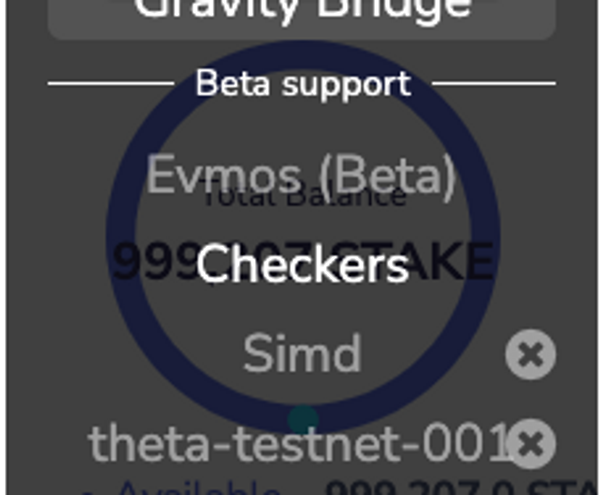
If you do not yet know your Keplr address on the checkers network, you will have to test in two passes. To test properly, you need to:

1. Run the initialization code by pretending to create a game. This makes Keplr prompt you to accept adding the checkers network and accessing your account. Accept both, but optionally reject the prompt to accept a transaction if your balance is zero.





1. Select *Checkers* in Keplr. Make a note of your address, for instance cosmos17excjd99u45c4fkzljwlx8eqyn5dplcujkwag8.



1. Use the faucet you started at the beginning of this section to put enough tokens in your Keplr *Checkers* account. "1000000stake" will satisfy by a 10x margin.



Copy

$ curl --request POST \

--header "Content-Type: application/json" \

--data '{"address":"cosmos17excjd99u45c4fkzljwlx8eqyn5dplcujkwag8","denom":"stake"}' \

localhost:4500/credit

1. Now start again to actually create a game. Accept the transaction, and you are redirected to the game page. This time the content of the game is from the blockchain.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#play-a-move) Play a move

In the GameContainer.tsx there are makeMove and saveGame functions. In a blockchain context, saveGame is irrelevant, as on each move done with a transaction the game will be automatically *saved* in the blockchain. So add index: -1 in the game to be saved to get past the compilation error.

Add a console.log to makeMove to demonstrate the format it uses:



Copy

const move: Position[] = keys.map((k: string): Position => k.split(",").map(Number) as Position)

console.log(move)

src /

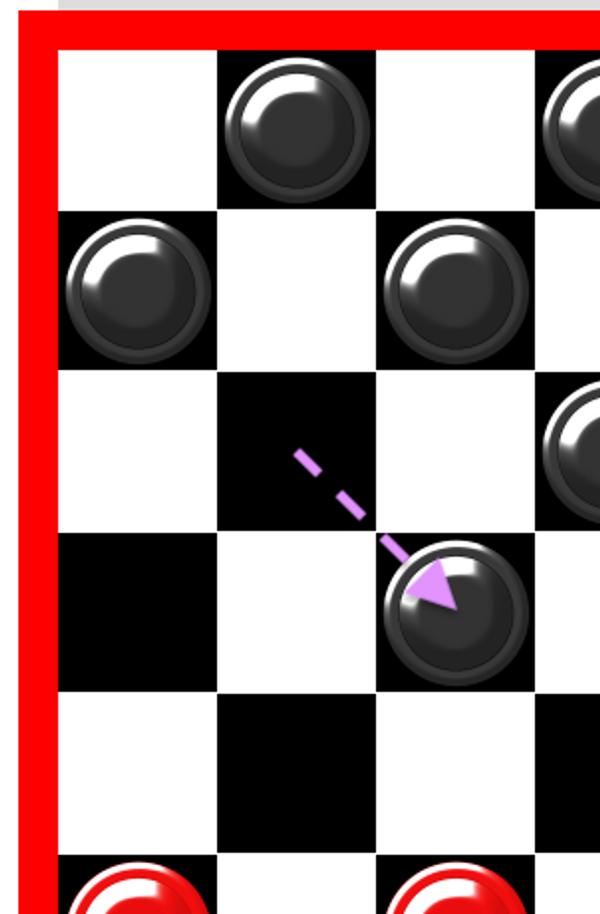
components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/unwired-gui/src/components/Game/GameContainer.tsx" \l "L172" \t "_blank)

Now play a move with the current interface. Move your first black piece:



In the blockchain code, this is fromX: 1, fromY: 2, toX: 2, toY: 3. However, the GUI prints:



Copy

[ [ 2, 1 ], [ 3, 2 ] ]

Evidently for each position X and Y are flipped. You can encapsulate this knowledge in a helper function:



Copy

export function guiPositionToPos(position: number[]): Pos {

return { x: position[1], y: position[0] }

}

src /

types /

checkers /

board.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/board.ts" \l "L45-L47" \t "_blank)

This is likely a familiar situation. Go back to the board and do as if your piece was moving twice. This time, the GUI prints an array with three positions:



Copy

[ [ 2, 1 ], [ 3, 2 ], [ 4, 3 ] ]

This may not be a legal move but it proves that the GUI lets you make more than one move per turn. You can use this feature to send a transaction with more than one move.

Do some more preparation:

1. In extensions-gui.ts declare an extension method to CheckersStargateClient to check whether the move is valid, with parameters as they are given in the GUI components:



Copy

declare module "../../checkers\_stargateclient" {

interface CheckersStargateClient {

...

+ canPlayGuiMove(

+ gameIndex: string,

+ playerId: number,

+ positions: number[][],

+ ): Promise<QueryCanPlayMoveResponse>

+ }

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L18-L22" \t "_blank)

Now define it:



Copy

CheckersStargateClient.prototype.canPlayGuiMove = async function (

gameIndex: string,

playerId: number,

positions: number[][],

): Promise<QueryCanPlayMoveResponse> {

if (playerId < 1 || 2 < playerId) throw new Error(`Wrong playerId: ${playerId}`)

return await this.checkersQueryClient!.checkers.canPlayMove(

gameIndex,

playerId === 1 ? "b" : "r",

guiPositionToPos(positions[0]),

guiPositionToPos(positions[1]),

)

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L50-L62" \t "_blank)

Note that due to the limits of the canPlayGuiMove function, you can only test the first move of a multi-move turn. That is, it uses only positions[0] and positions[1] of a potentially longer move. You cannot just test the next moves anyway because it will be rejected for sure. For it to be passed, the board would have to be updated first in the blockchain state.

1. Declare another extension method, this time for CheckersSigningStargateClient, to actually make the move with parameters as they are given in the GUI components:



Copy

declare module "../../checkers\_signingstargateclient" {

interface CheckersSigningStargateClient {

...

+ playGuiMoves(creator: string, gameIndex: string, positions: number[][]): Promise<(Pos | undefined)[]>

}

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L29" \t "_blank)

It returns the captured pieces. Now define it:



Copy

CheckersSigningStargateClient.prototype.playGuiMoves = async function (

creator: string,

gameIndex: string,

positions: number[][],

): Promise<(Pos | undefined)[]> {

const playMoveMsgList: MsgPlayMoveEncodeObject[] = positions

.slice(0, positions.length - 1)

.map((position: number[], index: number) => {

const from: Pos = guiPositionToPos(position)

const to: Pos = guiPositionToPos(positions[index + 1])

return {

typeUrl: typeUrlMsgPlayMove,

value: {

creator: creator,

gameIndex: gameIndex,

fromX: Long.fromNumber(from.x),

fromY: Long.fromNumber(from.y),

toX: Long.fromNumber(to.x),

toY: Long.fromNumber(to.y),

},

}

})

const result: DeliverTxResponse = await this.signAndBroadcast(creator, playMoveMsgList, "auto")

const logs: Log[] = JSON.parse(result.rawLog!)

return logs.map((log: Log) => getCapturedPos(getMovePlayedEvent(log)!))

}

src /

types /

checkers /

extensions-gui.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/types/checkers/extensions-gui.ts" \l "L74-L99" \t "_blank)



Note how it maps from the positions array except for the last position. This is to take the *moves* out of the *positions*.

With this done:

1. Repeat in GameContainer what you did in NewGameModal:
   1. Keep a [signing client and creator (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx#L40-L41) in the component's state.
   2. [Initialize (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx#L63-L64) it in the constructor.
   3. Add [a tuple (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx#L22-L25).
   4. Add a method to [lazily instantiate it (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx#L122-L145).
2. Change the declaration of makeMove to make it async:



Copy

public async makeMove(): Promise<void>

src /

components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx" \l "L202" \t "_blank)

1. In makeMove make sure that the move is likely to be accepted immediately after the existing code that extracts the move. It uses the read-only StargateClient, which allows players to look around without being asked to disclose their address:



Copy

const positions: Position[] = keys.map((k: string): Position => k.split(",").map(Number) as Position)

const client = await this.getStargateClient()

const canPlayOrNot = await client.canPlayGuiMove(

this.props.index,

this.state.board.current\_player,

positions,

)

if (!canPlayOrNot.possible) {

const error = `Cannot make this move ${canPlayOrNot.reason}`

alert(error)

throw new Error(error)

}

src /

components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx" \l "L211-L223" \t "_blank)



The move test is made with the *read-only* client. This is so that your users can poke around without connecting for as long as possible.

1. With this partial assurance, you can make an actual move:



Copy

const { creator, signingClient } = await this.getSigningStargateClient()

await signingClient.playGuiMoves(creator, this.props.index, positions).catch((e) => {

console.error(e)

alert("Failed to play: " + e)

})

src /

components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx" \l "L225-L229" \t "_blank)



The assurance is **partial** because what was tested with canPlayGuiMove is whether a player of a certain color can make a move or not. When making the move, it takes the Keplr address without cross-checking whether this is indeed the address of the colored player that tested a move. This **could be improved** either at the cost of another call to get the game, or better by adding the black and red addresses in the component's status.

The remaining code of the makeMove method can be deleted.

1. Finish with a reload of the game to show its new state:



Copy

const selected = Object.create(null)

this.setState({ selected })

return this.loadGame();

src /

components /

Game /

GameContainer.tsx

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/gui/src/components/Game/GameContainer.tsx" \l "L233-L235" \t "_blank)



There is a potentially hard-to-reproduce-in-production **race condition** bug here. The loadGame is done immediately after the transaction has completed. However, depending on the implementation of the RPC endpoint, the playGuiMoves and loadGame calls may hit two different servers on the backend. In some instances, the server that answers your loadGame may not have fully updated its store and may in fact serve you the **old** version of your game.   
  
As your GUI matures, you may want to show the *expected* state of the game before you eventually show its *finalized* state. Sometimes you may want to show the expected state of the game even before the transaction has completed, and add visual cues hinting at the fact that it is a **provisional** state.   
  
The same can happen when creating a game, where the second server may return null if it has not been updated yet.

Now you can play test in the GUI. Make sure to put your Keplr address as the black player, so that you start with this one. You will need a second account on Keplr with which to play red, otherwise you must play red from the command line. Alternatively, put the same Keplr address for both black and red.

Either way, it is now possible to play the game from the GUI. Congratulations!

If you started the chain in Docker, when you are done you can stop the containers with:



Copy

$ docker stop cosmos-faucet checkers

$ docker network rm checkers-net

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#further-exercise-ideas) Further exercise ideas

* Add pagination to the game list.
* Implement typical GUI features, like disabling buttons when their action should be unavailable, or adding a countdown to the forfeit deadline.
* Implement a Web socket to listen to changes. That would be useful when there are two players who cannot communicate otherwise (instead of polling).

synopsis

To summarize, this section has explored:

* How to prepare for and then integrate CosmJS and Keplr into the GUI of your checkers blockchain, including how to adjust the React app to be able to package CosmJS.
* How to integrate CosmJS, including working with the GUI's data structures, fetching games from the blockchain and displaying them, integrating with Keplr for browser-based players, creating a new game, and fetching a single game to be played.

previous

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/3-external-gui.html)

**[Get an External GUI](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/3-external-gui.html)**

up next

**[Backend Script for Game Indexing](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)**

[[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

Rate this Page

icon smile

icon meh

icon frown

Would you like to add a message?

Submit

Thank you for your Feedback!

[](https://ida.interchain.io/ida-course/discord-info.html)

On this page

[Prepare the integration with the checkers blockchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#prepare-the-integration-with-the-checkers-blockchain)

[Prepare Webpack](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#prepare-webpack)

[GUI data structures](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#gui-data-structures)

[Board converter](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#board-converter)

[Game converter](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#game-converter)

[Obtain a client](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#obtain-a-client)

[Pass RPC parameters](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#pass-rpc-parameters)

[Create the StargateClient](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#create-the-stargateclient)

[Show all games](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#show-all-games)

[All games container](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#all-games-container)

[A specific GUI method](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#a-specific-gui-method)

[All games call](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#all-games-call)

[Individual game menu container](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#individual-game-menu-container)

[Show one game](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#show-one-game)

[Integrate with Keplr](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#integrate-with-keplr)

[Identify the checkers blockchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#identify-the-checkers-blockchain)

[Prepare a signing client](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#prepare-a-signing-client)

[Create a new game](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#create-a-new-game)

[Play a move](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#play-a-move)

[Further exercise ideas](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html#further-exercise-ideas)

#### **Get Cosmos updates**

Unsubscribe at any time. [Privacy Policy↗](https://v1.cosmos.network/privacy)

     Next

Documentation

[Cosmos SDK](https://docs.cosmos.network/)[Cosmos Hub](https://hub.cosmos.network/)[CometBFT](https://docs.cometbft.com/)[IBC Protocol](https://ibc.cosmos.network/)

Community

[Interchain blog](https://blog.cosmos.network/)[Forum](https://forum.cosmos.network/)[Discord](https://discord.gg/cosmosnetwork)

Contributing

[Source code on GitHub](https://github.com/cosmos/sdk-tutorials)

[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)

**[](https://blog.cosmos.network/)[](https://twitter.com/cosmos)[](https://discord.gg/cosmosnetwork)[](https://www.linkedin.com/company/interchain-foundation/about/)[](https://reddit.com/r/cosmosnetwork)[](https://t.me/cosmosproject)[](https://www.youtube.com/c/CosmosProject)**



Dark mode

† This website is maintained by the Interchain Foundation (ICF). The contents and opinions of this website are those of the ICF. The ICF provides links to cryptocurrency exchanges as a service to the public. The ICF does not warrant that the information provided by these websites is correct, complete, and up-to-date. The ICF is not responsible for their content and expressly rejects any liability for damages of any kind resulting from the use, reference to, or reliance on any information contained within these websites.

Cosmos is a registered trademark of the [Interchain Foundation.](https://interchain.io/)[Privacy](https://v1.cosmos.network/privacy)