dKiT demo

Scaffold

Skip if you have an existing project.

To scaffold the project, we are going to use vite with the react-ts template. Feel free to pick a different vite template or replace vite with the build tool of your choice. The bottom line is that you are able to process javascript modules.

As for the package manager, in this example, we are going to use `bun`. Though you may as well opt for `npm`, `yarn`, `pnpm` or `deno`.
Make sure to replace `my-app` with your project name.

```
bun create vite my-app --template react-ts
```

Once the project is scaffolded, cd into the newly created directory.

```
cd my-app
```

Configure package resolutions

Pin the following dependency versions in `package.json` using resolutions. In the case of `npm` reference overrides. Either by manually updating the resolutions section in your `package.json`

```
{
  "resolutions": {
    "@noble/curves": "~1.6.0",
    "@noble/hashes": "~1.5.0"
  }
}
```

Or by running the command (which requires jq to be in the path):

```
cat <<< $(jq '.resolutions = {"@noble/curves":"~1.6.0","@noble/hashes":"~1.5.0"}' \
  package.json) > package.json
```

Install

In respect to the package manager you chose prior, install the `@doritokit/sdk` which will also install the rest of the project's dependencies.

```
bun install @doritokit/sdk
```

Configure polyfills and globals

Node.js globals `Buffer` and `global` need to be polyfilled. If you're using vite, it's as easy as installing and configuring vite-plugin-node-polyfills. Firstly, add the `vite-plugin-node-polyfills` package as a development dependency:

```
bun add -D vite-plugin-node-polyfills
```

For a concrete example reference the project's `vite.config.ts`, specifically, the plugins section:

```
import { defineConfig } from 'vite';
import { nodePolyfills } from 'vite-plugin-node-polyfills';

export default defineConfig = ({
  plugins: [
    nodePolyfills({ globals: { Buffer: true, global: true } }),
    // other plugins
  ],
  server: { // optionally configure the dev server port
    port: 3000,
  },
  // rest of the config
})
```

Configure tsconfig.json (if applicable)

The lowest supported `tsconfig.json` target is ES2022 and also requires lib ES2022 to be specified, due to usage of `Error.cause`.

Because of the usage of `enum`s, the erasableSyntaxOnly `compilerOption` in your `tsconfig.json` has to be turned off.

Below is a minimal example. For a more realistic configuration look up the project's `tsconfig.app.json`.

```json
{
  "compilerOptions": {
    "target": "ES2022",
    "erasableSyntaxOnly": false
  }
}
```

Start the dev server

In the context of a vite project, by convention the script to run the development server is called `dev`.

```
bun run dev
```

With the dev server running, navigate to `http://localhost:3000` where `3000` is to be replaced with the custom port number or the vite default `5173`.