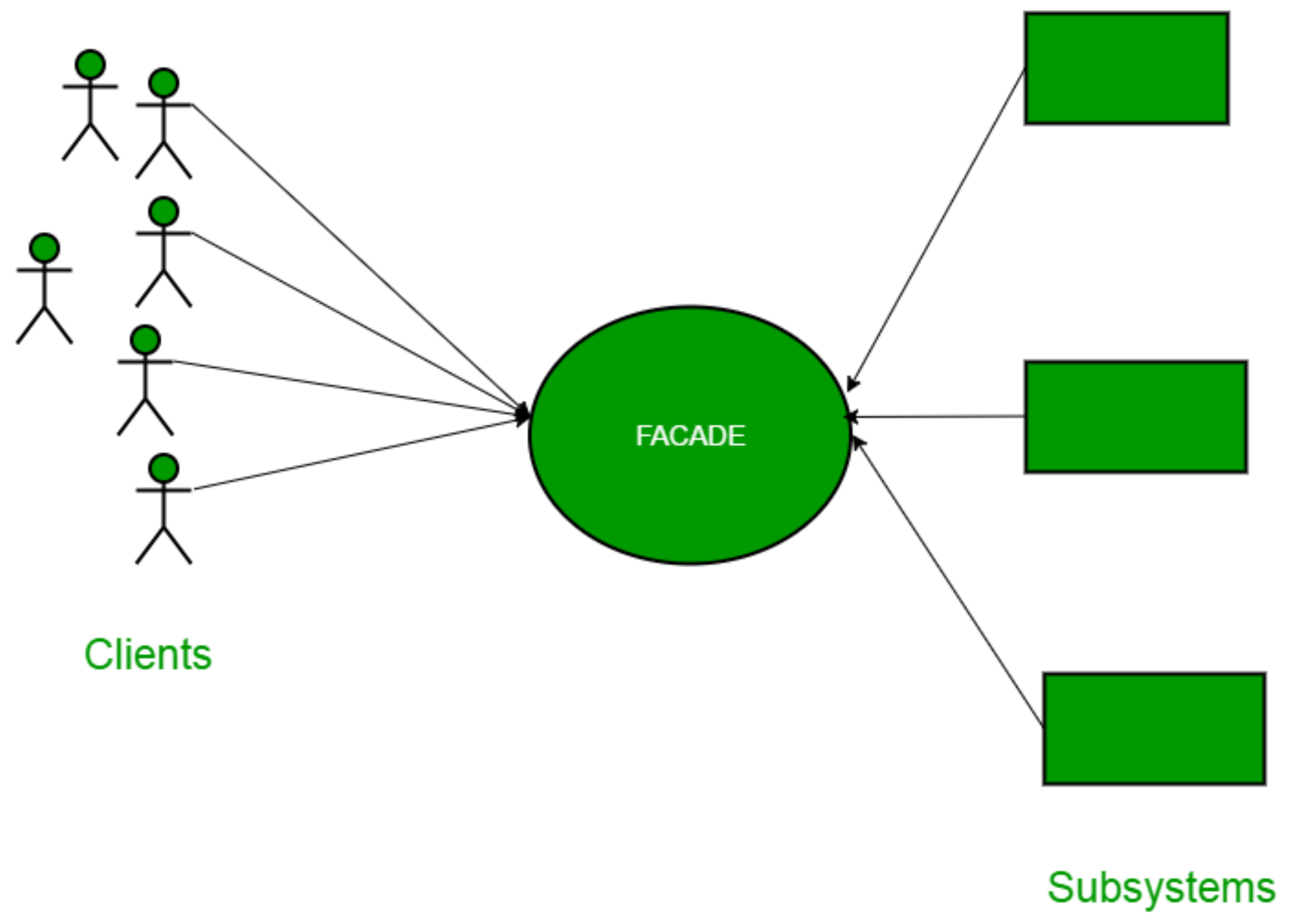
JSONStore

Full Stack Web Development

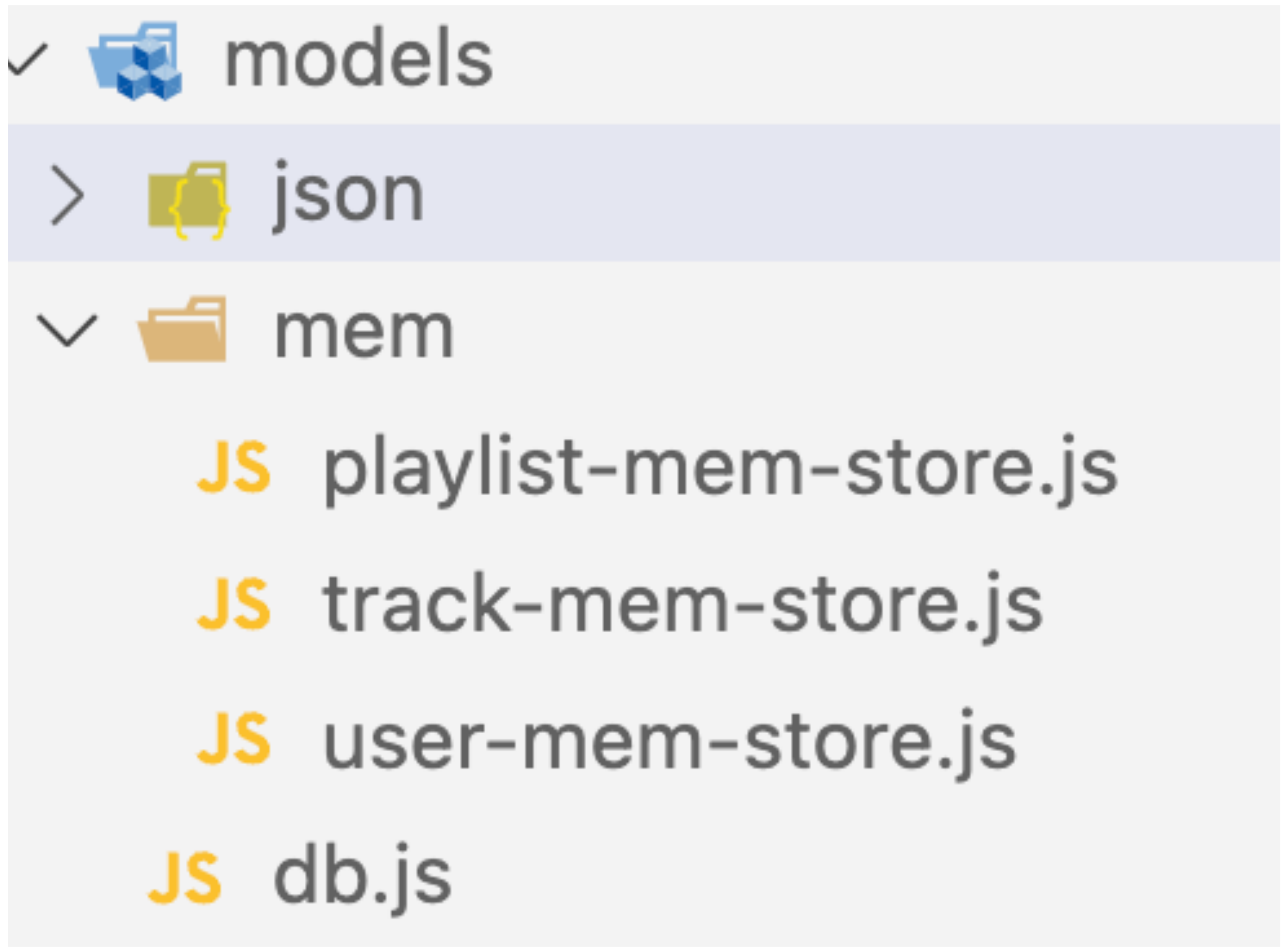
Facade Pattern

- The facade pattern (also spelled façade) is a software-design pattern commonly used in object-oriented programming.
- Analogous to a facade in architecture, a facade is an object that serves as a front-facing interface masking more complex underlying or structural code

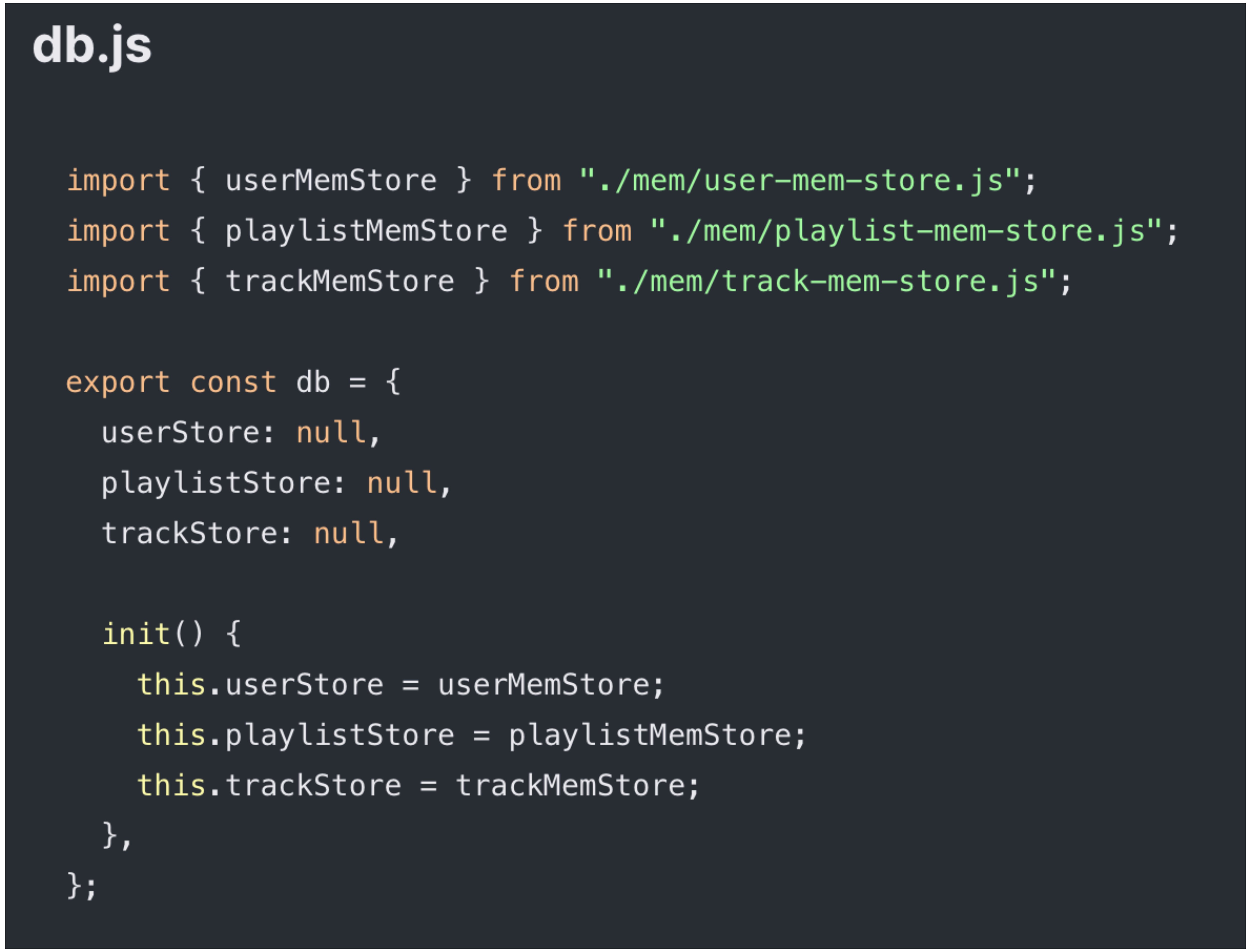


- improve the readability and usability of a software library by masking interaction with more complex components behind a single API
- provide a context-specific interface to more generic functionality
- serve as a launching point for a broader refactor of monolithic or tightly-coupled systems in favour of more loosely-coupled code

https://en.wikipedia.org/wiki/Facade_pattern



```
models
  json
  mem
    playlist-mem-store.js
    track-mem-store.js
    user-mem-store.js
  db.js
```

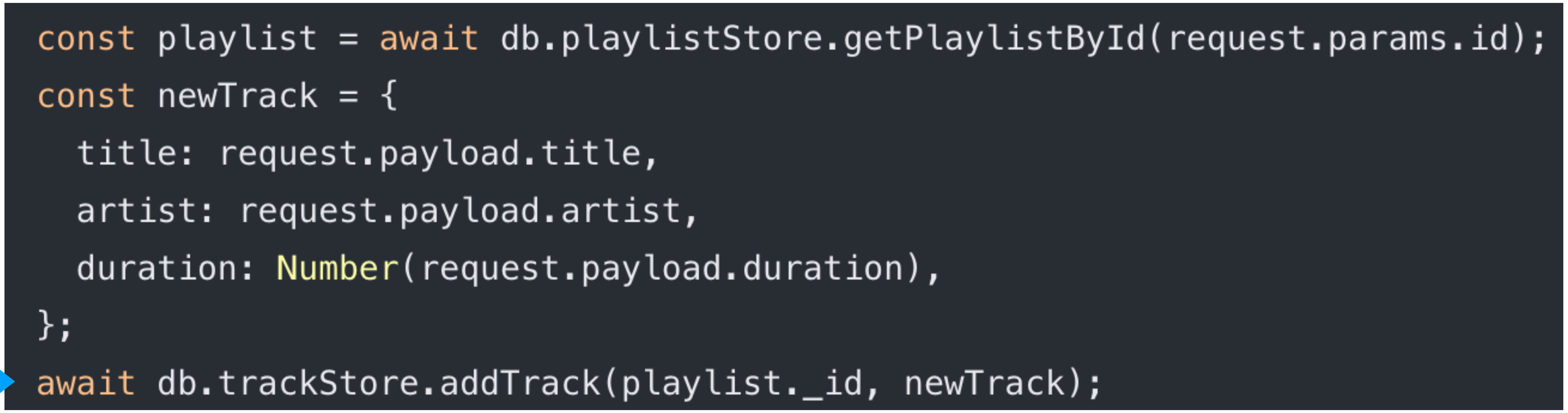


db.js

```javascript
import { userMemStore } from "./mem/user-mem-store.js";
import { playlistMemStore } from "./mem/playlist-mem-store.js";
import { trackMemStore } from "./mem/track-mem-store.js";

export const db = {
  userStore: null,
  playlistStore: null,
  trackStore: null,

  init() {
    this.userStore = userMemStore;
    this.playlistStore = playlistMemStore;
    this.trackStore = trackMemStore;
  },
};
```

- db.init() will initialise the database, associating userStore, playlistStore & trackStore with the MemStore implementations

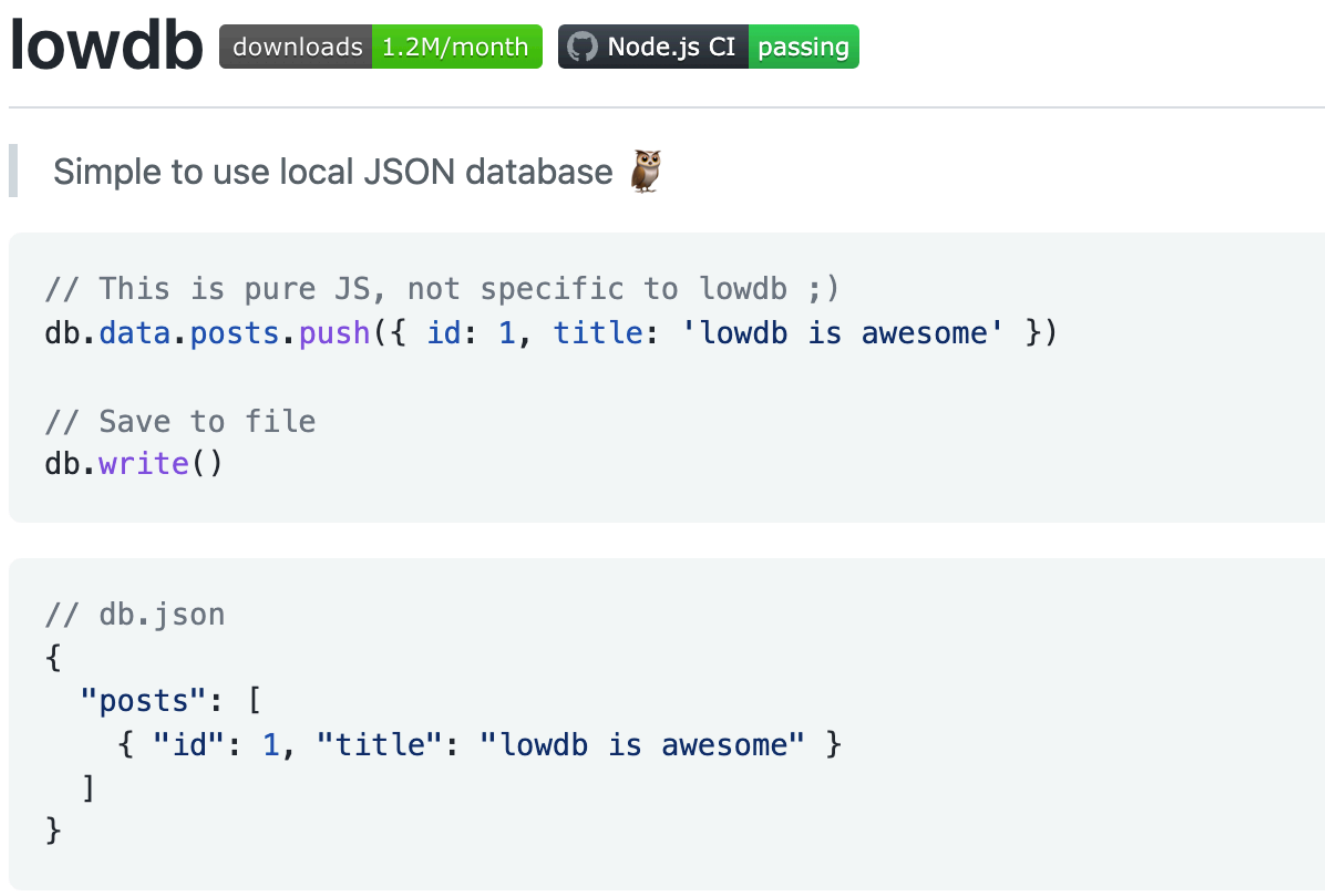
- The controllers will access the data stores via the db object. e.g:



```javascript
const playlist = await db.playlistStore.getPlaylistById(request.params.id);
const newTrack = {
  title: request.payload.title,
  artist: request.payload.artist,
  duration: Number(request.payload.duration),
};
await db.trackStore.addTrack(playlist._id, newTrack);
```

Lowdb

https://github.com/typicode/lowdb

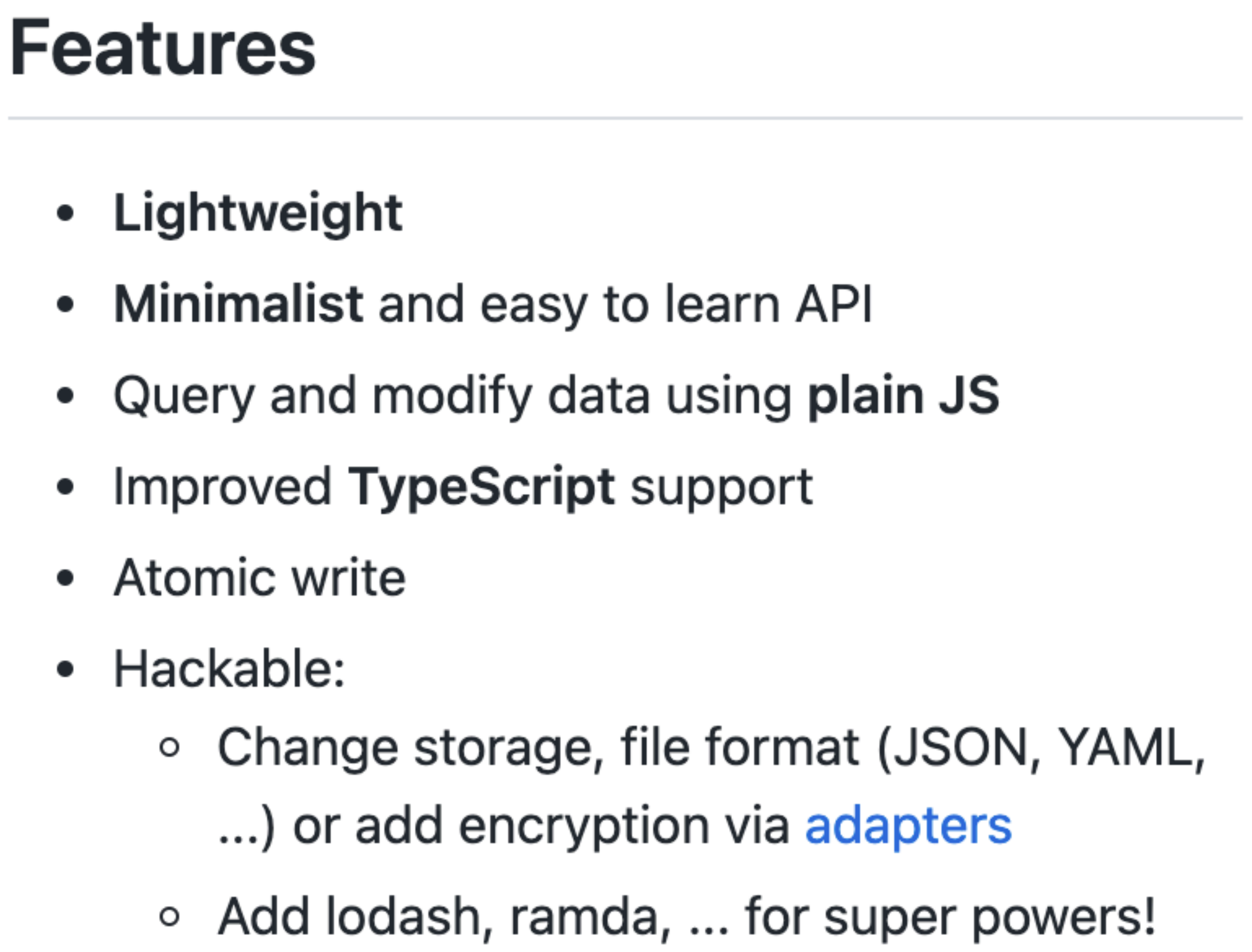


lowdb

> Simple to use local JSON database 🦉

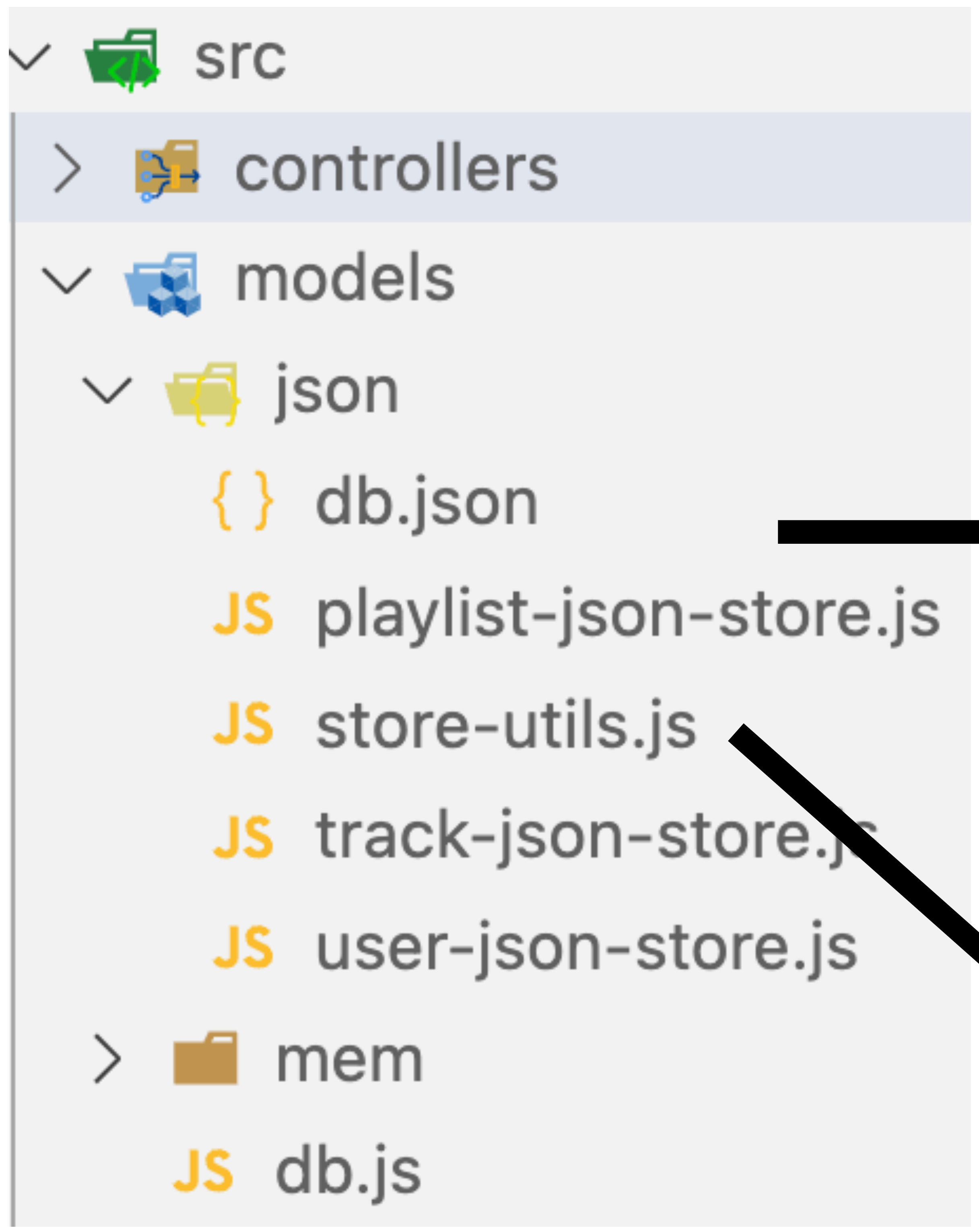
```js
// This is pure JS, not specific to lowdb ;)
db.data.posts.push({ id: 1, title: 'lowdb is awesome' })

// Save to file
db.write()
```

```
// db.json
{
  "posts": [
    { "id": 1, "title": "lowdb is awesome" }
  ]
}
```



Features

- **Lightweight**
- **Minimalist** and easy to learn API
- Query and modify data using **plain JS**
- Improved **TypeScript** support
- Atomic write
- Hackable:
  - Change storage, file format (JSON, YAML, ...) or add encryption via adapters
  - Add lodash, ramda, ... for super powers!

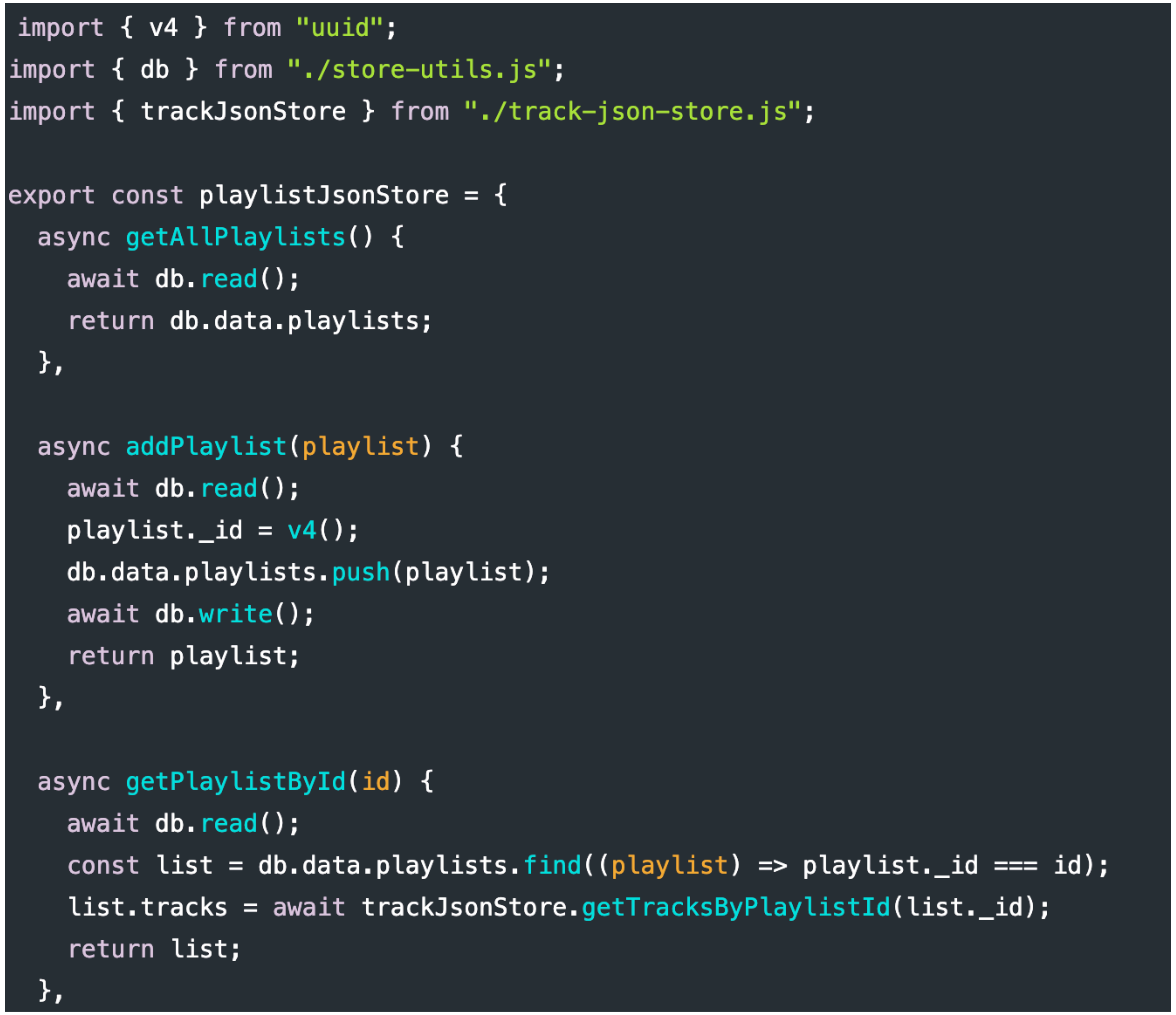


- Created by lowdb

```
{
  "users": [],
  "playlists": [],
  "tracks": []
}
```

- Store objects is JSON for
  - Users
  - Playlists
  - Tracks

```
import { JSONFilePreset } from "lowdb/node";

export const db = await JSONFilePreset("src/models/json/db.json", {
  users: [],
  playlists: [],
  tracks: [],
});
```

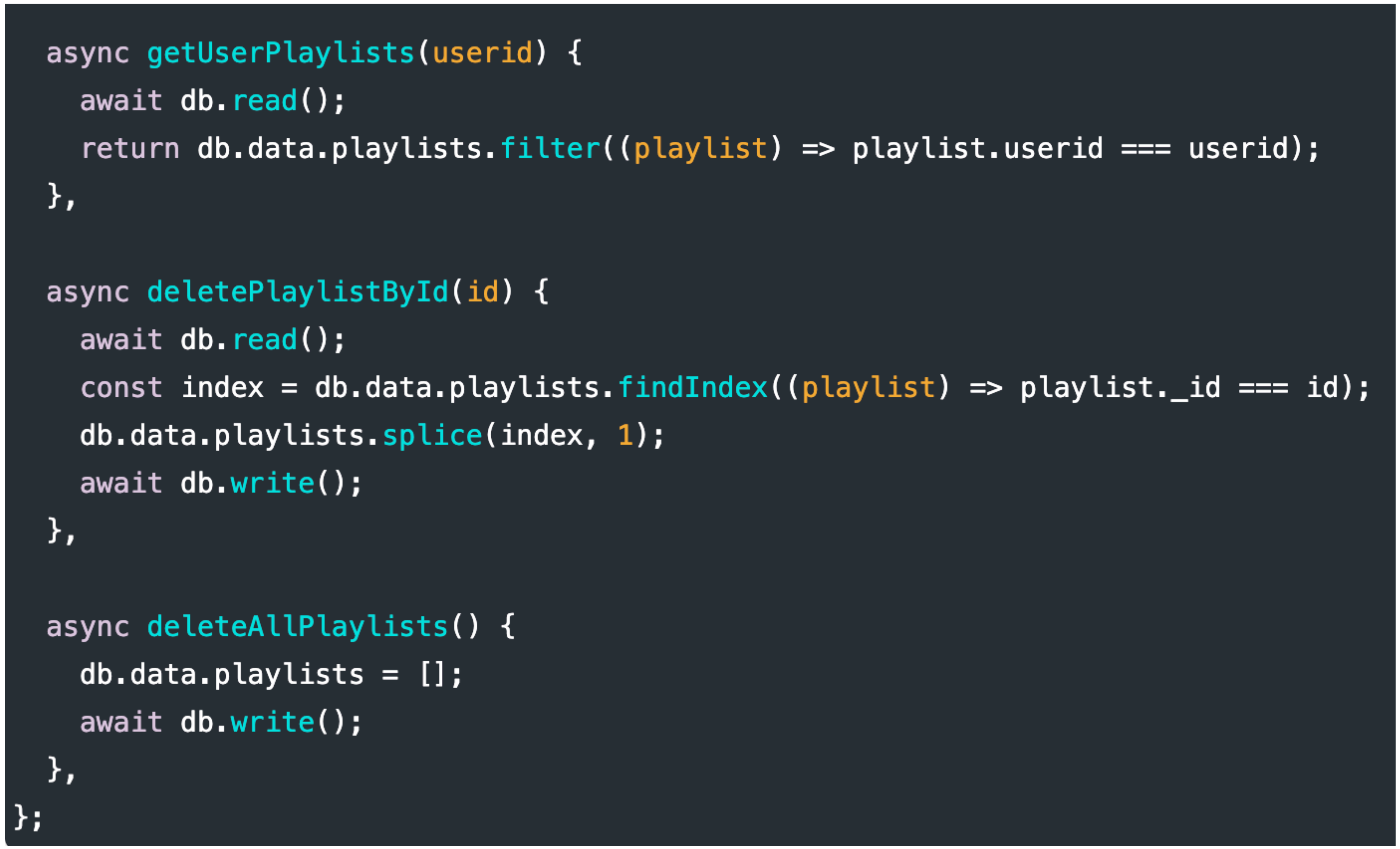


```
 import { v4 } from "uuid";
import { db } from "./store-utils.js";
import { trackJsonStore } from "./track-json-store.js";

export const playlistJsonStore = {
  async getAllPlaylists() {
    await db.read();
    return db.data.playlists;
  },

  async addPlaylist(playlist) {
    await db.read();
    playlist._id = v4();
    db.data.playlists.push(playlist);
    await db.write();
    return playlist;
  },

  async getPlaylistById(id) {
    await db.read();
    const list = db.data.playlists.find((playlist) => playlist._id === id);
    list.tracks = await trackJsonStore.getTracksByPlaylistId(list._id);
    return list;
  },
```

playlist-json-store.js

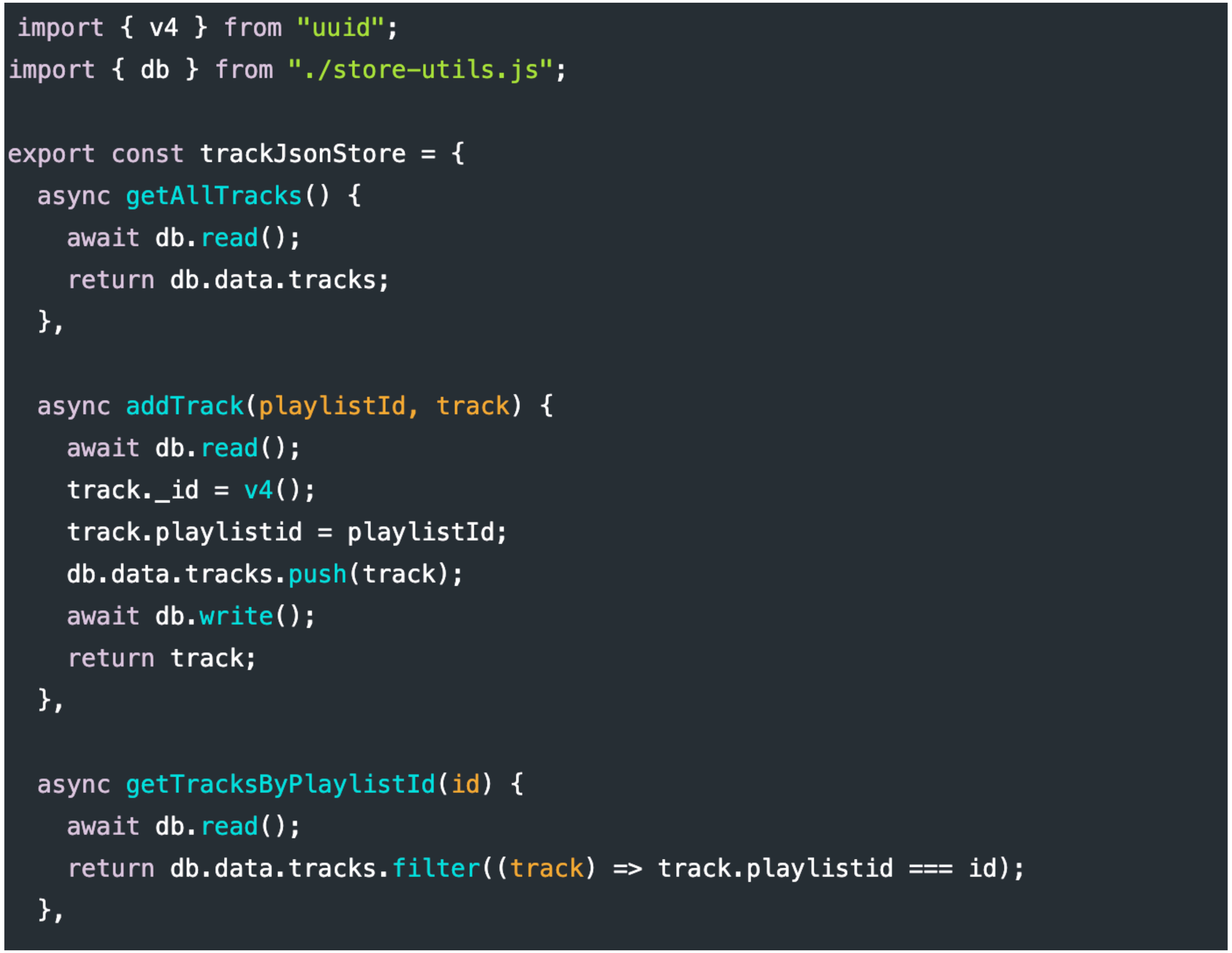
- Playlist implementation using Low DB library
- Similar implementation to MemStore version
  - Except db.read/write called to persist collections

playlist-json-store.js



```
  async getUserPlaylists(userid) {
    await db.read();
    return db.data.playlists.filter((playlist) => playlist.userid === userid);
  },

  async deletePlaylistById(id) {
    await db.read();
    const index = db.data.playlists.findIndex((playlist) => playlist._id === id);
    db.data.playlists.splice(index, 1);
    await db.write();
  },

  async deleteAllPlaylists() {
    db.data.playlists = [];
    await db.write();
  },
};
```

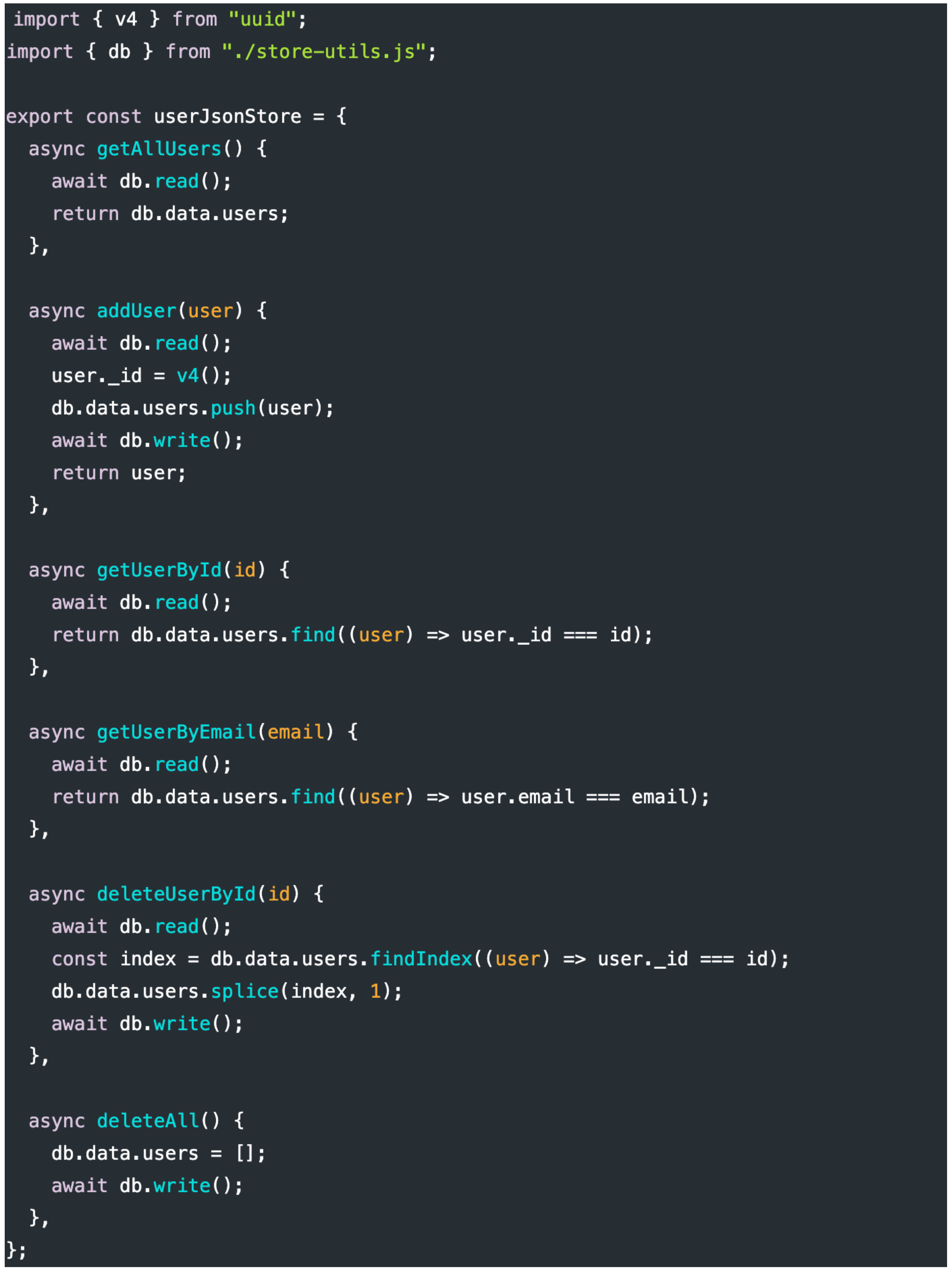
track-json-store.js



```javascript
 import { v4 } from "uuid";
import { db } from "./store-utils.js";

export const trackJsonStore = {
  async getAllTracks() {
    await db.read();
    return db.data.tracks;
  },

  async addTrack(playlistId, track) {
    await db.read();
    track._id = v4();
    track.playlistid = playlistId;
    db.data.tracks.push(track);
    await db.write();
    return track;
  },

  async getTracksByPlaylistId(id) {
    await db.read();
    return db.data.tracks.filter((track) => track.playlistid === id);
  },
```

track-json-store.js

```javascript
  async getTrackById(id) {
    await db.read();
    return db.data.tracks.find((track) => track._id === id);
  },

  async deleteTrack(id) {
    await db.read();
    const index = db.data.tracks.findIndex((track) => track._id === id);
    db.data.tracks.splice(index, 1);
    await db.write();
  },

  async deleteAllTracks() {
    db.data.tracks = [];
    await db.write();
  },

  async updateTrack(track, updatedTrack) {
    track.title = updatedTrack.title;
    track.artist = updatedTrack.artist;
    track.duration = updatedTrack.duration;
    await db.write();
  },
};
```

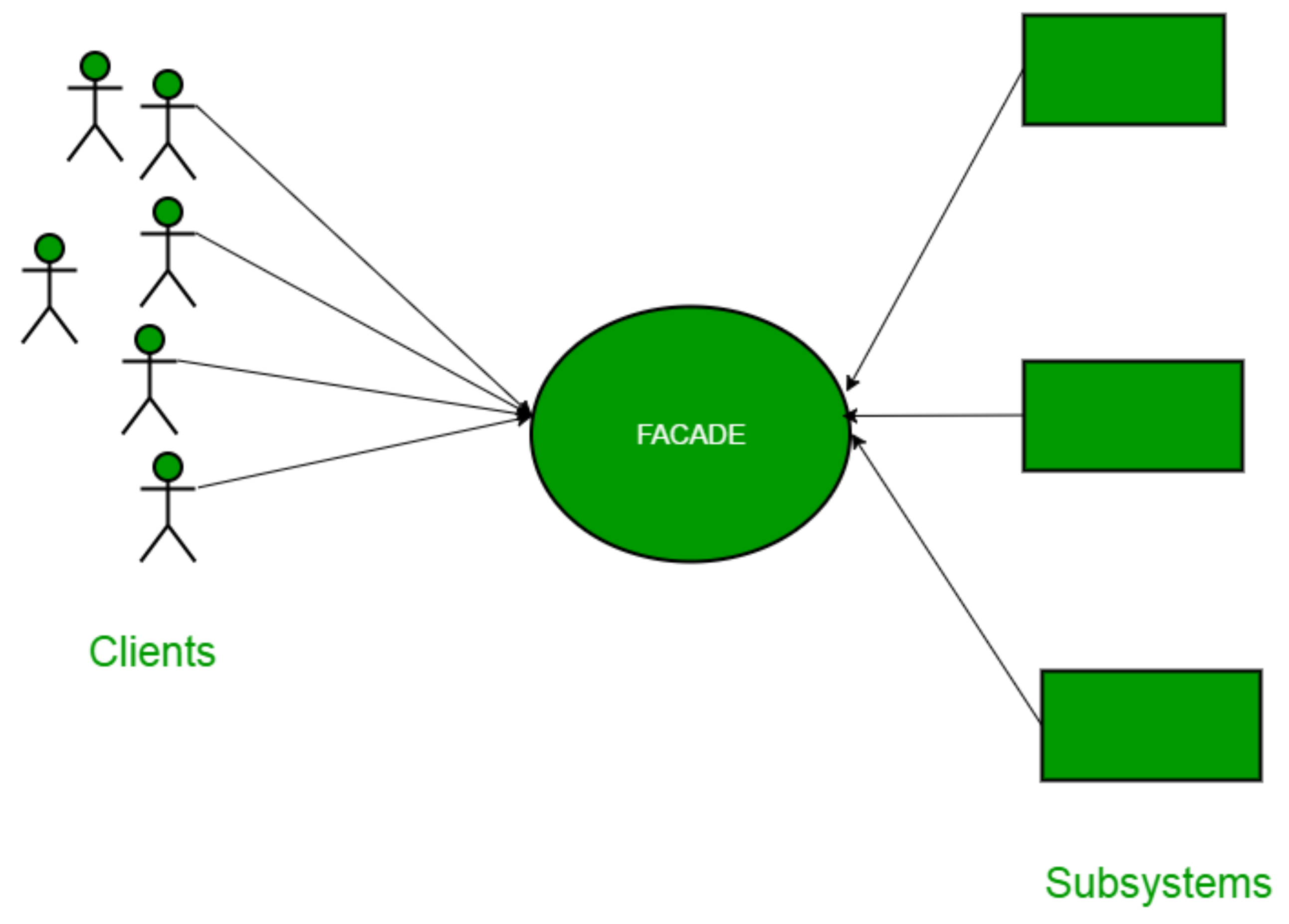


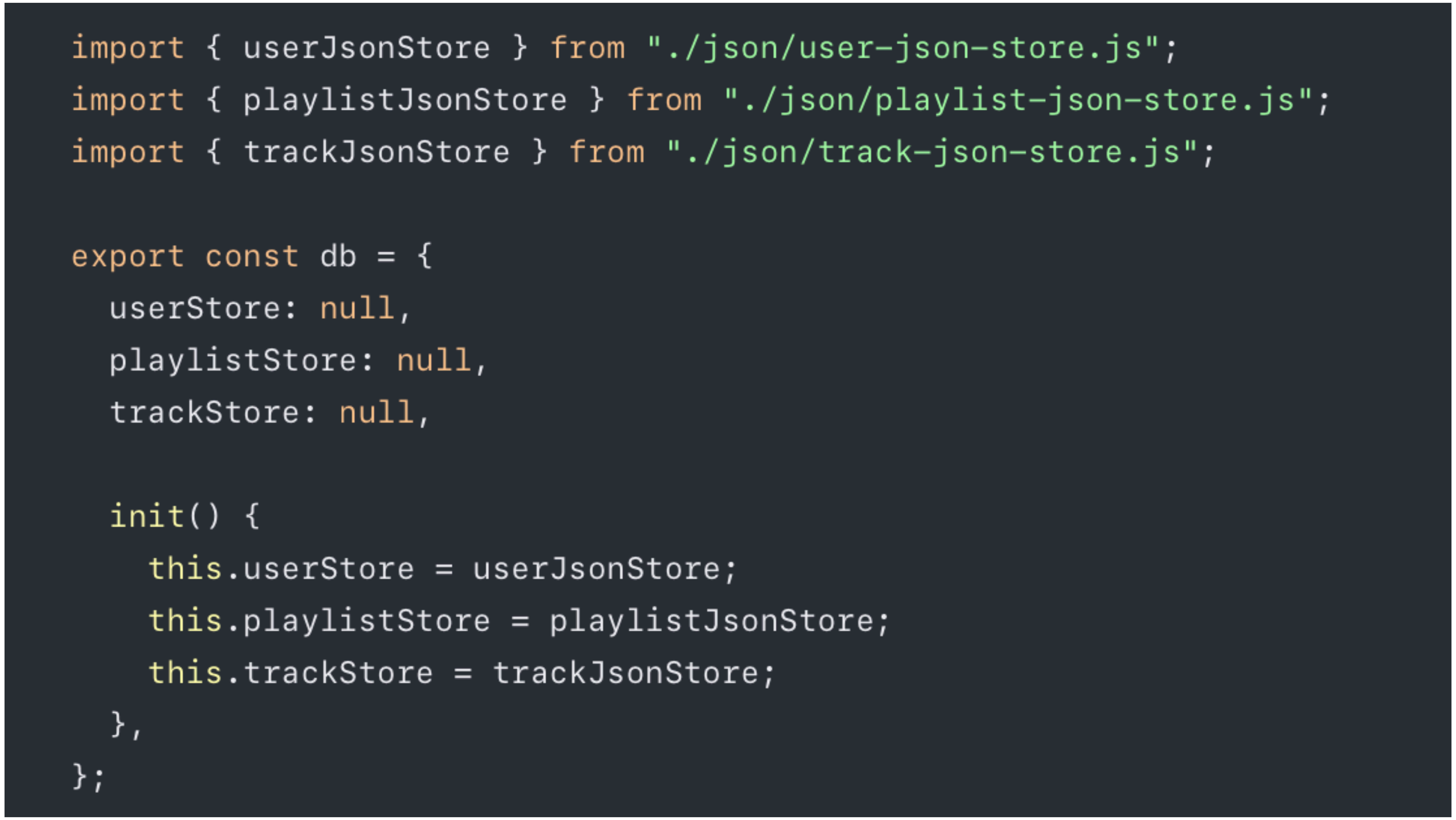
```
import { v4 } from "uuid";
import { db } from "./store-utils.js";

export const userJsonStore = {
  async getAllUsers() {
    await db.read();
    return db.data.users;
  },

  async addUser(user) {
    await db.read();
    user._id = v4();
    db.data.users.push(user);
    await db.write();
    return user;
  },

  async getUserById(id) {
    await db.read();
    return db.data.users.find((user) => user._id === id);
  },

  async getUserByEmail(email) {
    await db.read();
    return db.data.users.find((user) => user.email === email);
  },

  async deleteUserById(id) {
    await db.read();
    const index = db.data.users.findIndex((user) => user._id === id);
    db.data.users.splice(index, 1);
    await db.write();
  },

  async deleteAll() {
    db.data.users = [];
    await db.write();
  },
};
```

user-json-store.js

- Switch db Facade to connect to JSON store

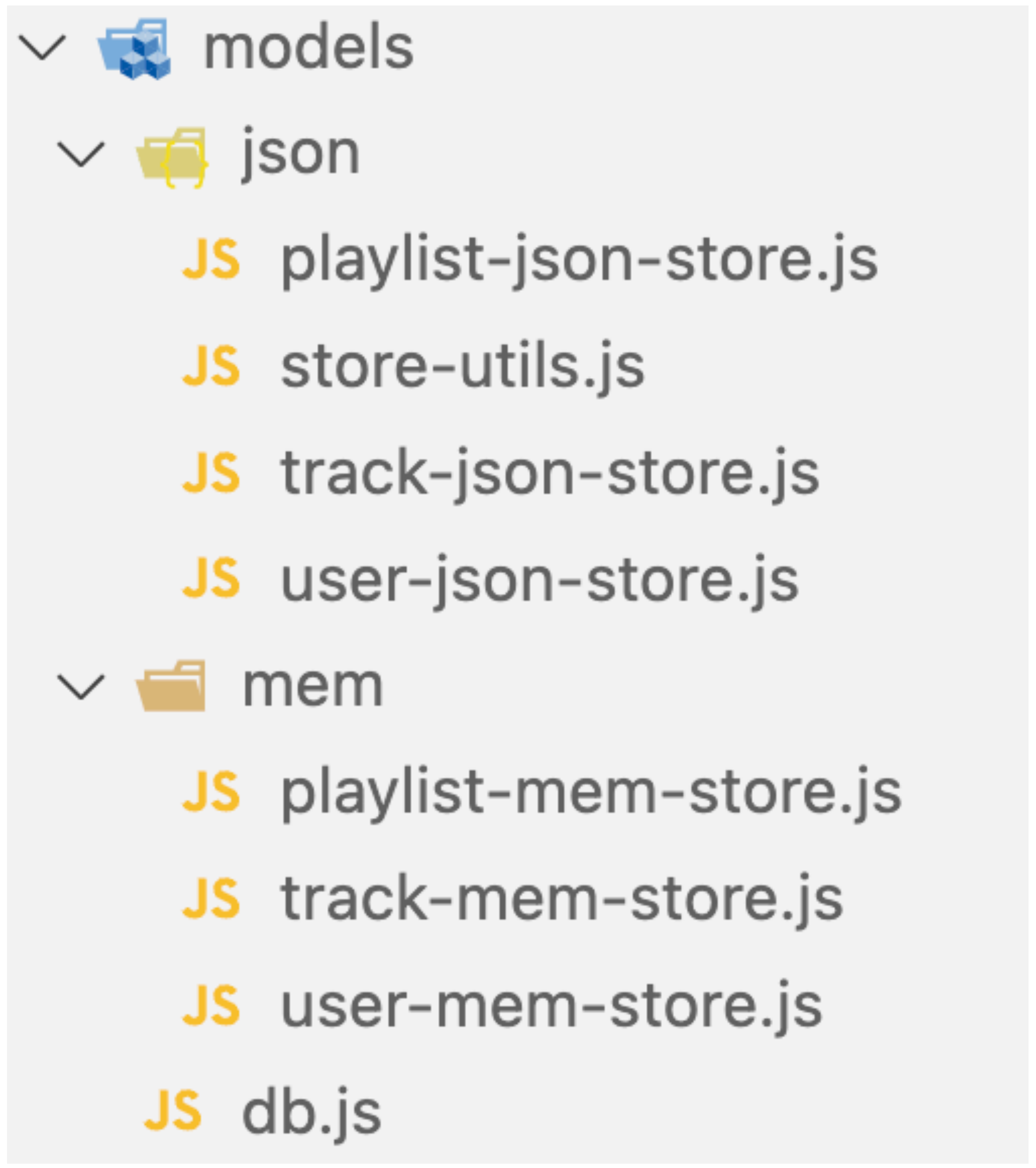
- No change to controllers - application should work as before.





```
import { userJsonStore } from "./json/user-json-store.js";
import { playlistJsonStore } from "./json/playlist-json-store.js";
import { trackJsonStore } from "./json/track-json-store.js";

export const db = {
  userStore: null,
  playlistStore: null,
  trackStore: null,

  init() {
    this.userStore = userJsonStore;
    this.playlistStore = playlistJsonStore;
    this.trackStore = trackJsonStore;
  },
};
```

db.js


```
models
  json
    playlist-json-store.js
    store-utils.js
    track-json-store.js
    user-json-store.js
  mem
    playlist-mem-store.js
    track-mem-store.js
    user-mem-store.js
  db.js
```

```
import { userJsonStore } from "./json/user-json-store.js";
import { playlistJsonStore } from "./json/playlist-json-store.js";
import { trackJsonStore } from "./json/track-json-store.js";

export const db = {
  userStore: null,
  playlistStore: null,
  trackStore: null,

  init() {
    this.userStore = userJsonStore;
    this.playlistStore = playlistJsonStore;
    this.trackStore = trackJsonStore;
  },
};
```

- db.init() will initialise the database, associating userStore, playlistStore & trackStore with the JsonStore implementations

- The controllers will access the data stores via the db object. e.g:

```
const playlist = await db.playlistStore.getPlaylistById(request.params.id);
const newTrack = {
  title: request.payload.title,
  artist: request.payload.artist,
  duration: Number(request.payload.duration),
};
await db.trackStore.addTrack(playlist._id, newTrack);
```

JSONStore

Full Stack Web Development