

# BorrowBox: turn a page into a working app

Track a borrowed drill, a book and a folding chair without losing the list on refresh.

Build

Check

Try independently

# The situation and your goal

## Situation

Your community lends objects. A resident needs to add an item, change it from available to borrowed, filter the list and recover the same state after refresh.

## Keep this artifact

A local BorrowBox app with add, filter, status change and refresh tests.

# Understand the mechanism

01

## State is the source of truth

Keep records in an array with stable IDs. Render the screen from those records. Reading the current DOM as your only database makes changes difficult to reason about.

02

## Persistence is separate from display

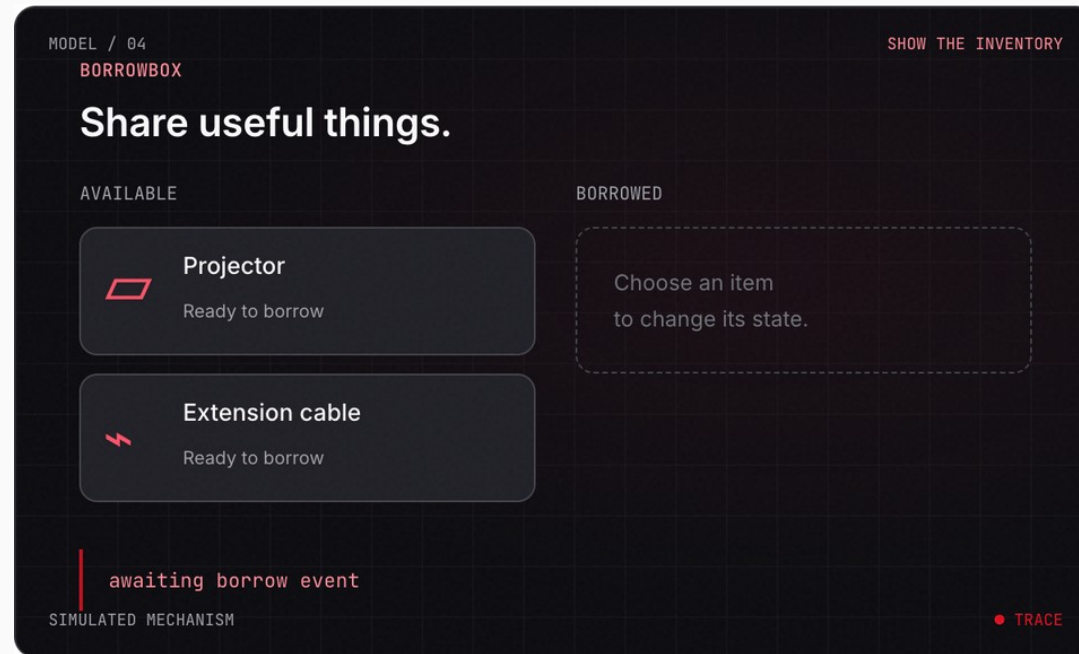
localStorage can save a JSON string for this browser origin. It does not create a shared community database or a cross-device backup.

03

## Inputs are untrusted text

Trim empty names, prevent duplicates according to your rule and render names using `textContent`. Treat an entered HTML tag as text, not executable markup.

# Explore before you build



# Define your record

- 1 Use borrowbox-items.json. Define id, name and status. Write the two allowed statuses and what happens for an empty name.
- 2 Check: You have an example valid record and invalid record.
- 3 Hint: Keep due dates as an extension until the basic state model works.

# Build add and render

- 1 Create an input, add button and list. Add a new record to state, then rerender. Render user text safely.
- 2 Check: Adding a name shows one new item exactly once.
- 3 Hint: Use stable IDs for updates, not changing array positions.

# Add transitions and filters

- 1 Implement available-to-borrowed and borrowed-to-available. Add an all, available and borrowed filter.
- 2 Check: Changing status updates the record and the filter results.
- 3 Hint: A filter should not delete records from state.

# Persist and restore

- 1 Serialize records into localStorage after changes. On load, parse inside try/catch and validate the shape before rendering.
- 2 Check: Refresh restores the same list on the same origin.
- 3 Hint: Serve on localhost for consistent tests. Private browsing or denied storage can fail.

# Break the happy path

- 1 Try an empty name, duplicate name, very long name and a literal HTML tag. Put invalid JSON in the app storage key using developer tools.
- 2 Check: The app explains invalid input and recovers from damaged storage.
- 3 Hint: Do not clear unrelated localStorage keys.

# Explain its boundary

- 1 Write a short handoff note: local device only, no account sync, how to export or reset your own records.
- 2 Check: Your partner understands why another phone sees a different list.
- 3 Hint: A real shared app needs a backend and access rules.

# When it goes sideways

01

## Refresh loses items

The app never writes state, or the origin changed. Try: Check the storage key, saved JSON and current page origin.

02

## Filter destroys hidden records

The filtered view replaced the original array. Try: Compute a visible subset and keep the complete state.

03

## Entered HTML changes the page

User input was inserted with innerHTML. Try: Render the item name with `textContent`.

# Review your evidence

1 Add, status change and three filters work.

2 Stable IDs identify records.

3 Refresh restores valid data.

# Review your evidence

- 1 Invalid JSON and denied storage have a visible recovery path.
- 2 HTML-like input stays literal text.
- 3 Local-only data limits are documented.

# Why can a second phone see an empty list?

- 1 localStorage belongs to a browser origin and device.
- 2 The first phone is definitely broken.
- 3 The app secretly deleted the community database.

# How should item names be rendered?

1 Concatenate them into innerHTML.

2 Use textContent.

3 Evaluate them as JavaScript.

# Build with a clear contract

Build a single-file BorrowBox lending tracker. Each record has a stable id, name and status (available or borrowed). Add records, switch status and filter without deleting hidden records. Trim and reject empty names; show duplicate warnings. Persist JSON in a namespaced localStorage key, recover visibly from invalid JSON or storage denial, and render input with textContent. Explain the state transitions and give manual tests. This is a local-only demo, not a shared database.

[Open copyable prompt](#)

# Add a due-date view

1

Filter does not modify stored records.

2

Invalid dates are handled.

3

Exported records can be inspected as JSON.

# Build independently, then teach it back

## Your next challenge

Introduce a due date and an overdue filter. Test the boundary at midnight and with a missing date.

## Explain your decisions

Show your artifact, one test that passed, one failure you repaired and one boundary the system respects.