Web apps that work offline and sync using React, Redux, and PouchDB

A Software Presentation From ZERRTECH

Jeremy Zerr

Site: http://www.zerrtech.com
LinkedIn: http://www.linkedin.com/in/jrzerr
Twitter: http://www.twitter.com/jrzerr
Introduction

- Designing an offline capable web/mobile product is hard. Local storage. Sync. Multiple platforms.
- A great way to do it is by using React, Redux, and PouchDB/CouchDB
- I will demonstrate some code using open source libraries that will add offline capability and syncing to an app
- This project is on Github at react-redux-pouchdb
What is the Problem?

☐ We design our web apps and mobile apps so differently.

☐ We always need to design mobile apps to work offline.

☐ But we rarely think of our web apps working offline.

☐ Multiple code bases, each handle state differently, users then get different experiences

☐ Web app offline capability and sync process can be hard to add later
Why do we need to solve this problem?

- If it wasn’t hard to design a web app to work offline, we would just design it that way from the beginning.

- One universal code base would reduce maintenance and easier to find developers with a particular set of skills.

- Sync is so tough, why does everybody roll their own? Hasn’t that been solved yet?
What would an ideal solution look like?

- Decouple in memory state changes from the API
- Our local store we use for persistence would be very similar to the server API store to make sync simple
- Ideally use someone else’s sync logic and engine so we don’t have to make it
- We should have the server notify clients of data changes
- Server keeps all document revisions to replay to clients
- Don’t write code in 3 different technologies for web, Android, iOS
- Oh yeah, and it should be free
Basics of offline design

- Data store changes happen in memory, but API might not always be available to sync
- Must decouple in memory state changes and API calls
- Introduce a persistent local store
- You sync all in memory store changes to a local persistent store
Basics of offline design - 2

- Persistent on “disk” means something like IndexedDB, actual filesystem on mobile devices, sqlite
- If internet up, sync local disk store with API
- Never have direct connection between memory and API
- App hydrates in memory store from local disk store
- Memory store, local store, server store all in sync
- Server “pushes” changes to clients to sync
How do people try to solve this problem today?

- Writing an API that cares about conflicts and can resolve them properly is a lot of work.
- Keeping track of revisions to reply is hard, usually just rely on timestamps to send back latest server copy.
- Use IndexedDB in JS or sqlite on Android/iOS, they are a LOT different storage systems, no easy way to compare.
- Communicate changes from server using socket.io or websockets.
- Firebase is one option, but closed source, no JS offline capabilities.

Meme courtesy of Zerrtech
CMO (Chief Meme Officer) Andrew Chumich
Open source solution

- Solve the multiple language/platform problems using React + Redux + React Native for mobile
- Use PouchDB for the local store, CouchDB for the server store
- Add Redux persistence library (redux-pouchdb) to write state changes to a local PouchDB store in a non-obtrusive way
- Set up the PouchDB local store to keep in sync with the CouchDB server store
- When a change hits CouchDB, the local PouchDB store should get the change, and then push back an action into Redux to set the state
What is this (P|C)ouchDB?

- Apache CouchDB is a database that you access using a HTTP/JSON REST API
- Document-oriented database, JSON docs
- Defined a CouchDB replication protocol, which allowed PouchDB to be created
- PouchDB is a JavaScript implementation of CouchDB, uses the same HTTP/JSON API
Why is CouchDB good at sync?

- Every document change creates a new rev
- When you do a document update, you need to provide the rev of the document you started with
- Exposes a changes feed that allows you out of the box to long poll to get changes as they happen
I built a sample app that would allow multiple users to increment a counter, offline, and will sync it to a server when online, one user sees the other’s changes quickly.

Demo is on Github: jzerr/react-redux-pouchdb

This is a React Redux app, where the counter state is in the Redux store using a Counter reducer.

Started with React Redux starter kit on Github: davezuko/react-redux-starter-kit

We will add in the npm library redux-pouchdb

yarn add redux-pouchdb

yarn add pouchdb

Very tiny diff here: Github diff
Reducers get initial state and changes from PouchDB via remote CouchDB
Counter Store Changes

Before

```
// Store Enhancers
const enhancers = []
```

After

```
import { persistentStore } from 'redux-pouchdb'

// Store Enhancers
const enhancers = [
  // set the store to persist to the local PouchDB
  persistentStore(db)
]
```

PouchDB persists parts of the store when it changes

Store Enhancers
import PouchDB from 'pouchdb'
// for debugging with PouchDB development tools
window.PouchDB = PouchDB

// Set up the PouchDB remote, local, and sync the two
const remotedb = new PouchDB('http://54.237.198.160:5984/counter');
const db = new PouchDB('counter')
db.sync(remotedb, {
  live: true,
  retry: true
})

Make the local PouchDB sync to remote CouchDB.
Just a single command
CouchDB Details

- Our server database: CouchDB
- Fire up an AWS EC2 instance
- Docker container: klaemo/couchdb
- docker pull klaemo/couchdb:latest
  - Gets you CouchDB 2.0
- docker run -d -p 5984:5984 -v $(pwd):/opt/couchdb/data klaemo/couchdb
Chrome Extensions

- Redux Devtools Chrome Extension
- Along with:
  - npm i --save-dev redux-devtools redux-devtools-log-monitor redux-devtools-dock-monitor
- PouchDB Inspector Chrome Extension
Thanks! Connect with Me! We would love to build your next app

A Software Presentation From

Jeremy Zerr

Site: http://www.zerrtech.com
LinkedIn: http://www.linkedin.com/in/jrzerr
Twitter: http://www.twitter.com/jrzerr