

ブックマーク管理システム: 動くア  
プリをとにかく早く安く作るう

# asocial-bookmark

- <https://github.com/azu/asocial-bookmark>
- **モチベーション:** はてなブックマーク APIが壊れた => 困った
- 金曜にAPIが500 Internal Server Error返すという報告した
  - CloudFrontがエラーを返してるので何かを設定ミス?

```
{
  body: '403 Forbidden Insufficient scope',
  headers: {
    'content-type': 'text/plain',
    'content-length': '32',
    connection: 'close',
    date: 'Fri, 31 May 2019 05:24:36 GMT',
    server: 'nginx',
    'set-cookie': [
      'b=$1$bTQbd484$ss/; expires=Thu, ' +
        '26-May-2039 05:24:36 GMT; domain=hatena.ne.jp; ' +
        'path=/'
    ],
    'x-dispatch': 'Boston::Web::Public::API::Legacy::Rest::My#tags',
    'x-varnish': '5020019',
    age: '0',
    via: '1.1 varnish-v4, 1.1 ' +
      'f5ea107910388dd712f11441721cd0ae.cloudfront.net ' +
      '(CloudFront)',
    'x-cache': 'Error from cloudfront',
    'x-amz-cf-id': 'hTBsI8X7-i6igkv8SvdaghasbJXXXXsh7YpqUXP7qNf1VFQ=='
  },
  statusCode: 403
}
```

# asocial-bookmark

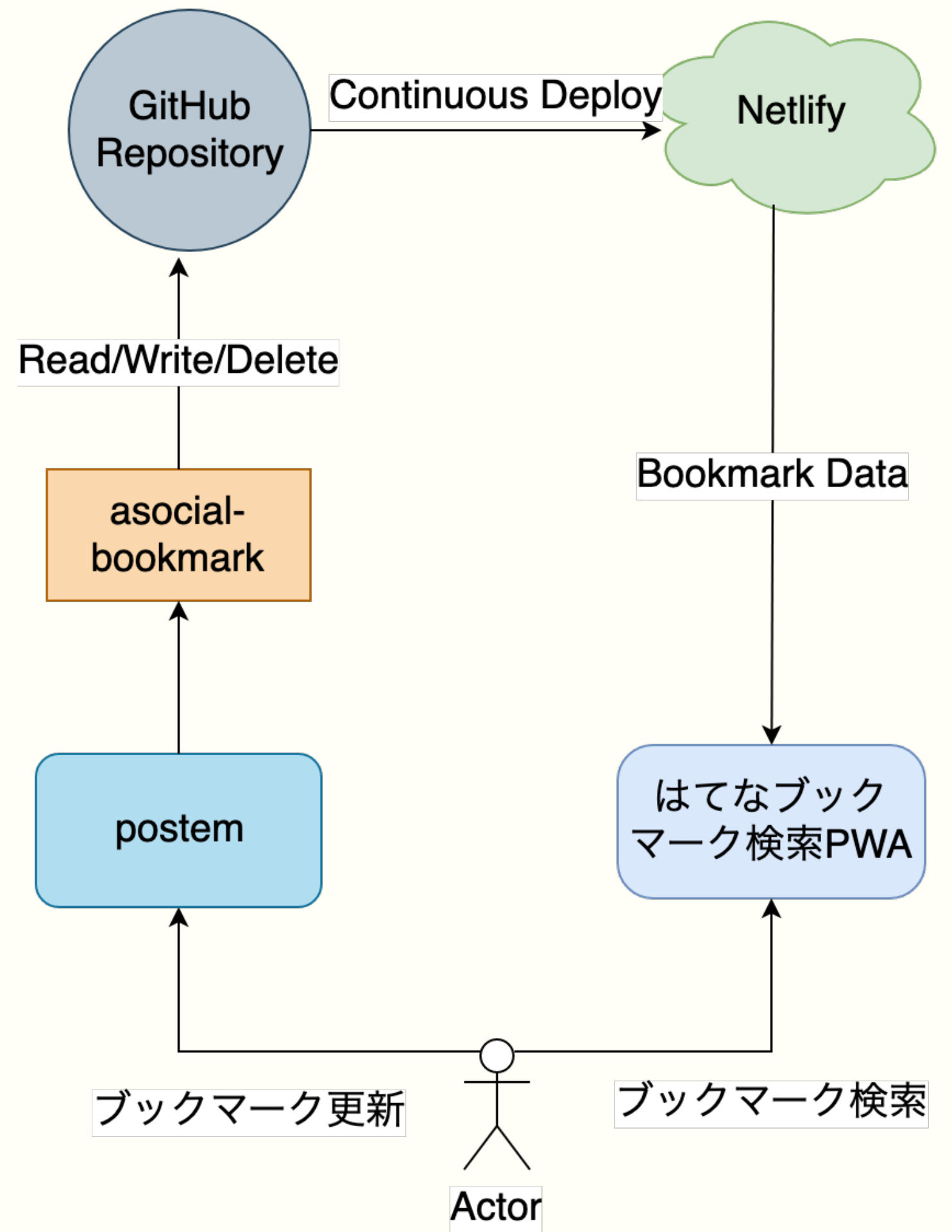
- 日曜の朝起きてもまだエラーおきてた
- => ブックマークできないのでブックマークサービス作る
- => 1日かけて作った

# asocial-bookmarkのコンセプト

- とにかく早く動かないとブラウザのタブが貯まるので早く作りたい
- [Pinboard](#)で良さそうと思ったけど、[Twitterアカウント](#)が活発過ぎて不安になった
- とりあえず自分で動くものを作ってみよう
- 維持費がかかるのは維持できなくなるので、維持コストはできるだけ低く設定する
- => 実はブックマークって大体静的なデータでいいのでは

asocial-bookmarkの全体像

- postemでブックマークを登録/更新
- 削除は手動で！
- ブックマークデータはGitHubにコミットされる
- GitHubにコミットされるとNetlifyにデプロイされる
- デプロイ時にビルドしたインデックスを作成する
- Netlifyからデータを取ってきて"はてなブックマーク検索PWA"で検索する



# asocial-bookmarkの全体像

- GitHub: データソース
- Netlify: CI/CD
- Client: postem(自作)
- Search: はてなブックマーク検索PWA(自作)



# 設計(作る前)

## asocial-bookmark(草案)

### ## 機能

- `/year/month/index.json`
- `/index.json`
- `/tags.json`

### ## 更新

- GitHub Action: `/index.json`
- GitHub API:
  - `/year/month/index.json`
  - `/year/month/id.json`

# 設計(作る前)

データ構造だけしか書いてなかった

# 実装

- 慣れた道具を使う
- 雑な実装をしていいが、モジュールは必ず守る
  - 泥団子はリファクタリングができない
- 必要なものはどんどん流用する
  - 流用して節約した時間で品質をあげる
  - 質が悪くなる流用は避ける

# 技術選定

- 早く作る場合に冒険心を抑える
- 普段から素振りしているものを使う
- [JavaScriptのトレンドを素振りして確認する方法 - Qiita](#)
- [JavaScriptの素振りする技術](#)
- ボトムアップに必要なものを手を止めずに作っていく

## 今回作ったもの/いじったもの

- <https://github.com/azu/korefile>
- <https://github.com/azu/asocial-bookmark>
- <https://github.com/azu/postem>
- <https://github.com/azu/hatebupwa>

# 技術選定

- TypeScript
  - 型
- なんかブラウザでも動きそうなものは動くように書く
- GitHubのAPIを叩くところはどっちでもいいはず

# korefile

- GitHubまたはLocalに対してRead/Write/Deleteを行うライブラリ
  - Writeは <https://github.com/azu/git-commit-push-via-github-api> の流用
  - AWS LambdaからGitHubでGit APIを使ってコミットpushする
  - Deleteは <https://github.com/jser/probot-jser-info/> の流用
  - Pull Requestにきたファイルのファイル名を変更するProbot(ファイルを消すコードがある)
- Read/Write/Deleteをそれぞれ実装した
- GitHubを叩くテストはexample.tsを作って実際にリポジトリで読み書きするものを書いた

# asocial-bookmark

- **korefile**を使ってGitHubリポジトリに対してブックマークデータを読み書きするライブラリ
  - ブックマークのデータ構造の定義を持っている
  - **korefile**が動くだろうと信じて想像で書く
- テストは以下同文
  - テストはexample.tsを作って実際にリポジトリで読み書きするものを書いた



# Migration Hatebu to asocial-bookmark

- はてなブックマークのデータをasocial-bookmarkに移行する
- [korefile](#)を作るときにfs adaptorを作っていた
  - GitHubをfsのように扱うというコンセプトだったので
- fsadaptorを使ってローカルのディレクトリに変換結果を作ってコミット！
- Adaptorという抽象実装を変えるだけで読み書きの先を変更できた



## モジュールの意識

- モジュールを分けることを意識するということは、どこで面を切るかを想像して書くこと
- 泥団子はすべて捨てるしかなくなるので、ここだけはちゃんと意識する
- ずっとコード読んだり書いたりしていると、切るパターンは見えてくる
- Node.js、ブラウザどちらでも動くモジュールを書こうとするとこれは常に意識する



# ブラウザとNode.jsでのモジュールの境界

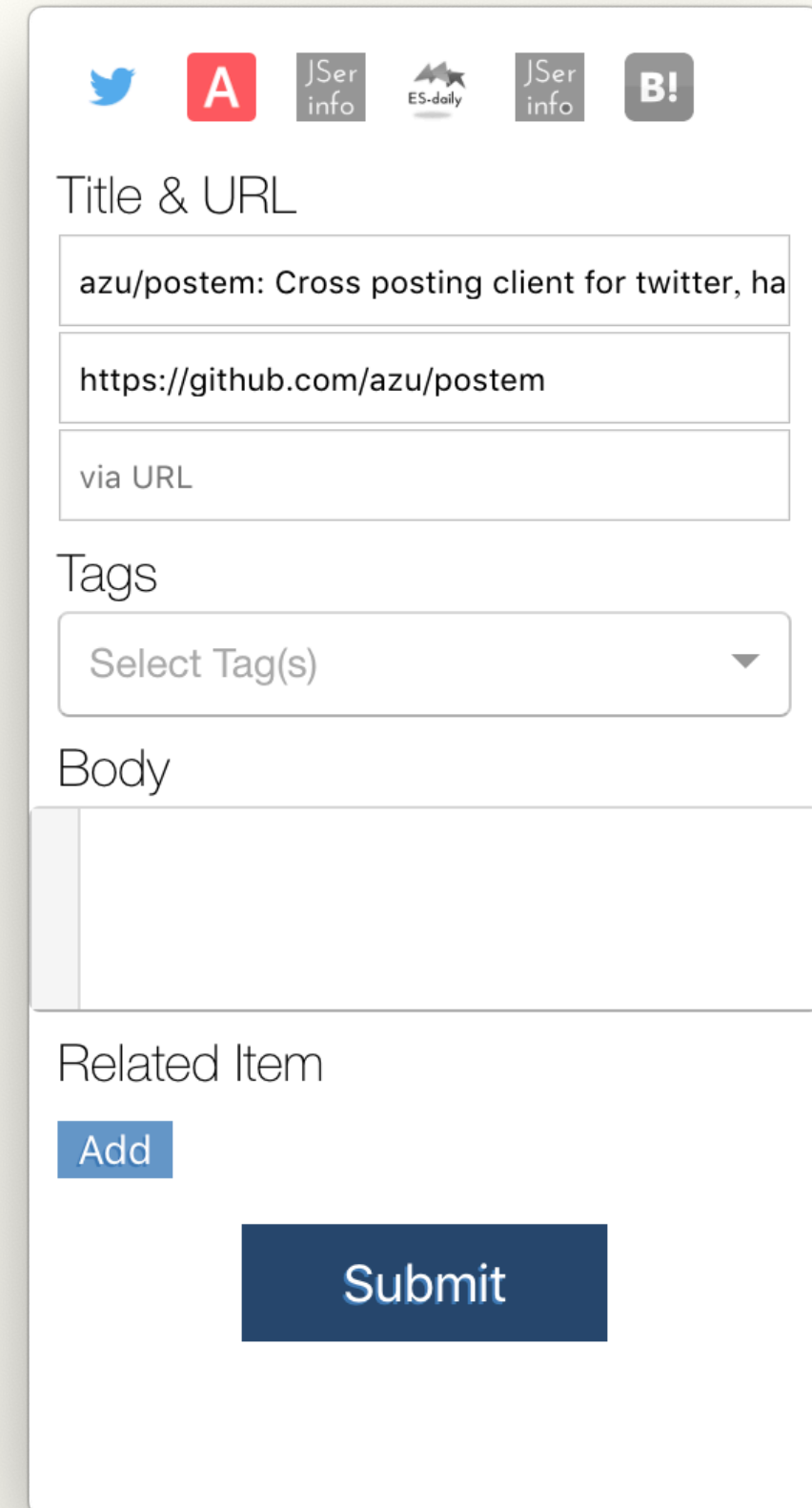
- ブラウザとNode.js
  - fs は一つの切れ目
    - fs に依存するところは外から差込可能なように作る or
    - fs に依存するところはコアから切り離す
  - そうしないとブラウザでは動かないモジュールができる
  - 今回はfsをadaptorとして外から差し込めるようにした

# GitHubのブックマークデータをURLとして取得できるように

- [Netlify](#)を使った
- デプロイ時にブックマークのインデックスデータ(全部を1つのJSONにしただけ！高速！)を作成
- Netlifyで配信すると自動的にCDNに乗るの
  - gh-pagesでも対しても問題ない(GitHub Actionを使えば大体手間は同じになる気がする)
  - 冒険より普段慣れているツールを使うことを優先
- ブックマークデータが `https://<your-domain>/index.json` で取得できるようになる

# postem

- pluginモデルではてなブックマークのプラグインがあった
- コピペしてURLを書き換えればほぼおわり
- ここでなんとなく投稿できるようになる
- 適当に投稿して試す
- GitHubへポストするロジックは [asocial-bookmark](#) に実装してたので意外とすんなり通る



The screenshot shows the postem web interface. At the top, there are social media icons for Twitter, a red 'A' logo, JSer info, ES-daily, another JSer info, and a 'B!' logo. Below these is the 'Title & URL' section with three input fields: the first contains 'azu/postem: Cross posting client for twitter, ha', the second contains 'https://github.com/azu/postem', and the third is labeled 'via URL'. The 'Tags' section has a dropdown menu labeled 'Select Tag(s)'. The 'Body' section is a large empty text area. At the bottom, there is a 'Related Item' section with an 'Add' button and a large dark blue 'Submit' button.

# はてなブックマーク検索 PWA

- ブックマークは検索したい
- インクリメンタルサーチをしたい
- => 以前からはてな向けのPWAを使って使ってた **モバイル/オフラインでも動作するはてなブックマーク検索のPWAを作った | Web Scratch**
- これが **asocial-bookmark** にも対応すればいい

はてなブックマーク検索

Input your hatena user name

https://azu-bookmarks.netlify.com/index.json 取得

Filter by words

ボトムアップドメイン駆動設計 後編   nrslib DDDの全体像の解説	2019-06-04 02:22
<a href="#">atomicojs/atomico: Small library for the creation of interfaces based on web-components, only using functions and hooks.</a> Web ComponentsベースのJSXでのDOM定義とReact HooksライクなAPIを提供するライブラリ	2019-06-03 18:42
<a href="#">What's New In DevTools (Chrome 76)   Web   Google Developers</a> Chrome 76の開発者ツールの変更点について。ネットワークパネル設定のUIの変更、WebSocketメッセージをHARファイルとしてエクスポートできるように、合計メモリ消費量のリアルタイム表示、Background Fetchのデバッグ機能など。	2019-06-03 18:40
<a href="#">Fishrock123/bob: 📦 binary data "streams+" via data producers, data consumers, and pull flow.</a> New Stream	2019-06-03 09:25
<a href="#">調べものリンク集   毎日ことば</a> 調べ物を調べるときのリンク集	2019-06-03 09:24
<a href="#">Zdog · Round, flat, designer-friendly pseudo-3D engine for canvas and SVG</a> 擬似的な3D描画を行うCanvas、SVGライブラリ。	2019-06-03 09:23
<a href="#">Portals · Issue #157 · mozilla/standards-positions</a> `<portal>`についてのMozilla Position	2019-06-03 09:20
<a href="#">TC39 - Specifying JavaScript</a> ECMAScriptについてのdiscourse/議論場所	2019-06-03 09:18

# はてなブックマーク検索 PWAのasocial-bookmark の対応

- 10行ぐらいで完了！
- ユーザー名にhttps://<your-bookmark>/index.jsonのURLを入れるとそのまま取得するように変更した！
- ウェブサービスではやっちゃダメだよ！(攻撃に使われることもあるので！)

```
7   export const fetchHatenaBookmark = (userName: string, sinceTime?: BookmarkDate)
8   +   // Support asocial-bookmark https://github.com/azu/asocial-bookmark
9   +   // TODO: undocument ways
10  +   const isURL = /^https?:\/\/\//.test(userName);
11  +   if (isURL) {
12  +       return fetch(userName)
13  +         .then(res => {
14  +           if (!res.ok) {
15  +             throw new Error("Can not fetch");
16  +           }
17  +           return res.json();
18  +         })
19  +         .then((json: AsocialBookmarkItem[]) => {
20  +           return json.map(item => {
21  +             return {
22  +               title: item.title,
23  +               comment: item.content,
24  +               url: item.url,
25  +               date: new Date(item.date)
26  +             };
27  +           });
28  +         });
29  +   }
30  +   const timeStamp = sinceTime ? `timestamp=${format(sinceTime.date, "YYYYMMDD"}`;
```

# memo

- 普通のウェブサービスだと、任意のURLを読み込ませることができるというリスクがある

```
<script src="https://example.com/secret.json">
```

- src先が利用できるかは関係なくブラウザはとりあえずロードするため、メモリ上にはデータが乗る(実行時にエラー)
- メモリにその(ログイン)ユーザーが読み込める秘密の情報を読み込ませる => Spectreでメモリにサイドチャネル攻撃して秘密の情報を取り出す



memo: そもそも意味ないロードは弾きたい

- [Cross-Origin Read Blocking \(CORB\)](#)という仕組みがあります
- `<script src="https://example.com/secret.json">` みたいな意味ないロードをそもそもブロックしてくれる
- CORBはサイドチャネル攻撃にとっても有効
  - Chrome, Firefox, Safariが実装
  - <https://www.chromestatus.com/feature/5629709824032768>

# ブックマークシステムの完成

- [postem](#)でブックマークを投稿
- [GitHub](#)にブックマークデータをコミット
- [Netlify](#)でブックマークのインデックスデータを配信
- [はてなブックマーク検索PWA](#)でブックマークを検索

# おわりに

- はてなブックマーク APIは月曜日に治りました！