

The background is an abstract composition of green and blue textures. On the left, there are dark, swirling patterns resembling water or ink. On the right, there are lighter, more uniform green areas with some small, white, bubble-like shapes. A white, rounded rectangular box is centered horizontally, containing the text.

SPA サンプルアプリ



サンプルの紹介

読書管理システム(簡易)

BookIndex

検索する



Webデザインの現場で使えるVue.jsの教科書

読んだ日: 2020-10-10

感想: おもしろかった。あああ




動かして学ぶ!Vue.js開発入門

読んだ日: 2020-10-11

感想: ためになった



使っている技術



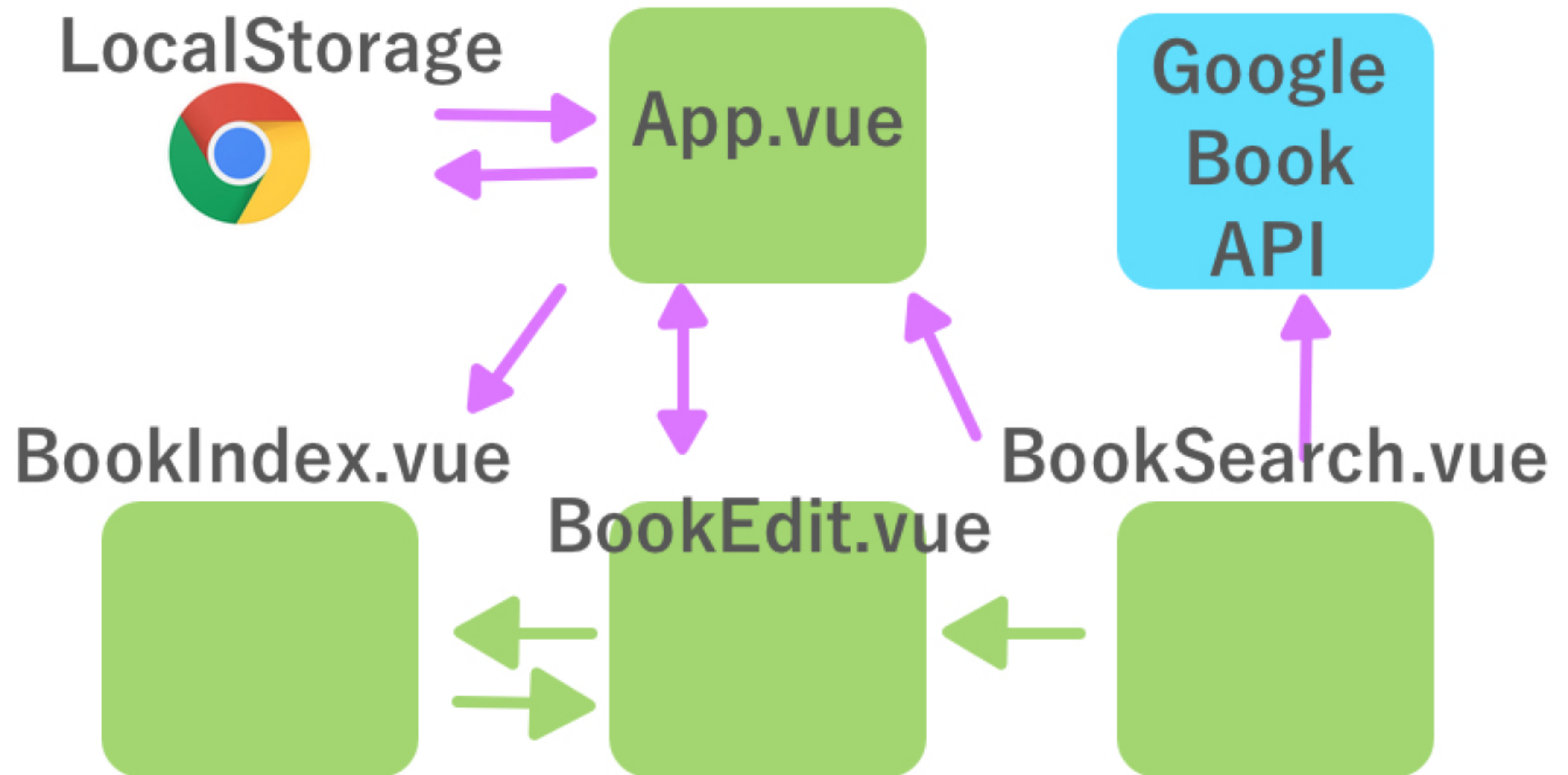
nuxt ^2.15.7

vuetify ^2.5.5


pwa ^3.3.5

GoogleBooksAPI /
WebStorage(LocalStorage)

簡単な構成図(Vue.js版)



VueJsのマニュアル



Vue.jsと検索

<https://jp.vuejs.org/index.html> (Vue2)

<https://v3.ja.vuejs.org/> (Vue3)



Google Books API

Google Books API



メリット

登録なしでも使える (1000件/日)

デメリット

検索が少し弱い(表示されない本も)

価格が表示されない・出版日が正確ではない
(AmazonAPIは条件厳しい
(30日以内に売上必要))

Google Books API



ベースのURL

[https://www.googleapis.com/books/v1/volumes?
q=検索語句](https://www.googleapis.com/books/v1/volumes?q=検索語句)


intitle: 本のタイトル

maxResults:40 検索表示数(10-40)

<https://developers.google.com/books>

Guides->Using the API-> Query parameter
reference

クエリーストリング



```
const baseUrl = 'https://www.googleapis.com/  
books/v1/volumes?'
```

```
const params = {  
  q: `intitle:${keyword}`,  
  maxResults:40  
}
```

```
queryParams = new URLSearchParams(params)  
fetch(baseUrl + queryParams)
```




WebStorage (LocalStorage)

Cookie & WebStorage

	サイズ	サーバー通信	有効期限	範囲
クッキー	4KB	毎回	指定期限まで	
Local Storage	1オリジンあたり 5MB	通信しない (必要時のみ)	なし	オリジン単位
Session Storage	1オリジンあたり 5MB	通信しない (必要時のみ)	ウィンドウを 閉じるまで	セッション単位

今回はLocalStorageで(DBの代わり)

LocalStorage



// 取得

`localStorage.getItem(key)`

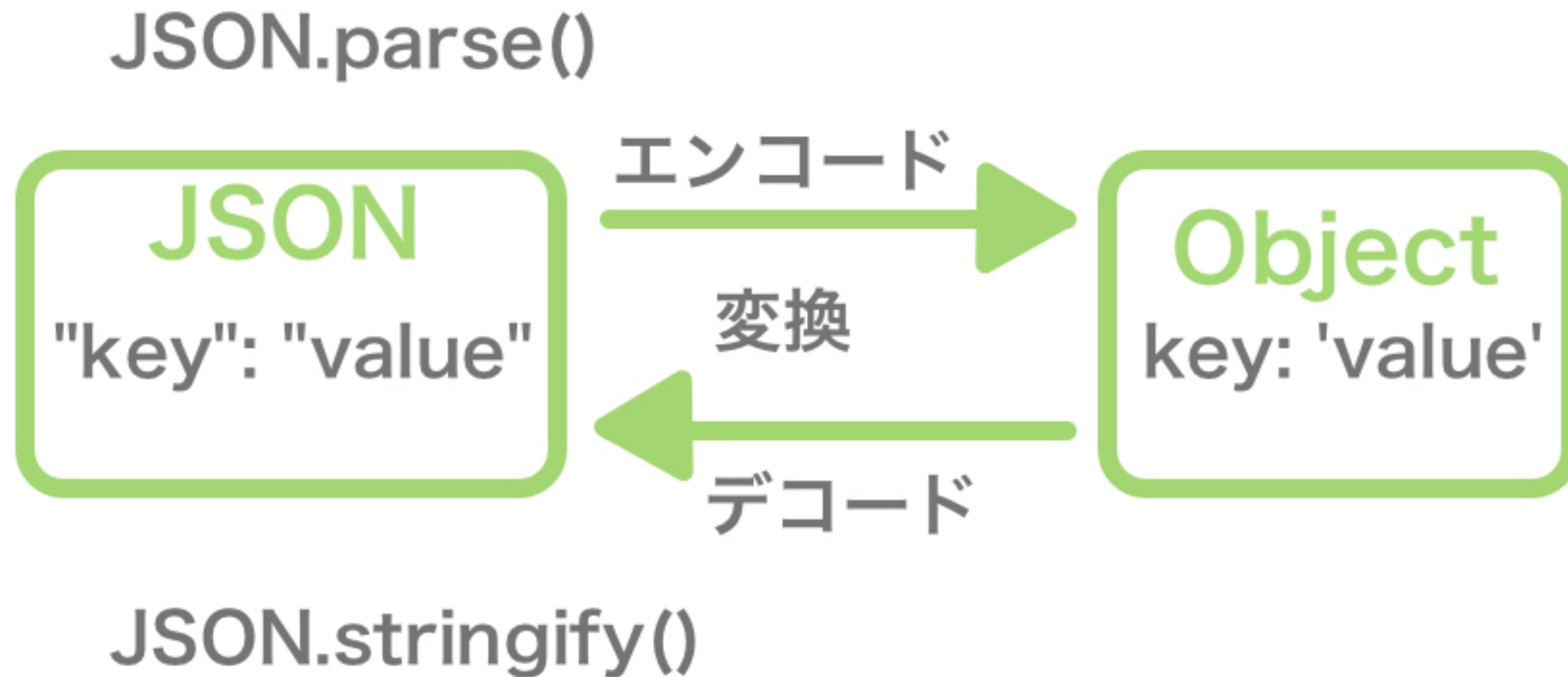
// 保存

`localStorage.setItem(key)`

// 削除

`localStorage.removeItem(key)`

JSONエンコード/デコード



LocalStorage と JSON



```
// 取得 JSON -> Object  
JSON.parse(localStorage.getItem(key))
```

```
// 保存 Object -> JSON  
const parsed = JSON.stringify(Object)  
localStorage.setItem(key, parsed)
```

Vue.jsガイド->クックブック->クライアントサイド
ストレージ



VueJs2のおさらい

Templateで使えるディレクティブ

v-if 条件分岐

v-show 表示非表示

v-for 繰り返し

v-bind (省略形 :) データ紐付け

v-on (省略形 @) イベント(クリックなど)

v-text, v-html テキスト、HTML

v-model フォーム用 双方向バインディング

OptionsAPI

data リアクティブなデータ

methods メソッド

computed 常時計算する算出プロパティ

watch 常時監視するオブジェクト

props down, event up

(コンポーネント間のやりとり)

ライフサイクル (created, mounted…)

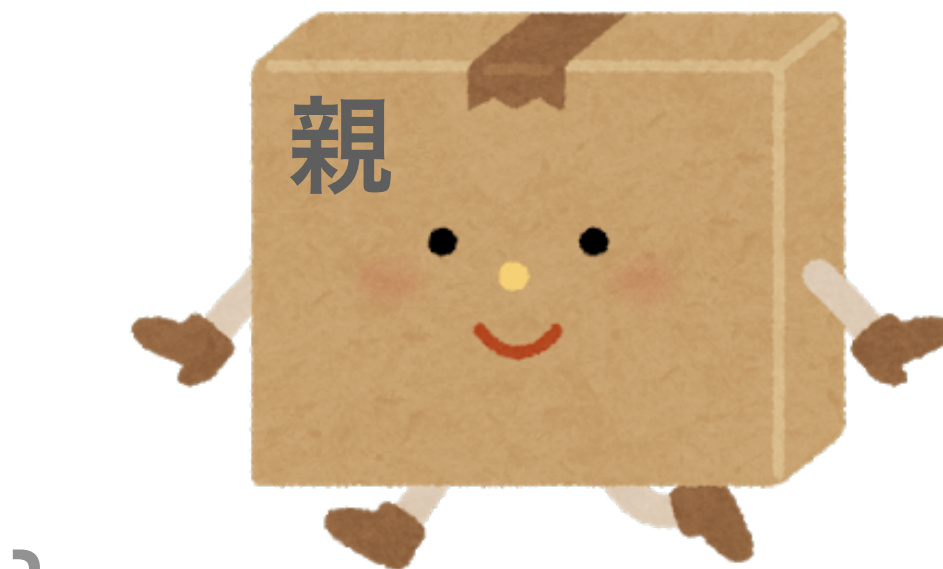
次ページはVueRouterの
ナビゲーションガード

No	タイミング	グローバル (アロー関数)	ルート単位(アロー関数)	コンポーネント内	用途
1	トリガ(クリック)				
2				beforeRouteLeave	本当に離れますか？
3		beforeEach			認証
4				beforeRouteUpdate	watch \$route の代用
5			beforeEnter		
6	非同期ルート を解決				
7				beforeRouteEnter	
8		beforeResolve			
9	ナビゲーション確定				
10		afterEach			
11	DOM更新				
12				beforeRouteEnter(next)	nextのcallbackを呼ぶ

Props Down Event Up

親側

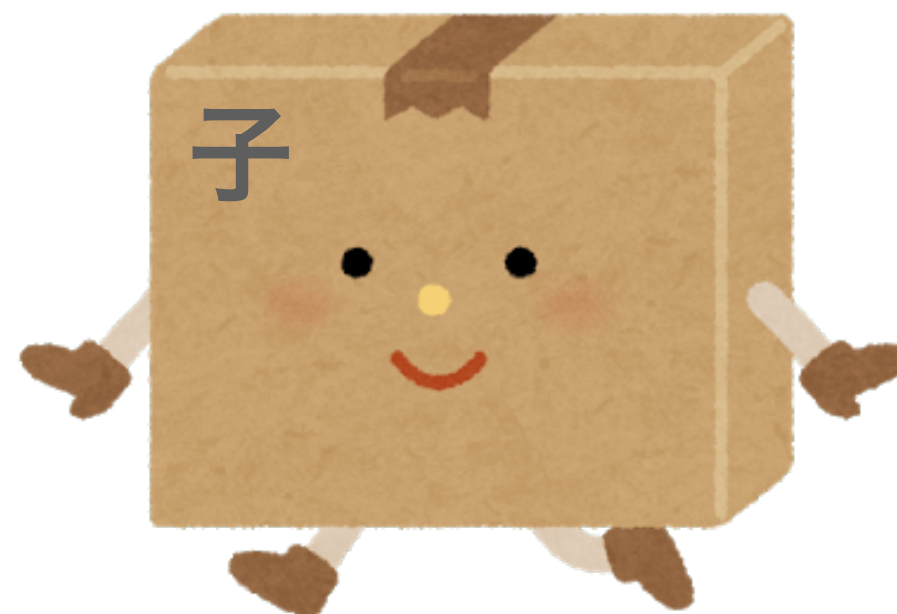
```
<v-btn :title=parent>  
data(){ return  
{ parent : '親データ' } }
```



```
<v-test @custom-event="親のメソッ  
ド名">  
methods:{  
  親のメソッド名(e){  
    console.log(e)  
  }  
}
```

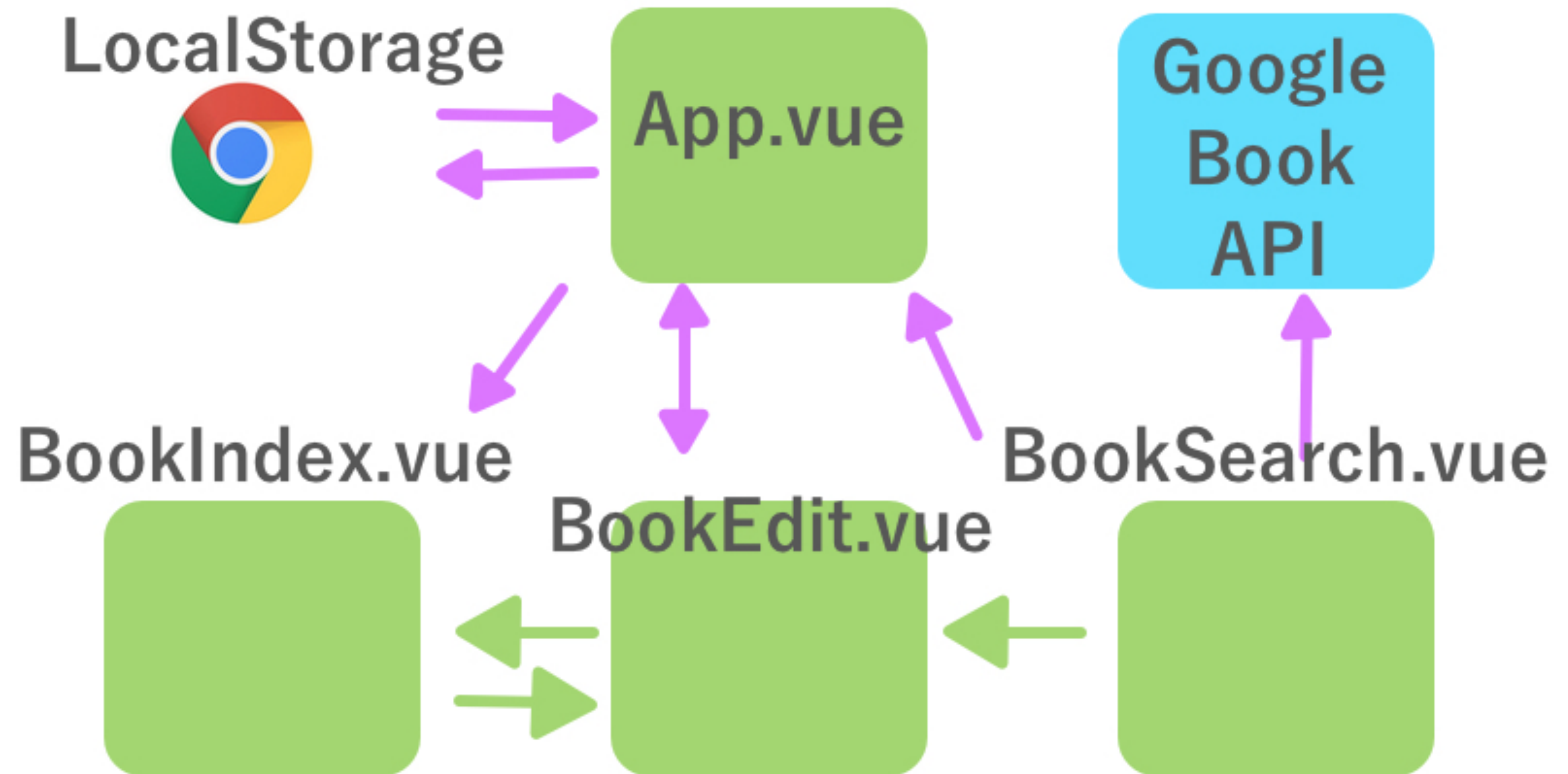
子側への通り道

```
props: {  
  title: {  
    type: String,
```

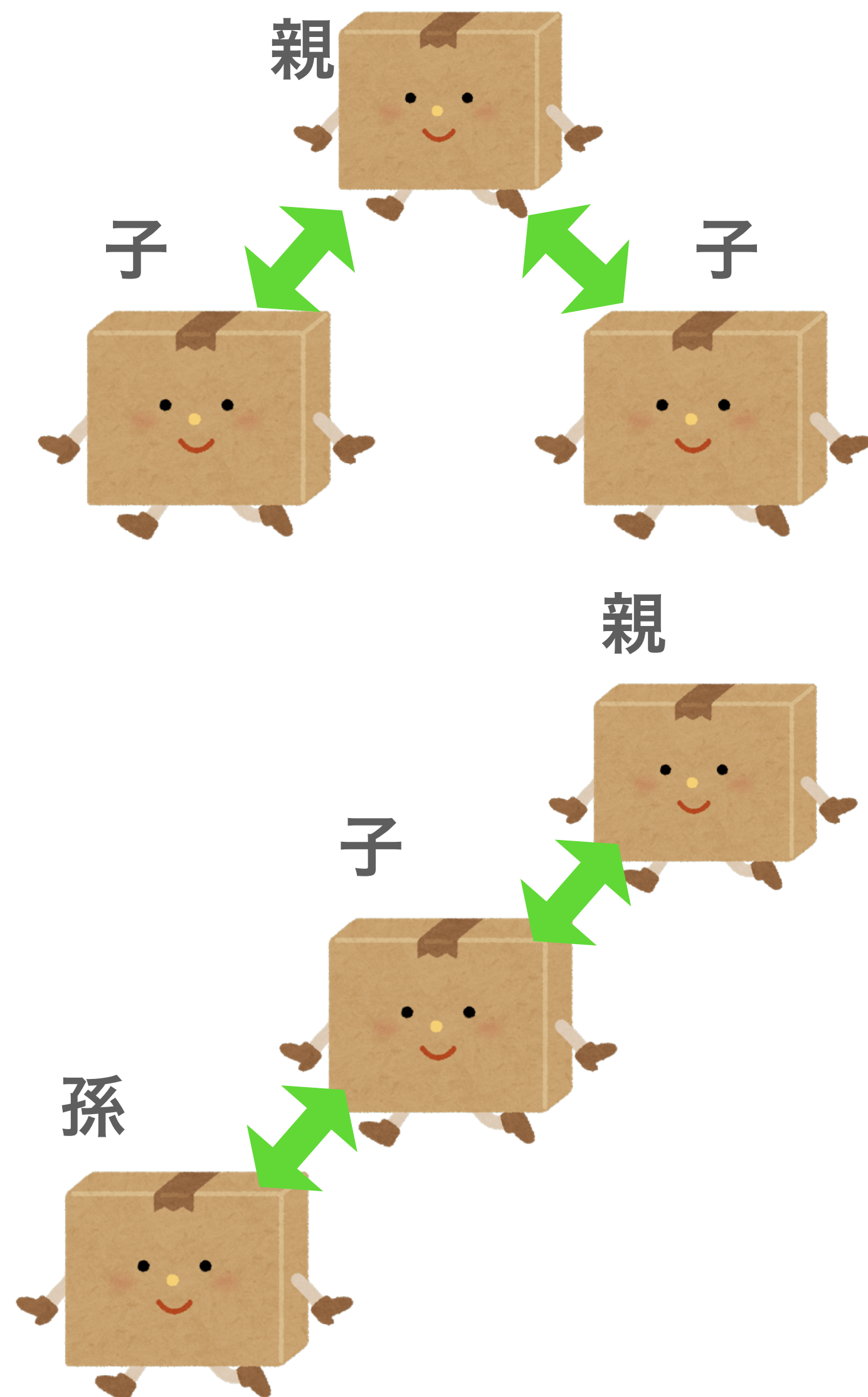


```
<div @click="子のメソッド名">  
methods:{  
  子のメソッド名(){  
    this.$emit('custom-event', 値)}
```

Vue.js講座時の構成図



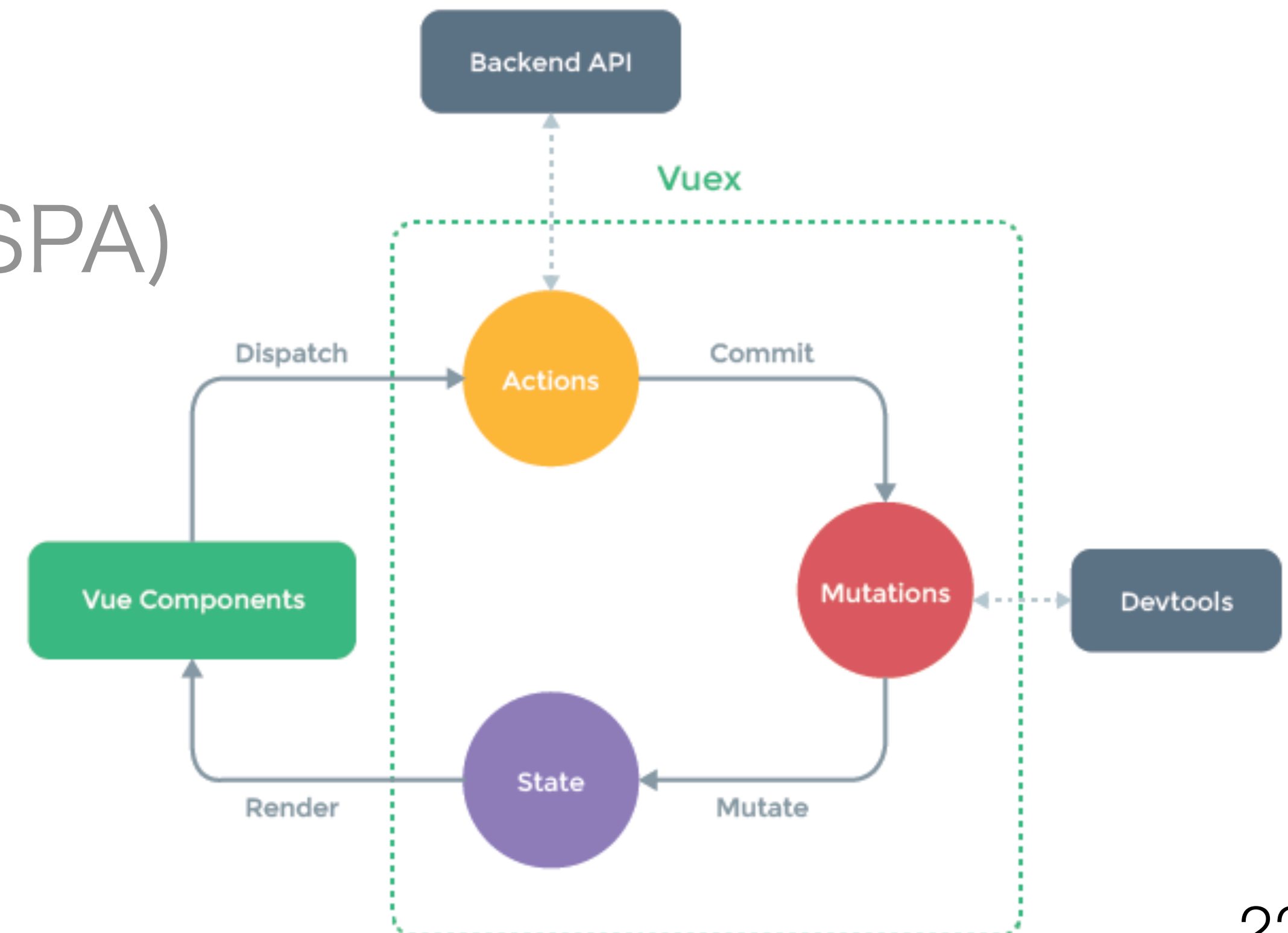
子<->子 親<->孫 など



EventBus



Vuex
(大規模SPA)



Vuetify v2

UIコンポーネント集

v-app (Vuetifyを扱う範囲)

v-main (メイン部分)

v-container (Grid有効範囲)

v-row (Gridの範囲)

v-col (12分割

cols="4" 12のうち4を使う

md="6" mdの幅以下なら6を使う

マニュアル群



Vuejs2

<https://jp.vuejs.org/>

Vue-Router

<https://router.vuejs.org/ja/>

Vuetify

<https://vuetifyjs.com/ja/>

もし表示されない場合は

<https://v2.vuetifyjs.com/ja/>



Nuxtインストール

create-nuxt-appでインストール

Project名: bookapp

Program言語: JavaScript

Package管理: npm

UIフレームワーク: **Vuetify**

Nuxt.jsモジュール: Axios, **PWA**

Lintingツール: ESLint, Prettier

Testツール: Jest

Renderingモード: Single Page App

公開先: Server

開発ツール: なし

CI: None

バージョンコントロール: Git

NuxtJs2 講座のバージョン

講座のバージョンと合わせる方法

package-lock.json ファイルを削除
node_modules フォルダを削除
package.json ファイルを修正
npm install を実行

[https://github.com/aokitashipro/
udemy_nuxt_test/tree/main/section3](https://github.com/aokitashipro/udemy_nuxt_test/tree/main/section3)

ファイル・フォルダ構成

layouts インストール時に生成

assets インストール時に生成

nuxt.config.js

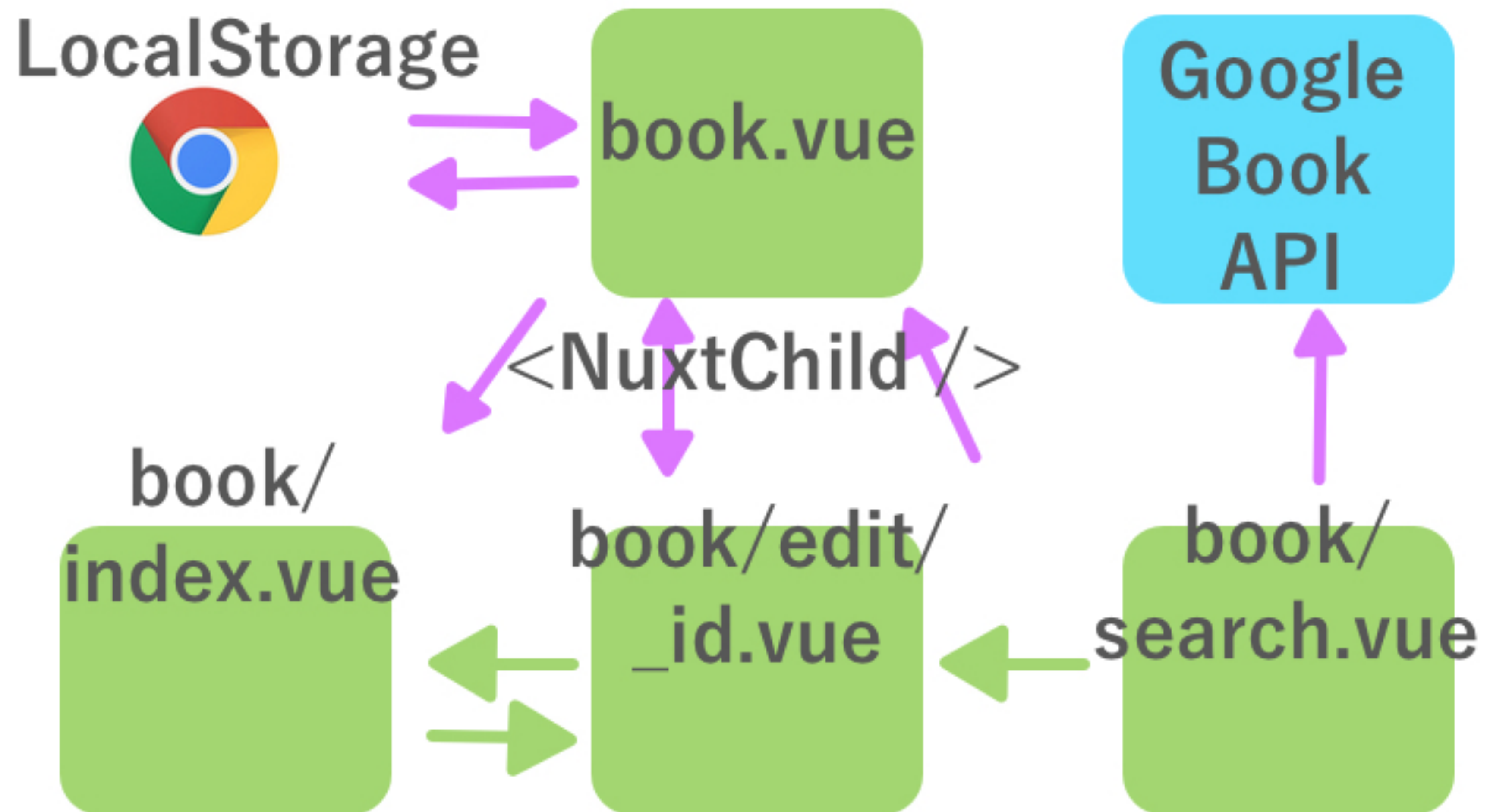
package.json

vuetifyとpwaが追記



BookApp

今回の構成図 Nuxt版



Layouts/default.vue

nuxt.config.js darkモードをfalse

Header, Footerコンポーネントに切り離し

不要なアイコン、dataは削除



NuxtChild

NuxtChild



Nuxtは自動でルーティング作成する
親子関係をつくるためにNuxtChildを使う

親のファイル名と同じフォルダを作成し
フォルダ内にファイルをつくる

親

pages/book.vue <NuxtChild />

子

pages/book/index.vue

pages/book/search.vue

Pages/book/edit/_id.vue



book/search.vue

book/search.vue (抜粋)

ESLintのチェックを無効にする
//eslint-disable-line のコメントを追記

methods内でdataにアクセスする
this.isFound = true

検索結果が0件の場合に表示する
<div v-show="!isFound">0件でした。 </div>

Vuetifyのブレイクポイント

Material Designブレイクポイント			
デバイス	コード	種類	解像度の範囲
 Extra small	xs	小型から大型のスマホ	< 600px
 Small	sm	小型から中型のタブレット	600px > < 960px
 Medium	md	大型タブレットからノートパソコン	960px > < 1264px*
 Large	lg	デスクトップ	1264px > < 1904px*
 Extra large	xl	4k、ウルトラワイド	> 1904px*
デスクトップにおけるブラウザスクロールバーの *-16px			

<v-col cols="12" sm="4">

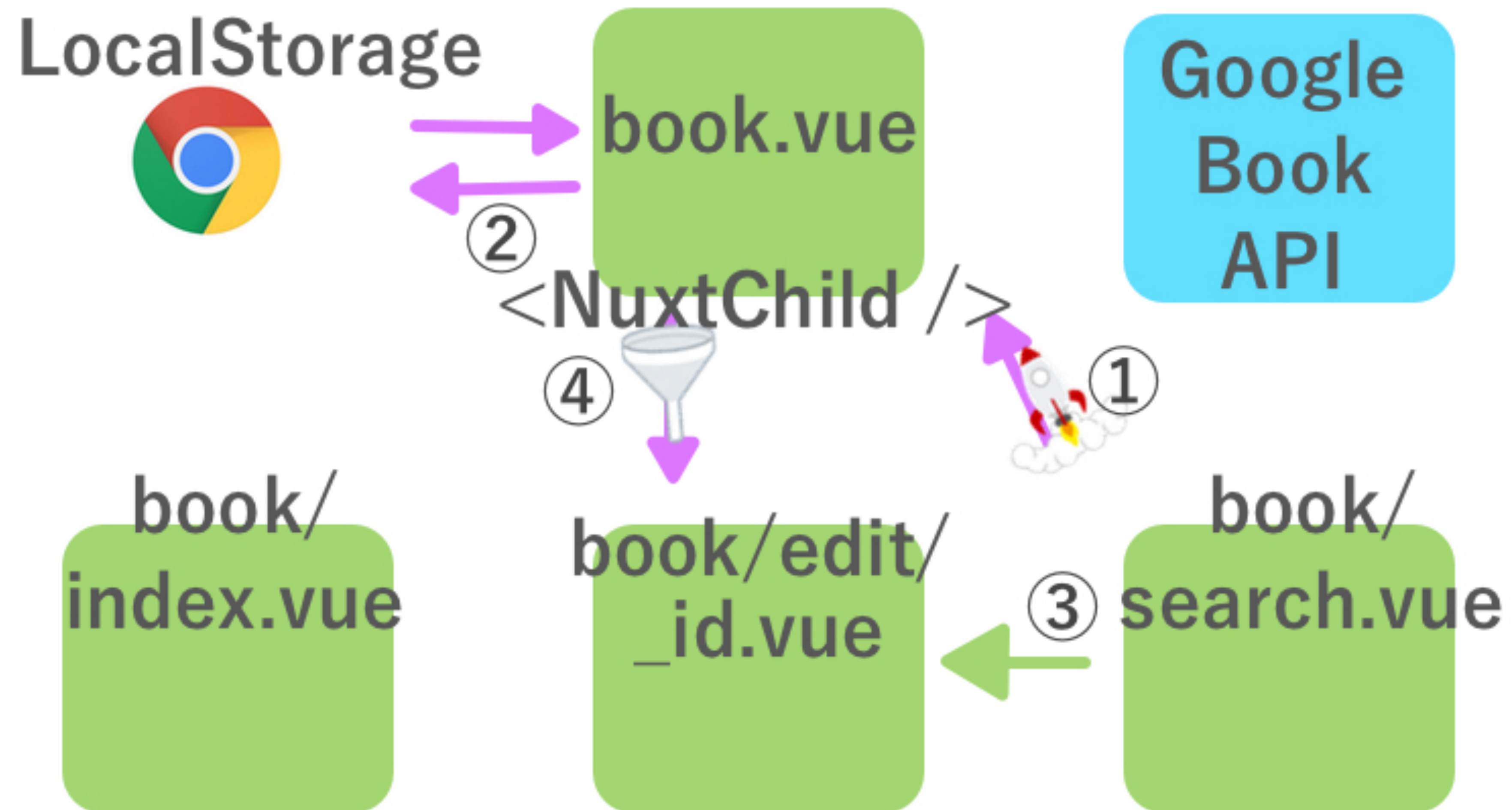
sm幅以上で4/12 ・ ・ 横に3つ並べる

v-forで繰り返し

引数2つめにindexで何番目かを取得できる
引数を複数設定する場合は()をつける
v-forには:keyもつけておく

```
<v-col  
  v-for="(book, index)" in searchResults" :key="book.index"  
  cols="1 2"  
  md="6"  
>  
<v-card>略</v-card>  
</v-col>
```

LocalStorageに追加



Idを取得してページ遷移

最後のid取得コード

```
//console.log(this.books.slice(-1)[0].id)  
this.goToEditPage(最後のid)
```

ページ遷移 (\$routerはvue-routerの機能)
動的に変わるのでテンプレート構文で書く
(`\${動的な値}`)
goToEditPage(id){
 this.\$router.push(`/book/edit/\${id}`)
}



book.vue

親->子へ情報を受け渡す

```
// book.vue
```

```
<NuxtChild :books=books />
```

```
// book/edit/_id.vue
```

```
template内 {{ books }}
```

```
script内
```

```
  props: {
```

```
    books: Array
```

```
}
```

コンポーネント表示は非同期

book.vueのmountedを
createdに変えてみる
(DOM更新前に実施)

※ asyncData()はthisが使えない事もあり
今回はcreatedで対応



book/edit/_id.vue

Propsにデフォルト値を指定

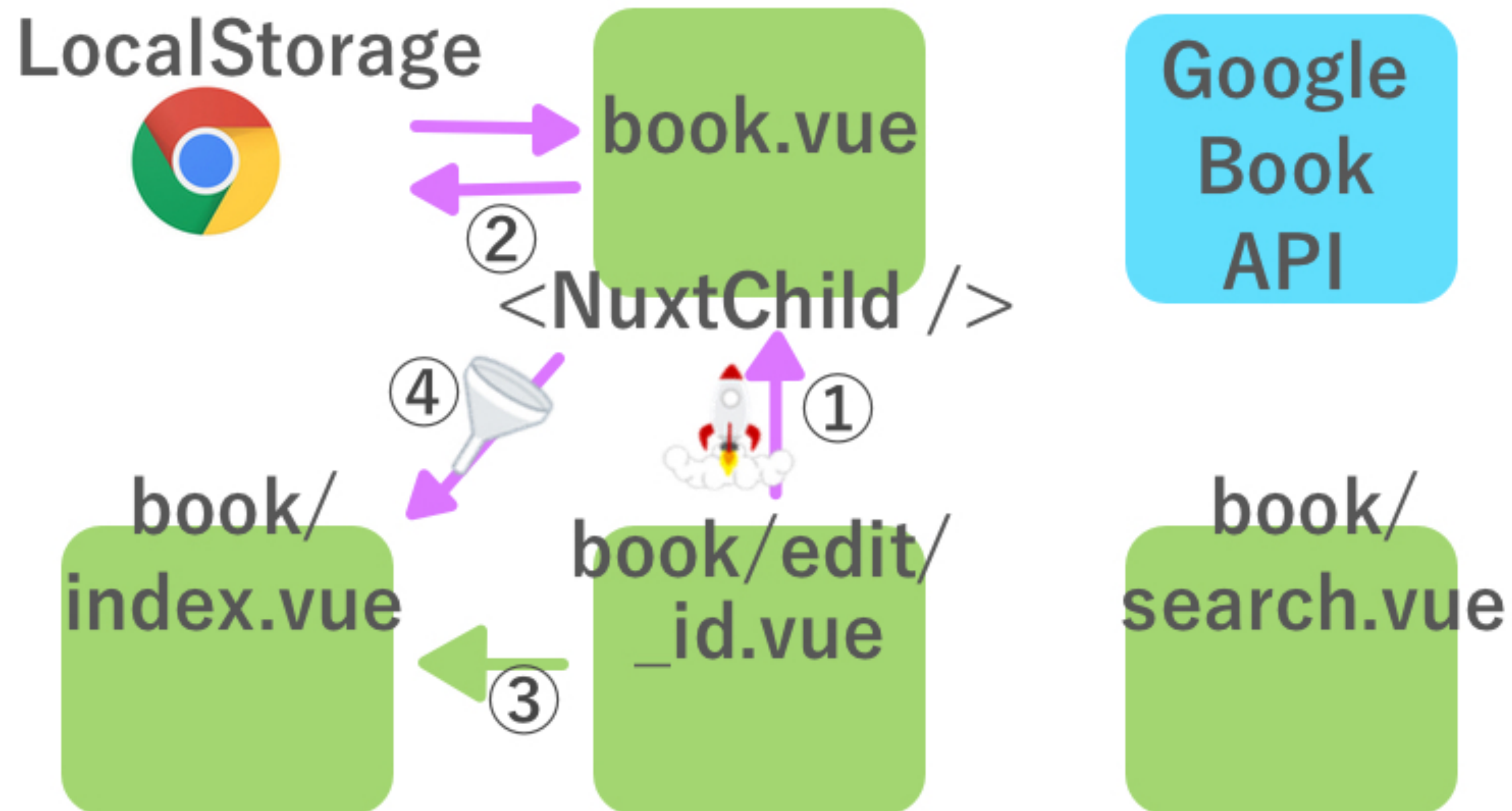
```
props: {  
  books: {  
    type: Array,  
    default: () => {}  
  }  
}
```


edit/_id.vueに本の情報を表示

ページ遷移時に実行したいので
vue-routerのナビゲーションガードを使う

```
beforeRouteEnter(to, from, next){  
  next( vm => {  
    vm.book = vm.books[vm.$route.params.id]  
  })  
},  
data(){  
  return { book: ""}  
}
```

edit/_id.vueで保存後indexに移動



Event Up

```
// Book.vue (親)
```

```
<NuxtChild @update-book-info="updateBookInfo" />
```

```
// book/edit/_id.vue (子)
```

```
<v-btn @click="updateBookInfo">
```

略

```
updateBookInfo(){
```

```
  //第1引数はイベント名、第2引数は親に渡す値
```

```
  this.$emit('update-book-info', {
```

```
    id: this.$route.params.id,
```

```
    readDate: this.date,
```

```
    memo: this.book.memo
```

```
  })
```

```
}
```

Book.vueにメソッド追加

```
// Book.vue (親)
updateBookInfo(e){ // e・・・子から渡ってくる値
  //先にオブジェクトを作る
  const updateInfo = {
    id: e.id,
    readDate: e.readDate,
    memo: e.memo,
    title: this.books[e.id].title,
    image: this.books[e.id].image,
    description: this.books[e.id].description
  }
  this.books.splice(e.id, 1, updateInfo) //idのオブジェクトを置換
  this.saveBooks()
  this.$router.push('/book')
}
```




book/index.vue

Book/index.vue 名前付きルート

抜粋

```
<v-btn  
  :to="{name: 'book-edit-id', params:  
    { id: book.id }}"  
  color="indigo"  
  fab small dark  
>  
<v-icon>mdi-pencil</v-icon>  
</v-btn>
```


読んだ日の修正

```
// book/edit/_id.vue
beforeRouteEnter(){
  next( vm => {
    略
    if( vm.book.readDate ) {
      vm.date = vm.book.readDate
    } else {
      vm.date = new Date(略)
    }
  })
}
```




LocalStorageの 削除

削除ボタンの追加

```
// book.vue (親)
<NuxtChild
  @delete-local-storage="deleteLocalStorage" />
```

```
// book/index.vue (子)
<v-btn @click="deleteLocalStorage"></v-btn>
略
methods:{
  deleteLocalStorage(){
    this.$emit('delete-local-storage')
  }
}
```

削除処理

```
// book.vue (親)
methods: {
  略
  deleteLocalStorage() {
    const isDeleted = '本当に削除してもよろしいですか?'
    if( window.confirm(isDeleted)){
      localStorage.setItem(STORAGE_KEY, '')
      localStorage.removeItem(STORAGE_KEY)
      this.books = []
      window.location.reload()
    } } }
```




補足

contextでリダイレクト

/ -> /book にリダイレクト

```
// index.vue  
asyncData({ redirect }){  
  redirect('/book')  
}
```


サンプルアプリの補足

改善案

- ・重複しないように ->Lodashの_searchなど
- ・削除機能 -> idがずれるのでインクリメントidを別で作成しておく
- ・初回登録 保存->編集 を 編集->保存に
- ・ソート機能->ソート番号を用意
- ・レイアウト機能