

Nuxt.js 基本

講座の紹介



section1,2 Nuxtの紹介

section3 GoogleBookAPI

section4 SPA x Firebase

section5以降 (後日追加予定)

SSR, SSG, PWA, Nuxt3, CompositionAPI

など



NuxtJsの概要

NuxtJs



Vueを拡張したフレームワーク
webアプリ構築によく使う機能が最初から備わっている
Vue, VueRouter, Vuex, VueServerRenderer
vue-meta

ライブラリ
Webpack, babel, etc…

ライブラリの拡張(create-nuxt-appで選択)
Vuetify, tailwindcss, jest, lint, prettier etc…

フロントエンドとバックエンド

クライアントサイドとサーバーサイド



クライアント



NUXTJS



サーバー



NuxtJsを使うメリット



ルーティングの自動生成

通信速度が早い (サイズ軽・コード分割・header管理)

静的サイト(HTML)を簡単に作れる

SSR (JSをサーバー側で実行する・難易度高)

PWA対応(スマホにダウンロードして扱える)が簡単

日本語ドキュメントが多い(Nuxt2)

NuxtJs導入事例



ZOZOテクノロジーズ, LINE,
DMM, note, マンガZERO,
一休com, Retty, さくらインターネット,
Gunosy, DeNA, MERY, クラウドワークス,
東京都 新型コロナ対策サイト
デジタル庁

Vue.jsの年表



2014/02 v0.8 リリース

2015/04 Laravel(PHP)で採用

2016/10 Vue2 リリース

2020/09 Vue3 リリース

NuxtJsの年表



2016/10 初期リリース

2018/01 v1.0.0 リリース

2018/09 v2.0.0 リリース

2021/06 v2.15.7 リリース

2021/10 Nuxt3 パブリックベータリリース

2022/Q1 Nuxt3 リリース予定

NuxtJsのホームページ



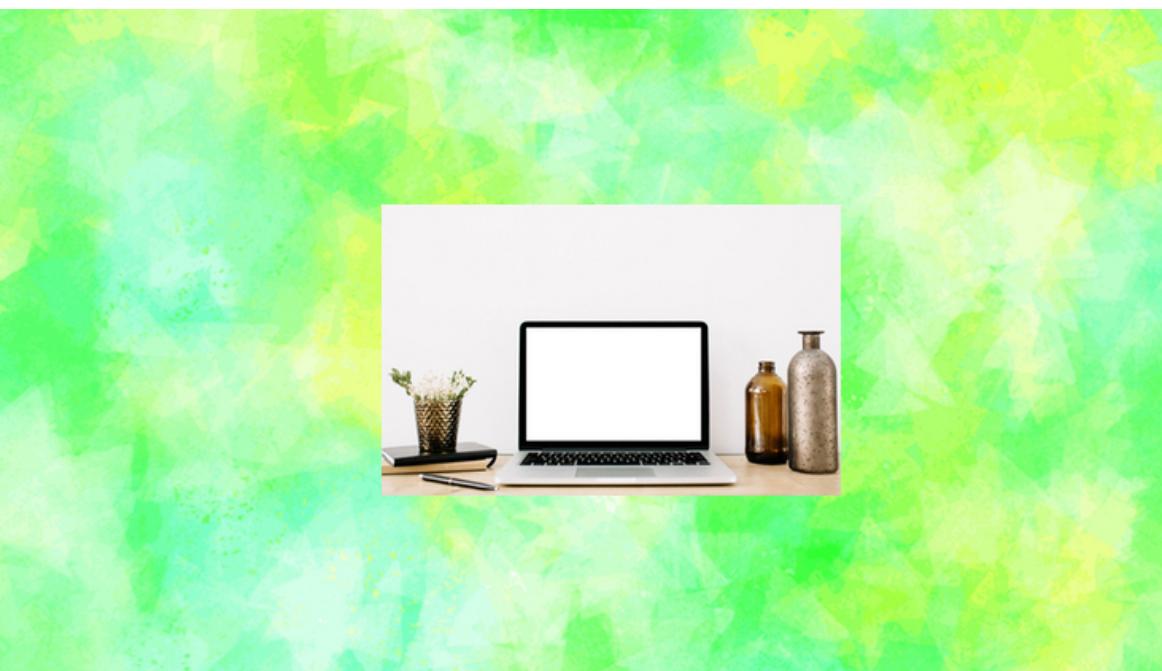
NuxtJs2 <https://nuxtjs.org/ja/>

NuxtJs3 <https://v3.nuxtjs.org/>

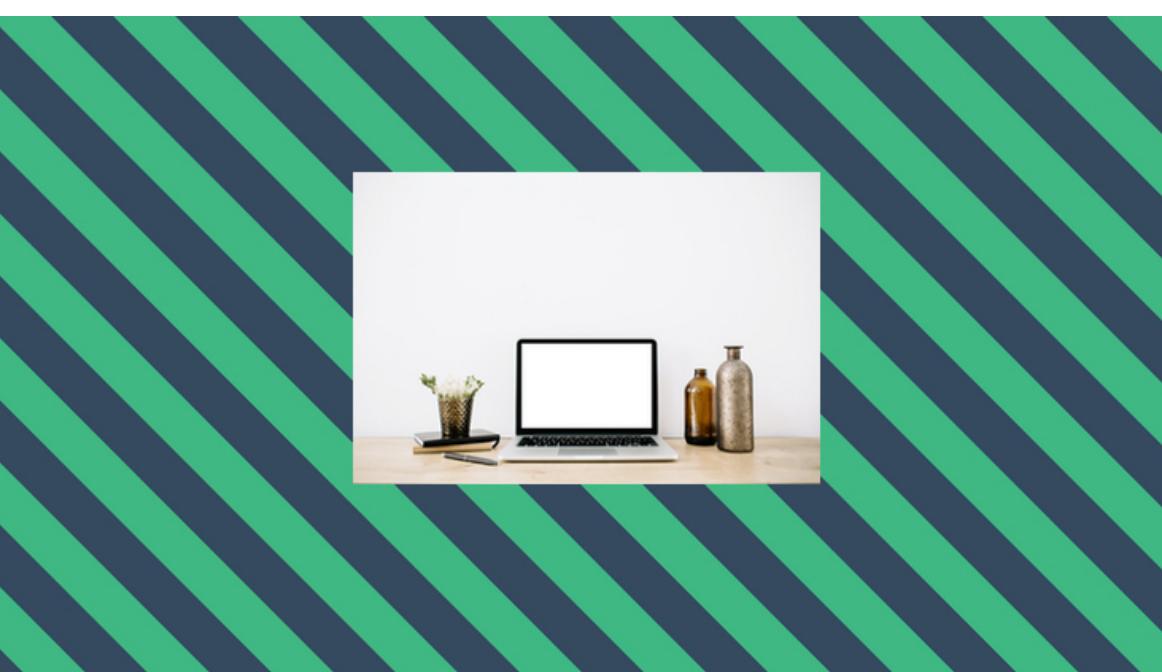
NuxtJs2と3 比較

Feature / Version	Nuxt 2	Nuxt Bridge	Nuxt 3
Stability	😊 Stable	😊 Semi-stable	😢 Unstable
Performance	🏎️ Fast	🚀 Faster	🚀 Fastest
Nitro Engine	✗	✓	✓
ESM support	🌙 Partial	👍 Better	✓
TypeScript	☑ Opt-in	🚧 Faster	✓
Composition API	⚠ Deprecated	✓	✓
Options API	✓	✓	✓ (not recommended)
Components Auto Import	✓	✓	✓
<code><script setup></code> syntax	✗	🚧 Partial	✓
Auto Imports	✗	✓	✓
Webpack	4	4	5
Vite	⚠ Partial	🚧 Partial	🚧 Experimental
Nuxi CLI	✗ Old	✓ nuxi	✓ nuxi
Static sites	✓	✓	🚧

関連する講座



【JavaScript】をとことん



【Vue.js2&Vue.js3対応】



【tailwindcss】
CSSが苦手な人向け



NuxtJs2 インストール

NuxtJs2 インストール



インストールの方法は2種類

nuxtのみインストール

```
npm init -y
```

```
npm install nuxt@2.15.7
```

ライブラリを選びまとめてインストール
(必要なファイル・フォルダ自動生成)
(最新バージョンがインストール)

```
npx create-nuxt-app <project-name>
```

Create-nuxt-app でインストール

Project名:

Program言語: JavaScript

Package管理: npm

UIフレームワーク: None

Nuxt.jsモジュール: Axios

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起動確認 開発用

```
cd nuxt_test  
npm run dev // 開発用  
http://localhost:3000
```

Nuxtの機能でホットリロード対応
(コード編集時 自動で再読み込み)

NuxtJs2起動確認 本番用



プロダクション用(本番用)

npm run build

// webpackでJSとCSSを
ミニファイ(圧縮)や分割(chunk)
(.nuxtフォルダ、distフォルダが生成される)

npm run start

// Nuxtサーバー起動

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/section2](https://github.com/aokitashipro/udemy_nuxt_test/tree/main/section2)



NuxtJs2

フォルダ構成

フォルダ・ファイル構成

.nuxt … ビルドフォルダ(自動生成)
components … コンポーネント用
dist … ビルド後のフォルダ
node_modules … 各種ライブラリ
pages … ページ(自動ルーティング)
static … robots.txtやfaviconなど
store … Vuex用
nuxt.config.js … Nuxtの設定ファイル
package.json … npmの設定ファイル

追加可能なフォルダ群



assets · · CSS, 画像, fontなど
layouts · · ヘッダー・フッターなど
content · · @nuxt/content使用時
middleware · · ユーザー認証など
plugins · · JSプラグイン
modules · · ビルドの動きを拡張

Nuxt側で自動認識するためフォルダ名は固定
もし変える場合は追加設定必要



拡張機能の インストール

VSCode プラグイン



Vetur · · Vueファイルを見やすく
<vue と書くと最低限のコードを生成

動画と合わせるなら · ·

Bracket Pair Colorizer2

Dracula Theme

Highlight Matching Tag

Japanese Language Pack for VS Code

Material Icon Theme

Vue.js devtools



GoogleChromeの拡張機能

Vue.js devtools 6.* · · · Vue.js3対応(beta)

Vue.js devtools 5.* · · · Vue.js2対応



ルーティング

ルーティング Vue.jsの場合



Vue.jsでルーティングする場合
(※VueCLIでインストール想定)

1. 表示したいコンポーネントを作成
2. routes/index.js に1をimport
3. routes/index.js にパスの情報を追記
4. リンクを貼る <router-link to="">
5. 描画 <router-view>

ルーティング Nuxtの場合



Nuxtのルールに合わせて作成すると、
自動でルーティングが設定される

1. pages フォルダ内にコンポーネントを作成
2. ビルド時にルーティング情報を自動で追記
3. リンクを貼る < NuxtLink to="" >
4. 描画 < Nuxt />

Nuxt router.jsについて



自動生成された内容は
.nuxt/router.js で確認可能

※ビルド時にコンパイルがかかるので
.nuxt/router.jsは修正しない

nuxt.config.js内に追記で上書き可能

```
router{
  extendRoutes(){}
}
```

URL設計の参考に REST API

Webシステムを外部から利用するための
呼び出し規約(API)の種類の一つ

※添付はLaravel(PHPフレームワーク)の資料ですが考え方は同じ
です

動詞	URI	アクション	ルート名
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

ルーティングをつくってみる



products/index.vue	・	・	一覧表示
products/create.vue	・	・	新規作成
products/_id/show.vue	・	・	詳細表示
products/_id/edit.vue	・	・	編集

動的に変わるファイル・フォルダは
頭に _ をつける

パラメータ取得は {{ \$route.params.id }}

リンクを張ってみる

products/index.vue

```
<NuxtLink to="/products/create">新規作成</NuxtLink>
```

```
<table>
  <tr>
    <td>1</td>
    <td>テスト</td>
    <td><NuxtLink to="/products/1/show">詳細</NuxtLink></td>
    <td><NuxtLink to="/products/1/edit">編集</NuxtLink></td>
  </tr>
</table>
```

ルーティング (拡張)



拡張したい場合は nuxt.config.js 内で
router.extendRoutes プロパティを使う

```
export default{
  router: { extendRoutes(){}
}
```

用途：
リダイレクト処理、
ルートコンポーネントにpropsを渡す



オートインポート

オートインポート機能



Nuxt v2.13で追加
nuxt.config.js内 components: true
の場合、
componentsフォルダ内のvueファイルが
自動でインポートされる

オートインポート機能 フォルダ名有無

Nuxt v2.15 (2021/2)

componentsフォルダ内にフォルダがある場合、フォルダも記載
components/base/Header.vue

<BaseHeader />

フォルダ名を省略する場合はnuxt.config.jsを編集

```
components:{  
  dirs: [  
    '~/components/',  
    '~/components/base'  
  ]  
}  
<Header />
```

重複する場合は下層のファイルが優先される



レイアウト

レイアウトのカスタマイズ



編集する場合は
layouts/default.vue を作成する

```
<template>
  <div>
    <BaseHeader />
    全ページに表示
    <Nuxt />
  </div>
</template>
```

別のレイアウトを指定する場合



layouts/blog.vue を作成する

pages/blog.vue の中で layoutを指定する

～略～

```
export default{
  layout: 'blog'
}
```

エラーページの編集方法



layouts/error.vue を追加して編集



ライフサイクル

ライフサイクル 簡易表

No.	特徴	クライアント (ブラウザ)	サーバー
1			ServerMiddleware
2	Vuexストア初期化		nuxtServerInit
3	認証関連 ルートパラメータ		RouteMiddleware
4	ページのバリデーション		validate
5		asyncData / fetch(引数あり)	
6	VueJs2はここから		beforeCreate
7	ここからthisが 使える		created
8			fetch
9		beforeMount	
10	DOM生成後	mounted	

各フックの呼び出しタイミング

Pages/index.vueで実験

```
asyncData(){
  console.log('asyncData')
},
created(){
  console.log('created')
},
fetch(){ // 引数なし
  console.log('fetch')
},
fetch( context ){ // 引数あり
  console.log( context.route )
}
```



context

context(文脈)

Nuxtで使える様々な情報を内包するオブジェクト
asyncData, fetch, plugins, middlewares, modules,
store/nuxtServerInit, などNuxtのライフサイクル内で使用可能
(Vuexのactionで使うcontextとは別物)

```
asyncData( context ){
  console.log( context.route )
}

// JS ES6 分割代入
asyncData({ route, isDev }){
  console.log( route.name )
  console.log( isDev )
}
```

Contextでよく使うキー

context.route · · ルーティング情報

context.store · · Vuex store内のデータ参照

context.error · · APIリクエスト失敗時の対応

context.redirect · · リダイレクト

context.query · · クエリパラメータ

context.isDev · · 開発中かどうか

context.req · · get/postの判定

context.app · · ルートのVueインスタンス

context.\$axios · · axios

プラグイン(plUGINS)の注入も可能



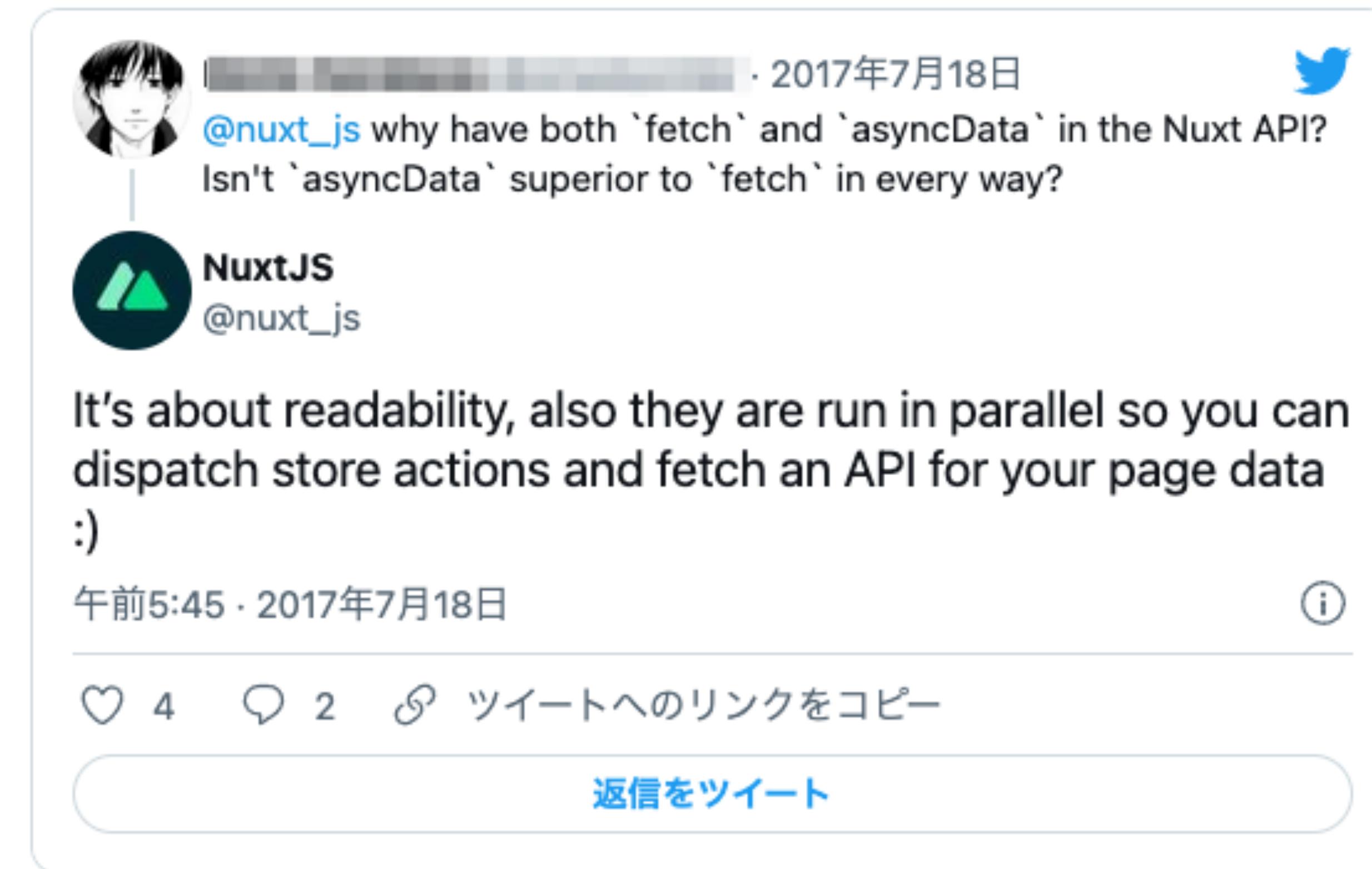
asyncData と
fetch

asyncDataとfetch

	asyncData	fetch
実行タイミング	createdの前	(引数有りの場合) createdの前
とれる引数	context	context
用途	取得したデータを ページコンポーネント内で 使う	取得したデータを Vuexストアに 入れる際に使う

動作的にはほぼ同じ
役割を分けてコードを読みやすくするため

Nuxt公式ツイッターにて



A screenshot of a Twitter conversation. The user (@nuxx_js) asks why both 'fetch' and 'asyncData' are in the Nuxt API, questioning if 'asyncData' is superior to 'fetch' in every way. The NuxtJS account (@nuxx_js) responds that it's about readability and parallel execution. Below the tweet, there are engagement metrics and a reply button.

2017年7月18日

@nuxx_js why have both 'fetch' and 'asyncData' in the Nuxt API?
Isn't 'asyncData' superior to 'fetch' in every way?

NuxtJS
@nuxx_js

It's about readability, also they are run in parallel so you can dispatch store actions and fetch an API for your page data :)

午前5:45 · 2017年7月18日

4 2 ⚡ ツイートへのリンクをコピー

返信をツイート



middleware

Middleware



ページ表示される前に実行

ex) ログイン済みユーザーかの確認

middleware/*.js を作成

- ・全てのページ・・nuxt.config.jsにmiddleware追記
- ・一部のページ(pages, layouts)
ファイル内に middleware: ‘ミドルウェア名’
を記載

呼び出し順 nuxt.config.js -> layouts -> pages

Middleware 作り方



middleware/middlewareCheck.js

middlewareにもcontextが使える

```
export default function ( { route } ) {
  console.log('middleware Check')
  console.log('middleware:', route.name)
}
```

Middleware 全てのページで実行



nuxt.config.js

```
router: {  
  middleware: 'middlewareCheck'  
}
```

複数指定する場合は配列で指定 [,]

Middleware 一部ページで実行



pagesやlayouts内ファイルにミドルウェア名追記

```
export default {  
  middleware: 'middlewareCheck'  
}
```

直接記入する方法もあり

```
export default {  
  middleware( context ){  
  }  
}
```

Ex) ログインしないとリダイレクト



middleware/auth.js

```
export default function ({ redirect, store, route }) {
  const user = store.state.user
  if(!user && route.path !== '/login') {
    redirect('/login')
  }
}
```



Plugins

Plugins 追加機能



dayjs (日付を表示する 軽量)

<https://www.npmjs.com/package/dayjs>

インストール

npm install dayjs —save

Plugins ファイルを作成



plugins/dayjs.js を作成

```
import 'dayjs/locale/ja' //日本時間をimport
import dayjs from 'dayjs'
dayjs.locale('ja') // 日本時間を設定
```

```
export default ({ app }, inject) => {
  inject('dayjs', string => dayjs(string))
})
```

Plugins コンフィグファイルに追記



nuxt.config.jsに追記

Vueインスタンス化する前に呼び出される

```
plugins: [  
  '@/plugins/dayjs',  
],
```

Plugins dayjsを使ってみる

Pages/About.vue

```
<template>{{ now }}</template>
<script>
export default {
  data(){
    return {
      now: null
    },
  }
}
```

```
mounted(){
```

```
  this.now = this.$dayjs().format('YYYY-MM-DD:HH:mm:ss')
}
```

```
</script>
```



Modules

モジュールとプラグインの違い



Modules

Nuxt起動時に呼び出される
ビルド時に読み込んだり
ビルドの動きを変えたい場合

Plugins

\$rootやcontextから呼び出して使う

用語集



全部 部品

プラグイン・・ソフトウェアに機能を追加する小さなプログラム(アドインともいう)

モジュール・・機能単位、交換可能な構成部分(axios, pwa …)

ライブラリ・・ある特定の機能をもったプログラムを、他のプログラムからりようできるように部品化し、1つのファイルにまとめたもの。

コンポーネント・・何らかの機能をもったプログラムの部品

Nuxt用のモジュール



<https://modules.nuxtjs.org/>

<https://github.com/nuxt-community/awesome-nuxt>
Officialなど

Modules ファイルを作成

modules/example.js を作成

```
export default function (){
  console.log('moduleのテスト')

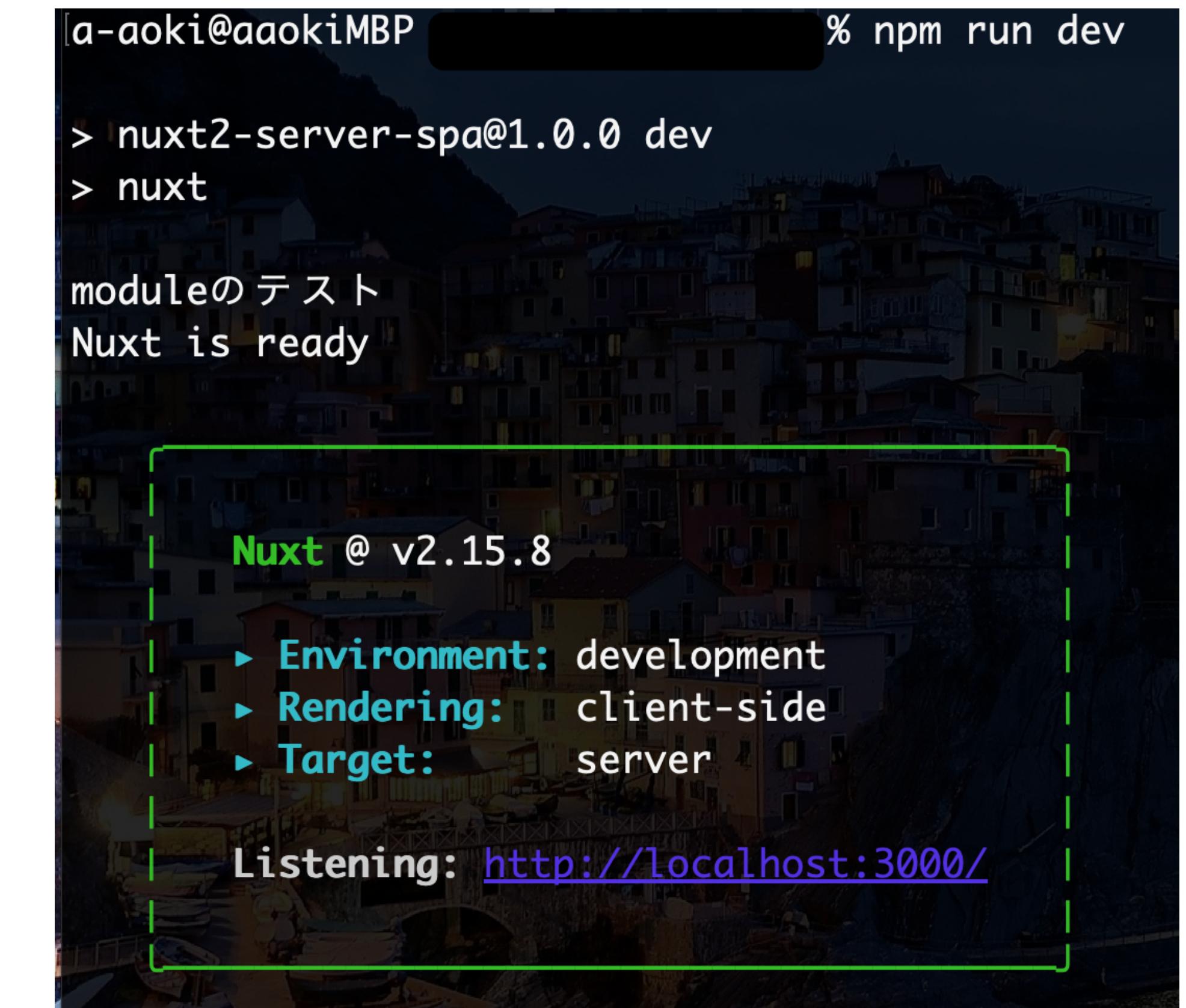
  // Hookでどのタイミングで実施するか指定もできる
  // このthisはModuleContainerと呼ばれるオブジェクト
  this.nuxt.hook('ready', async nuxt => {
    console.log('Nuxt is ready')
  })
}
```

Modules コンフィグファイルに追記

nuxt.config.js に追記

```
//開発限定の拡張機能  
buildModules: [  
  '@/modules/example'  
,
```

```
//本番環境でも実行  
modules: []
```



a-aoki@aokiMBP % npm run dev

> nuxt2-server-spa@1.0.0 dev
> nuxt

moduleのテスト
Nuxt is ready

Nuxt @ v2.15.8

- ▶ Environment: development
- ▶ Rendering: client-side
- ▶ Target: server

Listening: <http://localhost:3000/>

A terminal window showing the command "npm run dev" being executed. The output shows the "nuxt" module being installed and the "Nuxt is ready" message. A green dashed box highlights the "Listening" information at the bottom of the terminal output.

npm run dev 実行時ターミナル画面で確認できる

Modules hookの種類

- nuxt.hook('ready') ・・・ 動作準備完了時
- nuxt.hook('error') ・・・ フック呼出時のエラー
- nuxt.hook('close') ・・・ インスタンス正常終了の手前
- nuxt.hook('listen') ・・・ リッスンを始めた時
- nuxt.hook('modules:done') ・・・ モジュール読込後
- nuxt.hook('render:before') ・・・ レンダー前
- nuxt.hook('build:compile') ・・・ コンパイラ前
- nuxt.hook('generate:before') ・・・ ページ生成前



assets フォルダ

assets (資産) 画像

assets/images
/css
/fonts

assets/images 配下に画像配置

 で表示

assets (資産) CSS

assets/css/styles.css を作成

```
.red-b{  
    Border: 1px solid red;  
}
```

nuxt.config.js内のcssに追記

```
css: [  
    '@/assets/css/style.css'  
]
```

コンポーネント内で <div class="red-b">assets/cssのテスト</div>