

# Vue Router



# SPA

# Single Page Application



メリット

ページ移動がスムーズ

より高度なWeb表現

ネイティブアプリの代用(PWA)

デメリット

初回ページ読み込みに時間がかかる

実装コストがかかる

SEOが十分でない(改善中)

->対策 SSR SSG 他



# Vue Router



# Vue Router テンプレート側

---

インストール方法 CDN or NPM(VueCLI)  
<https://router.vuejs.org/ja/guide/#html>

リンク

```
<router-link to="/foo">Go to Foo</router-link>  
<router-link to="/bar">Go to Bar</router-link>
```

描画

```
<router-view></router-view>
```

# Vue Router JS側

//コンポーネント

```
const Foo = { template: '<div>foo</div>' }  
const Bar = { template: '<div>bar</div>' }
```

//ルート設定

```
const routes = [  
  { path: '/foo', component: Foo },  
  { path: '/bar', component: Bar }  
]
```

//ルーターのインスタンス化

```
const router = new VueRouter({  
  routes  
})
```

Vueインスタンス時にrouterも指定


```
const app = new Vue({  
  router  
}).$mount('#app')
```

# Vue CLI でのインストール

vue create xxx // CUI コマンド

history mode  
->urlに#がつかない

```
? Please pick a preset: Manually select features
? Check the features needed for your project:
  ● Choose Vue version
  ● Babel
  ○ TypeScript
  ○ Progressive Web App (PWA) Support
  > ● Router
  ○ Vuex
  ○ CSS Pre-processors
  ● Linter / Formatter
  ○ Unit Testing
  ○ E2E Testing
```



# ファイル・フォルダ 構成



# CDN時 / VueCLI時

//コンポーネント

```
const Foo = { template: '<div>foo</div>' }  
const Bar = { template: '<div>bar</div>' }
```

//ルート設定

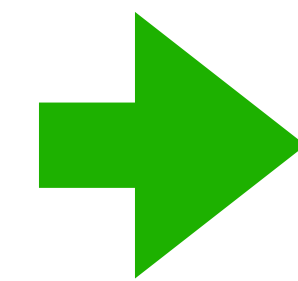
```
const routes = [  
  { path: '/foo', component: Foo },  
  { path: '/bar', component: Bar }  
]
```

//ルーターのインスタンス化

```
const router = new VueRouter({  
  routes  
})
```

Vueインスタンス時にrouterも指定

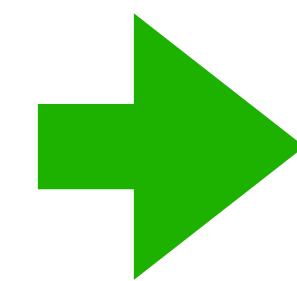
```
const app = new Vue({  
  router  
}).$mount('#app')
```



.vueファイル




router/index.js



main.js

# ファイル・フォルダ構成



src/

router/index.js ルーターの設定ファイル

views/About.vue

views/Home.vue

App.vue router-viewなど記載

main.js エントリーポイントでrouter読込



router-linkのprops

# router-linkのprops




to="" パス (router/index.jsで設定)

tag="" aタグ以外

activeClass クラス名設定

exactActiveClass 完全一致 クラス名設定



# \$router と \$route



# \$router と \$route

---

\$router VueRouterインスタンス(全体)  
ページ遷移で使う `$router.push()` など


\$route ルートオブジェクト(現在のページ)

現在のページ情報

ex) URLに含まれるパラメータ情報

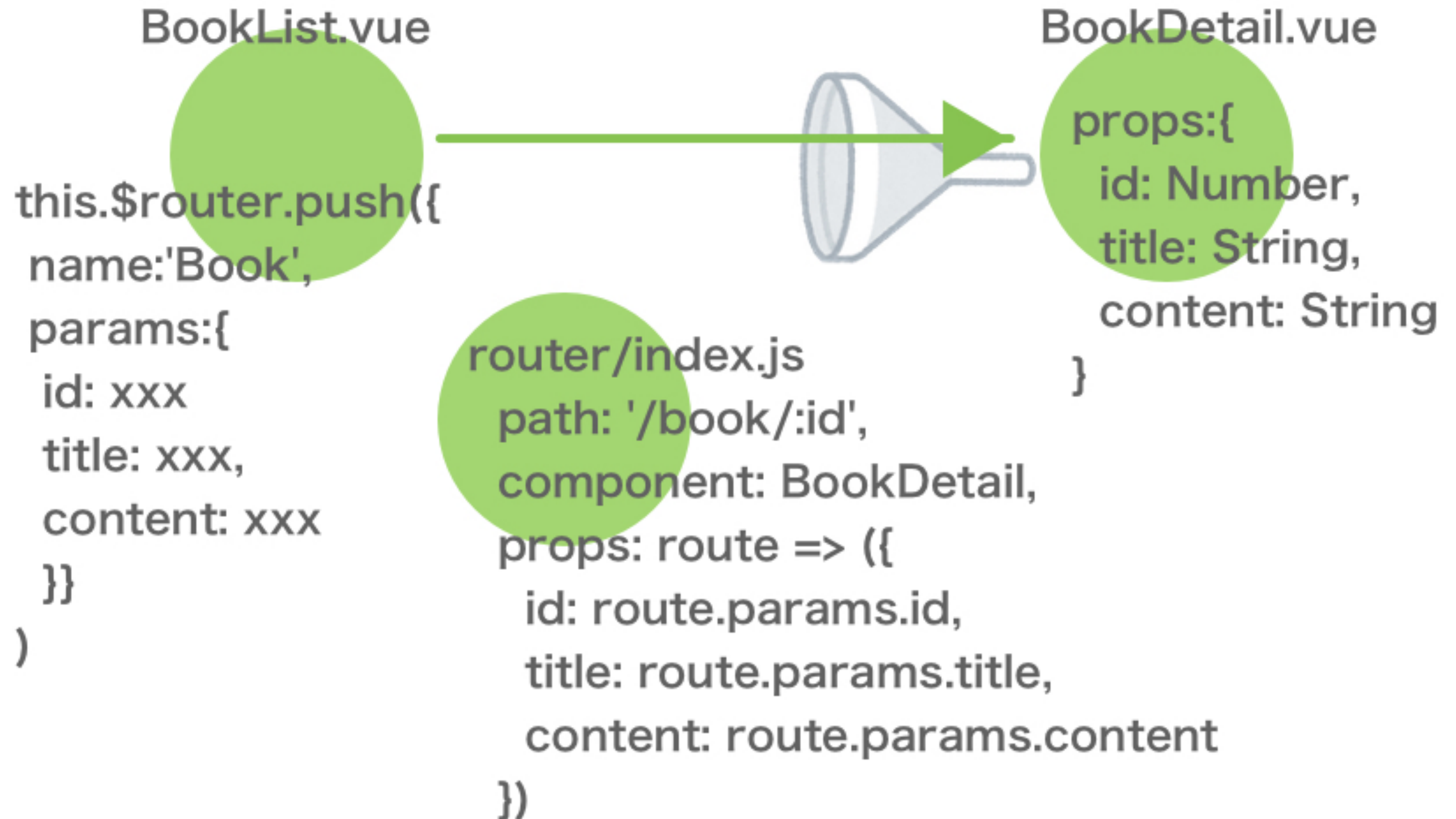
`$route.params`

`$route.params.id`



動的パラメータ  
+propsを使った  
ルーティング

# 動的パラメータ+Props



# 補足



コンポーネント間でパラメータを渡した後、  
再読み込みをするとデータが消える  
->一般的にはサーバー内に情報があり  
Vue.jsはAjaxで取得して表示するだけ  
どのコンポーネントでデータ取得するか要検討

Number(xxx) で数値に変換



# 動的パラメータ +watch




# 動的パラメータ+watch

同じコンポーネントを  
動的パラメータで切り替えると  
ライフサイクルフック(createdなど)が呼ばれない

```
watch:{  
  $route(to, from){  
}  
}
```

とすると\$routeオブジェクトを監視し  
変更時に処理をすることができる



# リダイレクト 404ページ

# Route/index.js で設定

---

```
path: '*' // マッチしないものの全て  
// redirect: '/' 指定したパスに移動  
component: NotFound
```

NotFound.vue を作成して  
そちらを表示させる方法もよく使われる



ネストされたルート

# ネストされたrouter-view

---

<router-view />の中に<router-view />  
router/index.jsで設定

```
component: User,  
children:[  
  {path:'profile', component:UserProfile},  
  {path:'post', component: UserPost}  
]
```





名前付きrouter-view

# 名前付き router-view

---

```
<router-view />
```

```
<router-view name="sub" />
```

```
router/index.js
```

```
components:{
```

```
  default: Home,
```


```
  Sub: HomeSub
```

```
}
```



# トランジション + router-view

# トランジション + router-view



```
<transition name="fade" mode="out-in">  
  <router-view />  
</transition>
```



# ナビゲーション ガード



# ナビゲーションガード

クリックしコンポーネントが切り替わる間に  
処理を実行できる機能

Before.vue



\$route

After.vue



\$route

`beforeEach(to, from, next){}`

`next()` 移動する `next(false)` 移動しない

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



# Historyモードの補足

# historyモードの補足

---

簡易サーバーは `public/index.html` に振り分けている

`dist` に吐き出す場合は自分で設定する必要がある

`index.js` に `base` を設定

`base: process.env.BASE_URL`

`vue.config.js` 内に `publicPath:`,

`public` フォルダに `.htaccess` を作成