

Vue.js Hands on

- 目次
 - Vue.jsとは
 - apiにブラウザからrequestを送る
 - responseをブラウザに表示する
 - いい感じにきれいにする

事前準備

for mac

- nodeのinstall

```
brew update  
brew install node
```

- vue/cliのinstall

```
npm install -g @vue/cli
```

- vue createしたあと

```
npm install --save element-ui  
npm install --save axios
```

Vue.jsとは

概要👤

- 2014年にリリースされた、JavaScriptのフレームワーク
 - [Vue.js公式サイト](#)
 - JavaScriptのフレームワークは他にも(React, AngularJS, jQuery...)
 - Adobeとか任天堂とかでも採用されてるらしい

Vue.jsのいいところ

- 公式Documentがちゃんとしてる(公式ページ一通りやると、いい感じに学べる)

やってみましょう 🧑

- **1** Vueプロジェクトを作成
- **2** 新しいComponentを作成
- **3** apiを呼んでみる
- **4** inputを反映してapiを呼んでみる

やってみましょう 🧑

- **1** Vueプロジェクトを作成

Vueプロジェクトを作成

- 好きなdirectoryを作って、そこでプロジェクト開始
- terminal

```
$ mkdir sample-vue-project  
$ vue create <folder-name> // <folder-name>がheaderとかになるよ
```

- 必要に応じて設定を選ぶ

Vue CLI v3.0.1

? Please pick a preset: Manually select features

? Check the features needed **for** your project:

- ☒ Babel
- ☐ TypeScript
- ☐ Progressive Web App (PWA) Support
- ☒ Router
- ☒ Vuex
- ☒ CSS Pre-processors
- ☒ Linter / Formatter
- ☒ Unit Testing
- ☐ E2E Testing

Vue CLI v3.0.1

? Please pick a preset: Manually select features

? Check the features needed **for** your project: Babel, Router, Vuex, CSS Pre-processors, Linter

? Use history mode **for** router? (Requires proper server setup **for** index fallback in production) **Yes**

? Pick a CSS pre-processor (PostCSS, Autoprefixer **and** CSS Modules are supported by default): SCSS/SASS

? Pick a linter / formatter config: Basic

? Pick additional lint features: Lint on save

? Where **do** you prefer placing config **for** Babel, PostCSS, ESLint, etc.? **In package.json**

? Save this as a preset **for** future projects? **No**

Vueプロジェクトを作成

- 設定を選び終わると、必要なpackageとかをvue/cliが準備してくれます

```
Vue CLI v4.3.1
✨ Creating project in /Users/sasakikensuke/marp-next/vue-handson/sample-prj/client.
⚙ Installing CLI plugins. This might take a while...
///
```

- 完了したらこんな感じになる

```
[4/4] 🔨 Building fresh packages...
success Saved lockfile.

$ cd <foldername>
$ yarn serve
```


Vueプロジェクトを作成

- お試しserverを起動してみる

```
$ cd <foldername>  
$ yarn serve
```

- `localhost:8080/` にサンプルページが立ち上がれば作成成功

やってみましょう 🧑

- **1** Vueプロジェクトを作成
- **2** 新しいComponentを作成

新しいComponentを作成

- 見た目をいい感じにするためのpackageを追加する [ElementUI](#)

```
$ npm install --save element-ui
```

新しいComponentを作成

1 routingを追加する

- router/index.js

```
Vue.use(VueRouter)

const routes = [
  //
  {
    path: '/search',
    name: 'Search',
    component: () => import(/* webpackChunkName: "about" */ '../views/Search.vue')
  },
]
```

新しいComponentを作成

2 viewを追加する

- views/Search.vue

```
<template>
  <div id="search" style="width:80%; margin:auto;">
    <h1>Search application</h1>
  </div>
</template>
```

新しいComponentを作成

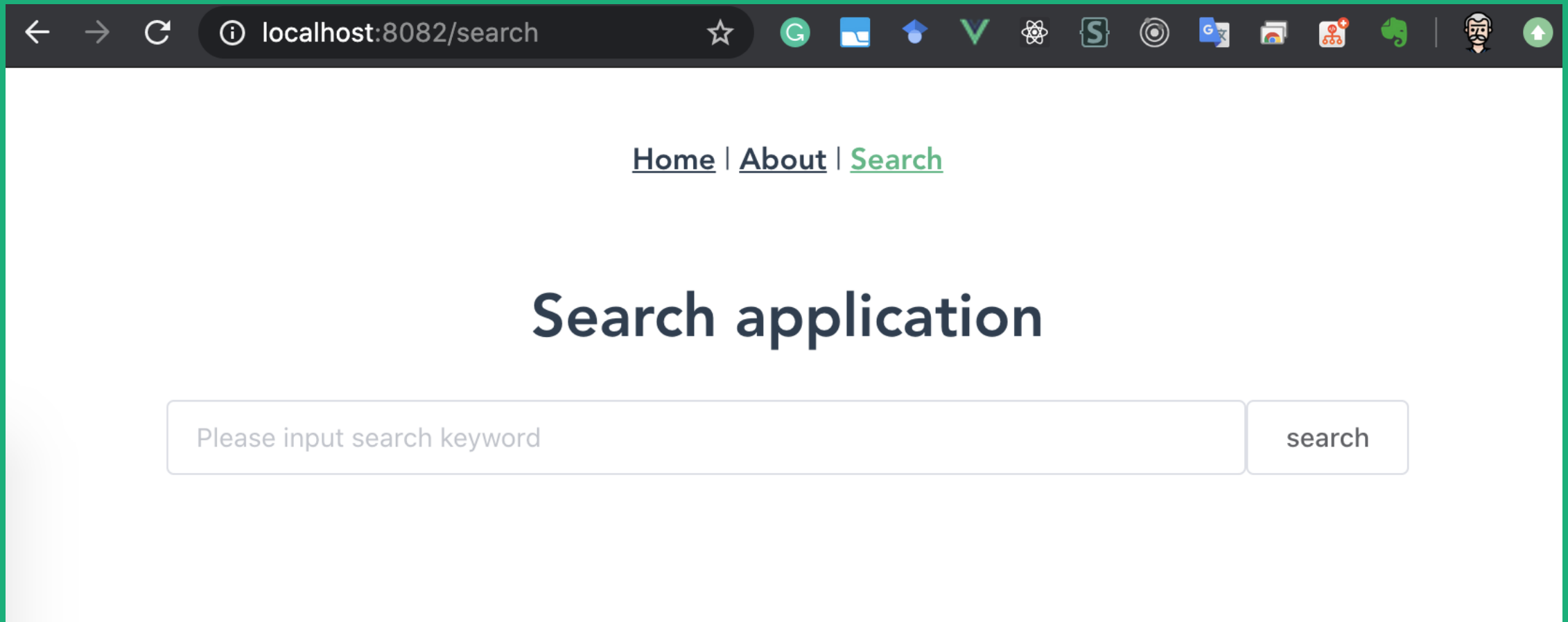
3 formとbuttonを追加する

- views/Search.vue

```
<template>
  <div id="search" style="width:80%; margin:auto;">
    <h1>Search application</h1>
    <div id="form-div" style="display:inline-flex; width:100%;">
      <el-input placeholder="Please input search keyword" v-model="input"></el-input>
      <el-button>search</el-button>
    </div>
  </div>
</template>
```

新しいComponentを作成

3 formとbuttonを追加する (こんなかんじ)



やってみましょう 🧑

- **1** Vueプロジェクトを作成
- **2** 新しいComponentを作成
- **3** apiを呼んでみる

apiを呼んでみる

1 methodを定義

- views/Search.vue

```
<template>
  //
</template>

<script>

import axios from 'axios'

export default {
  methods: {
    getAllTodos() {
      axios.get("https://jsonplaceholder.typicode.com/todos/")
        .then(response => {
          this.todos = response.data
          console.log(this.todos)
        })
    }
  }
}
```

apiを呼んでみる

2 dataを定義

- views/Search.vue

```
<script>
export default {
  methods: {
    getAllTodos() {
      //
    }
  },
  data() {
    return {
      todos: [], // responseを格納する配列を定義
    }
  }
}
</script>
```

apiを呼んでみる

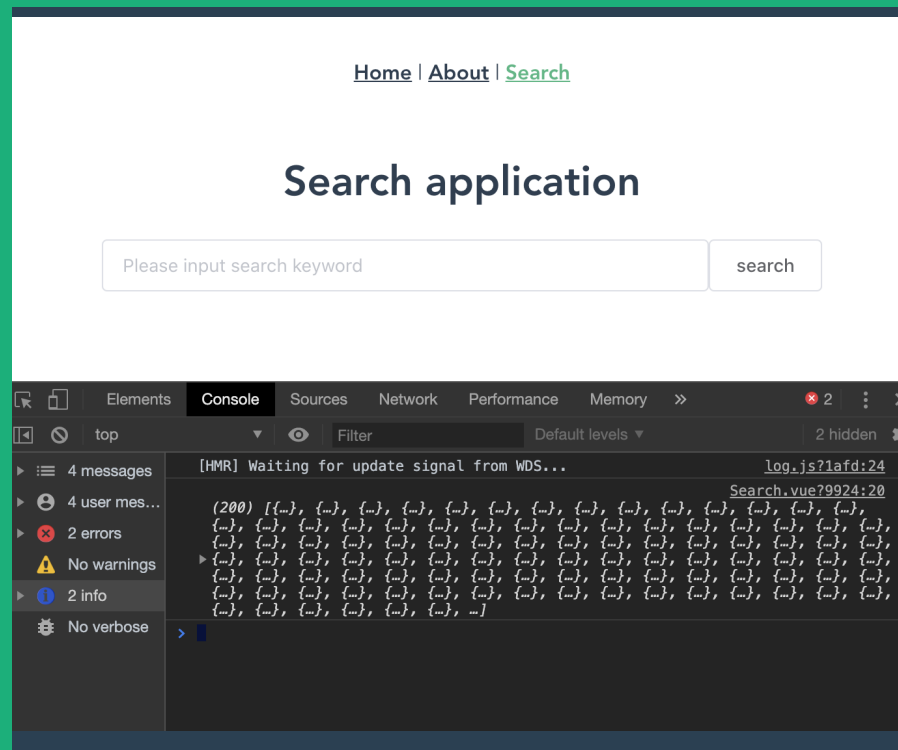
3 methodとbuttonのclickイベントを紐づける

- views/Search.vue

```
<template>
  <div id="search" style="width:80%; margin:auto;">
    <h1>Search application</h1>
    <div id="form-div" style="display:inline-flex; width:100%;">
      <el-input placeholder="Please input search keyword" v-model="input"></el-input>
      <el-button @click="getAllTodos">search</el-button>
    </div>
  </div>
</template>
```

apiを呼んでみる

4 ブラウザでconsoleを確認 → todosが呼べていることを確認



apiを呼んでみる

5 responseをtable形式で表示する

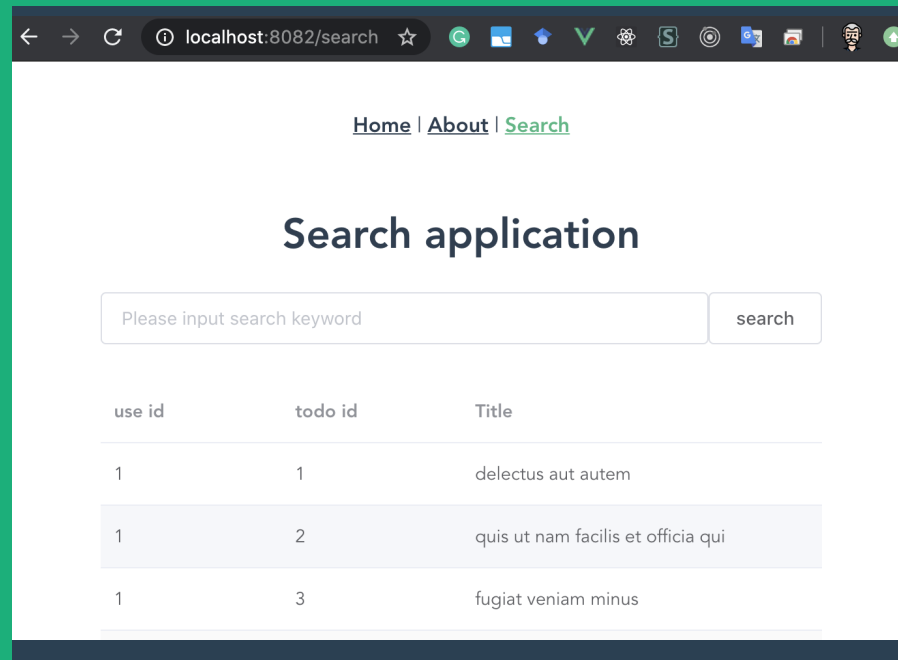
- ElementUI → Component → table から
- views/Search.vue

```
<template>
  <div id="search" style="width:80%; margin:auto;">
    <h1>Search application</h1>
    <div id="form-div" style="display:inline-flex; width:100%;">
      <el-input placeholder="Please input search keyword" v-model="input"></el-input>
      <el-button @click="getAllTodos">search</el-button>
    </div>

    <el-table :data="todos" stripe style="width: 100%; margin-top: 5%;">
      <el-table-column prop="userId" label="use id" :min-width="25"></el-table-column>
      <el-table-column prop="id" label="todo id" :min-width="25"></el-table-column>
      <el-table-column prop="title" label="Title" :min-width="50"></el-table-column>
    </el-table>
  </div>
</template>
```

apiを呼んでみる

6 ブラウザで表示確認 → tableにtodoの内容が表示できている



やってみましょう 🧑

- **1** Vueプロジェクトを作成
- **2** 新しいComponentを作成
- **3** apiを呼んでみる
- **4** inputを反映してapiを呼んでみる

inputを反映してapiを呼んでみる

1 ページが読まれた時点で getAllTodos() が呼び出されるようにする

- views/Search.vue

```
<script>
import axios from 'axios'

export default {
  methods: {
    //
  },
  data() {
    //
  },
  created() {
    this.getAllTodos()
  }
}
</script>
```


inputを反映してapiを呼んでみる

2 formのinputを反映して `getById()` を実行する

- `views/Search.vue`

```
<script>
export default {
  methods: {
    getById: function(input){
      const endpoint = "https://jsonplaceholder.typicode.com/todos/" + input
      axios.get(endpoint)
        .then(response => {
          this.todos = [response.data]
        })
    },
    //
  }
}
</script>
```

inputを反映してapiを呼んでみる

3 dataにinputのテキストを追加

- views/Search.vue

```
<script>
export default {
  methods: {
    //
  },
  data() {
    return {
      todos: [], // responseを格納する配列を定義
      input: '',
    }
  }
}
</script>
```

inputを反映してapiを呼んでみる

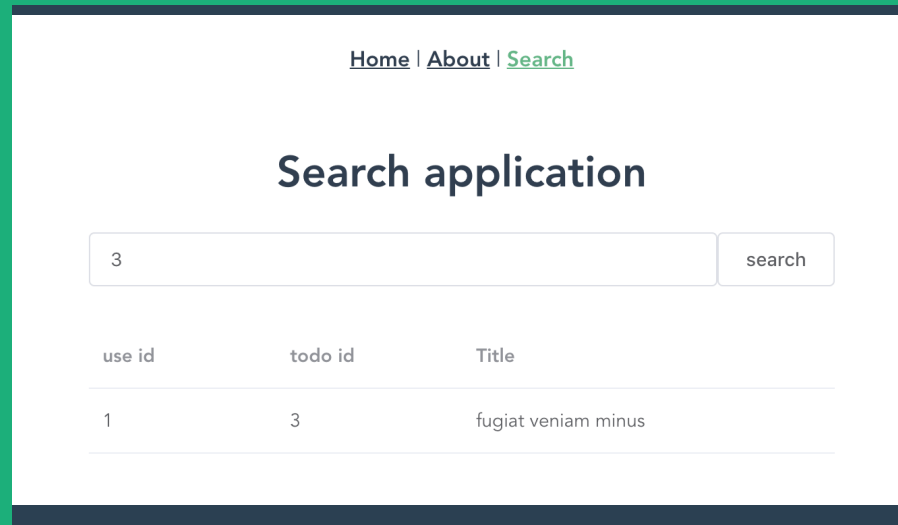
4 @clickに紐づいているmethodを更新

- views/Search.vue

```
<template>
  <div id="search" style="width:80%; margin:auto;">
    <h1>Search application</h1>
    <div id="form-div" style="display:inline-flex; width:100%;">
      <el-input placeholder="Please input search keyword" v-model="input"></el-input>
      <el-button @click="getById(input)">search</el-button>
    </div>
  </div>
</template>
```

inputを反映してapiを呼んでみる

5 idを入力して1件だけ帰ってくることを確認



Home | About | [Search](#)

Search application

3 search

use id	todo id	Title
1	3	fugiat veniam minus

おつかれさまでした 100

- **1** Vueプロジェクトを作成
- **2** 新しいComponentを作成
- **3** apiを呼んでみる
- **4** inputを反映してapiを呼んでみる

ソースコード

- -  の題名ごとにcommitしているので、途中経過を追ってください

参考

export defaultの中身

```
export default {  
  methods: {  
    // apiにリクエストするとかのmethodを書く  
  },  
  data() {  
    // viewsの中で使いたいdataを書く  
  },  
  created() {  
    // ライフサイクルフック  
    // 要素が生成した時に最初にやりたい処理を書く  
  },  
}
```