

# OpView 前端 Vue 說明文件

文件基於:

目錄:

- [Vue CLI使用及安裝](#)
- [Vue CLI建立後目錄結構](#)
- [Vue的應用及運作說明](#)
- [Vuex功能說明](#)

## Vue CLI使用及安裝

前置安裝:

安裝vue及vue相關套件都需使用NPM來進行。

一. [NPM 使用需安裝Node.js](#): [ctrl + 點我](#)

二. 安裝Vue cli: (以下安裝需使用cmd，建議使用VSCode做為開發IDE)

安裝	升級
<code>npm install -g @vue/cli</code>	<code>npm update -g @vue/cli</code>

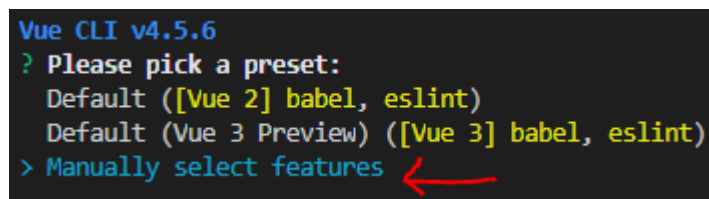
三. 創建專案: (如今天沒有要創建專案可直接跳到 [導入專案](#))

```
vue create projectName
```

註: projectName 可以自己取名

流程:

1. 輸入vue create projectName，選擇自訂安裝選項



```
Vue CLI v4.5.6
? Please pick a preset:
  Default ([Vue 2] babel, eslint)
  Default (Vue 3 Preview) ([Vue 3] babel, eslint)
> Manually select features
```

2. 建議選擇如圖選項，也可以依照所需選擇

注意:這邊操作是space選擇，enter為確定下一步

```
Vue CLI v4.5.6
? 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
```

3. 接下來會有一連串的選項，如無特別需求依照圖片選擇即可

```
Vue CLI v4.5.6
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with 2.x
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a linter / formatter config: Prettier
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? (y/N) N
```

四. 移動到專案內執行專案:

移動	執行
cd projectName	npmrunserve

註: 需先將路徑移動到專案內才能使用npmrunserve

五. 移動到專案內執行專案:

建議安裝套件: es-line (Coding Style)

導入專案:

如今天是導入其他已建立完成的vue專案的話則已以下方式執行專案。

一. 安裝專案所需套件

```
npminstall
```

註: 第一次引入專案都需要執行此步驟

二. 執行專案

```
npmrunserve
```

## Vue CLI建立後目錄結構

文件/資料夾	說明
node_modules	npm 套件讀取位置
public	index.html入口頁面及Logo放置

src	開發目錄(所有的程式及相關設定都在這) <ul style="list-style-type: none"> <li>assets: 放置靜態圖片及Logo</li> <li>components: 放置所有component (沒有可以自己創建不影響專案)</li> <li>mixins: component共用方法 (沒有可以自己創建不影響專案)</li> <li>router: vue router 設定檔 (一開始建立專案有選擇router才會出現)</li> <li>store: Vuex相關檔案 (一開始建立專案有選擇vuex才會出現)</li> <li>views: 完整頁面的vue放置處</li> <li>App.vue: 入口頁面</li> <li>main.js: vue的核心設定及全域套件引入的地方</li> </ul>
README.md	專案說明文件
package.json	專案的配置文件
.xxxx 文件	eslint, git等等的配置檔

## Vue的應用及運作說明

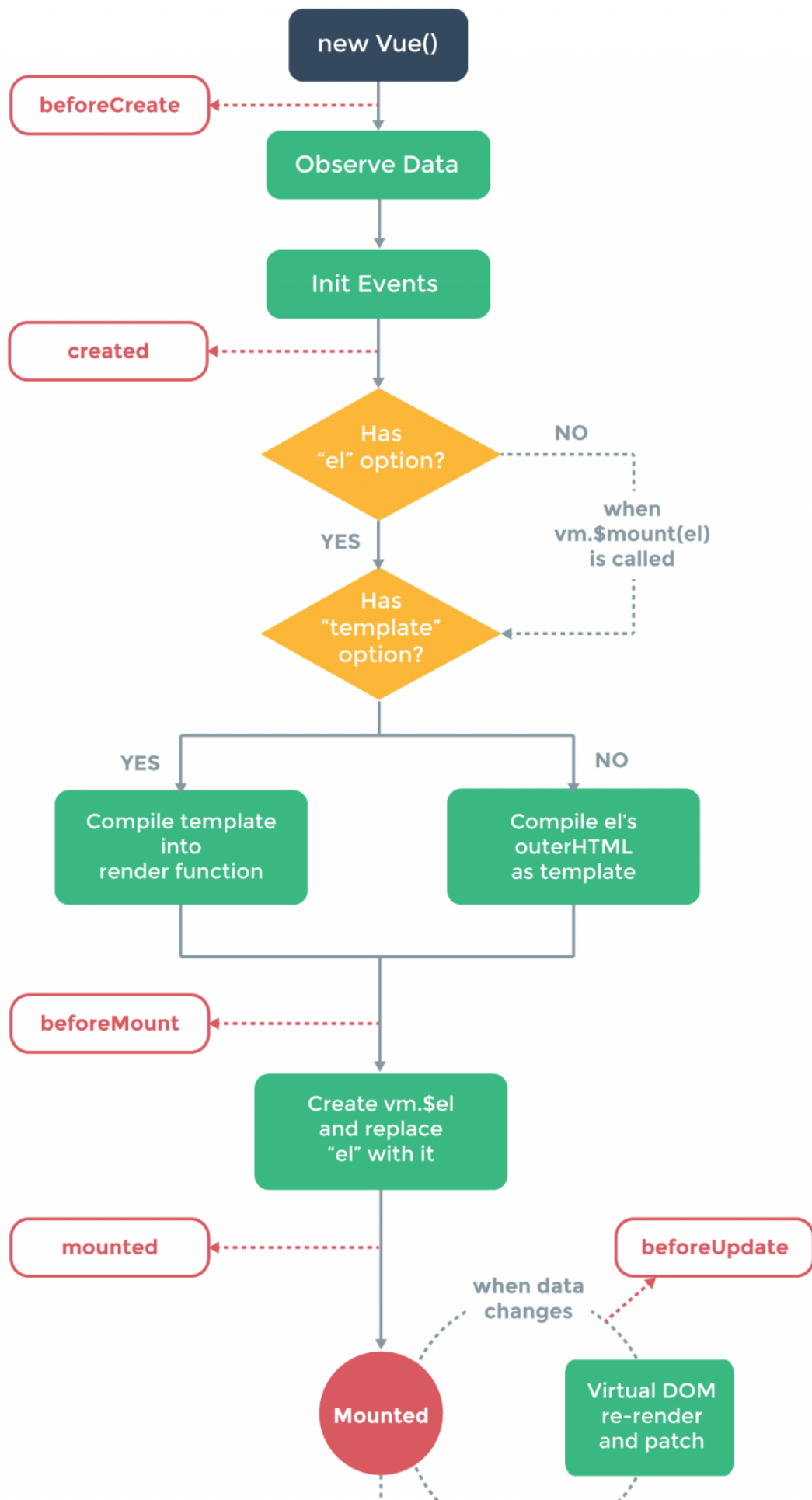
export default 可以使用的方法:

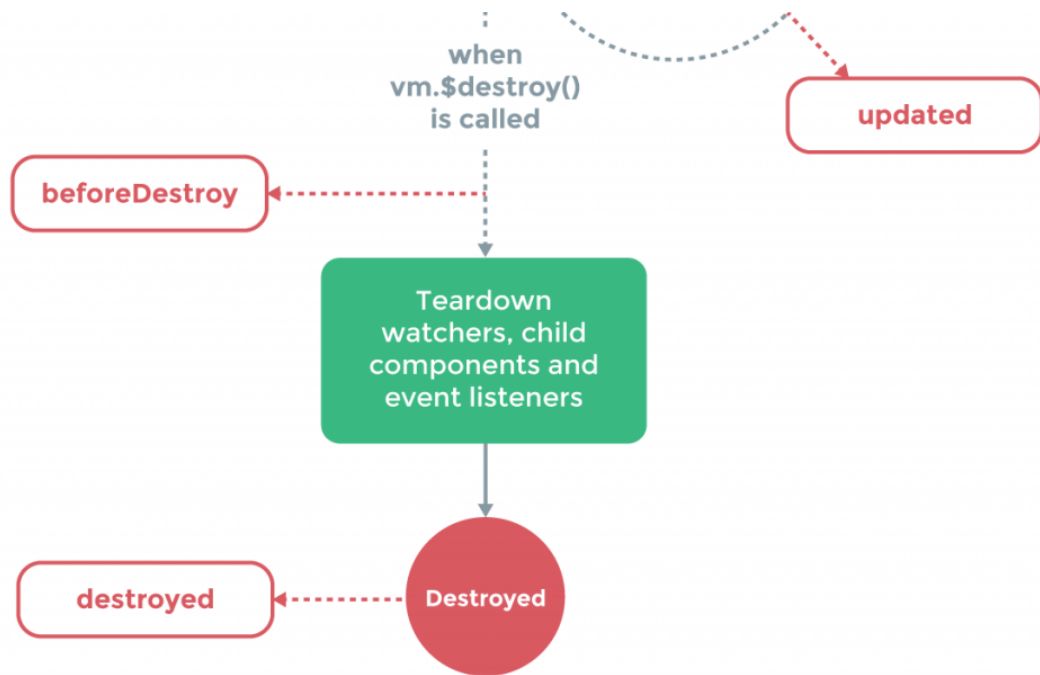
方法名稱	說明
name	這個元件實體的名稱。
components	載入其他的元件到這個元件實體內。
mixins	載入混合模式元件。
props	讀取外部傳入的屬性。
data()	設定預設值。
inheritAttrs	設定該元件是否繼承相關屬性，僅接受布林值。
directives	元件自身的自定義指令。
filters	元件自身定義的過濾器。
computed	設定計算屬性。
methods	設定元件內部方法。
watch	監聽工具。

與生命週期有關的函數：

方法名稱	說明
beforeCreate()	元件建立之前。
created()	元件建立完成。
beforeMount()	元件被放入 DOM 結構樹之前。
mounted()	元件放入 DOM 結構樹之後。
beforeUpdate()	元件實體更新之前。
updated()	元件實體更新之後。
beforeDestroy()	元件被銷毀（從 DOM 結構樹移除）之前。
destroyed()	元件被銷毀（從 DOM 結構樹移除）之後。

生命週期:





Computed Properties and Template Syntax: (Vue許多的應用，幾乎都是使用範例中的方法 有可能還會用到:class, v-show, v-model, v-if, v-html等等)

## example

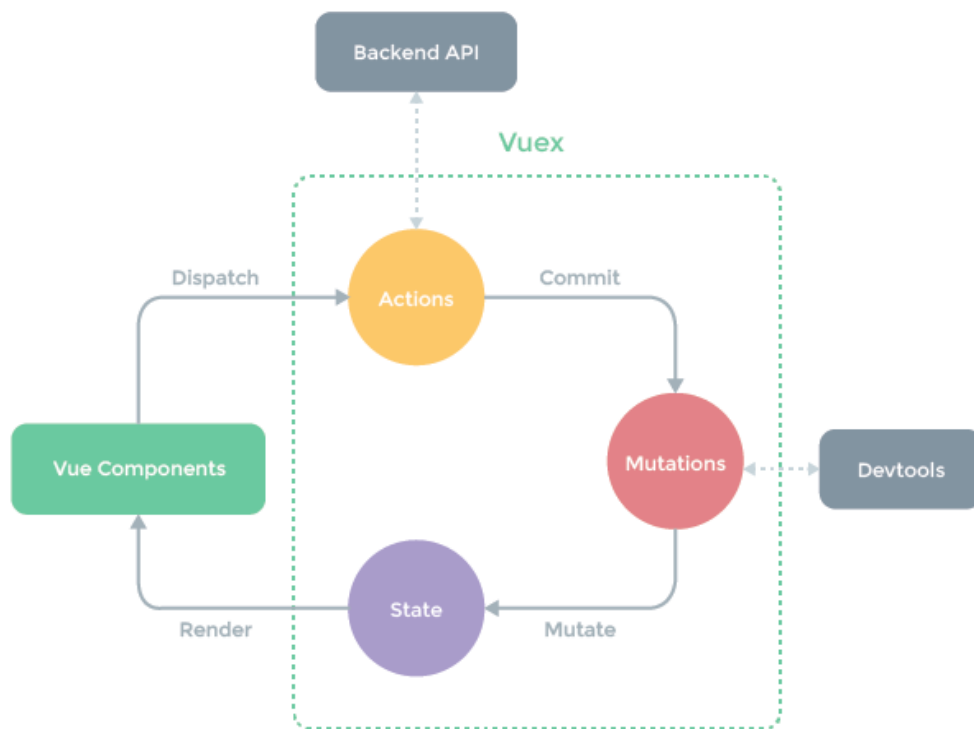
```
<template>
  <div id="example">
    <p>Original message: "{{ message }}"</p> <!-- Original message: Hello -->
    <p>Computed reversed message: "{{ reversedMessage }}"</p> <!-- Original message: olleH -->
    <a v-bind:href="href">Google</a> = <a :href="href">Google</a> = <a
href="https://google.com.tw">Google</a><br><br> <!--
前兩個方法對應data()裡的href，最後一個方法為原始html方法 -->
    <button v-on:click="doSomething">ADD</button> = <button @click="doSomething">ADD</button> <!--
點擊觸發methods.doSomething() -->
    <p>click add: {{ number }}</p> <!-- 透過methods.doSomething()做到number++ -->
  </div>
</template>

<script>
export default {
  data() {
    return {
      message: "Hello",
      number: 0,
      href: "https://google.com.tw",
    };
  },
  computed: {
    reversedMessage() {
      return this.message.split("").reverse().join("");
    },
  },
  methods: {
    doSomething() {
      this.number++;
    }
  },
};
</script>
```

## Vuex功能說明

vuex為狀態管理工具負責全專案的狀態變數傳遞。

運作原理:



常用方法:

方法名稱	說明
State	狀態儲存的物件，Vuex 使用 單一狀態樹 的方式來存放。
Getters	取得狀態資料的方法。
Mutations	更新狀態資料的方法。
Actions	類似 Mutations，但是 Actions 是呼叫 Mutations，且可支援非同步呼叫。
Modules	用於分割 Vuex 的區塊

取用方法:

方法名稱	說明
mapState	可以取得 state 裡面的資料。
mapMutations	可以取得 mutations 裡面的方法。
mapActions	可以取得 actions 裡面的方法。
mapGetters	可以取得 getters 裡面的方法。

執行方法:

方法名稱	說明
commit	這個函式是呼叫 mutation 裡面的方法。
dispatch	這個函式是呼叫 action 裡面的方法。
rootState	這個物件是整個 Store 的 state 資料。

state	這個物件是本地端 ( Local state ) 的資料。
rootGetters	這個物件是整個 Store 的 getters 資料。

範例:

store/modules/status.js (如沒有此路徑可以自己創建)

#### example

```
const state = {
  hello: "hello",
  age: 18,
};
const actions = {
  sayHello({
    commit,
  }) {
    commit('hello');
  },
  addAge({
    commit,
  }) {
    commit('incrementAge');
  },
};
const mutations = {
  hello() {
    if (state.hello === "hello world!") {
      state.hello = "hello";
    } else {
      state.hello = "hello world!";
    }
  },
  incrementAge() {
    state.age++;
  },
};
const getters = {
  getHello() {
    return state.hello;
  },
  getAge() {
    return state.age;
  },
};

export default {
  state,
  actions,
  mutations,
  getters,
};
```

store/index.js



#### example

```
import Vue from 'vue';
import Vuex from 'vuex';
import login from './modules/status';

Vue.use(Vuex);

export default new Vuex.Store({
  modules: {
    login,
  },
  strict: true,
});
```

view/example.vue

#### example

```
import { mapGetters, mapActions } from 'vuex';

export default {
  computed: {
    ...mapGetters({
      getHello: 'getHello',
      getAge: 'getAge',
    }),
    ...mapActions({
      addAge: 'addAge',
    })
  }
  mounted() { // 開啟頁面後會console hello
    console.log(this.getHello); // hello
    console.log(this.getAge); // 18
    console.log(this.addAge); // promise()
    console.log(this.getAge); // 19
  }
}
```