

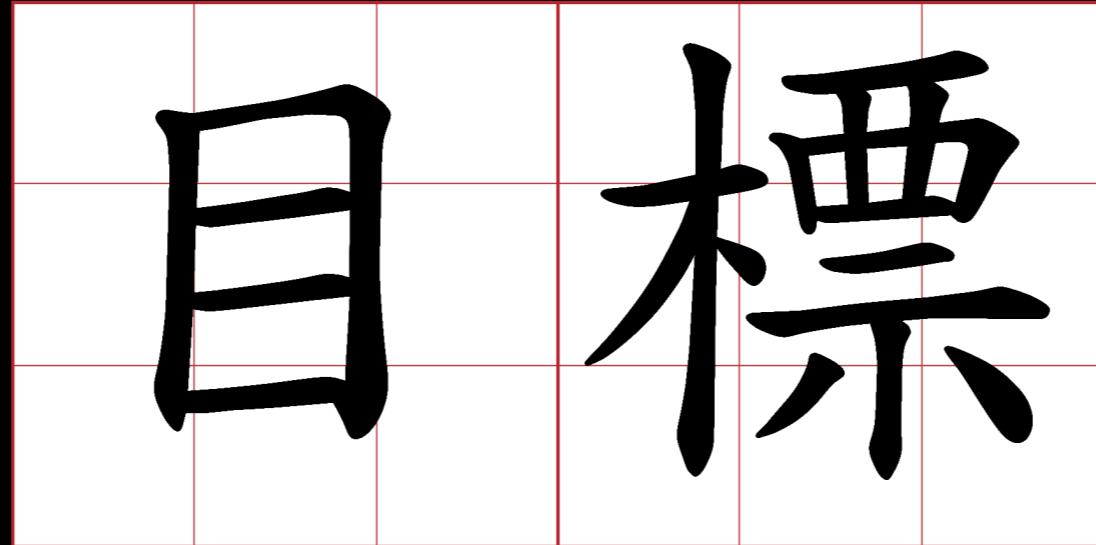
從 jQuery 到 Vue.js

Kuro Hsu / 2019-12-27



開始之前，我假設大家都對
jQuery 有一定程度的熟悉了

還不熟的歡迎報名下次 JavaScript 課程



分別透過 jQuery 與 Vue
打造一個圖片輪播器





jQuery 大家都很熟了
長話短說 直接看程式

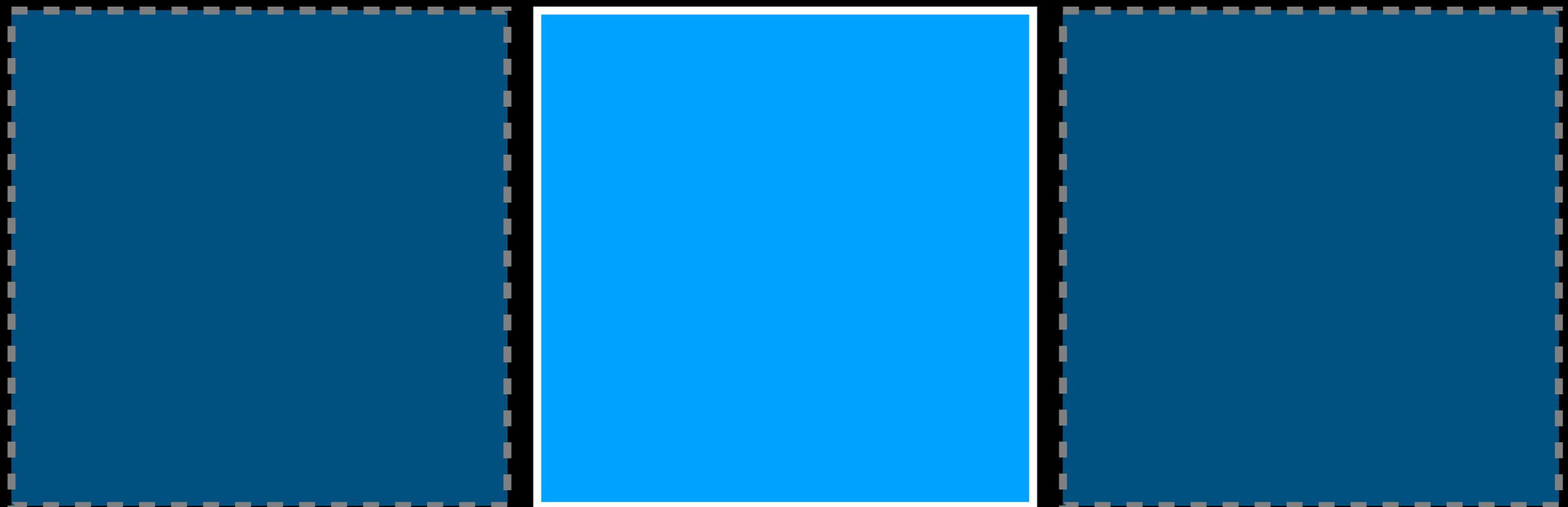


```
<div class="carousel">

<div class="carousel-items">
  <div></div>
  <div></div>
</div>

<div class="carousel-dots"></div>

</div>
```



3



2



1

$5 \times \{ \text{diamond} \} \cdot \text{tw}$

```
.carousel-items div {  
  box-sizing: border-box;  
  display: none;  
  width: 600px;  
  height: 400px;  
  top: 0;  
  left: 0;  
}
```

1

```
.carousel-items .active {  
  display: block;  
  position: absolute;  
  transition: all .9s ease;  
}
```

2

```
.carousel-items .active-enter {  
  transform: translateX(100%);  
}  
  
.carousel-items .active-leave {  
  transform: translateX(-100%);  
}
```

3

目前顯示照片

下一張

`translateX(100%);`

目前顯示照片

下一張



目前顯示照片

下一張



`translateX(-100%);`

`translateX(-100%);`

前一張

`translateX(100%);`

下一張

目前顯示照片



```
const setPhoto = (targetIdx) => {
  const currentImg = $('.active');
  const nextImg = (targetIdx !== undefined) ? $('.carousel-items div').eq(targetIdx) :
    (currentImg.next().length) ? currentImg.next() : $('.carousel-items div').first();

  targetIdx = targetIdx || nextImg.index();

  $('.carousel-dots .active').removeClass('active');
  $('.carousel-dots div').eq(targetIdx).addClass('active');

  // 下一張預備
  nextImg.addClass('active active-enter');

  // 開始動畫
  window.setTimeout(function () {
    currentImg.addClass('active-leave')
    nextImg.removeClass('active-enter');
  }, 0);

  // 結束後隱藏
  window.setTimeout(function () {
    currentImg.attr('class', '');
  }, 1000);

  if (intervalId === 0) {
    intervalId = window.setInterval(setPhoto, timer);
  }
};
```

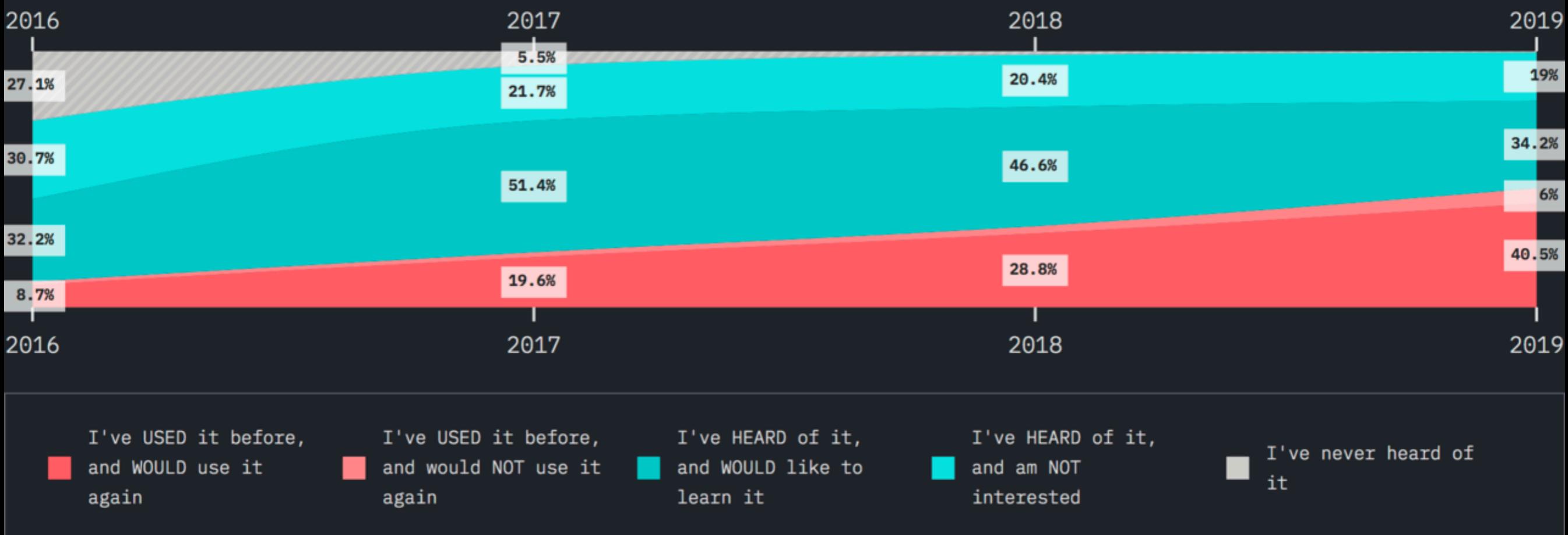


Vue.js

Vue.js

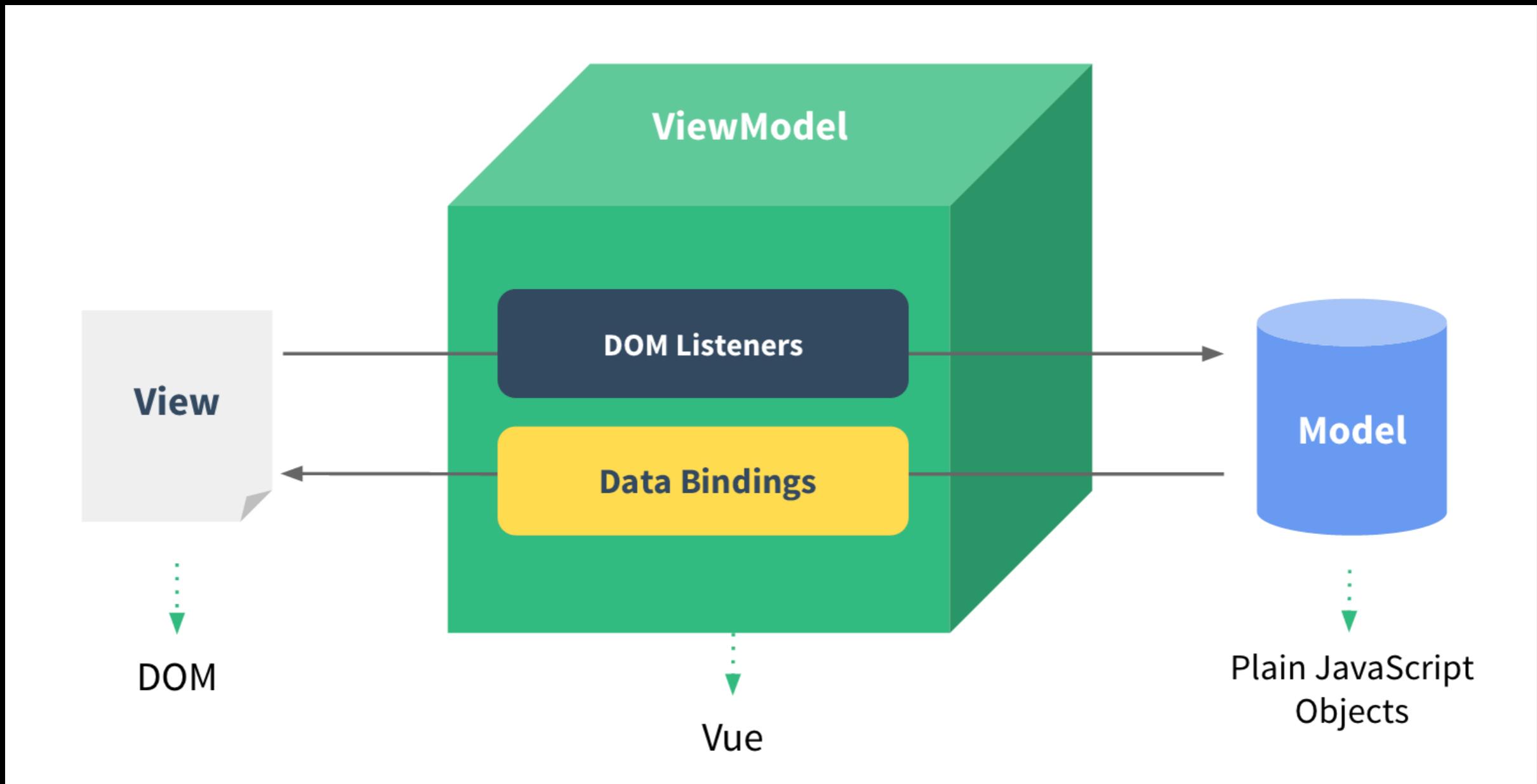
- 音同「View」
- 簡單輕量的 JS 漸進式框架，也能與其他前端框架與函式庫並存共用，目前最新的版本為 v2.6（v3.0 pre-alpha 階段）。
- 關注點分離：開發者只需專注在畫面元素，及其對應的資料變化即可。
- 每個元件各自代表獨立的單元，可自由組合，有各自的區塊模板及資料邏輯。也由於資料綁定的特性，無須手動操作 DOM，也減少發生錯誤的機會。

Respondent's experience with Vue.js.



https://2019.stateofjs.com/front-end-frameworks/vuejs/#vuejs_experience

M-V-VM



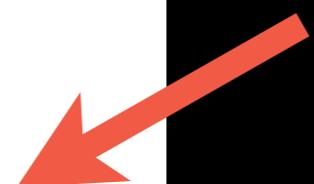
直接引入 vue.js

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Title</title>
</head>
<body>
</body>
</html>
```

```
<script src="scripts/vue.js"></script>
<script>

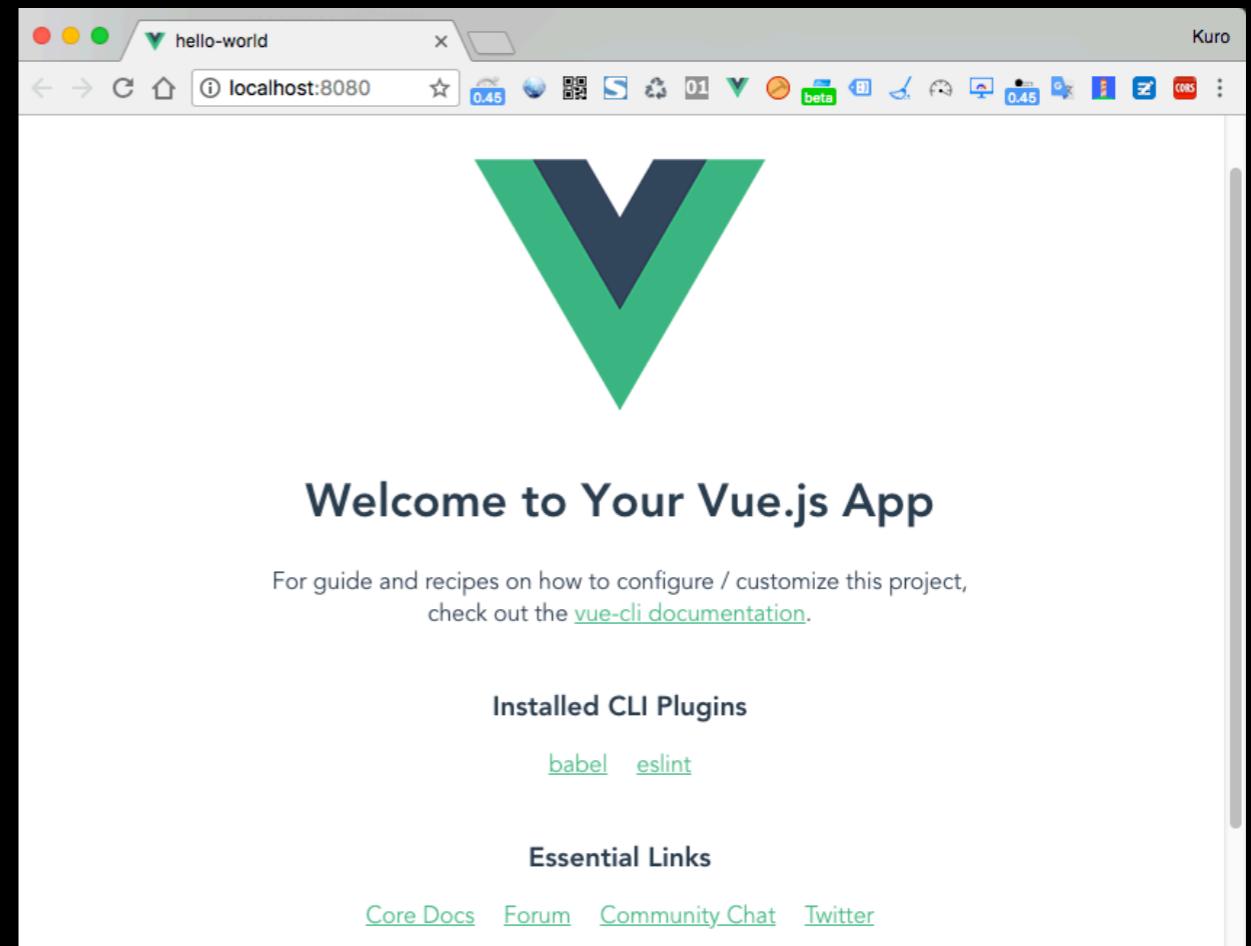
  new Vue({
    el: 'body'
  });

</script>
```



透過 vue-cli 建置新專案

```
# latest stable  
$ npm install vue
```



- 透過命令列工具的方式來快速建置專案
- CLI V3 之後更提供了圖型化介面來管理

Vue.js devtools

首頁 > 擴充功能 > Vue.js devtools

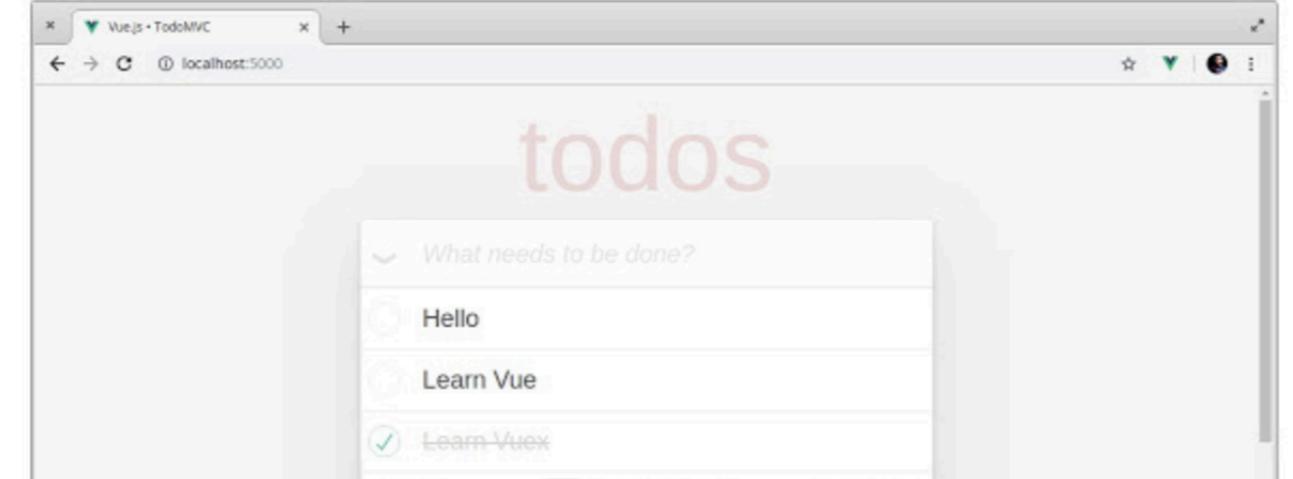
 **Vue.js devtools**

來源網站: <https://vuejs.org>

★★★★★ 1,546 | 開發人員工具 | 1,087,697 位使用者

從 Chrome 中移除

總覽 評論 支援 相關項目



Web Server for Chrome

200 OK! **Web Server for Chrome**

來源網站: chromebeat.com

★★★★★ (715) | 開發人員工具 | 117,394 位使用者

總覽 評論 支援 相關項目 G+1 335

200 OK!

Web Server for Chrome

可在離線狀態下執行
與你的裝置相容

A Web Server for Chrome, serves web pages from a local folder over the network, using HTTP. Runs offline.

Web Server for Chrome is an open source (MIT) HTTP server for Chrome.

It runs anywhere that you have Chrome installed, so you can take it anywhere. It even works on ARM chromebooks.

It now has the option to listen on the local network, so other computers can

網站 檢舉濫用情形

其他資訊

10 秒鐘看懂 vue.js

```
1 <div id="demo">  
2   <p>{{message}}</p>  
3   <input v-model="message">  
4 </div>
```

Hello Vue.js!

```
1 var demo = new Vue({  
2   el: '#demo',  
3   data: {  
4     message: 'Hello Vue.js!'  
5   }  
6 })
```

Hello Vue.js!

Vue 實體基本屬性

```
var vm = new Vue({  
  el: '#app',           // el: 用來掛載 Vue 實體的元素  
  data: {},             // data: 要綁定的資料  
  props: {},            // props: 用來接收外部資料的屬性  
  methods: {},          // methods: 用來定義在 Vue 實體內使用的函數（方法）  
  watch: {},            // watch: 用來觀察 Vue 實體內資料的變動  
  computed: {},          // computed: 自動為內部資料計算過的屬性  
  template: "...",       // template: 提供 Vue 實體編譯後的樣板  
  components: {}         // components: 用來定義子元件  
});
```

練習: Hello Vue.js

```
1 <div id="demo">  
2   <p>{{message}}</p>  
3   <input v-model="message">  
4 </div>
```

Hello Vue.js!

```
1 var demo = new Vue({  
2   el: '#demo',  
3   data: {  
4     message: 'Hello Vue.js!'  
5   }  
6 })
```

Hello Vue.js!

在 HTML 以 {{ }} 作為模板

```
1 <div id="demo">  
2   <p>{{message}}</p>  
3   <input v-model="message">  
4 </div>
```

透過 v-model 將 data 內的屬性與表單結合

Vue 模板

```
{{ 'Hello ' + name }}  
  
{{ number + 1 }}  
  
{{ ok ? 'YES' : 'NO' }}  
  
{{ msg.split(' ').reverse().join(' ') }}
```

以上哪些是合法的語法？

Vue 模板

```
{ { var a = 1 } }
```

```
{ { if (ok) { return message } } }
```

這些是不合法的模板語法

Vue 指令

由 Vue 所提供的特殊的 DOM 屬性，
Vue 內建的指令通常會以「v-」作為開頭。

- v-text
- v-html
- v-show
- v-if
- v-else
- v-else-if
- v-for

- v-on
- v-bind
- v-model
- v-pre
- v-cloak
- v-once

屬性綁定 v-bind

`{{ }}` 不可作為屬性使用。

如果需要在 HTML 標記內的屬性帶入 data 或運算後的內容時，要用 **v-bind** 來進行綁定，前面的 **v-bind** 可省略，只留下一個「:」

```
<div v-bind:id="tag_id"></div>
```

```
<button v-bind:disabled="someDynamicCondition">Button</button>
```

練習: v-bind 與 class

```
.red {  
  background-color: #f00;  
}  
  
.blue {  
  background-color: #00f;  
}
```

試著修改 data 裡的 blockClass，
當 blockClass 為 "red" 或 "blue" 時
會發生什麼事？

```
<div id="app">  
  ←— vue template here →  
  <div class="block" :class="blockClass"></div>  
</div>
```

事件 v-on

- Vue 提供 **v-on** 指令來對 DOM 監聽事件。
- 使用方式為 **v-on:[事件名]** 如 **v-on:click=""**，或也可簡寫為 **@click=""**

```
<button v-on:click="counter += 1">Add 1</button>
```

```
<button v-on:click="add()">Add 1</button>
```

練習: v-on 與 v-bind

```
<div id="app">
  ←— vue template here —→
  <div class="block" :class="blockClass"></div>

  <button>Red</button>
  <button>Blue</button>

</div>
```

試著為 Red 與 Blue 分別加上點擊 click 事件，讓他們點擊的時候改變 block 的顏色。

條件判斷 v-if / v-show

```
<div id="app">
  ←— vue template here —→
  <div class="block" v-if="count < 10"></div>

  <h1>count: {{ count }}</h1>

  <button @click="count++">Plus</button>
  <button @click="count--">Minus</button>

</div>
```

當 v-if 的條件成立時，指定的節點才會出現，
留意 v-if 與 v-show 的差別。

列表渲染 v-for

```
<tr>
  <td>{{ n.name }}</td>
  <td>{{ n.mail }}</td>
  <td><button class="btn btn-default del-profile">刪除</button></td>
</tr>
```

```
el: '#app',
data: {
  member: [
    {
      name: 'Adam',
      mail: 'adam@xxx.com'
    },
    {
      name: 'Jack',
      mail: 'jack@xxx.com'
    },
    {
      name: 'Candy',
      mail: 'candy@xxx.com'
    },
    {
      name: 'Louis',
      mail: 'loui@xxx.com'
    },
    {
      name: 'Lurry',
      mail: 'lurry@xxx.com'
    }
  ]
})
```

#	Name	Email	
1	Adam	adam@xxx.com	刪除
1	Jack	jack@xxx.com	刪除
1	Candy	candy@xxx.com	刪除
1	Louis	loui@xxx.com	刪除
1	Lurry	lurry@xxx.com	刪除
-	Name:	Email:	加入
-	<input type="text"/>	<input type="text"/>	

方法 methods

```
methods: {  
  hello: function () {  
    alert("是在哈囉！");  
  }  
}
```

```
<button @click="hello()" class="btn btn-default">>
```

雖然我們可以直接在 v-on 上面加入事件行為，
但如果遇到複雜或重複行為的時候，可以將它包裝至 methods 中。

練習: methods

#	Name	Email	
1	Adam	adam@xxx.com	刪除
1	Jack	jack@xxx.com	刪除
1	Candy	candy@xxx.com	刪除
1	Louis	loui@xxx.com	刪除
1	Lurry	lurry@xxx.com	刪除
-	Name:	Email:	加入
-	<input type="text"/>	<input type="text"/>	

完成畫面上的刪除與加入功能

Take a Break



```
    ...
  data: {
    imgList: [
      'https://i.picsum.photos/id/237/600/400.jpg',
      'https://i.picsum.photos/id/217/600/400.jpg',
      'https://i.picsum.photos/id/168/600/400.jpg',
      'https://i.picsum.photos/id/111/600/400.jpg',
      'https://i.picsum.photos/id/163/600/400.jpg',
      'https://i.picsum.photos/id/178/600/400.jpg',
      'https://i.picsum.photos/id/144/600/400.jpg',
      'https://i.picsum.photos/id/198/600/400.jpg',
      'https://i.picsum.photos/id/186/600/400.jpg',
      'https://i.picsum.photos/id/176/600/400.jpg',
      'https://i.picsum.photos/id/103/600/400.jpg']
  },
}
```

將圖片網址拆解、複製到 data 的 imgList 陣列內
重複的部分，並用 v-for 改寫

```
<div id="app" class="carousel">
  ←— vue template here —→

  <div class="carousel-items">
    <div v-for="i in imgList"></div>
  </div>

  <div class="carousel-dots"></div>

</div>
```

將圖片網址拆解、複製到 data 的 imgList 陣列內
重複的部分，並用 v-for 改寫



```
<div class="carousel-dots">
| <div v-for="i in imgList"></div>
</div>
```

用來表示目前順序的 `carousel-dots` 也是

```
<div class="carousel-dots">  
  <div v-for="i in imgList" @click="setPhoto( ? )"></div>  
</div>
```



```
<div class="carousel-dots">  
  <div v-for="(i, idx) in imgList" @click="setPhoto( idx )"></div>  
</div>
```

在 **carousel-dots** 的元素上加上點擊事件，
改變顯示照片

在 data 加入 currIdx，用來判斷目前顯示的圖片

```
data: {  
    currIdx: 0,  
    imgList: [
```

改寫 setPhoto 方法

```
setPhoto(targetIdx) {  
    this.currIdx = targetIdx;  
},
```

增加條件判斷

```
<div class="carousel-items">  
    <div v-for="i in imgList" v-if=?>  
          
    </div>  
</div>
```

在 carousel-dots 的節點上加上對應的 class

```
<div class="carousel-dots">
  <div
    v-for="(i, idx) in imgList"
    @click="setPhoto( idx )"
    :class="{ 'active': idx === currIdx }"></div>
</div>
```

到目前為止已經可以手動切換了。

在 data 加入 timer，用來定義自動轉換圖片的時間

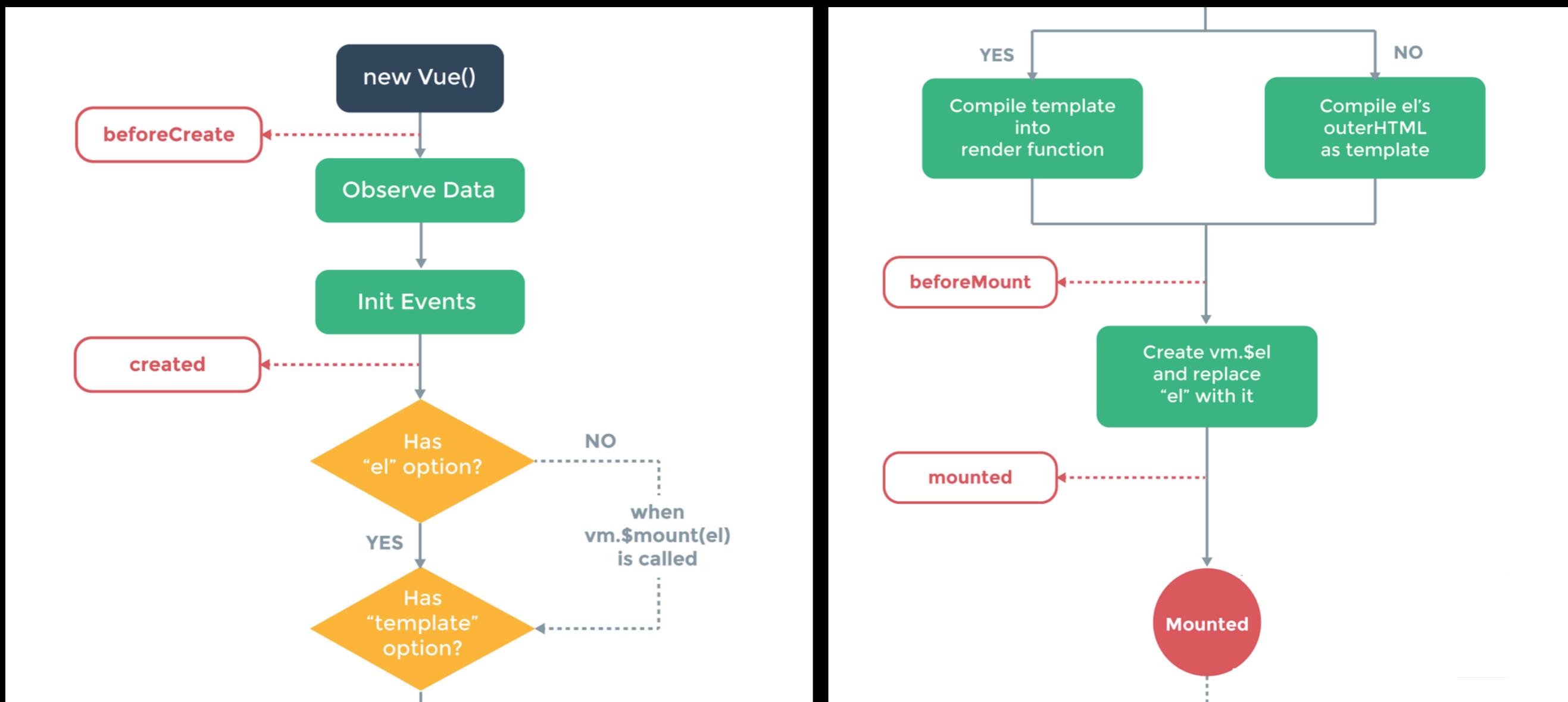
```
  data: {  
    currIdx: 0,  
    timer: 3500,  
    imgList: ['https://i.picsu  
      'https://i.picsum.photos
```

改寫 setPhoto 方法

```
methods: {  
  setPhoto(targetIdx) {  
    // 1. 判斷 targetIdx 有沒有值  
    // 2. 判斷是否為最後一張  
    this.currIdx = (targetIdx === undefined) ? targetIdx :  
      (this.currIdx < this.imgList.length - 1) ?  
        this.currIdx + 1 : 0;  
  
    window.setTimeout(this.setPhoto, this.timer);  
  },
```

想要一開始就自動啟動輪播？

Vue Lifecycle Hooks



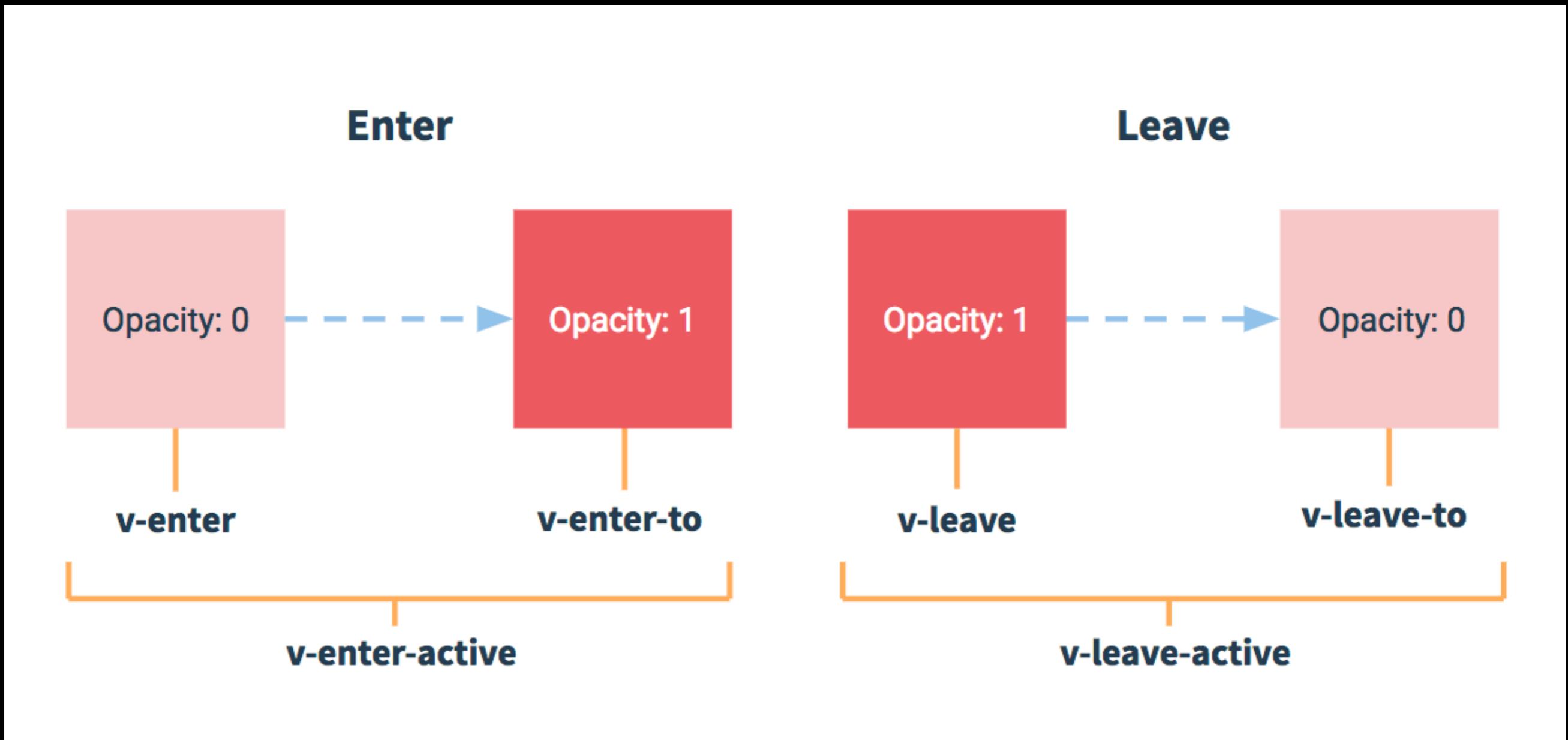
Vue.js hook	hook 說明
beforeCreate	元件實體剛被建立，屬性計算之前。
created	元件實體已建立，屬性已綁定，但 DOM 還沒生成。
beforeMount	模板 (template) 編譯或掛載至 HTML 之前
mounted	模板 (template) 編譯或掛載至 HTML 之後
beforeUpdate	元件被更新之前
updated	元件被更新之後
activated	keep-alive 用，元件被啟動時呼叫
deactivated	keep-alive 用，元件被移除時呼叫
beforeDestory	元件被銷毀前呼叫
destoryed	元件被銷毀後呼叫

改寫 setPhoto 方法，並且加入 mounted hook

```
methods: {
  setPhoto(targetIdx) {
    // 1. 判斷 targetIdx 有沒有值
    // 2. 判斷是否為最後一張
    this.currIdx = (targetIdx === undefined) ? targetIdx :
      (this.currIdx < this.imgList.length - 1) ?
        this.currIdx + 1 : 0;
  },
  startCarousel() {
    window.setInterval(this.setPhoto, this.timer);
  }
},
mounted() {
  this.startCarousel();
},
```

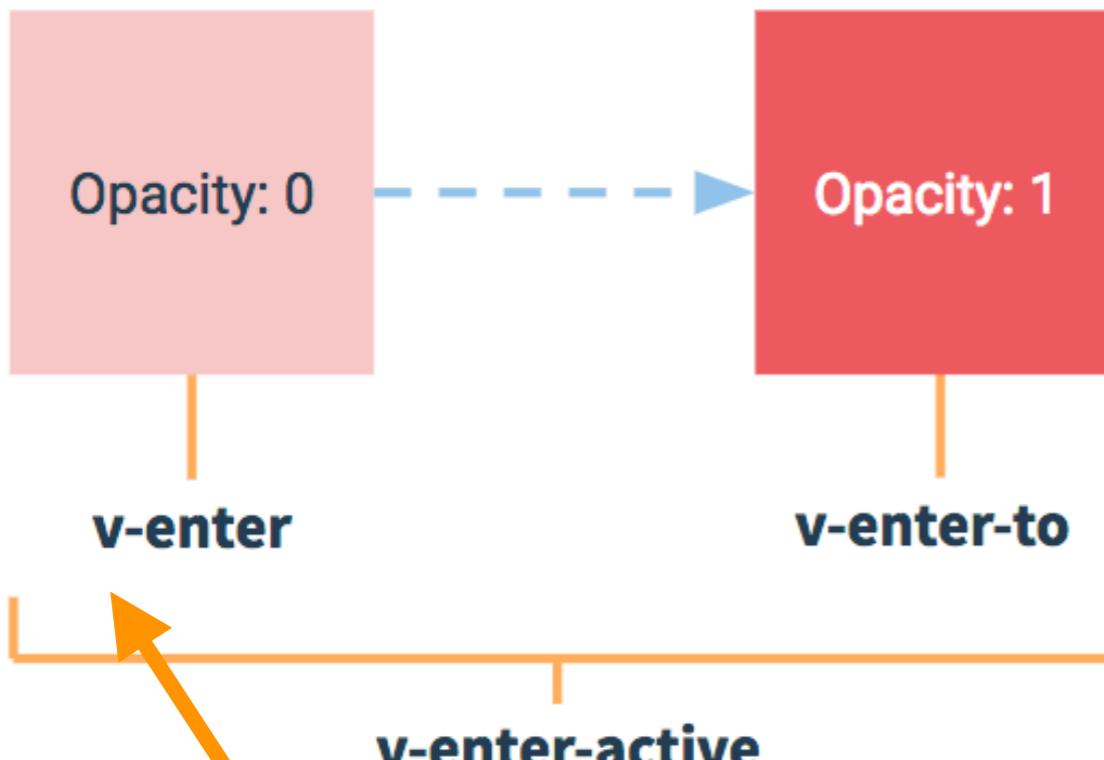
注意 mounted hook 不在 methods 裡面

漸變 - Transition

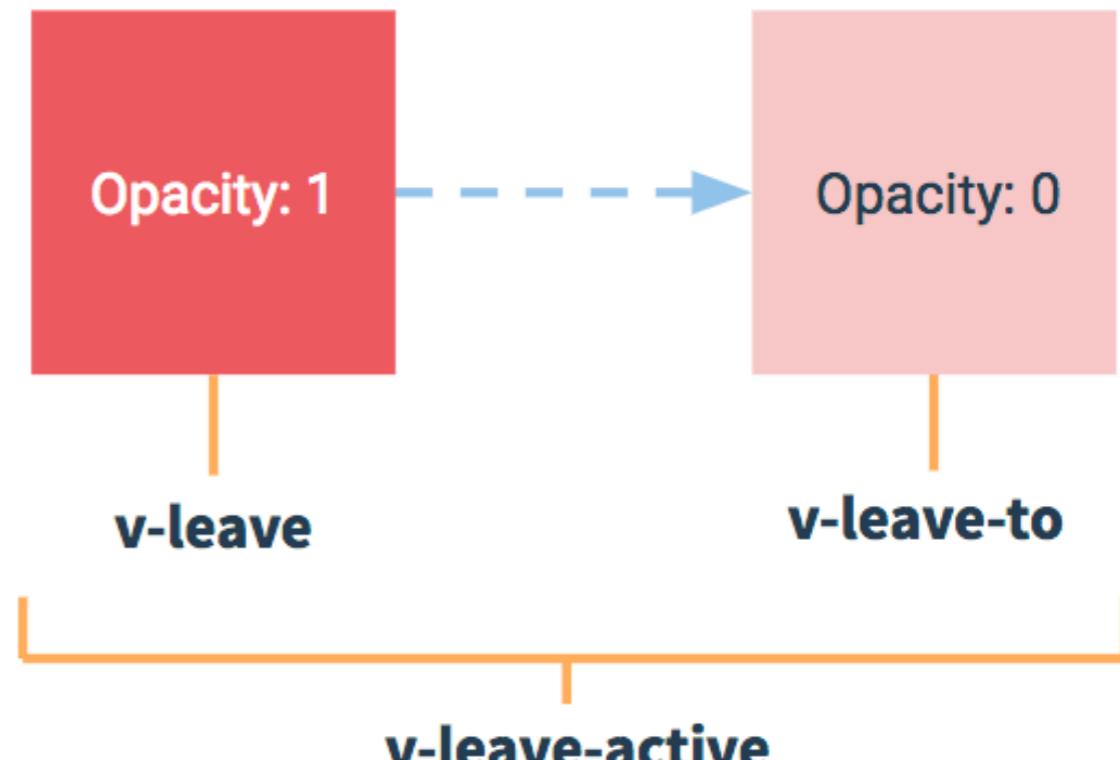


- Vue 將 CSS 的漸變透過 transition 包裝成一個類似元件的標記：
`<transition></transition>`
`<transition></transition>`
- transition 可與 v-if / v-show 以及動態的元件增減搭配組合。
- transition 由 name 來定義動畫的類別與名稱，Vue 則會透過 name 來取 CSS 的前綴名稱配合 hook Class。

Enter



Leave



前面的 v- 指的是前綴名

```
.slide-leave-active,  
.slide-enter-active {  
  transition: all .9s ease;  
}
```

```
.slide-enter {  
  transform: translateX(100%);  
}
```

```
.slide-leave-to {  
  transform: translateX(-100%);  
}
```

class 也要依照前綴名稱命名

```
<transition name="slide">  
  <div v-for="(i, idx) in imgList" v-if="idx === currIdx" :key="i">  
      
  </div>  
</transition>
```

這裡的 name="slide" 就是前綴名



目前為止已經有了動畫與輪播效果，
但是當使用者點擊指定的區塊，倒數仍然在執行...

在 data 新增一個 intervalId，用來記錄目前 timer

```
data: {  
    currIdx: 0,  
    intervalId: 0,  
    timer: 3500,  
    imgList: ['https://i.pics
```

啟動時，將 setInterval 的 id 指定給 intervalId

```
startCarousel() {  
    this.intervalId = window.setInterval(this.setPhoto, this.timer);  
}
```

改寫 setPhoto 方法

```
setPhoto(targetIdx) {  
  
    if (this.intervalId !== 0) {  
        // 結束目前等待中的動畫  
        window.clearTimeout(this.intervalId);  
        // 繼續下一次的自動輪播  
        this.startCarousel();  
    }  
  
    // 1. 判斷 targetIdx 有沒有值  
    // 2. 判斷是否為最後一張  
    this.currIdx = (targetIdx !== undefined) ? targetIdx :  
        (this.currIdx < this.imgList.length - 1) ?  
            this.currIdx + 1 : 0;  
},
```

完成 ! 🎉

工商服務



<https://5xruby.tw/talks/vue-js/>

相關資源

- **VueJS 官方文件 (中文) :**
<https://cn.vuejs.org/v2/guide/>
- **Awesome Vue.js:**
<https://github.com/vuejs/awesome-vue>
- **Vue.js 台灣 (Facebook 社團):**
<http://vue.tw>
- **Vue.js News (電子報):**
<https://www.getrevue.co/profile/vuenewsletter>
-