Vue技術點之間的聯繫

Vue.js技術棧開發前端PC端項目：

* Vue.js語法
* Vue-router
* Axios
* Webpack
* Element-UI
* Vuex

Vue.js技術棧開發前端移動端項目：

* Vue.js語法
* Vue-router
* Axios
* Webpack
* Vant-UI
* Vuex

Vue基礎

## 1 Vue概述

### （1）創始

尤雨溪是Vue.js的創建者

2014年2月，Vue.js正是發佈

2015年10月27日，正是發佈1.0.0

2016年4月27日，發佈2.0的預覽版本

官網：https://cn.vuejs.org/v2/guide/

Vue很高效：20kb運行大小，超快虛擬DOM。

### （2）Vue是漸進式JavaScript框架

#### ① 框架和庫的區別

框架以提供基礎性的服務為主。比如Vue有雙向數據綁定的支撐，可以減少實際開發時的代碼量。

庫以提供API為主。比如jQuery。

現在框架和庫的界限不是很明顯了。所以很多時候把兩者理解成一個東東。

Vue很靈活，在一個庫和一套完整框架之間自如伸縮。

#### ② 漸進式

聲明式渲染→組件系統→客戶端路由→集中式狀態管理→項目構建

## 2 Vue基本使用

  <body>

    <div id="app">

      <!-- {{}} 是插值表達式，作用是把數據填充到HTML標籤裡 -->

      <div>{{msg}}</div>

      <!-- 差值表達式支持基本的計算操作。比如下面的標籤裡顯示的數字是3。-->

      <div>{{1+2}}</div>

      <!-- 只要符合JS語法規則的表達式都可以放在這裡。 -->

      <div>{{msg + '------' + 123}}</div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      new Vue({

        el: "#app", //元素的掛載位置，值可以是CSS選擇器或者DOM元素。掛載是指把數據關聯到頁面的某個標籤裡。

        // data是模型數據，即開發中遇到的業務相關數據，值是一個對象。

        data: {

          msg: "Hello World by Vue",

        },

      });

    </script>

  </body>

Vue不需要直接操作DOM了。

Vue代碼運行原理分析

Vue.js把Vue代碼編譯為原生JS代碼，以便瀏覽器解析。

## 3 Vue模板語法

### （1）前端渲染

Vue是用來做前端渲染的。

前端渲染即把從數據庫請求回來的數據填充到HTML標籤中。

前端渲染的產物是靜態HTML內容，這內容最終被瀏覽器呈現給用戶。

### （2）前端渲染的方式

#### ① 原生js拼接字符串

缺點：不同開發人員的代碼風格差別很大；隨著業務越來越複雜，後期的維護變得逐漸困難起來。

#### ② 使用前端模板引擎，比如art-template

優點：大家都遵循同樣的規則寫代碼，代碼可讀性明顯提高，方便後期維護。

缺點：沒有專門提供事件機制。

#### ③ 使用vue特有的模板語法

模板語法概覽：

* 插值表達式
* 指令
* 事件綁定
* 屬性綁定
* 樣式綁定
* 分支循環結構

### （3）什麼是指令？

指令的本質就是自定義屬性。

比如下面的abc=123就是一個自定義屬性。標準屬性例子：a標籤中的href屬性。

<div id="app" abc=123>

指令的格式：以v-開始，後面跟個字符串，比如v-cloak。

### （4）v-cloak指令用法

* 差值表達式存在的問題：閃動（如果網絡不好，頁面會先顯示花括號，再把花括號替換為相關的數據內容，這個過程就叫做閃動，給用戶造成不好的體驗。）
* 如何解決該問題？使用v-cloak指令

    <style type="text/css">

      [v-clock] {

        display: none;

      }

    </style>

      <div v-cloak>{{msg}}</div>

* 解決該問題的原理：先通過樣式隱藏內容，然後在內存中進行值的替換，替換好值之後再顯示最終的結果。

### （5）數據綁定指令v-text，v-html，v-pre

1. v-text 填充純文本
   * 沒有閃動問題
   * 比插值表達式更加簡潔
   * 推薦用v-text而非插值表達式
2. v-html 填充HTML片段
   * 存在安全問題
   * 本網站內部數據可以使用，來自第三方的數據不可以用
3. v-pre 填充原始信息
   * 顯示原始信息，跳過編譯過程（分析編譯過程）

範例代碼：

  <body>

    <div id="app">

      <div v-text="msg"></div>

      <div v-html="msg1"></div>

      <div v-pre>{{msg}}</div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      new Vue({

        el: "#app",

        data: {

          msg: "Hello World by Vue",

          msg1: "<h1>HTML標籤h1</h1>",

        },

      });

    </script>

  </body>

顯示的樣子：



### （6）數據響應式和v-once

響應式類別：

* HTML5中的響應式（屏幕尺寸的變化導致樣式的變化）
* 數據的響應式（數據的變化導致頁面內容的變化）

數據綁定：把數據填充到標籤中（v-text，v-html，v-pre的功能，它們綁定的數據默認都是響應式的）

v-once只編譯一次，顯示內容之後不再具有響應式功能，可以提高性能。

### （7）雙向綁定數據v-model

在input中添加v-model功能，可以修改data中msg的值，進而影響div中的顯示內容

  <body>

    <div id="app">

      <div>{{msg}}</div>

      <input type="text" v-model="msg" />

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      new Vue({

        el: "#app",

        data: {

          msg: "Hello World by Vue",

        },

      });

    </script>

  </body>

### （8）與雙向數據綁定相關的MVVM設計思想

* M（model）
  + Plain JavaScript Objects，比如上面的例子中data中的數據
* V（view）
  + 本質上是DOM元素
* VM（view-model）
  + 實現控制邏輯，把上面兩者結合在一起。



### （9）事件綁定

v-on指令用法

<input type="button" v-on:click="num++" />

v-on指令簡寫

<input type="button" @click="num++" />

事件函數的調用方式：

* 直接綁定函數名稱

<button v-on:click="say">Hello</button>

* 調用函數（要傳遞參數的話，只能用這種方式）

<button v-on:click="say()">Hi</button>

函數定義在methods屬性中

示例代碼

  <body>

    <div id="app">

      <div>{{num}}</div>

      <button v-on:click="say">Hello</button>

      <button v-on:click="say()">Hi</button>

      <button v-on:click="inc">點擊+1</button>

      <button v-on:click="inc()">點擊+1</button>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          msg: "Hello World by Vue",

          num: 1,

        },

        methods: {

          say: function () {

            console.log("LOL");

          },

          inc: function () {

            // this === vm。在這裡，this指的是Vue的實例對象vm

            this.num++;

          },

        },

      });

    </script>

  </body>

### （10）事件函數參數傳遞

<button v-on:click="say('hi', $event)">Hi</button>

如果沒用到$event，傳參數的時候可以不寫$event。

下面這種直接綁定的方式默認了會傳遞$event參數

<button v-on:click="say">Hello</button>

示例代碼

  <body>

    <div id="app">

      <!-- 傳遞1個參數 -->

      <button v-on:click="say1('LOL1')">Hi 1個參數</button>

      <!-- 傳遞多個參數 -->

      <button v-on:click="say2('LOL2','lala2', $event)">Hi 多個參數</button>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          msg: "Hello World by Vue",

          num: 1,

        },

        methods: {

          say1: function (*p1*) {

            console.log(p1);

          },

          say2: function (*p1*, *p2*, *event*) {

            console.log(p1, p2, event);

          },

        },

      });

    </script>

  </body>

### （11）事件修飾符

.stop阻止冒泡

<a v-on:click.stop="handle">跳轉</a>

.prevent阻止默認行為

<a href="http://www.bing.com" v-on:click.prevent="handle">跳轉</a>

事件修飾符可以連著寫。寫的先後順序不同，有時會有不同的效果。

既阻止冒泡又阻止默認行為

<a href="http://www.bing.com" v-on:click.stop.prevent="handle">跳轉</a>

### （12）按鍵修飾符

.enter回車鍵

<input type="text" v-on:keyup.enter="handle" />

.delete刪除鍵

<input type="text" v-on:keyup.delete="handle" />

### （13）自定義按鍵修飾符

全局config.keyCodes對象

比如，按下a，會觸發handle函數

<input type="text" v-on:keyup.65="handle" />

### （14）屬性綁定v-bind

<a v-bind:href="url">跳轉</a>

縮寫形式

<a :href="url">跳轉</a>

url是寫在data裡面的數據

  <body>

    <div id="app">

      <a :href="url">跳轉</a>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          url: "http://www.bing.com",

        },

        methods: {

          handle: function () {

            console.log("handle");

          },

        },

      });

    </script>

  </body>

### （15）v-model底層實現原理

  <body>

    <div id="app">

      <!-- 下面兩種寫法都可以 -->

      <input type="text" v-on:input="handle" />

      <input type="text" v-on:input="msg=$event.target.value" />

      <div>{{msg}}</div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          msg: "天朗氣清",

        },

        methods: {

          handle: function (*event*) {

            this.msg = event.target.value;

          },

        },

      });

    </script>

  </body>

### （16）樣式綁定

#### ① 對象語法

<div v-bind:class="{active:isActive}"></div>

示例代碼

    <style type="text/css">

      .active {

        width: 100px;

        height: 50px;

        border: 5px solid skyblue;

      }

      .error {

        background-color: yellow;

      }

    </style>

  <body>

    <div id="app">

      <div v-bind:class="{active:isActive1}">1</div>

      <div v-bind:class="{active:isActive, error:isError}">2</div>

      <button v-on:click="handle1">操縱邊框1</button>

      <button v-on:click="handle2">操縱邊框2</button>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          isActive1: true,

          isActive: true,

          isError: true,

        },

        methods: {

          handle1: function () {

            this.isActive1 = !this.isActive1;

          },

          handle2: function () {

            this.isActive = !this.isActive;

            this.isError = !this.isError;

          },

        },

      });

    </script>

  </body>

#### ② 數組語法

<div v-bind:class="[activeClass, errorClass]"></div>

示例代碼

    <style type="text/css">

      .active {

        width: 100px;

        height: 50px;

        border: 5px solid skyblue;

      }

      .error {

        background-color: yellow;

      }

    </style>

  <body>

    <div id="app">

      <div v-bind:class="[activeClass, errorClass]"></div>

      <button v-on:click="handle">操作背景色</button>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          activeClass: "active",

          errorClass: "error",

        },

        methods: {

          handle: function () {

            this.errorClass = "";

          },

        },

      });

    </script>

  </body>

#### ③ 對象綁定和數組綁定可以結合使用

    <style type="text/css">

      .active {

        width: 100px;

        height: 50px;

        border: 5px solid skyblue;

      }

      .error {

        background-color: yellow;

      }

      .test {

        color: salmon;

      }

    </style>

  <body>

    <div id="app">

      <div v-bind:class="[activeClass, errorClass,{test: isTest}]">test</div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          activeClass: "active",

          errorClass: "error",

          isTest: true,

        },

        methods: {},

      });

    </script>

  </body>

#### ④ class綁定的值可以簡化操作

    <style type="text/css">

      .active {

        width: 100px;

        height: 50px;

        border: 5px solid skyblue;

      }

      .error {

        background-color: yellow;

      }

      .test {

        color: salmon;

      }

    </style>

  <body>

    <div id="app">

      <div v-bind:class="arrClasses">test</div>

      <div v-bind:class="objClasses">test</div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          arrClasses: ["active", "error", "test"],

          objClasses: {

            active: true,

            error: true,

            test: true,

          },

        },

        methods: {},

      });

    </script>

  </body>

#### ⑤ 處理默認的class

原有的class不會被覆蓋。在原有的class基礎上追加class。

    <style type="text/css">

      .active {

        width: 100px;

        height: 50px;

        border: 5px solid skyblue;

      }

      .error {

        background-color: yellow;

      }

      .test {

        color: salmon;

      }

      .base {

        font-size: 28px;

      }

    </style>

  <body>

    <div id="app">

      <div class="base" v-bind:class="arrClasses">test</div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          arrClasses: ["active", "error", "test"],

        },

        methods: {},

      });

    </script>

  </body>

#### ⑥ style樣式處理

方式1

  <body>

    <div id="app">

      <div

        v-bind:style="{border: borderStyle, width:widthStyle, height:heightStyle}"

      >

        test

      </div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          borderStyle: "1px solid skyblue",

          widthStyle: "200px",

          heightStyle: "100px",

        },

        methods: {},

      });

    </script>

  </body>

方式2

  <body>

    <div id="app">

      <div v-bind:style="objStyles">test</div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          objStyles: {

            border: "1px solid skyblue",

            width: "200px",

            height: "100px",

          },

        },

        methods: {},

      });

    </script>

  </body>

方式3，用數組的方式把樣式對象羅列上，前面有的樣式會被覆蓋，沒有的會加上

  <body>

    <div id="app">

      <div v-bind:style="[objStyles,overrideStyles]">test</div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>

    <script type="text/javascript">

      var vm = new Vue({

        el: "#app",

        data: {

          objStyles: {

            border: "1px solid skyblue",

            width: "200px",

            height: "100px",

          },

          overrideStyles: {

            border: "5px solid #355c7d",

            backgroundColor: "pink",

          },

        },

        methods: {},

      });

    </script>

  </body>

### （17）分支結構

分支結構：

* v-if
* v-else
* v-else-if
* v-show

v-if不會渲染標籤；v-show會把標籤渲染出來，如果不需要顯示，則把標籤設置為display:none，如果需要顯示，則不給標籤設置display:none。

### （18）循環結構

v-for遍歷數組

<li v-for=’item in list’>{{item}}</li>

<li v-for=’(item, index) in list’>{{item}} + ‘---’ + {{index}}</li>

（其中item和index都是自定義的，愛叫啥叫啥）

key的作用：幫助vue區分不同的元素，從而提高性能

<li :key=’item.id’ v-for=’(item, index) in list’>{{item}} + ‘---’ + {{index}}</li>

<li :key=’index’ v-for=’(item, index) in list’>{{item}} + ‘---’ + {{index}}</li>

v-for遍歷對象

<div v-for=’(value, key, index) in object’></div>

v-if和v-for結合使用遍歷對象：滿足了某個條件才渲染標籤。同理，v-if可以和上述的v-for遍歷數組結合使用。

<div v-if=’value===12’ v-for=’(value, key, index) in object’></div>

（其中value、key、index都是自定義的，愛叫啥叫啥）

## 基礎案例

Vue常用特性

綜合案例