1. 深入vue组件

**I. 组件**

一、组件基础

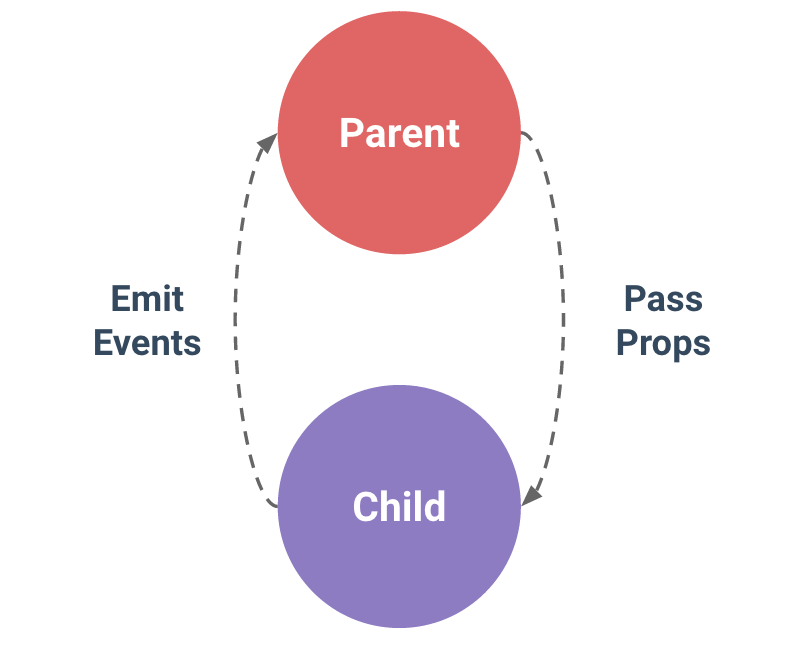
1、组件名、属性在template中最好把驼峰格式转成中线格式。

2、可以通过:is=””方法在template中使用组件。

二、组件之间的通信 -- 主要有三种：属性、emit、slot插槽

父向子传 用props，即标签属性

子向父传 用emit，即通过去触发一个事件，这个事件只有emit触发和on监听两种方法



（1）父传子之静态属性传递

**父组件App.vue中：**

<template>

<div>

<comp-a **number-to-do** = 78></comp-a>

</div>

</template>

<script>

import compA from './components/a'

export default {

components: {

compA

},

data () {

return {}

}

}

</script>

**子组件a.vue中：**

<template>

<div>

{{ hello }}

{{ **numberToDo** }}

</div>

</template>

<script>

export default {

props: ['**number-to-do**'],

data () {

return {

hello: 'I am a component A!'

}

}

}

</script>

PS：a) 注意以上三处加粗的代码的名字转换。

b) 在子组件中通过props接收父组件传过来的数据，就相当于在自己的data中添加了该数据，在data中就不必重新定义。

c) props除了可以接收数组外，还可以接收对象，对象里可以定义接收的数据类型。

父组件中：

<comp-a number-to-do = 78></comp-a>

子组件中：

props: {

'number-to-do': [Number, String]

},

PS：父组件中的78不管加不加引号，都是String类型，如果子组件中props未指定该类型，就会报错，指定了就不会。

1. 父传子之动态属性传递

**父组件App.vue中：**

<template>

<div>

<input type="text" v-model="myVal">

<comp-a :my-value="myVal"></comp-a>

</div>

</template>

<script>

import compA from './components/a'

export default {

components: {

compA

},

data () {

return {

myVal: ''

}

}

}

</script>

**子组件a.vue中：**

<template>

<div>

{{ hello }}

{{ myValue }}

</div>

</template>

<script>

export default {

props: {

'my-value': [Number, String]

},

data () {

return {

hello: 'I am a component A!'

}

}

}

</script>

PS：注意属性如:my-value、事件如@my-event只能用中划线，不能用驼峰方式。

1. 父传子之插槽

如果父组件想向子组件插入一个模板标签元素，要借助<slot>标签。

**父组件App.vue中：**

<template>

<div>

<comp-a :my-value="myVal">

**<p>123</p>**

</comp-a>

</div>

</template>

**子组件a.vue中：**

<template>

<div>

{{ hello }}

{{ myValue }}

**<slot></slot>**

</div>

</template>

PS：上例中就把父组件放到子组件标签中的p标签插到了子组件的slot位置。

**1）插槽**的便捷之处：

a) 常用场景：

比如子组件<comp-a>是个dialog，标签里可以插入任何元素<p>、<span>之类的，这样就可以实现弹框内容不同了。

b) 比如子组件<comp-a>中没有内容，可以在插槽<slot>标签中写内容，就会被显示，若子组件有内容，就只展示子组件的内容，而不展示插槽的。

1. 关于具名slot -- 可以知道不同部分的插入

**父组件中**：

<comp-a :my-value="myVal">

<p>123</p>

<p slot="header">xxx header</p>

<p slot="footer">yyy footer</p>

</comp-a>

**子组件中**：

<slot></slot>

<slot name="header">no header</slot>

balabala

<slot name="footer">no footer</slot>

1. 子传父 使用emit和v-on:，参照第三章P16第五事件绑定。
2. 动态组件
3. 通过:is=＂变量如：currentView＂，将某个组件渲染到某个位置，我们就可以通过currentView来载入不同的组件到那个位置。
4. 常用场景：

比如tab切换，tab切换下面会展现不同的小组件，这个小组件由一个组件切换成另一个组件，所以我们只要在切换的方法中更新currentView就会将组件同步更新到相应的位置。

1. 动态组件有些类似路由，换句话说<router-view>就相当于一个动态组件。

跟路由相似，动态组件也支持<keep-alive>内置标签，可以缓存组件，提高组件的加载速度。

<template>

<keep-alive>

<p :is = "currentView"></p>

</keep-alive>

</template>

<script>

import comA from './components/a'

export default {

components: {comA},

data () {

return {

currentView: 'com-a'

}

}

}

</script>

PS：加粗的com-a就是子组件变量，该子组件一定要先import再注册，否则就会报错。

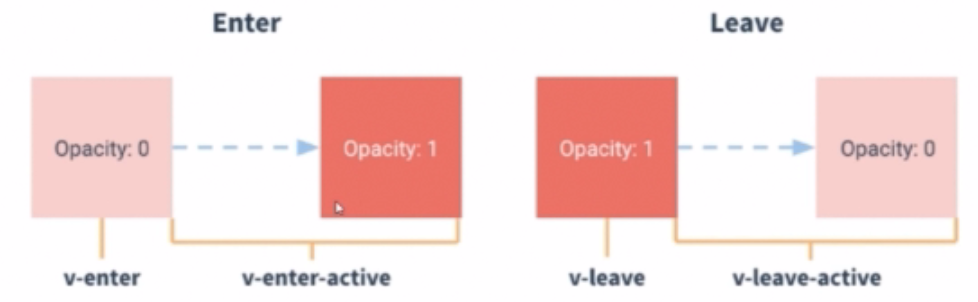
1. 总结



**II. Vue.js的高级功能**

1. 过渡/动画
2. css实现过渡

css阶段类名：



通过<transition>标签与v-show或v-if指令以及style样式配合，样式中一般都是opacity/transform样式：

（1）淡入淡出

<template>

<button @click="show = !show">toggle</button>

<div>

<transition name="**fade**">

<p v-show="show">this is toggle show</p>

</transition>

</div>

</template>

<script>

export default {

data () {

return {

show: true

}

}

}

</script>

<style>

**.fade**-enter,

**.fade**-leave-active {

opacity: 0;

}

**.fade**-enter-active,

**.fade**-leave-active {

transition: all .5s ease-out;

}

</style>

PS：style中的.fade-和<transition>标签的name是对应的。

1. 自定义动画，如位移

<template>

<button @click="show = !show">toggle</button>

<div>

<transition name="my-trans">

<p v-show="show">this is toggle show</p>

</transition>

</div>

</template>

<script>

同上

</script>

<style>

.my-trans-enter-active,

.my-trans-leave-active {

transition: all 1s ease-out;

}

.my-trans-enter {

transform: translateY(-500px);

opacity: 0;

}

.my-trans-leave-active {

transform: translateY(500px);

opacity: 0;

}

</style>

（3）接受的过渡状态、多元素过渡

a) 方式一

<template>

<button @click="toggleCom">toggle</button>

<transition name="**fade**" mode="out-in">

<div :is="currentView"></div>

</transition>

</template>

<script>

import comA from './components/a'

import comB from './components/b'

export default {

components: {comA, comB},

data () {

return {

currentView: 'com-b'

}

},

methods: {

toggleCom () {

if (this.currentView === 'com-a') {

this.currentView = 'com-b'

} else {

this.currentView = 'com-a'

}

}

}

}

</script>

<style>

**.fade**-enter,

**.fade**-leave-active {

opacity: 0;

}

**.fade**-enter-active,

**.fade**-leave-active {

transition: all .5s ease-out;

}

</style>

PS：a) style中的.fade-和<transition>标签的name是对应的。

b) 与直接用css过渡不同之处在于，标签上不用加v-show/v-if指令。

c) <transition>标签默认的mode为in-out即新的先进来旧的再出去，这样体验很不好，所以修改为out-in即旧的先出新的再进。

b) 方式二

<template>

<button @click="show = !show">toggle</button>

<transition name="fade" mode="out-in">

<p v-if="show">I am show</p>

<div v-else>I am not show</div>

</transition>

</template>

<script>

import comA from './components/a'

import comB from './components/b'

export default {

components: {comA, comB},

data () {

return {

currentView: 'com-b',

show: true

}

}

}

</script>

<style>

**.fade**-enter,

**.fade**-leave-active {

opacity: 0;

}

**.fade**-enter-active,

**.fade**-leave-active {

transition: all .5s ease-out;

}

</style>

PS：在vue中如果<transition>中的两个元素是相同标签如都是<p>，则不会执行该动画，需要添加key区分两个标签，动画才能正常执行。

<transition name="fade" mode="out-in">

<p v-if="show" key="0">I am show</p>

<p v-else key="1">not in show</p>

</transition>

1. JS过渡

— 同样也分为一些过渡阶段，与css过渡不同的是，js通过v-on来监听这些事件钩子。

<template>

<transition

@before-enter="beforeEnter"

@enter="enter"

@leave="leave"

:css="false">

<p class="animate-p" v-show="show">I am show</p>

</transition>

<br><br><br><br><br>

<button @click="show = !show">toggle</button>

</template>

<script>

import $ from 'jquery'

export default {

data () {

return {

show: true

}

},

methods: {

beforeEnter (el) {

$(el).css({

left: '-500px',

opacity: 1

})

},

enter (el, done) {

$(el).stop().animate({

left: 0,

opacity: 1

}, {

duration: 1500,

complete: done

})

},

leave (el, done) {

$(el).stop().animate({

left: '500px',

opacity: 0

}, {

duration: 1500,

complete: done

})

}

}

}

</script>

<style>

.animate-p {

position: absolute;

left: 0;

top: 0;

}

</style>

注意：（1）<transition>标签不需要name，但需要绑定属性:css=”false”,避免该标签去检查css中指定的选项。

1. 每个方法其实都要操作传过来的参数el，这个el是<transition>体内的DOM元素。
2. 其中的两个方法enter、leave除了参数el外，还有个done（必须调用），是在两个动作执行完了执行。
3. 用js操作时可能需要用到jquery等公用库，最好把它在index.html中引入，因为那儿的代码不会经过打包和处理资源，只会作为一个入口的页面往里加东西。

*PS：如何在项目中引入jquery？*

在当前项目的目录下（就是package.json），运行命令 cnpm install jquery --save-dev  这样就将jquery安装到了这个项目中。然后修改webpack.base.conf.js(在build文件下)两个地方：

小注：上面的--save是为了更新package.json，使得别人在用的时候也是更新后的。

1. 加入

var webpack=require('webpack');

b) 在module.exports的里面加入

plugins: [

new webpack.optimize.CommonsChunkPlugin('common.js'),

new webpack.ProvidePlugin({

jQuery: "jquery",

$: "jquery"

})

]

c) 最后在main.js中加入import $ form 'jquery',完成jquery的引入

1. 自定义指令

通过选项directives定义如v-color指令，在directtives指令选项中定义color函数，传el和binding两个参数，el就代表v-color指令所在的元素，binding是个对象，指它绑定的一些变量或计算表达式等，binding.value指的是指令接收到值。

1、局部指令 -- 把该指令定义到普通组件的directives里

<template>

<p v-color="'red'">这是一个段落</p>

<p v-custom="'40px'">第二段</p>

</template>

<script>

export default {

directives: {

color (el, binding) {

el.style.color = binding.value

},

**custom: {**

**bind (el, binding) {**

**},**

**inserted (el, binding) {**

**el.style.fontSize = binding.value**

**}**

**}**

}

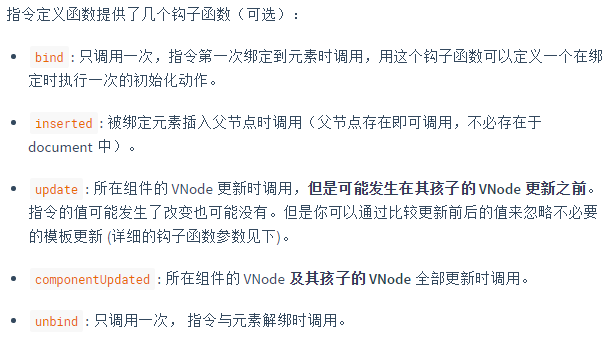
}

</script>

1. 全局指令 -- 把该指令定义到main.js的根组件的directives里，定义方法同上



PS：如上例中，color指令方法为合写，custom指令方法为分写，详细介绍参见下面：



1. 自定义指令适用场景

实现该元素一插入到页面就获取焦点：

<template>

<input type="text" v-focus>

</template>

<script>

export default {

directives: {

focus: {

inserted (el, binding) {

el.focus()

}

}

}

}

</script>

1. 插件 -- 功能完整的一套组件，独立于vue而存在的

1、例如vue-router，如何安装并引用呢？

第一步，在当前项目的目录下（就是package.json），运行命令cnpm install vue-router --save

第二步，如果全局使用，就在main.js中import VueRouter from ‘vue-router’，等同于var VueRouter = require(‘vue-router’)

第三步，注册使用该插件Vue.use(VueRouter)

第四步，实例化该插件let router = new VueRouter()

第五步，正式使用，在根组件中直接使用上面定义的router

以下为入口文件main.js：

import Vue from 'vue'

import App from './App'

import VueRouter from 'vue-router'

// 注册

Vue.use(VueRouter)

// 实例化

va

let router = new VueRouter()

/\* eslint-disable no-new \*/

new Vue({

router, // 正式使用

el: '#app',

render: h => h(App)

})

1. 例如vue-resource，它的安装同vue-router，在main.js中引入时直接注册即可，不需要实例化，这样在每个子组件的方法中都可以直接采用this.$http.get()方法。

更多插件详见https://github.com/vuejs

1. 单文件组件
2. mixins混合