我们前面的教程一直在使用组件，下面是我们导出的一个组件

<template>

  <div id="app"></div>

</template>

<script>

export default {

  name: "app",

  data() {

    return {

      selected: ""

    };

  }

};

</script>

vue-loader会将上面的代码转为对象

{

  template: '<div id="app"></div>',

  data() {

    return {

      selected: ""

    };

  }

}

注册一个全局组件语法格式如下：

Vue.component(tagName, options)

注册后，我们可以使用以下方式来调用组件：

<tagName></tagName>

**全局组件**

所有实例都能用全局组件。

import Vue from 'vue/dist/vue.esm.js'

// 注册全局

Vue.component('App', {

  template: '<h1>自定义组件!</h1>'

})

// 将 id 为 container 的元素替换成我们 template 代码

var vm = new Vue({

  el:"#container",

  template:'<App/>'

})

**局部组件**

我们也可以在实例选项中注册局部组件，这样组件只能在这个实例中使用：

import Vue from 'vue/dist/vue.esm.js'

// 注册局部

let App = {

  template: '<h1>自定义组件!</h1>'

};

// 将 id 为 container 的元素替换成我们 template 代码

var vm = new Vue({

  el:"#container",

  template:'<App/>',

  components:{ App }

})

**Prop**

prop 是父组件用来传递数据的一个自定义属性。

父组件的数据需要通过 props 把数据传给子组件，子组件需要显式地用 props 选项声明 "prop"：

import Vue from 'vue/dist/vue.esm.js'

// 注册局部

let App = {

  // 声明 props

  props: ['message'],

  template: '<h1>{{ message }}</h1>'

};

// 将 id 为 container 的元素替换成我们 template 代码

var vm = new Vue({

  el:"#container",

  template:'<App message="传给子组件的数据" />',

  components:{ App }

})

**动态 Prop**

类似于用 v-bind 绑定 HTML 特性到一个表达式，也可以用 v-bind 动态绑定 props 的值到父组件的数据中。每当父组件的数据变化时，该变化也会传导给子组件：

import Vue from 'vue/dist/vue.esm.js'

// 注册局部

let App = {

  // 声明 props

  props: ['message'],

  template: '<h1>{{ message }}</h1>'

};

// 将 id 为 container 的元素替换成我们 template 代码

var vm = new Vue({

  el:"#container",

  template:'<App v-bind:message="childMessage" />',

  components:{ App },

  data: {

    childMessage: "传给子组件的数据"

  }

})

**Prop 验证**

组件可以为 props 指定验证要求。

当 prop 验证失败的时候，(开发环境构建版本的) Vue 将会产生一个控制台的警告。

// 注册局部

let App = {

  // 声明 props

  props: {

    // 基础的类型检查 (`null` 和 `undefined` 会通过任何类型验证)

    propA: Number,

    // 多个可能的类型

    propB: [String, Number],

    // 必填的字符串

    propC: {

      type: String,

      required: true

    },

    // 带有默认值的数字

    propD: {

      type: Number,

      default: 100

    },

    // 带有默认值的对象

    propE: {

      type: Object,

      // 对象或数组默认值必须从一个工厂函数获取

      default: function () {

        return { message: 'hello' }

      }

    },

    // 自定义验证函数

    propF: {

      validator: function (value) {

        // 这个值必须匹配下列字符串中的一个

        return ['success', 'warning', 'danger'].indexOf(value) !== -1

      }

    }

  },

  template: '<h1>{{ propA }}</h1>'

};

**自定义事件**

父组件是使用 props 传递数据给子组件，但如果子组件要把数据传递回去，就需要使用自定义事件！

import Vue from 'vue/dist/vue.esm.js'

// 注册局部

let App = {

  // 声明 props

  props: ['total'],

  template: '<button v-on:click="buttonClick">{{total}}</button>',

  methods: {

    buttonClick(){

      // 触发组件的 incrementTotal 事件

      this.$emit("incrementTotal");

    }

  }

};

// 将 id 为 container 的元素替换成我们 template 代码

var vm = new Vue({

  el:"#container",

  template:'<App v-bind:total="total" v-on:incrementTotal="incrementTotal" />',   // 监听组件的 incrementTotal 事件

  components:{ App },

  data: {

    total: 0

  },

  methods: {

    incrementTotal: function () {

      this.total += 1

    }

  }

})

**子组件data 必须是一个函数**

// 注册局部

let App = {

  // 声明 props

  props: ['total'],

  template: '<button v-on:click="buttonClick">{{total}}</button>',

  data(){

    return {

      count: 0

    }

  }

};