

Vue预习课

Vue预习课

拓展知识——可复用性

过滤器

自定义指令

渲染函数

基础

虚拟DOM

createElement参数

函数式组件

混入

插件

插件声明

插件使用

拓展知识——可复用性

过滤器

Vue.js 允许你自定义过滤器，可被用于一些常见的文本格式化。过滤器可以用在两个地方：**双花括号插值和 v-bind 表达式** (后者从 2.1.0+ 开始支持)。过滤器应该被添加在 JavaScript 表达式的尾部，由“管道”符号指示：

```
<!-- 在双花括号中 -->
{{ message | capitalize }}

<!-- 在 `v-bind` 中 -->
<div v-bind:id="rawId | formatId"></div>
```

范例：course-list显示价格使用货币符号

```
{{ c.price | currency('RMB') }}
```

```
filters: {
  currency(value, symbol = '¥') {
    return symbol + value;
  }
}
```

自定义指令

除了核心功能默认内置的指令 (`v-model` 和 `v-show`), Vue 也允许注册自定义指令。注意, 在 Vue2.0 中, 代码复用和抽象的主要形式是组件。然而, 有的情况下, 你仍然需要对普通 DOM 元素进行底层操作, 这时候就会用到自定义指令。

范例: 输入框获取焦点

```
Vue.directive('focus', {
  inserted(el) {
    el.focus()
  }
})
```

使用, cource-add

```
<input v-focus>
```

钩子函数

范例: 按钮权限控制

```
const role = 'user'
Vue.directive('permission', {
  inserted(el) {
    if (role !== 'admin') {
      el.parentElement.removeChild(el)
    }
  }
})
```

使用

```
<div class="toolbar" v-permission="'admin'">
```

渲染函数

Vue 推荐在绝大多数情况下使用模板来创建你的 HTML。然而在一些场景中, 你真的需要 JavaScript 的完全编程的能力。这时你可以用**渲染函数**, 它比模板更接近编译器。

基础

```
render: function (createElement) {
  // createElement函数返回结果是VNode
  return createElement(
    tag,    // 标签名称
    data,  // 传递数据
    children // 子节点数组
  )
}
```

范例：用render实现heading组件

```
Vue.component('heading', {
  props: ['level', 'title'],
  render(h) {
    return h(
      'h' + level,
      this.$slots.default
    )
  }
})
```

虚拟DOM

Vue 通过建立一个**虚拟 DOM** 来追踪自己要如何改变真实 DOM。

范例：输出虚拟DOM观察期结构

```
const vnode = h(
  'h' + level,
  { attrs: { title: this.title } }, // 之前省略了title的处理
  this.$slots.default
)
console.log(vnode);
```

createElement参数

接下来你需要熟悉的是如何在 `createElement` 函数中使用模板中的那些功能。这里是 `createElement` 接受的参数：

```
// @returns {VNode}
createElement(
  // {String | Object | Function}
  // 一个 HTML 标签名、组件选项对象，或者
  // resolve 了上述任何一种的一个 async 函数。必填项。
  'div',

  // {Object}
  // 一个与模板中属性对应的数据对象。可选。
  {
    // (详情见下一节)
  },
)
```

```
// {String | Array}
// 子级虚拟节点 (vNodes), 由 `createElement()` 构建而成,
// 也可以使用字符串来生成“文本虚拟节点”。可选。
[
  '先写一些文字',
  createElement('h1', '一则头条'),
  createElement(MyComponent, {
    props: {
      someProp: 'foobar'
    }
  })
]
)
```

范例：处理title、添加icon

```
Vue.component('heading', {
  props: ['level', 'title', 'icon'],
  render(h) {
    let children = [];
    // 添加图标功能
    // <svg><use xlink:use="#icon-xxx"></use></svg>
    if (this.icon) {
      children.push(h(
        'svg',
        { class: 'icon' },
        [h('use', { attrs: { 'xlink:href': '#icon-' + this.icon } })])
      children = children.concat(this.$slots.default)
    }
    vnode = h(
      'h' + level,
      { attrs: { title } }, // 之前省略了title的处理
      children
    )
    console.log(vnode);
    return vnode
  }
})
```

函数式组件

组件没有管理任何状态，也没有监听任何传递给它的状态，也没有生命周期方法时，可以将组件标记为 `functional`，这意味它无状态 (没有响应式数据)，也没有实例 (没有 `this` 上下文)。

```
Vue.component('heading', {
  functional: true, // 标记函数式组件
  props: ['level', 'title', 'icon'],
  render(h, context) { // 上下文传参
    let children = [];
    // 属性获取
    const { icon, title, level } = context.props
    if (icon) {
      children.push(h(
        'svg',
```

```

        { class: 'icon' },
        [h('use', { attrs: { 'xlink:href': '#icon-' + icon } })]]))
    // 子元素获取
    children = children.concat(context.children)
  }
  vnode = h(
    'h' + level,
    { attrs: { title } },
    children
  )
  console.log(vnode);
  return vnode
}
})

```

混入

混入 (mixin) 提供了一种非常灵活的方式，来分发 Vue 组件中的可复用功能。一个混入对象可以包含任意组件选项。当组件使用混入对象时，所有混入对象的选项将被“混合”进入该组件本身的选项。

```

// 定义一个混入对象
var myMixin = {
  created: function () {
    this.hello()
  },
  methods: {
    hello: function () {
      console.log('hello from mixin!')
    }
  }
}

// 定义一个使用混入对象的组件
Vue.component('comp', {
  mixins: [myMixin]
})

```

插件

插件通常用来为 Vue 添加全局功能。插件的功能范围一般有下面几种：

1. 添加全局方法或者属性。如: [vue-custom-element](#)
2. 添加全局资源: 指令/过滤器/过渡等。如 [vue-touch](#)
3. 通过全局混入来添加一些组件选项。如 [vue-router](#)
4. 添加 Vue 实例方法, 通过把它们添加到 `Vue.prototype` 上实现。
5. 一个库, 提供自己的 API, 同时提供上面提到的一个或多个功能。如 [vue-router](#)

插件声明

Vue.js 的插件应该暴露一个 `install` 方法。这个方法的第一个参数是 `vue` 构造器, 第二个参数是一个可选的选项对象:

```
MyPlugin.install = function (Vue, options) {  
  // 1. 添加全局方法或属性  
  Vue.myGlobalMethod = function () {}  
  
  // 2. 添加全局资源  
  Vue.directive('my-directive', {})  
  
  // 3. 注入组件选项  
  Vue.mixin({  
    created: function () {  
      // 逻辑...  
    }  
  })  
  
  // 4. 添加实例方法  
  Vue.prototype.$myMethod = function (methodOptions) {}  
}
```

范例: 修改heading组件为插件

```
const MyPlugin = {  
  install (Vue, options) {  
    Vue.component('heading', {...})  
  }  
}  
  
if (typeof window !== 'undefined' && window.Vue) {  
  window.Vue.use(MyPlugin)  
}
```

插件使用

使用Vue.use即可引入插件

```
Vue.use(MyPlugin)
```

