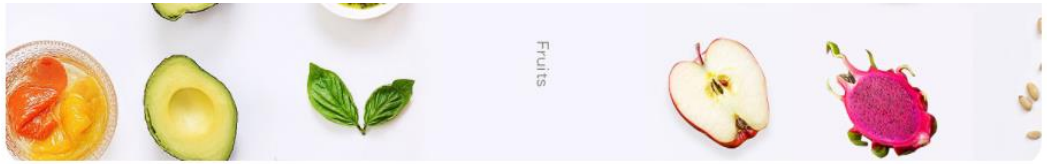
Vue 核心技术与实战

day02

目录

Contents

综合案例：水果购物车



/ 购物车

选中	图片	单价	个数	小计	操作
☐		20	- 11 +	220	删除
☑		3	- 10 +	30	删除
☐		34	- 20 +	680	删除

☐ 全选　　总价：¥ 30　结算(10)

目录

Contents

指令修饰符

通过 ”.” 指明一些指令 后缀，不同 后缀 封装了不同的处理操作 → 简化代码

① 按键修饰符

@keyup.enter → 键盘回车监听

② v-model修饰符

v-model.trim → 去除首尾空格

v-model.number → 转数字

③ 事件修饰符

@事件名.stop → 阻止冒泡

@事件名.prevent → 阻止默认行为

目录

Contents

v-bind 对于样式控制的增强

为了方便开发者进行样式控制， Vue 扩展了 v-bind 的语法，可以针对 class 类名 和 style 行内样式 进行控制 。

class 类名

style 行内样式

v-bind 对于样式控制的增强 - 操作class

语法 :class = "对象/数组"

① 对象 → 键就是类名，值是布尔值。如果值为 true，有这个类，否则没有这个类

```
<div class="box" :class="{ 类名1: 布尔值, 类名2: 布尔值 }"></div>
```

适用场景：一个类名，来回切换

② 数组 → 数组中所有的类，都会添加到盒子上，本质就是一个 class 列表

```
<div class="box" :class="[ 类名1, 类名2, 类名3 ]"></div>
```

适用场景：批量添加或删除类

```
:class="['pink', 'big']"
```

案例：京东秒杀 tab 导航高亮


```
▼<ul> flex
  ▼<li>
      <a href="#" class="active"> 京东秒杀 </a> == $0
    </li>
  ▼<li>
      <a href="#" class> 每日特价 </a>
    </li>
  ▼<li>
      <a href="#" class> 品类秒杀 </a>
    </li>
</ul>
```

```
list: [
  { id: 1, name: '京东秒杀' },
  { id: 2, name: '每日特价' },
  { id: 3, name: '品类秒杀' }
]
```

核心思路：

1. 基于数据动态渲染 tab → v-for
2. 准备下标记录高亮的是哪一个 tab → activeIndex
3. 基于下标，动态控制 class 类名 → v-bind:class

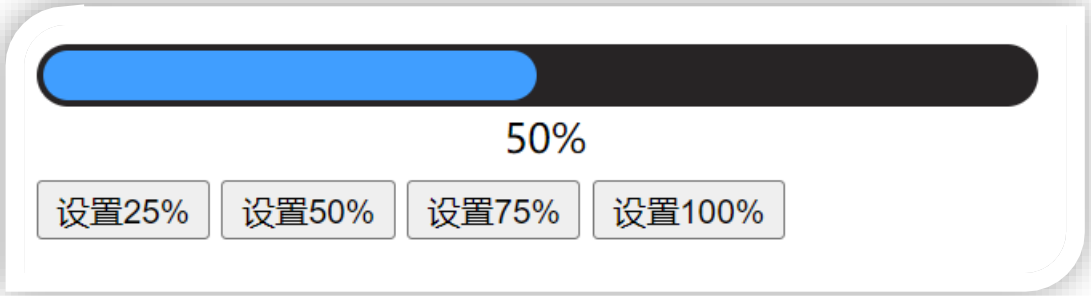
所谓切换高亮，其实就是改下标

v-bind 对于样式控制的增强 - 操作style

语法　:style = "样式对象"

```
<div class="box" :style="{ CSS属性名1: CSS属性值, CSS属性名2: CSS属性值 }"></div>
```

适用场景：某个具体属性的动态设置



目录

Contents

v-model 应用于其他表单元素

常见的表单元素都可以用 v-model 绑定关联 → 快速 获取 或 设置 表单元素的值

它会根据 控件类型 自动选取 正确的方法 来更新元素

输入框 input:text → value

文本域 textarea → value

复选框 input:checkbox → checked

单选框 input:radio → checked

下拉菜单 select → value

...

目录

Contents

计算属性

概念：基于现有的数据，计算出来的新属性。 依赖的数据变化，自动重新计算。

语法：

① 声明在 computed 配置项中，一个计算属性对应一个函数

② 使用起来和普通属性一样使用 {{ 计算属性名 }}

计算属性 → 可以将一段 求值的代码 进行封装

小黑的礼物清单

名字	数量
篮球	1个
玩具	2个
铅笔	5个

礼物总数：？个 {{ 计算属性名 }}

```
data: {
  list: [
    { id: 1, name: '篮球', num: 1 },
    { id: 2, name: '玩具', num: 2 },
    { id: 3, name: '铅笔', num: 5 },
  ]
},
```

```
computed: {
  计算属性名 () {
    基于现有数据，编写求值逻辑
    return 结果
  }
},
```

目录

Contents

computed 计算属性 vs methods 方法

computed 计算属性：

作用：封装了一段对于数据的处理，求得一个结果。

语法：

① 写在 computed 配置项中

② 作为属性，直接使用 → this.计算属性 {{ 计算属性 }}

methods 方法：

作用：给实例提供一个方法，调用以处理业务逻辑。

语法：

① 写在 methods 配置项中

② 作为方法，需要调用 → this.方法名( ) {{ 方法名( ) }} @事件名="方法名"

缓存特性（提升性能）：

计算属性会对计算出来的结果缓存，再次使用直接读取缓存，

依赖项变化了，会自动重新计算 → 并再次缓存

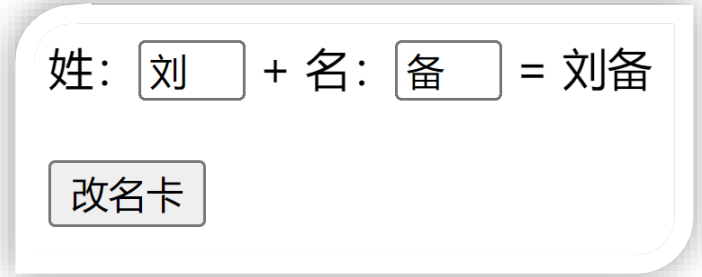
目录

Contents

计算属性完整写法

计算属性默认的简写，只能读取访问，不能 "修改"。

如果要 "修改" → 需要写计算属性的完整写法。




```
computed: {

  计算属性名 () {
    一段代码逻辑（计算逻辑）
    return 结果
  }

}
```

```
computed: {
 计算属性名: {
    get() {
      一段代码逻辑（计算逻辑）
      return 结果
    },
    set(修改的值) {
      一段代码逻辑（修改逻辑）
    }
  }
}
```

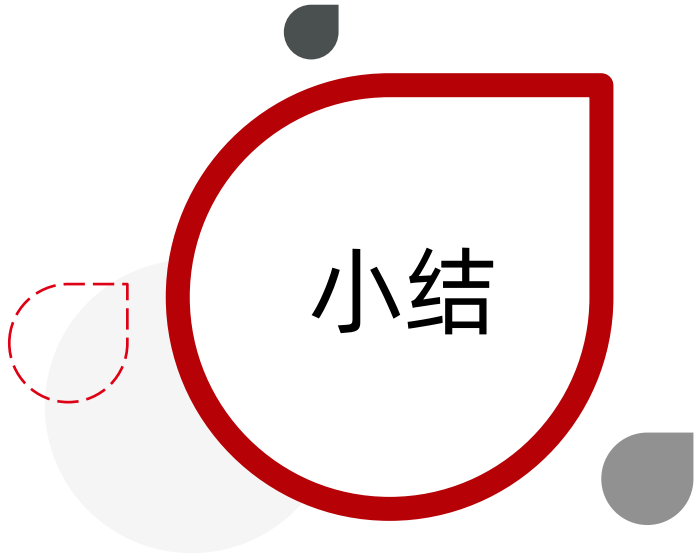
目录

Contents

综合案例 - 成绩案例

需求说明：

1. 渲染功能
2. 删除功能
3. 添加功能
4. 统计总分，求平均分



业务技术点总结：

1. 渲染功能（不及格高亮）

   v-if v-else　　v-for　　v-bind:class

2. 删除功能

   点击传参　filter过滤覆盖原数组

   .prevent 阻止默认行为

3. 添加功能

   v-model　v-model修饰符(.trim　.number)

   unshift 修改数组更新视图

4. 统计总分，求平均分

   计算属性　reduce求和

目录

Contents

watch 侦听器（监视器）

作用：监视数据变化，执行一些 业务逻辑 或 异步操作。

语法：

① 简单写法 → 简单类型数据，直接监视

② 完整写法 → 添加额外配置项

需求：输入内容，实时翻译

```
data: {
  words: '苹果',
  obj: {
    words: '苹果'
  }
},

watch: {
  // 该方法会在数据变化时，触发执行
  数据属性名 (newValue, oldValue) {
    一些业务逻辑 或 异步操作。
  },
  '对象.属性名' (newValue, oldValue) {
    一些业务逻辑 或 异步操作。
  }
}
```

watch 侦听器（监视器）

② 完整写法 → 添加额外配置项

(1) deep: true 对复杂类型深度监视

(2) immediate: true 初始化立刻执行一次handler方法

需求：输入内容，修改语言，都实时翻译

```
data: {
  obj: {
    words: '苹果',
    lang: 'italy'
  },
},

watch: {// watch 完整写法
  数据属性名: {
    deep: true, // 深度监视
    handler (newValue) {
      console.log(newValue)
    }
  }
}
```

watch 侦听器（监视器）

② 完整写法 → 添加额外配置项

(1) deep: true 对复杂类型深度监视

(2) immediate: true 初始化立刻执行一次handler方法

需求：默认文本，一进入页面，立刻翻译一次

```
data: {
  obj: {
    words: '苹果',
    lang: 'italy'
  },
},

watch: {// watch 完整写法
  数据属性名: {
    deep: true, // 深度监视
    immediate: true, // 是否立刻执行一次handler
    handler (newValue) {
      console.log(newValue)
    }
  }
}
```

小结

watch侦听器的语法有几种？

① 简单写法 → 监视简单类型的变化

```
watch: {
  数据属性名 (newValue, oldValue) {
    一些业务逻辑 或 异步操作。
  },
  '对象.属性名' (newValue, oldValue) {
    一些业务逻辑 或 异步操作。
  }
}
```

② 完整写法 → 添加额外的配置项 (深度监视复杂类型，立刻执行)

```
watch: {// watch 完整写法
  数据属性名: {
    deep: true, // 深度监视(针对复杂类型)
    immediate: true, // 是否立刻执行一次handler
    handler (newValue) {
      console.log(newValue)
    }
  }
}
```

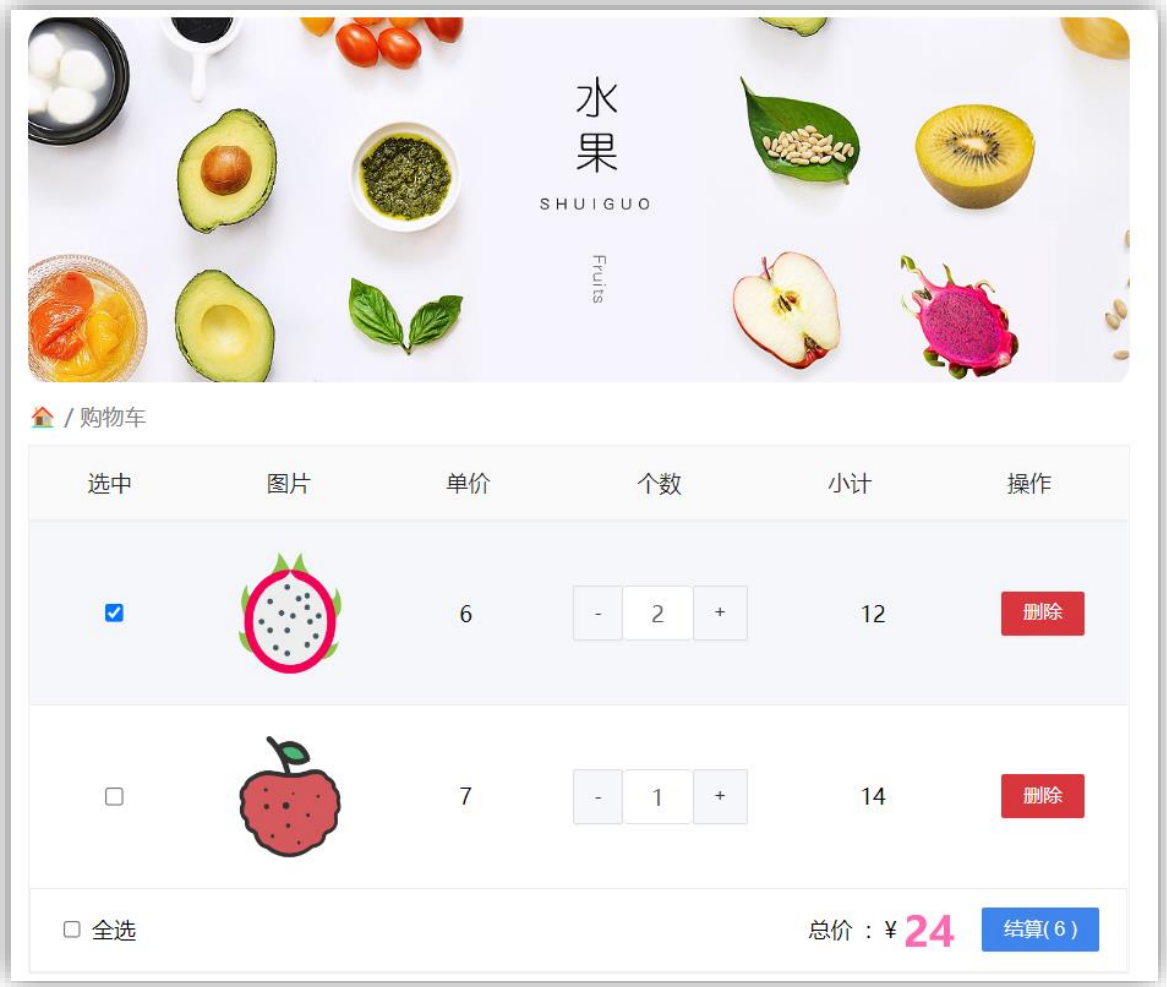
目录

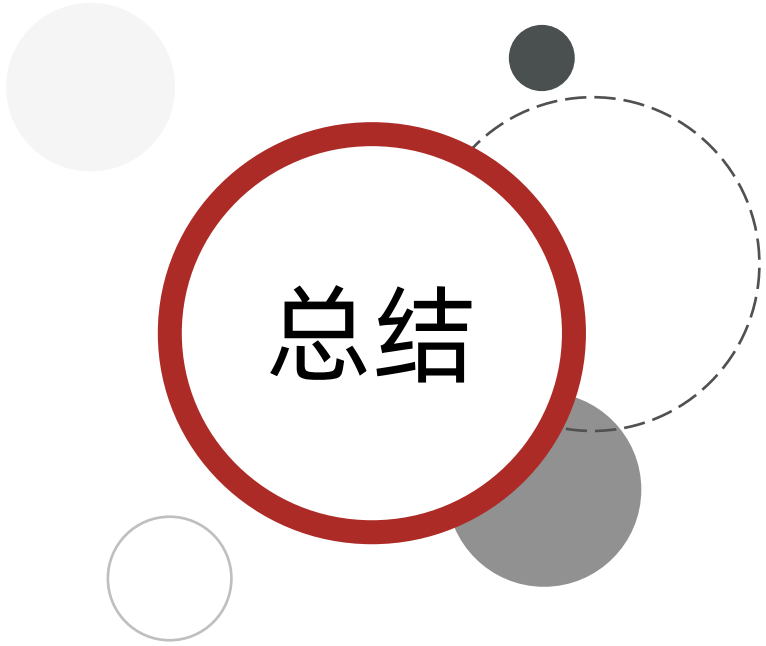
Contents

综合案例 - 水果购物车

需求说明：

1. 渲染功能
2. 删除功能
3. 修改个数
4. 全选反选
5. 统计 选中的 总价 和 总数量
6. 持久化到本地





业务技术点总结：

1. 渲染功能： v-if/v-else　v-for　:class

2. 删除功能：点击传参　filter过滤覆盖原数组

3. 修改个数：点击传参　find找对象

4. 全选反选：计算属性computed　完整写法 get/set

5. 统计选中的总价和总数量: 计算属性computed　reduce条件求和

6. 持久化到本地： watch监视，localStorage，JSON.stringify，JSON.parse

传智教育旗下高端IT教育品牌