计算属性computed

写法是在computed中去定义，默认是只读的，修改一个计算属性时，你会收到一个运行时警告

```
export default {
  data() {
    return {
      author: {
        name: 'John Doe',
        books: [
          'Vue 2 - Advanced Guide',
          'Vue 3 - Basic Guide',
          'Vue 4 - The Mystery'
        ]
      }
    }
  },
  computed: {
    // 一个计算属性的 getter
    publishedBooksMessage() {
      // `this` 指向当前组件实例
      return this.author.books.length > 0 ? 'Yes' : 'No'
    }
  }
}
```

可修改的计算属性，通过同时提供 getter 和 setter 来创建

```
export default {
  data() {
    return {
      firstName: 'John',
      lastName: 'Doe'
    }
  },
  computed: {
    fullName: {
      // getter
      get() {
        return this.firstName + ' ' + this.lastName
      },
      // setter
      set(newValue) {
        // 注意：我们这里使用的是解构赋值语法
        [this.firstName, this.lastName] = newValue.split(' ')
      }
    }
  }
}
```

特性

计算属性值会基于其响应式依赖被缓存

```
<p>{{ calculateBooksMessage() }}</p>
```

```
methods: {
  calculateBooksMessage() {
    return this.author.books.length > 0 ? 'Yes' : 'No'
  }
}
```

若我们将同样的函数定义为一个方法而不是计算属性，两种方式在结果上确实是完全相同的，然而，不同之处在于**计算属性值会基于其响应式依赖被缓存。**一个计算属性仅会在其响应式依赖更新时才重新计算。这意味着只要 author.books 不改变，无论多少次访问 publishedBooksMessage 都会立即返回先前的计算结果，而不用重复执行 getter 函数。

计算属性的 getter 应只做计算而没有任何其他的副作用，这一点非常重要，请务必牢记。举例来说，**不要改变其他状态、在 getter 中做异步请求或者更改 DOM**！

侦听器watch

watch 选项期望接受一个对象，其中键是需要侦听的响应式组件实例属性 (例如，通过 data 或 computed 声明的属性)——值是相应的回调函数。该回调函数接受被侦听源的新值和旧值。
除了一个根级属性，键名也可以是一个简单的由点分隔的路径，例如 a.b.c。注意，这种用法不支持复杂表达式——仅支持由点分隔的路径。如果你需要侦听复杂的数据源，可以使用命令式的 $watch() API。
值也可以是一个方法名称的字符串 (通过 methods 声明)，或包含额外选项的对象。当使用对象语法时，回调函数应被声明在 handler 中。额外的选项包含：
immediate：在侦听器创建时立即触发回调。第一次调用时，旧值将为 undefined。
deep：如果源是对象或数组，则强制深度遍历源，以便在深度变更时触发回调。详见深层侦听器。
flush：调整回调的刷新时机。详见回调的触发时机及 watchEffect()。
onTrack / onTrigger：调试侦听器的依赖关系。详见侦听器调试。
声明侦听器回调时避免使用箭头函数，因为它们将无法通过 this 访问组件实例。

```
watch: {
  // 侦听根级属性
  a(val, oldVal) {
    console.log(`new: ${val}, old: ${oldVal}`)
  },
  // 字符串方法名称
  b: 'someMethod',
  // 该回调将会在被侦听的对象的属性改变时调动，无论其被嵌套多深
  c: {
    handler(val, oldVal) {
      console.log('c changed')
    },
    deep: true
  },
  // 侦听单个嵌套属性:
  'c.d': function (val, oldVal) {
    // do something
  },
  // 该回调将会在侦听开始之后立即调用
  e: {
    handler(val, oldVal) {
      console.log('e changed')
    },
    immediate: true
  },
  // 你可以传入回调数组，它们将会被逐一调用
  f: [
    'handle1',
    function handle2(val, oldVal) {
      console.log('handle2 triggered')
    },
    {
      handler: function handle3(val, oldVal) {
        console.log('handle3 triggered')
      }
      /* ... */
    }
  ]
},
```

计算属性和watch的区别

- 计算属性存在有缓存，页面下次更新时，如果依赖没有发生变化，计算属性不会触发
- 计算属性内部存在有return，不适用于异步请求或者更改 DOM
- 计算属性会在页面首次加载时就触发，watch需要配置immediate:true
- 计算属性对于数据是深度监听，watch需要配置deep:true