VUE3基础

1.安装VUE + Ts项目

在创建Vite项目之前，需要先安装Vite

```
npm install -g create-vite
```

接着使用 `npm create vite@latest`初始化项目,输入项目名称后选择Vue和TypeScript。选择后，Vite会自动生成一个项目模板。

也可以在命令行中直接制定具体项目格式

```
npm create vite@latest my-vue -- --template vue
```

最后使用 `npm intall`安装依赖就可以运行 `npm run dev`启动开发服务器了

通常会在 `http://localhost:5173`上运行

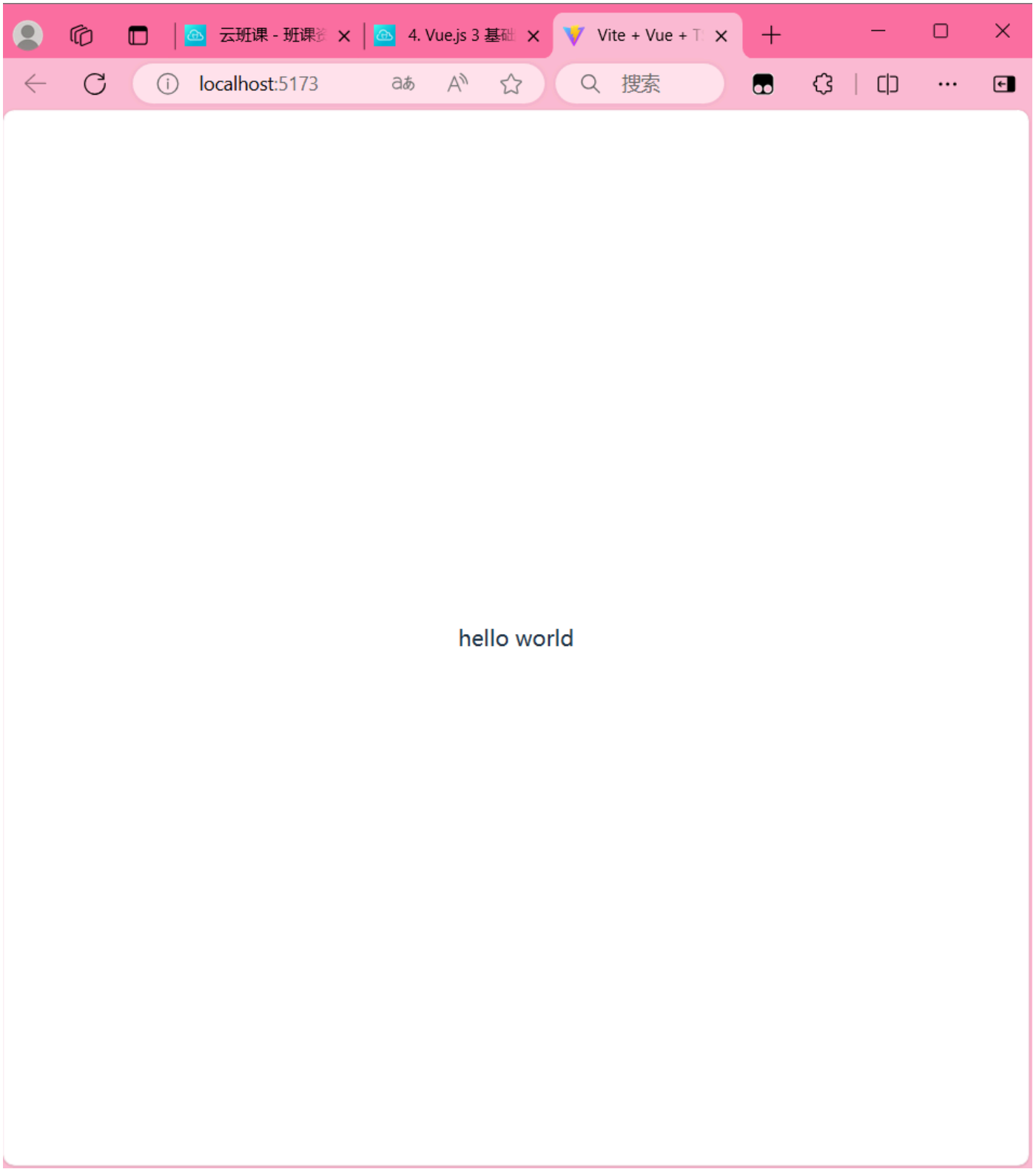
2.模板语法

2.1声明式渲染

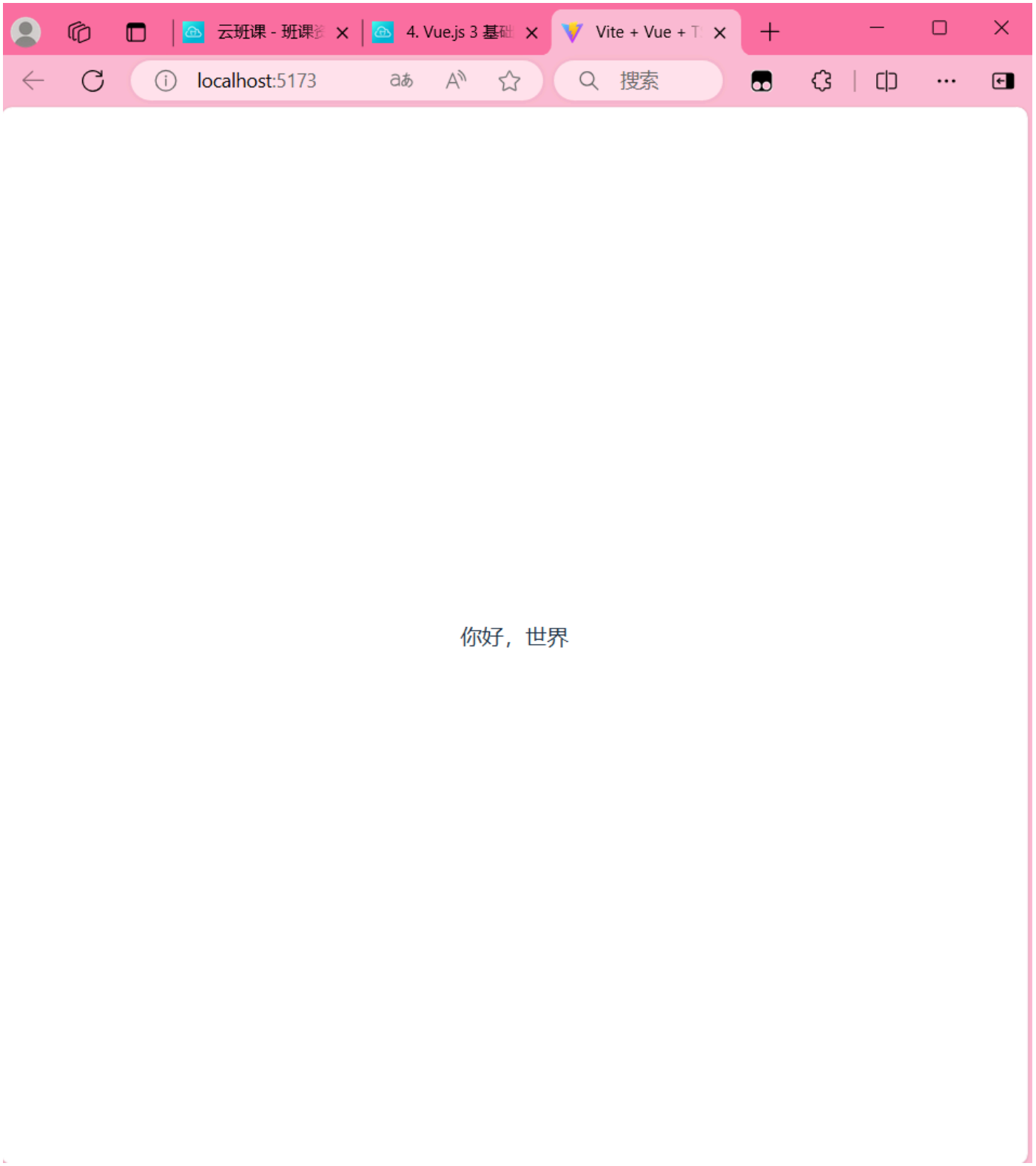
创建一个template组件

```vue
<template>
  <div>
    <Render/>
  </div>
</template>

<script setup lang="ts">
import SunWuKong from './components/SunWuKong .vue';
import WuKongSkills from './components/WuKongSkills.vue';
import WuKongHP from './components/WuKongHP.vue';
import WuKongAttack from './components/WuKongAttack.vue';
import Gear from './components/Gear.vue';
import Dom from './components/Dom.vue';

import Render from './components/vue3基础/基础/Render.vue';
import Mustache from './components/vue3基础/基础/Mustache.vue';
import AttibuteBinding from './components/vue3基础/绑定/AttibuteBinding.vue';
import ClassBinding from './components/vue3基础/绑定/ClassBinding.vue';
import StyleBinding from './components/vue3基础/绑定/StyleBinding.vue';
import CustomBinding from './components/vue3基础/绑定/CustomBinding.vue';
import Ref from './components/vue3基础/基础/Ref.vue';
import Reactive from './components/vue3基础/绑定/Reactive.vue';
import MethodsNoArg from './components/vue3基础/基础/MethodsNoArg.vue';
import MethodsWithArg from './components/vue3基础/基础/MethodsWithArg.vue';
import StateAndMethods from './components/vue3基础/基础/StateAndMethods.vue';
import BasicCondition from './components/vue3基础/渲染/BasicCondition.vue';
import ArrayListRender from './components/vue3基础/渲染/ArrayListRender.vue';
import ObjectArrayRender from './components/vue3基础/渲染/ObjectArrayRender.vue';
import IndexRender from './components/vue3基础/渲染/IndexRender.vue';
import ConditionAndListRender from './components/vue3基础/渲染/ConditionAndListRender.vue';
import ComputedPrice from './components/vue3基础/计算属性/ComputedPrice.vue';
import ComputedTypeDeclare from './components/vue3基础/计算属性/ComputedTypeDeclare.vue';
import WatchInput from './components/vue3基础/监听器/WatchInput.vue';
import WatchMultiState from './components/vue3基础/监听器/WatchMultiState.vue';
import DeepWatch from './components/vue3基础/监听器/DeepWatch.vue';
import ImmediatelyWatch from './components/vue3基础/监听器/ImmediatelyWatch.vue';
import ClickEvent from './components/vue3基础/事件处理/ClickEvent.vue';
import EventObject from './components/vue3基础/事件处理/EventObject.vue';
import PreventSubmit from './components/vue3基础/事件处理/PreventSubmit.vue';
```

将其导入App.vue中,这样元素组件就被嵌入index.html的div中，最后在浏览器上显示



单击这个div元素，触发方法



2.2组件系统

组件的代码结构从整体上来看可以分为3个区域

1. `<template></template>`：模板区域，基于HTML，声明式地绑定数据，表示页面结构；
2. `<script></script>`: 脚本区域，提供主要的数据和方法等;
3. `<style></style>`:样式区域，用来修饰模板的样式.

3.Vue.js的基础概念

3.1插值语法

```vue
<template>
    <div>
        <p>当前用户:{{ userName }}</p>
        <p>2+2的结果是:{{ 2 + 2 }}</p>
        <p>用户全名:{{ firstName + ' ' + lastName
    </div>
</template>

<script setup lang="ts">

import { ref } from 'vue'

const userName = ref<string>('张三')
const firstName = ref<string>('张')
const lastName = ref<string>('三')

</script>
```

{{}}可以动态的显示其中变量的值

当前用户:张三

也可以插入js表达式

2+2的结果是:4

用户全名:张 三

3.2属性绑定

使用 v-bind可以绑定一个元素的属性的值到变量上，借此操控该元素属性,v-bind也可以简写成 :

```vue
<template>
    <div>
        <button :disabled="isDisabled">点击我</bu
        <button v-bind="buttonattrs">点击我</butt
        <img :src="url"/>
    </div>
</template>

<script setup lang="ts">

import { ref } from 'vue'
import { reactive } from 'vue';

const isDisabled = ref<boolean>(false)
const buttonattrs = reactive({
    disabled: false,
    title: '按钮已禁用',
    id: 'unique-button'
})
const url = ref<string>(
    'https://img-s-msn-com.akamaized.net/tenant/a
)

</script>
```

也可以一次性将多个属性动态绑定到元素上，这通常通过一个对象来实现

```
▼ setup
  ▼ buttonattrs: Reactive
      disabled: false
      id: "unique-button"
      title: "按钮已禁用"
    isDisabled: false (Ref)
    url: "https://img-s-msn-com.akamaized.net/tenant/amp/entityid/AA
```

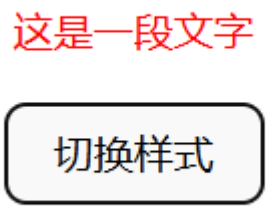
`v-bind`也可以绑定class和style，动态的切换类和样式

```vue
<template>
    <div>
        <p :class="pClass">这是一段文字</p>
        <button @click="toggleClass">切换样式</bu
    </div>
</template>

<script setup lang="ts">

import { ref } from 'vue'

const isHighLight = ref<boolean>(false)
const pClass = ref<{ [key: string]: boolean}>({
    isHighLight: isHighLight.value,
    normal: !isHighLight.value
})

const toggleClass = () => {
    isHighLight.value = !isHighLight.value
    pClass.value = {
        isHighLight: isHighLight.value,
        normal: !isHighLight.value
    }
}

</script>

<style scoped>
.isHighLight {
    color: red;
}
.normal {
    color: black;
}
</style>
```

这是一段文字

切换样式

点击按钮后切换类名，更换样式



通过绑定元素的style来操纵样式

```vue
<template>
    <div>
        <p :style="pStyle">这是一段有样式文字</p>
        <button @click="toggleStyle">切换样式</bu
    </div>
</template>

<script setup lang="ts">

import { ref } from 'vue'

const isBold = ref<boolean>(false)
const pStyle = ref<{
    fontWeight: string;
    color: string;
}>({
    fontWeight: 'normal',
    color: 'black'
})

const toggleStyle = () => {
    isBold.value = !isBold.value
    pStyle.value = {
        fontWeight: isBold.value ? 'bold' : 'norm
        color: isBold.value ? 'blue' : 'black'
    }
}

</script>

<style scoped>

</style>
```

这是一段有样式文字

切换样式

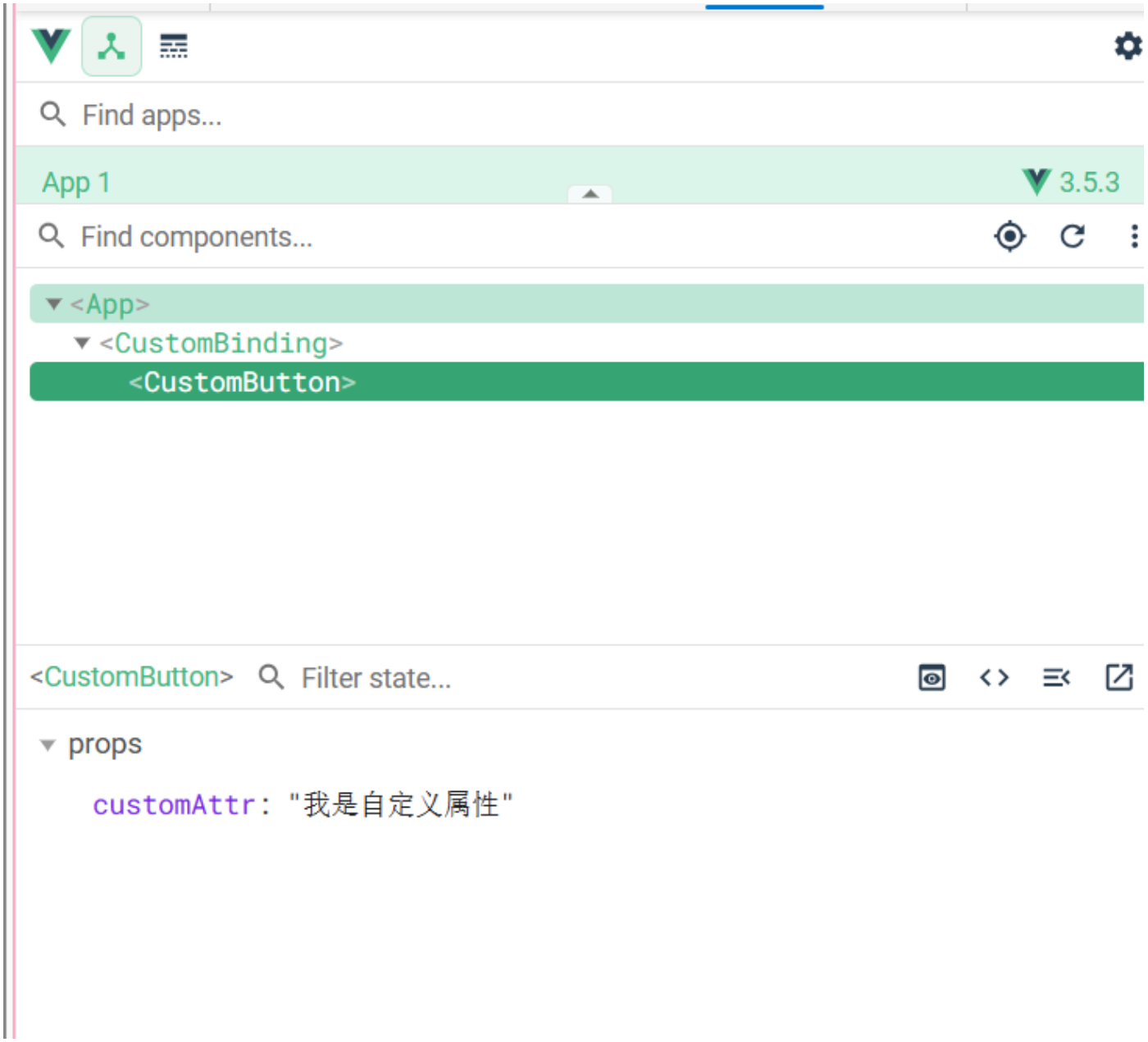
这是一段有样式文字

切换样式

v-bind也可以绑定自定义属性到任意html元素

```vue
<template>
    <div>
        <CustomButton :custom-attr="CustomValue">
    </div>
</template>

<script setup lang="ts">

import { ref } from 'vue'
import CustomButton from './CustomButton.vue';

const CustomValue = ref<string>('我是自定义属性')

</script>

<style scoped>

</style>
```

运行结果如下



3.3状态和方法

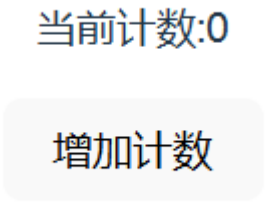
状态是组件内部存储和管理的数据，它在组件生命周期内变化，并驱动视图的更新。

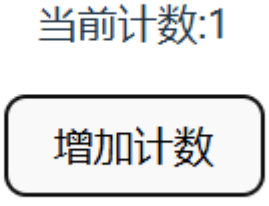
状态可以通过`ref`或`reactive`进行定义

- `ref`:用于基本数据类型以及简单对象,并通过 `.value`来访问包装的值
- `reactive`:适用于复杂对象（如数组、对象),`reactive`将整个对象转换为响应式的，当对象的任一属性发生改变时，Vue会自动追踪这些变化

使用ref定义一个简单状态

```vue
<template>
    <div>
        <p>当前计数:{{ count }}</p>
        <button @click="increment">增加计数</butt
    </div>
</template>

<script setup lang="ts">

import { ref } from 'vue';

const count = ref<number>(0)

const increment = (): void => {
    count.value ++
}

</script>

<style scoped>

</style>
```





通过按钮修改状态后，vue会自动更新视图

使用reactive定义复杂对象状态

```vue
<template>
    <div>
        <p>用户名: {{ user.name }}</p>
        <p>年龄: {{ user.age }}</p>
        <button @click="increaseAge">增加年龄</bu
    </div>
</template>

<script setup lang="ts">

import { reactive } from 'vue';

const user = reactive<{ name: string; age: number
    name: '张三',
    age: 25
})

const increaseAge = (): void => {
    user.age ++
}
</script>

<style scoped>

</style>
```