

第9周

组合式API

什么是组合式API

组合式 API 是 Vue 3 中的一项新特性，旨在改进 Vue 应用程序中的代码组织和逻辑复用。在 Vue 2.x 版本中，我们主要使用选项式 API，该 API 要求我们根据功能（如 `data`、`methods`、`computed` 等）组织代码。然而，随着应用程序的复杂度增加，这种方法可能导致代码的组织和维护变得更加困难。

组合式 API 的引入旨在解决这个问题，它允许我们根据逻辑关系组织代码，而不是功能。这意味着我们可以将相关的状态和函数放在一起，使得代码更加模块化和可维护。组合式 API 的核心是一组新的函数，如 `ref`、`reactive`、`computed` 和 `watch` 等，它们使我们能够更好地控制和管理 Vue 组件的状态和逻辑。

组合式API的优点

更好的逻辑关注点分离：组合式 API 允许我们根据逻辑关系组织代码，这有助于提高代码的可读性和可维护性。

更好的逻辑复用：通过将逻辑组织成可复用的函数，我们可以轻松地在不同的组件之间共享和复用代码。

更好的类型推导：由于组合式 API 的函数式本质，它为 TypeScript 提供了更好的类型推导支持，从而提高了代码的健壮性。

按需引入：组合式 API 允许我们只引入所需的功能，这有助于减小最终构建的大小。

选项式API

data

组合式API

```
<template>
| <button>{{ number }}</button>
</template>

<script>
export default {
  data() {
    return {
      number: 0,
    };
  },
};
</script>
```

```
<template>
| <button>{{ number }}</button>
</template>

<script setup>
import { ref } from "vue";

const number = ref(0);
</script>
```

选项式API

methods

组合式API

```
<template>
| <button @click="addNumber">{{ number }}</button>
</template>

<script>
export default {
  data() {
    return {
      number: 0,
    };
  },
  methods: {
    addNumber() {
      this.number++;
    },
  },
};
</script>
```

```
<template>
| <button @click="addNumber">{{ number }}</button>
</template>

<script setup>
import { ref } from "vue";

const number = ref(0);

function addNumber() {
  number.value++;
}
</script>
```

选项式API

watch

组合式API

```
<template>
| <button @click="addNumber">{{ number }}</button>
</template>

<script>
export default {
  data() {
    return {
      number: 0,
    };
  },
  methods: {
    addNumber() {
      this.number++;
    },
  },
  watch: {
    number() {
      alert("变了");
    },
  },
};
</script>
```

```
<template>
| <button @click="addNumber">{{ number }}</button>
</template>

<script setup>
import { ref, watch } from "vue";
const number = ref(0);
function addNumber() {
  number.value++;
}

watch(number, () => {
  alert(number.value);
});
</script>
```

选项式API

computed

组合式API

```
<template>
| <span>{{ biggerThanZero }}</span>
</template>

<script>
export default {
  // 声明一个数组
  data() {
    return {
      books: ["books1", "books2", "books3"],
    };
  },
  computed: {
    // 一个计算属性
    biggerThanZero() {
      // 数组的元素数量大于0返回Yes, 反之返回No
      return this.books.length > 0 ? "Yes" : "No";
    },
  },
};
</script>
```

```
<template>
| <span>{{ biggerThanZero }}</span>
</template>

<script setup>
import { ref, computed } from "vue";

// 声明一个数组
const books = ref(["book1", "book2", "book3"]);


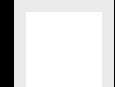
// 一个计算属性
const biggerThanZero = computed(() => {
  // 数组的元素数量大于0返回Yes, 反之返回No
  return books.value.length > 0 ? "Yes" : "No";
});
</script>
```

第9周

Element-plus

UI组件库


```
<template>  
| <button class="btn">Danger</button>  
</template>
```

```
<style>  
.btn {  
  background-color:  dodgerblue;  
  color:  white;  
  font-weight: bold;  
  border: none;  
  padding: 10px 15px 10px 15px;  
  border-radius: 18px;  
}  
</style>
```

A blue rounded rectangular button with the word "Primary" in white text.

Primary

Primary

```
<el-button type="primary">Primary</el-button>
```

Primary

```
<el-button type="primary" plain>Primary</el-button>
```

Primary

```
<el-button type="primary" round >Primary</el-button>
```



01

什么是UI组件库



什么是UI组件库

UI 组件库是一组**预制**的用户界面元素和交互组件，通常用于构建 Web 应用程序和移动应用程序。这些组件包括常用的 UI 控件和组件，如按钮、输入框、下拉框、表格、对话框等，以及其他交互元素，如进度条、轮播图、时间线等。

UI 组件库的作用在于提供了一套统一的视觉风格和交互行为，减少了前端开发者编写样式和交互代码的工作量。同时，UI 组件库还能够提高应用程序的可维护性和可扩展性，因为所有的组件都遵循相同的设计原则和代码风格，便于开发人员进行重用和修改。

Vue常见的UI组件库

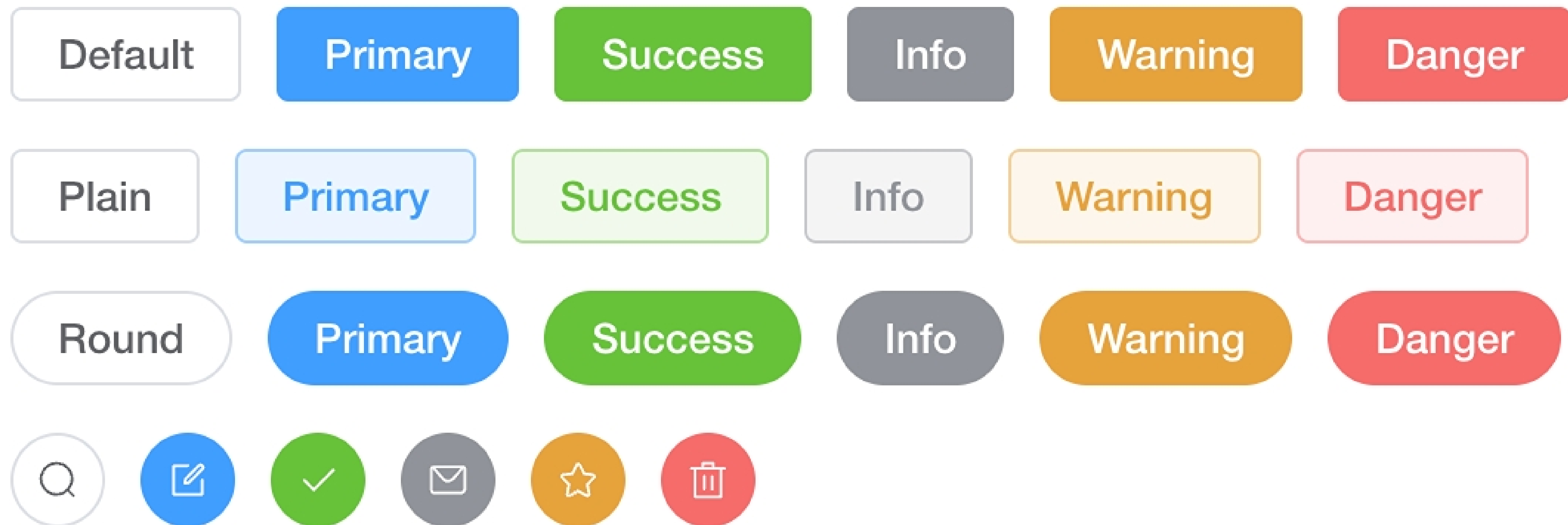
Element Plus: <https://element-plus.org/>

Ant Design Vue: <https://antdv.com/>

Vuetify: <https://vuetifyjs.com/>

Bootstrap Vue: <https://bootstrap-vue.org/>
Quasar Framework: <https://quasar.dev/>

Element Plus



Element Plus 的安装

```
npm install element-plus --save
```

■ Element Plus 的全局注册

main.js

JS main.js X

src > JS main.js

```
1  import { createApp } from "vue";
2  import App from "./App.vue";
3  import ElementPlus from "element-plus"; ←
4  import "element-plus/dist/index.css"; ←
5
6  import "./assets/main.css";
7
8  createApp(App).use(ElementPlus).mount("#app");
```




THANKS

THE END

