TO DO LIST

模块使用

import { ref, onMounted, computed, watch} from 'vue'

组合式 API (Composition API) 是一系列 API 的集合，使我们可以使用函数而不是声明选项的方式书写 Vue 组件。

- 响应式 API
  - 创建响应式引用 — ref()
  - reactive()
- 生命周期钩子
  - 生命周期钩子 — onMounted()
  - onUnmounted()
- 依赖注入 — provide() 和 inject()
- 计算属性函数 — computed
- 侦听器和观察者函数
  - watch
    - 选择一个对象，观察其变化

```
watch(data, (newValue, oldValue) => {
  console.log(`Data changed to: ${newValue} (from ${oldValue})`);
});
```

```
watch(data, (newValue, oldValue) => {
  console.log(`Data changed to: ${newValue}`);
}, {
  immediate: true
});
```

    - 后面还可以跟选项
      - immediate：一个布尔值，如果设置为 true，则在开始观察时立即触发回调函数。
      - deep：一个布尔值，如果设置为 true，则进行深度观察，意味着内部对象或数组的变化也会触发回调。
      - flush：指定副作用的调度时机。
  - watchEffect
- 副作用函数 — nextTick

使用工具

- VUE3
- VITE
  - Vite是一种新型的前端构建工具，它能显著改善前端开发体验。
  - 拓展
    - CommonJS：模块同步 — 运用在node.js
    - AMD：模块异步 — 依赖前置，加载完依赖后直接执行依赖，依赖加载成功后执行回调 — 预加载
    - CMD：模块异步 — 延迟执行，先加载所有依赖模块，运行时才执行require内容，并按顺序执行 — 懒加载
    - ESM
      - JavaScript 语言官方的模块系统，由ECMAScript 规范定义
      - 静态分析

变量声明

```
//定义变量
let theme=ref('normal') //定义响应式主题变量
const todos = ref([])   //存储输入的todo的中转变量
const name = ref('')    //定义用户名字的变量
const input_content = ref('') //输入的todo内容变量
const input_category = ref(null) //输入的类别变量
```

```
//计算属性，对所有todo进行排序
const todos_asc = computed(() => todos.value.sort((a, b) => {
  return a.createdAt - b.createdAt
}))
```

sort()数组排序函数，它通过跟着回调函数的返回值确定排序的规则，谁小谁在前面

监视变量

```
//监视所有todos的改变
watch(todos, newVal => {
  localStorage.setItem('todos', JSON.stringify(newVal))
}, {
  deep: true
})
```

将所有更改过的todo的所有属性（deep)存储在本地仓库中

```
//监视所有name的变化，将name存储到本地仓库
watch(name, (newVal) => {
  localStorage.setItem('name', newVal);
})
```

```
//监视所有theme的变化，将theme存储到本地仓库
watch(theme, (newVal) => {
  localStorage.setItem('theme', newVal)
})
```

```
//生命周期钩子函数，从本地仓库取出我们所需要的值
onMounted(() => {
  name.value = localStorage.getItem('name') || ''
  todos.value = JSON.parse(localStorage.getItem('todos')) || []
  theme.value = localStorage.getItem('theme') || 'normal'
})
```

取出本地存储值

功能函数

```
//添加tooo函数
const addTodo = () => {
  //判断
  if (input_content.value.trim() === '' || input_category.value === null) {
    return
  }
  //添加
  todos.value.push({
    content: input_content.value,
    category: input_category.value,
    done: false,
    editable: false,
    createdAt: new Date().getTime()
  })
  //清空输入框
  input_category.value=null
  input_content.value=''
}
```

添加todo

```
//删除todo函数
const removeTodo = (todo) => {
  todos.value = todos.value.filter((t)=>t !==todo)
}
```

删除todo

filter是数组的方法，用于创建新数组，包含通过测试的所有元素,后面是一个回调函数，筛选包含的元素

```
//改变颜色方法
const changeColor = async (event) => {
  if (event.target.id === 'color1') {
      theme.value = 'normal'
    }
  else if(event.target.id === 'color2'){
    theme.value = 'teal'
  }else if(event.target.id === 'color3'){
    theme.value = 'amber'
  }
};
```

改变主题颜色函数

HTML结构

```
<div class="nav">
  <ul>
    <li @click="changeColor($event)" id="color1">Normal</li>
    <li @click="changeColor($event)" id="color2">Teal</li>
    <li @click="changeColor($event)" id="color3">Amber</li>
  </ul>
</div>
```

导航条，包含主题颜色选择

```
<!-- 开场问候部分 -->
<section class="greeting">
  <h2 class="title">
    GET READY?
    <input
      type="text"
      id="name"
      placeholder="name here"
      v-model="name">
  </h2>
</section>
```

注意双向数据绑定

```
<!-- 创建todo部分 -->
<section class="create-todo">
  <h3>CREATE A TODO</h3>
  <form id="new-todo-form" @submit.prevent="addTodo">
    <h4>Get start by a todo first</h4>
    <input type="text" name="content" id="content" placeholder="e.g. do my homework" v-model="input_content" />
    <h4>PICK A CATEGORY</h4>
    <div class="options">
      <label for="bussiness">
        <input type="radio" name="category" value="bussiness" id="bussiness" v-model="input_category">
        <span class="bubble bussiness"></span>
        <div>Bussiness</div>
      </label>
      <label for="personal">
        <input type="radio" name="category" value="personal" id="personal" v-model="input_category">
        <span class="bubble personal"></span>
        <div>Personal</div>
      </label>
    </div>
    <input type="submit" value="Add todo" />
  </form>
</section>
```

```
<!-- todo列表展示 -->
<section class="todo-list">
  <h3>TODO LIST BELOW</h3>
  <div class="list" id="todo-list">
    <div v-for="todo in todos_asc" :class="`todo-item ${todo.done && 'done'}`">
      <label>
        <input type="checkbox" v-model="todo.done" />
        <span :class="`bubble ${todo.category == 'business'
            ? 'business'
            : 'personal'
          }`"></span>
      </label>
      <div class="todo-content">
      <input type="text" v-model="todo.content">
      </div>
      <div class="actions">
        <button class="delete" @click="removeTodo(todo)">Delete</button>
      </div>
    </div>
  </div>
</section>
```

CSS

使用变量法，对每一个类名下的变量重新定义，实现主题切换