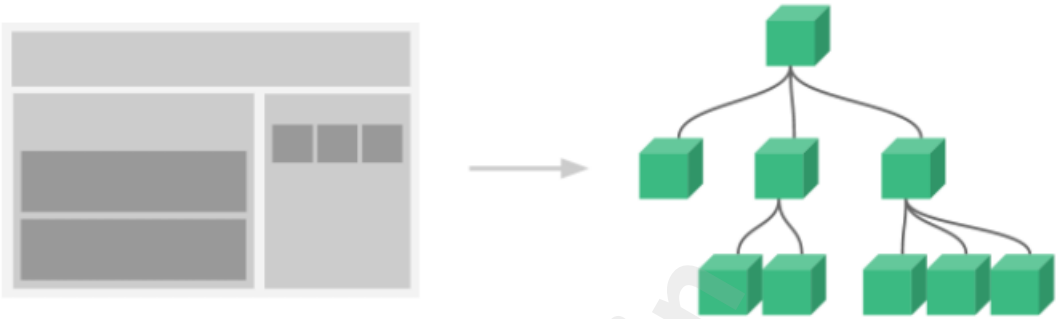
组件基础

Vue构建大规模前端页面采用了组件拼装的技术，即 我们可以把页面切割成大大小小不同的模块，每个模块又各自可以分割，这样依次循环下去。这样做避免了编写前端页面时重复造轮子，可以大大提高效率。



组件的使用方法是先定义组件，再注册组件

定义组件

```
//定义组件
const blog={
  data(){
    return {
      id:'1',
      title:'hhh',
    }
  },
  computed:{
    link(){
      return `/post/${this.id}`
    }
  },
  template: `
    <li><a :href="link">{{title}}</a></li>
  `,
}
```

组件可以复用，和根组件没什么不同，都可以定义data ,methods,computed,watched等等这些属性，案例中的模板后期会用文件代替，现在只作为案例。

注册组件

```
const app = Vue.createApp({
  data() {
    return {
      posts: [
```

```
            { id: 1, title: '红楼梦' },
            { id: 2, title: '三国演义' },
            { id: 3, title: '西游记' },
            { id: 4, title: '水浒传' },
          ]
        }
      },
    })
    //注册组件
    app.component('blogPost',blog)

    const vm = app.mount('#root')
```

通过app.component方法注册组件，其中组件名将作为自定义标签名使用在父模板中

模板

```
  <div id="root">
    <ul>
      <blog-post v-for="item in posts" :title="item.title" :id="item.id"></blog-
post>
    </ul>
  </div>
```

使用组件

你可以将组件进行任意次数的复用：

```
  <div id="root">
    <ul>
      <post></post>
      <post></post>
      <post></post>
      <post></post>
    </ul>
  </div>
```

`<post></post>` 这个标签即自定义的组件名 `app.component('post',blog)`

组件使用注意事项

在HTML中标签是不区分大小写的，如果注册的标签名是驼峰式，如blogPost，那么对应在html标签中应该写成 `<blog-post>`

```
<body>
  <div id="root">
    <ul>
      <blog-post v-for="item in posts" :title="item.title" :id="item.id">
</blog-post>
    </ul>
  </div>
  <script src="../vue.js"></script>
  <script>
    //定义组件
```

```
    const blog={
      props: ['title','id'],
      computed:{
        link(){
          return `/post/${this.id}`
        }
      },
      template: `
        <li><a :href="link">{{title}}</a></li>
      `,
    }

    const app = Vue.createApp({
      data() {
        return {
          posts: [
            { id: 1, title: '红楼梦' },
            { id: 2, title: '三国演义' },
            { id: 3, title: '西游记' },
            { id: 4, title: '水浒传' },
          ]
        }
      },
    })
    //注册组件
    app.component('blogPost',blog)

    const vm = app.mount('#root')

  </script>
</body>
```


组件的数据传递

父组件可以把数据传递给子组件

首先，子组件通过props声明接收

```
    //定义组件
    const blog={
      props: ['title','id'],
      computed:{
        link(){
          return `/post/${this.id}`
        }
      },
      template: `
        <li><a :href="link">{{title}}</a></li>
      `,
    }
```

父组件通过v-bind 来动态传递数据。属性必须和子组件里定义的props保持一致！

```
  <div id="root">
    <ul>
      <post v-for="item in posts" :title="item.title" :id="item.id"></post>
    </ul>
  </div>
```

ToDoList实战续

使用组件的方式改造项目，并且增加两个功能

功能1：添加勾选按钮，勾选当前任务表示完成。

功能2：添加移除单个任务的按钮，点击清除当前任务。

先做组件化改造

```
<body>
  <div id='todoapp'>
    <header>
      <h1>小海记事</h1>
      <input type="text" placeholder="请输入任务" v-model.trim="task"
@keyup.enter="add(task)">
    </header>
    <div class="main">
      <ul class="todo-list">
        <todo v-for="item in todo_list" :item="item"></todo>
      </ul>
    </div>
    <div class="footer">
      <span class="todo-count">3</span>
      <button class="clear-completed" @click="empty">清空</button>
    </div>
  </div>

  <script src="../vue.js"></script>
  <script>

    //创建vue对象
    const app = Vue.createApp({
      data(){
        todo_list = ['吃饭','睡觉','上班','学习']
        task=''
        return {
          todo_list,
          task
        }
      },
      methods:{
        add(task){
          if(task){
            this.todo_list.push(task)  // 将任务保持到任务列表
            this.task=''               // 清空文本框
          }
        },
        empty(){
```

```
          this.todo_list=[]
        }
      }
    })
    //定义组件
    const todo = {
      props:['item'],
      template: `
      <li class="todo">{{item}}</li>
      `
    }
    //注册组件
    app.component('todo',todo)
    const vm = app.mount('#todoapp') // 返回组件实例
  </script>
</body>
```

添加勾选功能

功能1：添加勾选按钮，勾选当前任务表示完成。

```
<body>
  <div id='todoapp'>
    <header>
      <h1>小海记事</h1>
      <input type="text" placeholder="请输入任务" v-model.trim="task"
@keyup.enter="add(task)">
    </header>
    <div class="main">
      <ul class="todo-list">
        <todo v-for="(item,index) in todo_list" :item="item" :index="index">
</todo>
      </ul>
    </div>
    <div class="footer" v-show="todo_list.length != 0">
      <span class="todo-count"><strong>{{todo_list.length}} tasks left</strong>
</span>
      <button class="clear-completed" @click="empty">清空</button>
    </div>
  </div>

  <script src="../vue.js"></script>
  <script>

    //创建vue对象
    const app = Vue.createApp({
      data(){
        todo_list = ['吃饭','睡觉','上班','学习']
        task=''
        return {
          todo_list,
          task
        }
      },
      methods:{
        add(task){
          if(task){
```

```
            this.todo_list.push(task)  // 将任务保持到任务列表
            this.task=''               // 清空文本框
          }
        },
        empty(){
          this.todo_list=[]
        }
      }
    })
    //定义组件
    const todo = {
      props:['item','index'],
      template: `
      <li class="todo">
        <div class="view">
          <input type="checkbox" :id="index" :value="item">
          <span class="index">{{index+1}}. </span>
          <label :for="index">{{item}}</label>
        </div>
      </li>
      `
    }
    //注册组件
    app.component('todo',todo)
    const vm = app.mount('#todoapp') // 返回组件实例
  </script>
</body>
```

移除当前任务功能

在每个任务后增加一个按钮，用以移除当前任务。

其中，父组件定义remove方法用于移除当前任务，再通过props把方法传给子组件，否则子组件调用不到该方法。

```
<body>
  <div id='todoapp'>
    <header>
      <h1>小海记事</h1>
      <input type="text" placeholder="请输入任务" v-model.trim="task"
@keyup.enter="add(task)">
    </header>
    <div class="main">
      <ul class="todo-list">
        <todo v-for="(item,index) in todo_list" :item="item" :index="index"
:remove="remove"></todo>
      </ul>
    </div>
    <div class="footer" v-show="todo_list.length != 0">
      <span class="todo-count"><strong>{{todo_list.length}} tasks left</strong>
</span>
      <button class="clear-completed" @click="empty">清空</button>
    </div>
  </div>

  <script src="../vue.js"></script>
  <script>
```

```
    //创建vue对象
    const app = Vue.createApp({
      data(){
        todo_list = ['吃饭','睡觉','上班','学习']
        task=''
        return {
          todo_list,
          task
        }
      },
      methods:{
        add(task){
          if(task){
            this.todo_list.push(task)  // 将任务保持到任务列表
            this.task=''               // 清空文本框
          }
        },
        empty(){
          this.todo_list=[]
        },
        remove(index){
          this.todo_list.splice(index,1)
        }
      }
    })
    //定义子组件
    const todo = {
      props:{
        item:{
          type: String,
          default: ''
        },
        index:{
          type: Number,
          default: NaN
        },
        remove:{
          type: Function,
          default: null,
        }
      },
      template: `
      <li class="todo">
        <div class="view">
          <input type="checkbox" :id="index" :value="item">
          <span class="index">{{index+1}}. </span>
          <label :for="index">{{item}}</label>
          <button class="destory" @click="remove(index)">X</button>
        </div>
      </li>
      `
    }
    //注册组件
    app.component('todo',todo)
    const vm = app.mount('#todoapp') // 返回组件实例
  </script>
</body>
```