Vue 高级特性

- 自定义v-model
- $nextTick
- ref
- slot 插槽
- 动态组件
- 异步组件
- 缓存组件(keep-alive)
- mixin 混入

1. 自定义 v-model

自定义组件实现双向数据绑定。 在父组件中引入子组件，并定义 `v-model` 指令，在子组件中定义 `input[type="text"]` `input[type="checkbox"]` `input[type="radio"]` 等元素，元素中包含 `:value="content"` 属性，`@input="'customEvent', $emit($event.target.value)"` 事件，并在子组件中声明 `model: { prop: 'content', event: 'customEvent' }`。

2. $nextTick

Vue 更新 DOM 是异步执行的，并且多项数据更新后并非多次更新渲染 DOM，而是一次性更新 DOM，这样做优化了性能。 数据更新如果引起了 DOM 的重绘、重排，需要等到 DOM 渲染完成后才能正确获取 DOM 节点相关的数据。这时候通过 `$nextTick` 来判断完成 DOM 渲染了，`$nextTick` 返回一个 `Promise`，所以可以使用 `await this.$nextTick()` 后，在获取 DOM 节点相关数据。

3. ref

Vue 是一门数据驱动视图的框架，不需要我们直接操作 DOM，如果一定要获取 DOM 节点相关的数据，可使用 `Vue.$refs` 来获取指定 DOM 节点的数据，比如：在元素中设置 `<ul ref="list"></ul>` 属性，则可以通过 `this.$refs.list` 获取到 `ul` 节点，进而获取需要的数据。

4. slot

slot 插槽，Vue 提供了内容分发的 API，就是 v-slot 指令，`slot` 元素接收父组件传来的的内容。

4.1. 后备内容

如果父组件没有传递任何内容，则使用子组件自身的 `<slot></slot>` 元素中的内容。

4.2. 具名插槽

在父组件中使用 `<template></template>` 模版声明 `v-slot:` 定义名称，然后在子组件的 `<slot></slot>` 元素中定义 `name` 属性，接收指定名称的插槽内容。`v-slot:` 可以缩写为 `#default`。

4.3. 作用域插槽

子组件中使用的数据都是父组件传递过来的，如果需要使用自身的数据，则需要用到作用域插槽。

父组件中使用 `<template v-slot:default="slotProps">{{ slotProps.info.firstName }}</template>`，子组件中使用 `<slot :info="user">{{ user.first }}</slot>` 获取到 `user` 对象的 `firstName` 的值，这个对象是子组件自身定义的数据。

5. 动态组件

Vue 内置了 `<component></component>` 组件，使用 `:is` 属性接收组件名，则可以根据不同的状态，动态接受不同的组件。

6. 异步组件

`import xxx from 'path'` 是同步引入的组件，而在 components 对象中使用 `xxx => import('path')` 则是异步引入组件，默认不引入，在需要使用 `xxx` 组件时才会引入。

7. 缓存组件(keep-alive)

Vue 3 中是大写 `<KeepAlive></KeepAlive>`，Vue 2 中是小写 `<keep-alive></keep-alive>`。

keep-alive 用于缓存组件，当切换到其它组件时，当前组件不会被卸载（不会调用 `beforeDestroy` 和 `destroyed` 钩子函数），当下一次切换到当前组件时，不会触发挂载（不会调用 `beforeCreate`, `created`, `beforeMount` 和 `mounted`钩子函数）。因为它已经被缓存了。

8. mixin

当多个组件有公共的逻辑时，可以将公共逻辑部分抽离出来，抽离出来的部分作为混入内容引入到组件中，这种操作方式就是混入 mixin。

```
<!-- index.vue -->
import common from 'common.js';

mixin: [common]
```

```
// common.js
export default {
    data() {
        content: ''
    },
    methods: {
        getData() {
            console.log("content", this.content);
        }
    }
}
```

混入之后，组件即拥有了公共逻辑中的变量和函数。