依赖注入

当我们需要从父组件向子组件传递数据时，会使用 `props`。如果有一些多层级嵌套的组件，形成了一颗巨大的组件树，而某个深层的子组件需要一个较远的祖先组件中的部分数据。在这种情况下，如果仅使用 `props` 则必须将其沿着组件链逐级传递下去，这会非常麻烦。

`provide` 和 `inject` 可以帮助我们解决这一问题。一个父组件相对于其所有的后代组件，会作为依赖提供者。任何后代的组件树，无论层级有多深，都可以注入由父组件提供给整条链路的依赖。

provide（提供）

要为组件后代提供数据，需要使用到 provide() 函数：

```
<script setup>
import { provide } from "vue";
provide(/* 注入名 */ "message", /* 值 */ "hello!");
</script>
```

除了在一个组件中提供依赖，我们还可以在整个应用层面提供依赖：

```
import { createApp } from "vue";
const app = createApp({});
app.provide(/* 注入名 */ "message", /* 值 */ "hello!");
```

inject（注入）

要注入上层组件提供的数据，需使用 inject() 函数：

```
<script setup>
import { inject } from "vue";
const message = inject("message");
</script>
```

一个完整的示例：

```
// App.vue

<script setup>
import { ref, provide } from "vue";
import Child from "./Child.vue";

// by providing a ref, the GrandChild
// can react to changes happening here.
```

```
const message = ref("hello");
provide("message", message);
</script>

<template>
  <input v-model="message" />
  <Child />
</template>
```

```
// Child.vue

<script setup>
import { ref, inject } from "vue";
import GrandChild from "./GrandChild.vue";
const value = inject("message");
</script>

<template>
  <GrandChild />
  <div>Child: {{ value }}</div>
</template>
```

```
// GrandChild.vue

<script setup>
import { inject } from 'vue'

const message = inject('message')
</script>

<template>
  <p>
    Message to grand child: {{ message }}
  </p>
</template>
```

Message to grand child: hello2

Child: hello2

如果想确保提供的数据不能被注入方的组件更改，你可以使用 `readonly()` 来包装提供的值。

```
<script setup>
import { ref, provide, readonly } from 'vue'
```

```
const count = ref(0)
provide('read-only-count', readonly(count))
</script>
```