slot 插槽

1. 插槽内容

插槽是将 元素作为承载分发内容的出口。像这样合成组件：

```
<navigation-link url="/profile">
  Your Profile
</navigation-link>
```

在 `<navigation-link>` 的模板中写为：

```
<a
  v-bind:href="url"
  class="nav-link"
>
  <slot></slot>
</a>
```

当组件渲染的时候，`<slot></slot>` 将会被替换为 “Your Profile”。插槽内可以包含任何模板代码，包括 HTML：

```
<navigation-link url="/profile">
  <!-- 添加一个 Font Awesome 图标 -->
  <span class="fa fa-user"></span>
  Your Profile
</navigation-link>
```

甚至其它的组件：

```
<navigation-link url="/profile">
  <!-- 添加一个图标的组件 -->
  <font-awesome-icon name="user"></font-awesome-icon>
  Your Profile
</navigation-link>
```

如果 `<navigation-link>` 的 `template` 中没有包含一个 `<slot>` 元素，则该组件起始标签和结束标签之间的任何内容都会被抛弃。

2. 后备内容

有时为一个插槽设置具体的后备 (也就是默认的) 内容是很有用的，它只会在没有提供内容的时候被渲染。例如在一个 `<submit-button>` 组件中：

```
<!-- <submit-button /> 组件 -->
<button type="submit">
  <slot></slot>
</button>
```

为了将 “Submit” 作为后备内容，我们可以将它放在 标签内：

```
<!-- <submit-button /> 组件 -->
<button type="submit">
  <slot>Submit</slot>
</button>
```

在一个父级组件中使用 `<submit-button>` 并且不提供任何插槽内容时：

```
<!-- <submit-button /> 组件的父组件 -->
<submit-button></submit-button>
```

后备内容 “Submit” 将会被渲染：

```
<button type="submit">Submit</button>
```

但是如果我们提供内容：

```
<submit-button>Save</submit-button>
```

则这个提供的内容将会被渲染从而取代后备内容：

```
<button type="submit">Save</button>
```

父组件中不提供任何插槽内容时，子组件中 `<slot>xxx</slot>` 的内容 `xxx` 将会被渲染。如果父组件提供了插槽内容，子组件中 `<slot>yyy</slot>` 的内容将会被替代。

3. 具名插槽

有时我们需要多个插槽。例如对于一个带有如下模板的 组件：

```
<div class="container">
  <header>
    <!-- 我们希望把页头放这里 -->
  </header>
  <main>
    <!-- 我们希望把主要内容放这里 -->
  </main>
  <footer>
    <!-- 我们希望把页脚放这里 -->
  </footer>
</div>
```

对于这样的情况，`<slot>` 元素有一个特殊的 attribute：`name`。这个 attribute 可以用来定义额外的插槽：

```
<div class="container">
  <header>
    <slot name="header"></slot>
  </header>
  <main>
    <slot></slot>
  </main>
  <footer>
    <slot name="footer"></slot>
  </footer>
</div>
```

一个不带 name 的 出口会带有隐含的名字“default”。 `<slot></slot>` 等价于 `<slot name="default"></slot>`。

在向具名插槽提供内容的时候，我们可以在一个 `<template>` 元素上使用 `v-slot` 指令，并以 `v-slot` 的参数的形式提供其名称：

```
<base-layout>
  <template v-slot:header>
    <h1>Here might be a page title</h1>
  </template>

  <p>A paragraph for the main content.</p>
  <p>And another one.</p>

  <template v-slot:footer>
    <p>Here's some contact info</p>
  </template>
</base-layout>
```

现在 `<template>` 元素中的所有内容都将会被传入相应的插槽。 **任何没有被包裹在带有 `v-slot` 的 `<template>` 中的内容都会被视为默认插槽的内容。**

如果希望更明确一些，仍然可以在一个 `<template>` 中包裹默认插槽的内容：

```
<base-layout>
  <template v-slot:header>
    <h1>Here might be a page title</h1>
  </template>
  <template v-slot:default>
    <p>A paragraph for the main content.</p>
    <p>And another one.</p>
  </template>
  <template v-slot:footer>
    <p>Here's some contact info</p>
  </template>
</base-layout>
```

具名插槽可以缩写：`v-slot:` 替换为 `#`。

```
<base-layout>
  <template #header>
    <h1>Here might be a page title</h1>
  </template>
  <template #default>
    <p>A paragraph for the main content.</p>
    <p>And another one.</p>
  </template>
  <template #footer>
    <p>Here's some contact info</p>
  </template>
</base-layout>
```

任何一种写法都会渲染出：

```
<div class="container">
  <header>
    <h1>Here might be a page title</h1>
  </header>
  <main>
    <p>A paragraph for the main content.</p>
    <p>And another one.</p>
  </main>
  <footer>
    <p>Here's some contact info</p>
  </footer>
</div>
```

`v-slot` **只能添加在** `<template>` **上。**

- 一个不带 name 的 <slot> 出口会带有隐含的名字“default”。**<slot></slot> 等价于 <slot name="default"></slot>**;
- 一个不带 v-slot 的 template 标签，即：**<template></template> 等价于 <template v-slot="default"></template>**;
- **任何没有被包裹在带有 v-slot 的 <template> 中的内容都会被视为默认插槽的内容。**即：入口 <template name="default"> paragraph </template>、 paragraph 、<template> paragraph </template> 都将替换 <slot></slot> 或 <slot name="default"></slot>。
- v-slot 也有缩写，即把参数之前的所有内容 (v-slot:) 替换为字符 #。例如 v-slot:header 可以被重写为 #header。

4. 作用域插槽

有时让插槽内容能够访问子组件中才有的数据是很有用的。

为了让 user 在父级的插槽内容中可用，我们可以将 user 作为 <slot> 元素的一个 attribute 绑定上去：

```
<!-- <current-user> 组件 -->
<span>
  <slot :info="user">
    {{ user.lastName }}
  </slot>
</span>

<!-- script -->
data() {
    user: {
        firstName: 'child-firstName'
    }
}
```

绑定在 <slot> 元素上的 attribute 被称为插槽 prop。现在在父级作用域中，我们可以使用带值的 v-slot 来定义我们提供的插槽 prop 的名字：

```
<current-user>
  <template v-slot:default="slotProps"> <!-- <template #default="slotProps"> -->
    {{ slotProps.info.firstName }} <!-- 这里的 info 需要和子组件中属性名 info 一致 -
->
  </template>
</current-user>
```

上面最终渲染的是子组件中的内容 child-firstName。

将包含所有插槽 prop 的对象命名为 slotProps，也可以使用任意的其它名字。