可以分为 静态传递 或者 使用 v-bind动态传递：

例如：

给 prop 传入一个静态的值：

```
<blog-post title="My journey with Vue"></blog-post>
```

也知道 prop 可以通过 v-bind 动态赋值，例如：

```
<!-- 动态赋予一个变量的值 -->
<blog-post v-bind:title="post.title"></blog-post>

<!-- 动态赋予一个复杂表达式的值 -->
<blog-post v-bind:title="post.title + ' by ' + post.author.name"></blog-post>
```

在上述两个示例中，我们传入的值都是字符串类型的，但实际上任何类型的值都可以传给一个 prop。

子组件接受：

```
export default {
  props : ["title"]
 }
 //或者
 export default {
  props : {
        title:{
        type:string,
        default:""
    }
  }
 }
```

详解：

传入一个数字：

```
<!-- 即便 `42` 是静态的，我们仍然需要 `v-bind` 来告诉 Vue -->
<!-- 这是一个 JavaScript 表达式而不是一个字符串。-->
<blog-post v-bind:likes="42"></blog-post>

<!-- 用一个变量进行动态赋值。-->
<blog-post v-bind:likes="post.likes"></blog-post>
```

传入一个布尔值：

```
<!-- 包含该 prop 没有值的情况在内，都意味着 `true`。-->
<blog-post is-published></blog-post>

<!-- 即便 `false` 是静态的，我们仍然需要 `v-bind` 来告诉 Vue -->
<!-- 这是一个 JavaScript 表达式而不是一个字符串。-->
<blog-post v-bind:is-published="false"></blog-post>

<!-- 用一个变量进行动态赋值。-->
<blog-post v-bind:is-published="post.isPublished"></blog-post>
```

传入一个数组：

```
<!-- 即便数组是静态的，我们仍然需要 `v-bind` 来告诉 Vue -->
<!-- 这是一个 JavaScript 表达式而不是一个字符串。-->
<blog-post v-bind:comment-ids="[234, 266, 273]"></blog-post>

<!-- 用一个变量进行动态赋值。-->
<blog-post v-bind:comment-ids="post.commentIds"></blog-post>
```

传入一个对象：

```
<!-- 即便对象是静态的，我们仍然需要 `v-bind` 来告诉 Vue -->
<!-- 这是一个 JavaScript 表达式而不是一个字符串。-->
<blog-post v-bind:author="{ name: 'Veronica', company: 'Veridian Dynamics'
}"></blog-post>

<!-- 用一个变量进行动态赋值。-->
<blog-post v-bind:author="post.author"></blog-post>
```

传入一个对象的所有属性：

> 如果你想要将一个对象的所有属性都作为 prop 传入，你可以使用不带参数的 v-bind (取代 v-bind:prop-name)。

例如，对于一个给定的对象 post：

```
post: {
  id: 1,
  title: 'My Journey with Vue'
}
```

下面的模板：

```
<blog-post v-bind="post"></blog-post>
```

等价于：

```
<blog-post
  v-bind:id="post.id"
  v-bind:title="post.title"
></blog-post>
```