# 1.组件通信-父传子p179

|  |
| --- |
|  |
|  |
|  |

# 2.组件通信-子传父p181

|  |
| --- |
|  |

# 3.组件通信-其他p182

# 演练1.父传子

## 1>用模板来创建一个项目,叫做lesson18-component-comm-dad-to-son,然后安装依赖,再输入npm run dev先把项目运行起来

|  |
| --- |
|  |

## 2.在src/components文件夹下面新建一个子组件叫做ChildComp,此时是空白的,不过没有问题,我们需要使用它

|  |
| --- |
|  |

## 3>把App.vue作为父组件.我们在里面导入子组件,然后定义一些响应式数据

|  |
| --- |
|  |

## 4>然后我们在App.vue的HTML模板里面使用子组件,标签把数据以自定义属性的方式传递给子组件

|  |
| --- |
|  |

## 5>然后我们回到子组件,使用defineProps函数来接收父组件的数据,注意这个变量会自动解包

|  |
| --- |
|  |

## 6>然后在子组件的html模板里面渲染父组件的数据,注意props会自动解包

|  |
| --- |
|  |

### 刷新页面,效果如下

|  |
| --- |
|  |

# 演练2:子传父

## 1>用模板创建一个项目,起名: lesson19-data-son-to-dad,然后进入目录,安装依赖,然后输入npm run dev使项目运行起来

|  |
| --- |
|  |

## 2>然后在components文件夹下面新建一个子组件ChildComp,在子组件里面创建一个按钮,给他绑定一个事件,这里需要用到一个defineEmits函数定义一个emits函数,然后在按钮的点击事件处理函数里面用这个emits函数来发送事件和数据

|  |
| --- |
|  |

## 3>然后在App.vue里面导入子组件,并且定义一个针对子组件发送的事件的处理函数

|  |
| --- |
|  |

### 运行程序,效果如下

|  |
| --- |
|  |

## 4>我们可以来试一试,传递一个对象看看

|  |
| --- |
|  |

### 发现也是可以获取到的

|  |
| --- |
|  |