****vue的父子组件之间是如何传值的？****  
首先呢，需要说说的是，vue既然有双向绑定，那为何会有父子组件之间的传值问题？这个问题也简单，vue的组件会供其他的vue页面进行调用，如果数组都是双向绑定的话，那么就容易混乱了，比如a,b页面绑了一个num=10，那b，c页面又绑了num=5,那vue实例的num到底听谁的？所以，这就是vue官网为什么说  
****组件之间的数据只能是单项流通的，而且由父组件传递给子组件****  
好，接下来就话不多说了，父子组件是如何传值的，而且谁是父谁是子呢？  
例子1：先写一个组件放在component文件夹下叫son.vue好了（有点剧透的命名...）

<template>

<div>

<button class="test-btn" @click="add">+</button>

<button class="test-btn" @click="minu">-</button>

<p class="text-link">这里是son的num：{{num}}</p>

</div>

</template>

<script>

export default {

*//props:["num"],//接收父组件传递过来的值，这里我先写上*

data () {

return {

num:0

}

},

methods:{

add(){*//es6的语法相当于add:function(){}*

this.num++;

},

minu(){

this.num--;

}

}

}

</script>

这个son.vue的组件相信大家都看得懂，加减num的组件。接下来写一个index.vue调用son.vue

//index.vue

<template>

<div>

<son v-bind:num="num"></son>//传递一个值给son.vue，这时候可以把son.vue的props那个注释注销掉了

<p class="text-link">这里是index的num：{{num}}</p>

</div>

</template>

<script>

import son from './../components/son'

export default {

data () {

return {

num:10

}

},

components:{

son

}

}

</script>

这时候两个num都是10。再次点击加减按钮，我们会发现，‘son的num’一直有变化，而‘index的num’一直是10，这就是数据的单项流通。那么我们如何点击按钮然后改变‘index的num’呢？这时候，需要$emit干活了。

我们需要在index.vue里改动一下代码  
首先：

<son v-bind:num="num" v-on:add="icr" v-on:minu="der"></son>*//v-on:add="icr"就是绑定一个自定义事件*

再增加

methods:{

icr(){

this.num++;

},

der(){

this.num--;

}

}

然后在son.vue中methods变成

methods:{

add(){

this.$emit("add");*//$emit("add")就是触发父组件中的add方法*

},

minu(){

this.$emit("minu");

}

}

所以，****$emit("xxx")触发了父组件的函数，改变了父组件的data的num值，父组件再通过props传值给子组件。从而实现数据传递，父子组件通信****。  
这是son.vue和index.vue的完整代码

//son.vue

<template>

<div>

<button class="test-btn" @click="add">+</button>

<button class="test-btn" @click="minu">-</button>

<p class="text-link">这里是{{num}}</p>

</div>

</template>

<script>

export default {

props:["num"],

data () {

return {

num:10

}

},

methods:{

add(){

this.$emit("add");

},

minu(){

this.$emit("minu");

}

}

}

</script>

//index.vue

<template>

<div>

<son v-bind:num="num" v-on:add="icr" v-on:minu="der"></son>

<p class="text-link">父{{num}}</p>

</div>

</template>

<script>

import son from './../components/son'

export default {

data () {

return {

num:10

}

},

components:{

son

},

methods:{

icr(){

this.num++;

},

der(){

this.num--;

}

}

}

</script>

****二、说说vuex以及他的state、actions、getters、mutations、modules、store****  
首先，vuex官网上说是一个vue的状态管理工具。可能状态比较难理解，大家可以简单地把状态理解成为vue的data里面的变量。当组件之间的data变量关系复杂一点的时候，就把其中的变量抽离出来管理。刚好大家可以看看上面，父子组件之间的num之间的通信是不是比较麻烦，改变数据还要用$emit。如果有一个地方跟仓库一样就存放着num的值，谁要用谁去请求num的值，谁想改就改该多好是吧，vuex就是干这个的，有点全局变量的意思。任何组件需要拿，改东西，都可以找他。

1、首先state是惟一的数据载体，跟仓库一样。  
2、而mutations是唯一可以改变state的值的东东，使用commit等。  
这两个是vuex最最基础缺一不可的。简单的vuex管理就使用这两个就行，如何使用vuex？看这里[https://segmentfault.com/a/11...](https://segmentfault.com/a/1190000015392842?_ea=3918042)  
3、getters的官方说明：派生出新的状态，这个比较难理解。简单来说，就是****过滤，组合！****  
比如说state里面存了一个数组，数组有好多个数据，而我只想要用status：0的那些个，就可以用getters。是不是有点过滤的意思。所以getters有时候还很好用，很必要！。  
4、actions是用来提交mutations，为什么不用actions直接修改state值呢？其实大多数的管理工具都是这样做的，比如redux，mobx。actions定义修改的动作并且进行数据请求，通过commit再去触发对应的mutations。是很重要的一个属性，也符合解耦的思想。  
5、modules也是辅助方法。比如modulesA有一个完整的state、actions、getters、mutations；modulesB也可以有一个完整的state、actions、getters、mutations，他就是将store分割成模块，避免混淆。