# 父子组建的通信方式

:class=[]

:style={'':'','':''}

prop是单项数据绑定，父级变化会传递到子级，但子级变化不会引起父级变化

验证：type:String,required:true,default: 'a',validator:function(value){return value>10}

type: String,Number,Boolean,Function,Object,Array

自定义事件：将子组建的数据传递给父组建

# Vuex

Npm install vuex

Vuex是一个基于vue的localstorage的存储机制

This.$store.commit是同步的mutation触发方式，this.$store.dispatch是异步的action的触发方式

使用vuex而不适用localstorage的优点在于，vuex在状态变更时可以驱动所有注册了它的组建。而localstorage只是存值，值变更后不会主动驱动变更，而必须被动取值来引起变更

Vue的优点不仅仅是主动触发变更，还在于他的异步机制，异步机制的action使它可以分发mutation，譬如，dispath一个事件，action中可以根据情况异步处理，多个异步处理还可触发不同的mutation。Action是它的又一亮点。

# Cookie,session,localstorage,sessionstorage的区别

总结：

Session存于服务器端，其余存于客户端

Cookie在http每次会话都会携带，基于域内共享，最大4k，只保存字符串。

Sessionstorage是浏览器的本地存储，基于浏览器的会话期间有效，最大5M

Localstorage是浏览器本地存储，浏览器关闭后也会持久保存，最大5M

1. Cookie

cookie的内容包括名字，值，过期时间，路径，域。路径和域一起构成cookie的作用范围，若不设置时间，则表示这个cookie的生命期为浏览器会话期间，关闭浏览器，cookie就会消失。这种生命周期为浏览器会话期的cookie被称为会话cookie。会话cookie一般不保存在硬盘而是保存在内存里，当然这个行为并不是规范规定的，若设置了过期时间，浏览器会把cookie保存到硬盘上，关闭后再打开浏览器这些cookie仍然有效直到超过设定的过期时间。对于保存在内存里的cookie，不同浏览器有不同的处理方式

1. Session

当程序需要为某个客户端的请求创建一个session时，服务器首先检查这个客户端的请求里是否已包含了一个session标识（称为session id），如果已包含则说明以前已经为此客户端创建过session，服务器就按照session id把这个session检索出来使用（检索不到，会新建一个），如果客户端请求不包含session id，则为客户端创建一个session并且生成一个与此session相关联的session id，session id的值应该是一个既不会重复，又不容易被找到规律以仿造的字符串，这个session id将被在本次响应中返回给客户端保存。保存这个session id的方式可以采用cookie，这样在交互过程中浏览器可以自动的按照规则把这个标识发送给服务器

1. **cookie和session的区别：**
2. cookie数据存放在客户的浏览器上，session数据放在服务器上
3. cookie不是很安全，别人可以分析存放在本地的cookie并进行cookie欺骗，考虑\*到安全应当使用session
4. session会在一定时间内保存在服务器上，当访问增多，会比较占用你服务器的性能，考虑到减轻服务器性能方面，应当使用cookie
5. 单个cookie保存的数\*据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie
6. 建议将登录信息等重要信息存放为session，其他信息如果需要保留，可以放在cookie中
7. session保存在服务器，客户端不知道其中的信心；cookie保存在客户端，服务器能够知道其中的信息
8. session中保存的是对象，cookie中保存的是字符串
9. session不能区分路径，同一个用户在访问一个网站期间，所有的session在任何一个地方都可以访问到，而cookie中如果设置了路径参数，那么同一个网站中不同路径下的cookie互相是访问不到的\*
10. **web Storage和Cookie的区别**

Web Storage的概念和cookie相似，区别是它是为了更大容量存储设计的，cookie的大小是受限的，并且每次请求一个新的页面的时候cookie都会被发送过去，这样无形中浪费了带宽，另外cookie还需要指定作用域，不可跨域调用。   
除此之外，web storage拥有setItem,getItem,removeItem,clear等方法，不像cookie需要前端开发者自己封装setCookie，getCookie。   
但是cookie也是不可或缺的，cookie的作用是与服务器进行交互，作为http规范的一部分而存在的，而web Storage仅仅是为了在本地“存储”数据而生   
sessionStorage、localStorage、cookie都是在浏览器端存储的数据，其中sessionStorage的概念很特别，引入了一个“浏览器窗口”的概念，sessionStorage是在同源的同窗口中，始终存在的数据，也就是说只要这个浏览器窗口没有关闭，即使刷新页面或进入同源另一个页面，数据仍然存在，关闭窗口后，sessionStorage就会被销毁，同时“独立”打开的不同窗口，即使是同一页面，sessionStorage对象也是不同的

Web Storage带来的好处：   
1、减少网络流量：一旦数据保存在本地之后，就可以避免再向服务器请求数据，因此减少不必要的数据请求，减少数

据在浏览器和服务器间不必要的来回传递   
2、快速显示数据：性能好，从本地读数据比通过网络从服务器上获得数据快得多，本地数据可以及时获得，再加上网

页本身也可以有缓存，因此整个页面和数据都在本地的话，可以立即显示   
3、临时存储：很多时候数据只需要在用户浏览一组页面期间使用，关闭窗口后数据就可以丢弃了，这种情况使用sessionStorage非常方便

1. **浏览器本地存储与服务器端存储的区别**

其实数据既可以在浏览器本地存储，也可以在服务器端存储   
浏览器可以保存一些数据，需要的时候直接从本地存取，sessionStorage、localStorage和cookie都是由浏览器存储在本地的数据   
服务器端也可以保存所有用户的所有数据，但需要的时候浏览器要向服务器请求数据。   
1、服务器端可以保存用户的持久数据，如数据库和云存储将用户的大量数据保存在服务器端   
2、服务器端也可以保存用户的临时会话数据，服务器端的session机制，如jsp的session对象，数据保存在服务器上，

实际上，服务器和浏览器之间仅需传递session id即可，服务器根据session id找到对应用户的session对象，会话数据仅在一段时间内有效，这个时间就是server端设置的session有效期

服务器端保存所有的用户的数据，所以服务器端的开销较大，而浏览器端保存则把不同用户需要的数据分别保存在用户各自的浏览器中，浏览器端一般只用来存储小数据，而非服务可以存储大数据或小数据服务器存储数据安全一些，浏览器只适合存储一般数据

1. **sessionStorage、localStorage和cookie的区别**

共同点：都是保存在浏览器端、且同源的   
区别：   
1、cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递，而sessionStorage和localStorage不会自动把数据发送给服务器，仅在本地保存。cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下   
2、存储大小限制也不同，cookie数据不能超过4K，同时因为每次http请求都会携带cookie、所以cookie只适合保存很小的数据，如会话标识。sessionStorage和localStorage虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大   
3、数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭之前有效；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie：只在设置的cookie过期时间之前有效，即使窗口关闭或浏览器关闭   
4、作用域不同，sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；localstorage在所有同源窗口中都是共享的；cookie也是在所有同源窗口中都是共享的   
5、web Storage支持事件通知机制，可以将数据更新的通知发送给监听者   
6、web Storage的api接口使用更方便

1. **sessionStorage与页面js数据对象的区别**

页面中一般的js对象的生存期仅在当前页面有效，因此刷新页面或转到另一页面这样的重新加载页面的情况，数据就不存在了   
而sessionStorage只要同源的同窗口中，刷新页面或进入同源的不同页面，数据始终存在，也就是说只要浏览器不关闭，数据仍然存在

# [NPM install -save 和 -save-dev 傻傻分不清](https://www.cnblogs.com/limitcode/p/7906447.html)

### 回顾 npm install 命令

最近在写Node程序的时候，突然对 npm install 的-save和-save-dev 这两个参数的使用比较混乱。其实博主在这之前对这两个参数的理解也是模糊的，各种查资料和实践后对它们之间的异同点略有理解。遂写下这篇文章避免自己忘记，同时也给node猿友一点指引。

我们在使用 npm install 安装模块的模块的时候 ，一般会使用下面这几种命令形式：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | npm install moduleName # 安装模块到项目目录下    npm install -g moduleName # -g 的意思是将模块安装到全局，具体安装到磁盘哪个位置，要看 npm config prefix 的位置。    npm install -save moduleName # -save 的意思是将模块安装到项目目录下，并在package文件的dependencies节点写入依赖。    npm install -save-dev moduleName # -save-dev 的意思是将模块安装到项目目录下，并在package文件的devDependencies节点写入依赖。 |

那么问题来了，在项目中我们应该使用四个命令中的哪个呢？这个就要视情况而定了。下面对这四个命令进行对比，看完后你就不再这么问了。

### npm install moduleName 命令

1. 安装模块到项目node\_modules目录下。  
2. 不会将模块依赖写入devDependencies或dependencies 节点。  
3. 运行 npm install 初始化项目时不会下载模块。

### npm install -g moduleName 命令

1. 安装模块到全局，不会在项目node\_modules目录中保存模块包。  
2. 不会将模块依赖写入devDependencies或dependencies 节点。  
3. 运行 npm install 初始化项目时不会下载模块。

### npm install -save moduleName 命令

1. 安装模块到项目node\_modules目录下。  
2. 会将模块依赖写入dependencies 节点。  
3. 运行 npm install 初始化项目时，会将模块下载到项目目录下。  
4. 运行npm install --production或者注明NODE\_ENV变量值为production时，**会**自动下载模块到node\_modules目录中。

### npm install -save-dev moduleName 命令

1. 安装模块到项目node\_modules目录下。  
2. 会将模块依赖写入devDependencies 节点。  
3. 运行 npm install 初始化项目时，会将模块下载到项目目录下。  
4. 运行npm install --production或者注明NODE\_ENV变量值为production时，**不会**自动下载模块到node\_modules目录中。

### 总结

devDependencies 节点下的模块是我们在开发时需要用的，比如项目中使用的 gulp ，压缩css、js的模块。这些模块在我们的项目部署后是不需要的，所以我们可以使用 -save-dev 的形式安装。像 express 这些模块是项目运行必备的，应该安装在 dependencies 节点下，所以我们应该使用 -save 的形式安装。

# iView在自定义主题时报错 Inline JavaScript is not enabled. Is it set in your options?

解决方案是按下列网址处理的

<https://blog.csdn.net/weixin_41652495/article/details/83626651>

# vue iview Input标签编译报错闭合标签报错，解决方法：

.eslintrc.js文件中，rules，增加

"vue/no-parsing-error": [2, { "x-invalid-end-tag": **false** }]

# Eslint script标签内部 import置顶的问题，

解决方法：

<https://segmentfault.com/q/1010000013682009?sort=created>

关于eslint规范过程中遇到的问题

起初安装的eslint是针对vue的，所以配置文件中是

plugins: [  
 'vue'  
],

然后出现了，script标签中，export default 缩进问题，webstorm会自动缩进一个tab，但是eslint报错，认为应该无缩进，然后找到下列方法解决：

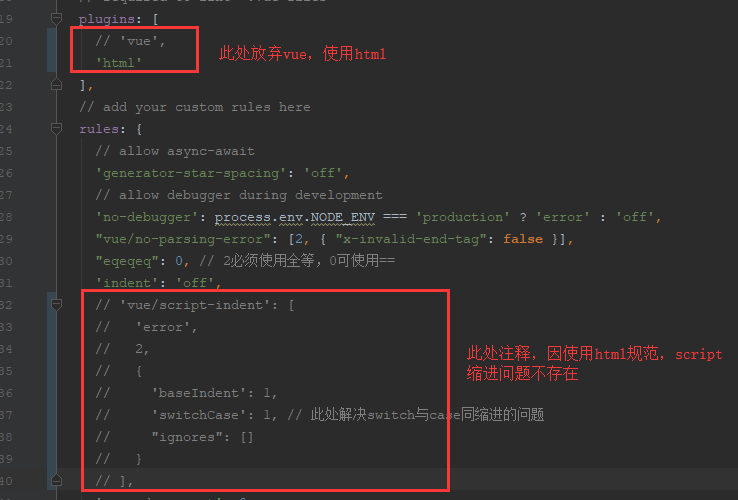
在.eslintrc.js配置文件中增加：

'vue/script-indent': [  
 'error',  
 2,  
 {  
 'baseIndent': 1,  
 'switchCase': 1, // 此处解决switch与case同缩进的问题  
 "ignores": []  
 }  
 ],

得以解决

但后续又发现问题，ctrl+/注释代码时，vue文件中//符号是顶格的，于是eslint又报错了，认为//符号应该按正常的缩进来

然后找到了<https://blog.csdn.net/weixin_37219302/article/details/79685351>，才发现，wenstorm中的vue是按html规范来的，于是修改.eslintrc.js配置文件为



问题解决。