1. 对于事件的冒泡中的event.stopPropagation和event.preventDefault和return false的区别；

event.stopPropagation：这是阻止事件的冒泡，阻止事件向document上蔓延，没有阻止默认事件

event.preventDefault：只是阻止默认事件的发生

Return false：是做的比较绝，就是把上面两个事件都做了，他会同事阻止事件冒泡也会阻止默认事件

1. 在 angular1的项目中，实现 按需动态懒加载的方式，通过一个插件 oclazyload ，配置看对应的 <http://blog.csdn.net/qq_26744901/article/details/54238032>

当然这个还没试过，要用个项目试一波；

1. 对于 postCss 生态圈，主要作用如：

https://web-design-weekly.com/2016/06/04/postcss-what-it-is-and-what-it-can-do/

比如****lint****（译者注2：一种用来检测CSS代码的工具），添加****vendor prefixes****（译者注3：添加浏览器内核前缀，可以使用浏览器的一些独有特性），允许使用最新的CSS特性，在你的****CSS****里提供统计数据，或者是允许你使用****Sass****，****Less****或是****Stylus****等****CSS****预处理器。

当然用到对应的功能 ，需要下载不同的插件 比如 [Autoprefixer](https://github.com/postcss/autoprefixer" \o "Autoprefixer)，这个是给不同的浏览器添加 前缀的；

当然这个还没试过，要用个项目试一波；

同时这里还要看对应的 正则和 字符串的 replace方法

1. 对于JavaScript中的 Number 和 parseInt两个方法的共同点和不同点：

两个函数 Number(params); parseInt(params,几进制);（但是这个几进制，优先级低于 传入的参数对应的特征 比如：parseInt(011,10)，按照8进制来计算）；

共同点：这两个函数都不会一直按照十进制来进行运算，他通过传入的参数设置来进行判断；

parseInt("012"); //12

parseInt(012); //10

parseInt("012",8); //10

如果是数值：0开头为八进制 ，0x开头为16进制，否则为10进制，书写规范的情况下

如果为字符串：parseInt("0x12");//18 parseInt("012",8);//10

parseInt("012");//12

不同点：parseInt在转换含有不是数字特征以外的其他字符串的时候，可以截取前面一节；而Number就会直接转换成 NaN

1. 对于js脚本 修改 keyframes的用法，常规的 dom节点操作不行，需要获取 样式表；

方法如下：

/\*用js 控制 keyframes\*/  
//这里看清楚是 哪个对象 含有 deleteRule 和 insertRule 方法  
/\*  
\* document.styleSheets[0].deleteRule(index); 对应的 索引  
\* document.styleSheets[0].insertRule("string样式",position); 也是对应的 索引  
\* \*/  
**function** changeKeyframes(name,newCss,position){  
 **let** styleSheets = document.styleSheets[0];  
 /\*这里有两种 , 分辨 keyframes 的用 name （就是 animation 的名字）,如果是普通的css 分辨通过 selectorText的值（对应的 选择器值 ）\*/  
 **for**(**let** i=0;i<styleSheets.cssRules.length;i++){  
 **if**(styleSheets.cssRules[i].type == 7 && styleSheets.cssRules[i].name == name){  
 styleSheets.deleteRule(i);  
 styleSheets.insertRule(newCss,position);  
 console.log(document.styleSheets);  
 }  
 }  
}

changeKeyframes("try",`@keyframes try {   
 0% {  
 transform :translateX(100%) scale(1);  
 opacity:0;  
 }  
 100% {  
 transform :translateX(100px) scale(2);  
 opacity:1;  
 }  
}`,1);

1. 对于在页面上浏览 要上传的图片的问题；(不兼容低版本,ie10以上)

**I. URL.createObjectURL()**静态方法会创建一个 [DOMString](https://developer.mozilla.org/zh-CN/docs/Web/API/DOMString" \o "一个UTF-16字符串，JavaScript 正是使用了这种编码的字符串，所以 DOMString 直接被映射为（is mapped directly to）String。)，它的 URL 表示参数中的对象。这个 URL 的生命周期和创建它的窗口中的 [document](https://developer.mozilla.org/zh-CN/docs/Web/API/Document" \o "Document 接口代表在浏览器及服务器中加载任意 web 页面，也作为 web 页面内容（DOM tree， 包含如 <body> 和 <table> 等 element）的入口。其也为文档（document）提供了全局性的函数，例如获取页面的URL、在文档中创建新的 element 的函数。) 绑定。  
II.在每次调用 **createObjectURL()** 方法时，都会创建一个新的 URL 对象，即使你已经用相同的对象作为参数创建过。当不再需要这些 URL 对象时，每个对象必须通过调用 [URL.revokeObjectURL()](https://developer.mozilla.org/zh-CN/docs/Web/API/URL/revokeObjectURL" \o "URL.revokeObjectURL() 静态方法用来释放一个之前通过调用 URL.createObjectURL() 创建的已经存在的 URL 对象。当你结束使用某个 URL 对象时，应该通过调用这个方法来让浏览器知道不再需要保持这个文件的引用了。) 方法来释放。浏览器会在文档退出的时候自动释放它们，但是为了获得最佳性能和内存使用状况，你应该在安全的时机主动释放掉它们。

例：

<a href="javascript:;" class="file pick-btn">  
 <img ng-src="{{userMessage.homeProperty || 'img/bg-img.jpg'}}" alt="" style="width:50%;">  
 <input type="file" accept="image/\*" class="userGas" id="homeProperty" style="display:none;">  
</a>

**var** propertyOwner1 =document.getElementById("propertyOwner1");

Var filePropertyOwner1;

propertyOwner1.addEventListener("change",**function**(){  
 $scope.userMessage.propertyOwner1 = URL.createObjectURL(propertyOwner1.files[0]);  
 filePropertyOwner1 = propertyOwner1.files[0];  
 $scope.$digest();  
});

**var** formData = **new** FormData(); //相当于用js模拟表单提交

formData.append("filePropertyOwner1",filePropertyOwner1);

//最后把formData 用post的方法 提交给后台

1. 在jQuery中有一个 $.proxy(function,”函数执行的环境”);

这个可以用于 改变 当前函数的执行环境；例子：

(function($){

Let index = {

Init:function(){

let me = $(this);

Me.dom = $(“test”);

},

Render:function(){

Let me = $(this);

Me.on(“click”,$.proxy(me[“\_do”],me)); // 这里访问 对象的 属性或者方法用[] 比 . 执行更快

},

\_do:function(e){

//do something

//在他的事件执行环境里，通过 上面的方法 可以 获取到 index 这个this，而不是 事件的 执行环境 的 $(“test”)这个对象;获取这个对象可以通过 e.target 获取他的信息

}

}

})($); //这种写法可以 释放缓存，并且 让 局部变量获取比全局快