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);