# 调试JS代码

## 1. 说明



有时候在生产环境中跟踪问题时，会需要在 JavaScript 代码的指定位置添加 debugger 断点，对代码进行调试的情况。但通常情况下，我们又无法直接修改生产环境的 JS 代码并重新部署系统。这时候我们可以用下述两种方法进行调试。

> ⚠ 注意：如果要调试smartbi，最好在url中增加debug=true参数（例如http://192.168.1.10:16000/smartbi/vision/index.jsp?debug=true），否则前后端交互是加密了的。

## 2. 使用Charles工具调试

具体步骤如下：

1、使用新版本的 Charles 工具。
2、点击 Tools→"No Caching..."并启用。
3、关闭并重新打开浏览器删除缓存，刷新页面。
4、在 Charles 中找到需要修改的请求，工具-》Map Local...或Map Remote...
5、在 Local Path 中填入本地文件路径，或在 Remote 中填写 localhost 8080 映射自己机器上的代码。
6、重新刷新浏览器，Charles 会自动使用本地文件替换成请求的文件内容方便调试。
7、新版本的 Charles 中还包含 Breakpoints... 功能，允许你随意修改发出的请求与返回。

另，若由于前端使用 resolveMany 导致一次性请求多个文件，这样会造成不方便修改调试，可以首先将 jsloader 设置为 Map Local 并修改

```
resolveMany : function(names) {
        return;
        var namesToGet = [];
         ......
}
```

## 3. 使用控制台调试

除了上一种方法，还可以使用下面的方法在指定位置添加 debugger 断点，使用起来可能比修改 Charles 要方便一点，操作步骤如下：
1、先清空浏览器缓存文件。
2、打开登录页面时附带 debug 参数，使服务端对当前客户端的 JS 请求内容不进行压缩。

```
http://192.168.186.233:18080/smartbi/vision/index.jsp?debug=true
```

3、打开浏览器"JS控制台"，执行下面的示例代码。即可在相应 js 类(上例中的QueryView与Dashboard，需根据实际情况修改)的相应方法(上例中的openQueryViewEx等方法，需根据实际情况修改)的执行前后都插入debugger断点。

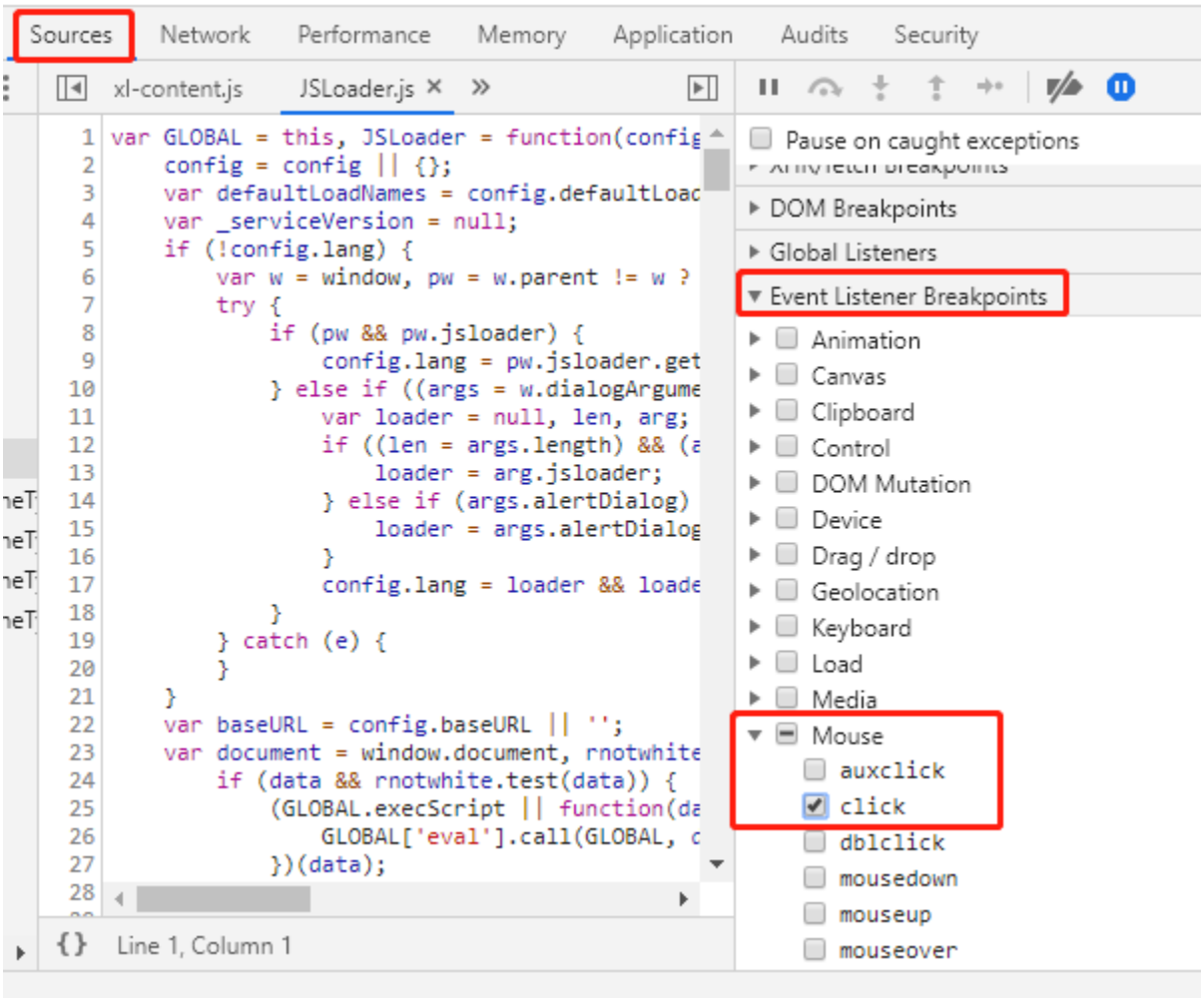
```
( function(jsloader,  breakPoints) {
    var  log = window.console ? window.console.log : function(msg){};
    if  (!jsloader) { log('jsloader not found!'); return; }
    var  clzName, Clazz, Proto, methodNames, len, i;
    for  (clzName in breakPoints) {
       Clazz  = jsloader.resolve(clzName);
       if  (Clazz && (Proto = Clazz.prototype)) {
           methodNames  = breakPoints[clzName];
           for (i  = 0, len = methodNames.length; i < len; i++) {
              (   function(methodName, Proto) {
                  var  tmpName = '__breakpoint_' + methodName;
                  if  (tmpName in Proto) {
                     Proto[methodName]  = Proto[tmpName];
                     Proto[tmpName]  = null;
                     delete  Proto[tmpName];
                     log('Detach  breakpoint: ' + clzName + '.prototype.' +  methodName);
                  }   else if  (Proto[methodName]) {
                     Proto[tmpName]  = Proto[methodName];
                     Proto[methodName]  = function() {
                         debugger;
                         Proto[tmpName].apply(this,  Array.prototype.slice.apply(arguments));
                         debugger;
                     };
                     log('Attach  breakpoint: ' + clzName + '.prototype.' +  methodName);
                  }
              })(methodNames[i],  Proto);
           }
       }
    }
})(window.jsloader, {
    'freequery.query.QueryView' : 'openQueryViewEx,openQueryViewHybrid,refreshData'.split(','),
    'bof.decisionpanel.dashboard.Dashboard' : 'setRefreshChartSize'.split(',')
});
```

4、恢复浏览器环境：重现执行一次第3步中的代码或刷新浏览器即可。

## 4.使用Event  Listener调试

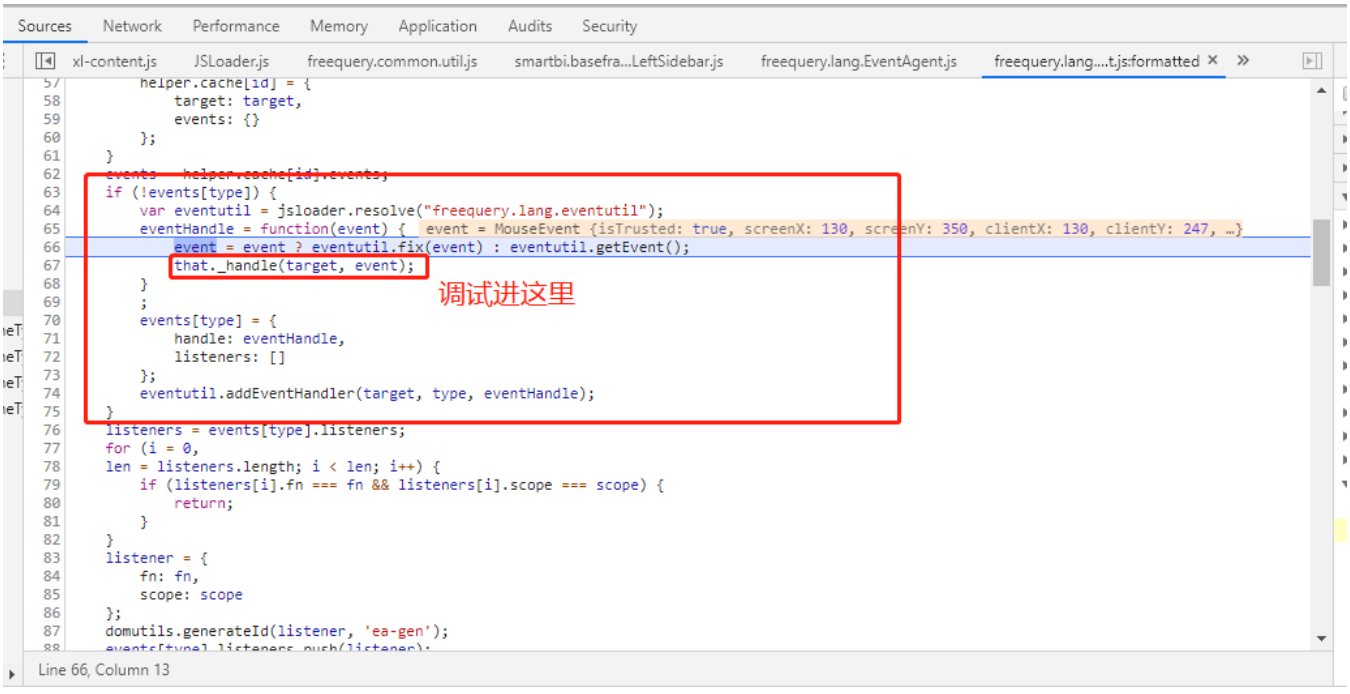
类似想知道点击了某个按钮，会执行什么操作之类，利用chrome开发者工具中事件断点（在网页上发生鼠标交互时会停在断点处的原理）。

1、设置鼠标监听，此处是设置了鼠标单击事件的监听。

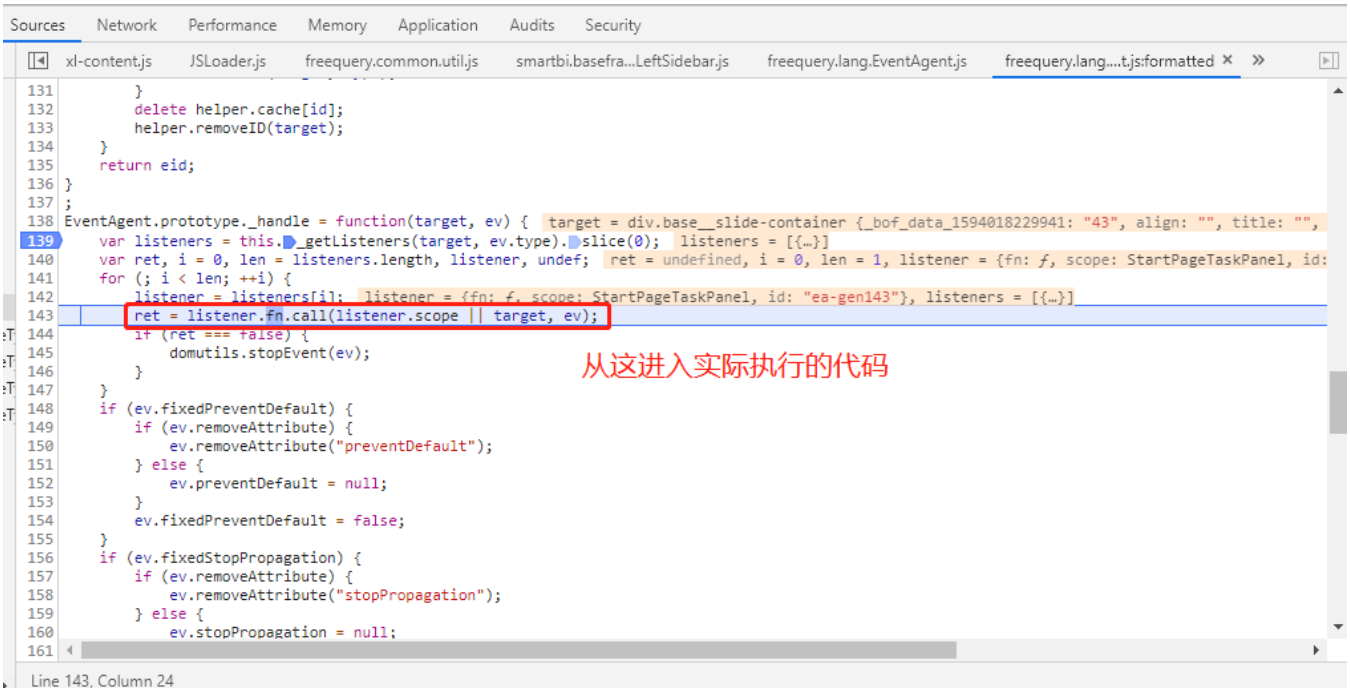



2、进入鼠标事件处理总入口

在页面单击后会进入到下图代码处，但是此时还不是实际的处理逻辑，下图代码为Smartbi鼠标事件的总入口，此时单步调试进去。



3、进入到dom元素的事件监听处理逻辑，每个dom元素都可以绑定多个鼠标单击事件监听，此处代码会遍历所有的监听是将并执行.




4、调试找到对应的代码后，回看堆栈：