错误处理与调试

管理科学与工程学科
耿方方

主要内容

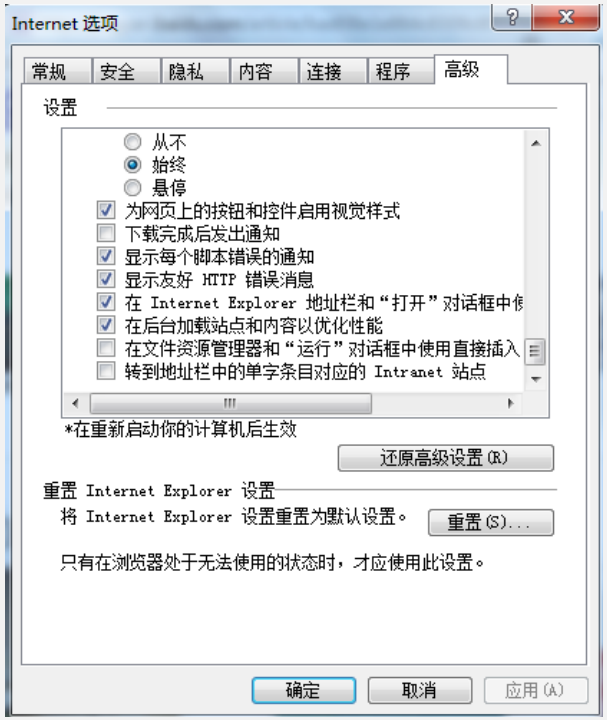
- 浏览器报告的错误
- 错误处理
- 调试技术
- 常见的IE错误

- 由于JavaScript本身是动态语言，而且多年来一直没有固定的开发工具，因此人们普遍认为它是一种最难于调试的编程语言。为了解决这个问题，引入了try-catch和throw语句以及一些错误类型，意在让开发人员能够适当的处理错误。2008年以来，大多数浏览器都已具备一些调试能力。

浏览器报告的错误

IE

- 每种浏览器都有JavaScript错误报告机制，只是报告方式不同而已。如果需要IE浏览器弹出错误报告对话框，需要设置IE浏览器，具体步骤为：选择IE浏览器菜单栏中“工具”/“Internet选项”，选择“高级”选项卡“显示每个脚本错误的通知”。



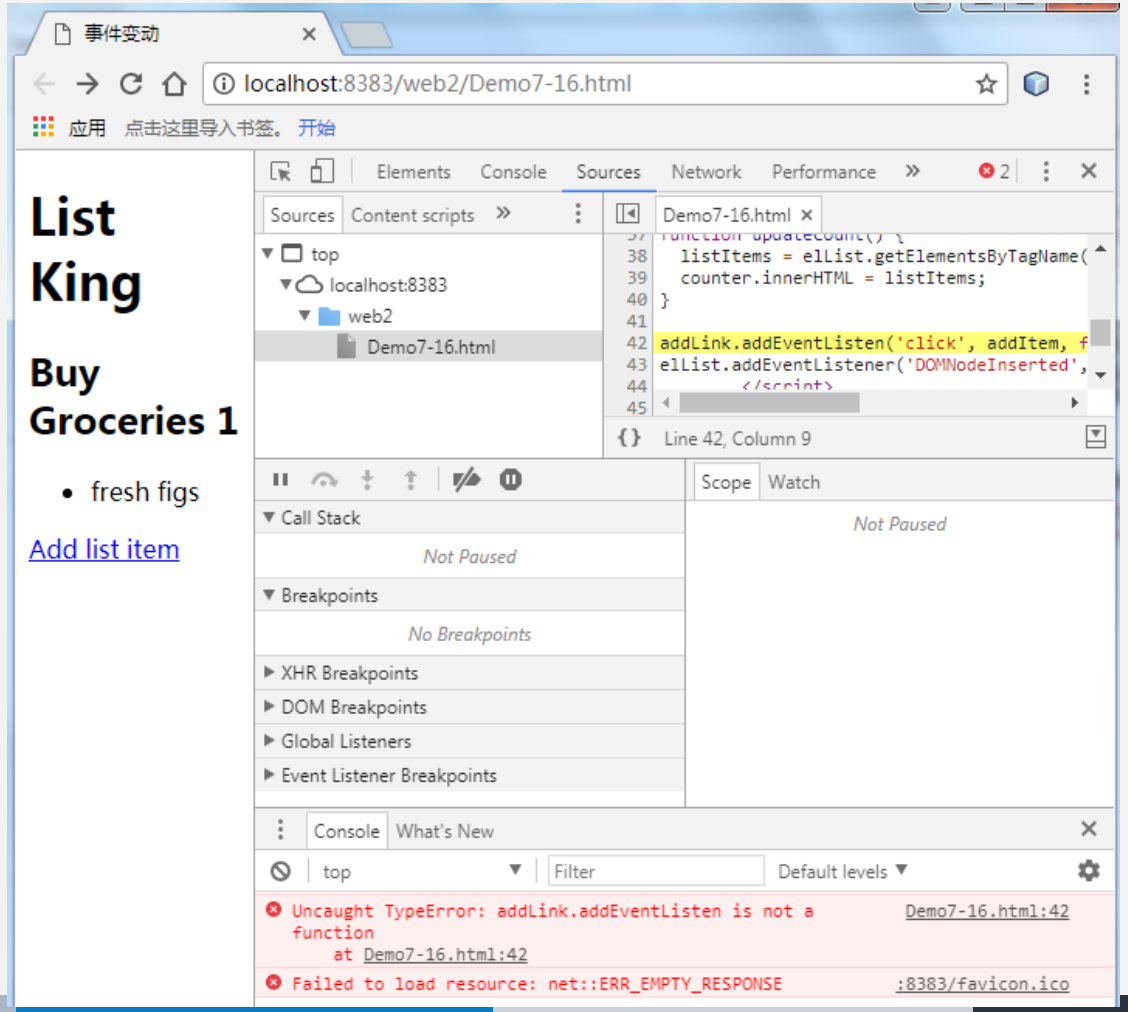
浏览器报告的错误

- 目前，最流行的Firefox插件Firebug，已经成为开发人员必备的JavaScript纠错工具。

浏览器报告的错误

Chrome

- Chrome在默认情况下也会隐藏JavaScript错误。所有错误都被记录到控制台中。找到“更多工具”/“开发者工具”，见下图。



错误处理

- 常见的错误类型有3中，分别为“语法异常”、“运行时的异常”和“逻辑异常”，其中“语法异常”通常是在程序员输入一些编译器无法识别的代码后发生的；“运行时的异常”实在运行时碰到一个错误时发生的，它与“语法异常”的区别在于它不一定是JavaScript语言的错误引发的异常；“逻辑异常”往往发生在程序设计时，程序没有按照预先设计的方式运行。
- JavaScript处理异常的方式通常有两种:一种方式使用onerror事件,该事件可以在window对象或图像对象上触发,而另一种方式是使用try...catch...finally模型。

错误处理

Error对象

- Error对象在创建后包含以下属性：

属性	描述
name	异常类型
message	描述
fileNumber	JavaScript文件名称
lineNumber	错误所在代码行数

- JavaScript有7种内置错误对象。

对象	描述
Error	一般错误——其他错误的基础
SyntaxError	语法未遵循规范
ReferenceError	尝试引用未在作用域内声明的变量
TypeError	意外的数据类型，无法进行自动转换
RangeError	数字超出了可接受的范围
URIError	encodeURI（）、decodeURI（）以及类似的方法会在出错时使用此类型
EvalError	eval（）函数会在出错时使用此类型

错误处理

onerror事件

- 触发onerror事件是最早用于处理JavaScript异常的机制，页面出现异常时，将触发onerror事件，该事件在window对象上触发。
- 语法：

```
<script>
window.onerror=function(){
        somestatements;
        return true;
}
</script>
```

- 注意：如果没有return true语句，在弹出错误提示对话框后，浏览器的错误报告会显示出来，为了隐藏此错误报告，函数需要返回true。

错误处理

onerror事件

- 除了window对象可以出发onerror事件之外，图像对象也可以触发onerror事件。
- 语法：

```
<script>
document.images[0].onerror=function(){
        somestatements;
        return true;
}
</script>
```

- 使用onerror事件处理异常除了可以捕捉异常之外，还可以提供如下3种信息来确定发生异常的详细信息。
- 异常信息：获取异常信息；
- URL：获取发生异常的文件的绝对路径；
- 行号：给定文件程序中发生异常的行号。

错误处理

onerror事件

- 案例1:
- <script language="javascript">
- window.onerror=function(msg,url,line){
- alert("您调用的函数不存在\n"+msg+"\n"+url+"\n"+line+"\n");//弹出错误提示对话框
- return true;
  //返回true
- }
- function ImgLoad(){
- document.images[0].onerror=function(){
- alert("您调用的图像不存在\n");
- };
- document.images[0].src="test.gif";
- }
- </script>

```
<body onload="ImgLoad()">
    <script language="javascript">
onHave();
            //调用不存在的onHave()函数
</script>
<img/>

    </body>
```

错误处理

try...catch

- JavaScript从Java语言中引入了try...catch...finally功能,语法:

```
<script language="javascript">
try{
        somestatements;
}
catch(exception e){
        somestatements;
}finally{
        somestatements;
}
</script>
```

- 即把所有可能会抛出错误的代码都放在try语句块中，把用于错误处理的代码放在catch块中。

错误处理

try...catch

- 即把所有可能会抛出错误的代码都放在try语句块中，把用于错误处理的代码放在catch块中。
- 例如:
- try{
- window.someNonexistentFunction();

} catch(error){ alert('an error happened!')};

finally子句在 try...catch语句中是可选的，一经使用，代码都会被执行。

例如: function testFinally(){

try{ return 2;}

catch(error){return 1;}

finally{return 0;}

}

错误处理

try...catch

- 案例2

```
<body>
<script>
try{
        document.forms.input.length;
}catch(exception){
        alert("没有定义表单以及文本框");
}finally{
        alert("结束try...catch...finally语句");
}
</script>
    </body>
```

错误处理

try...catch的嵌套

- 如果在catch区域发生异常，可以在catch区域中再使用一组try...catch语句，即嵌套使用try...catch语句。
- 语法:
- try{
- somestatements;
- }
- catch(exception){
- try{somestatements;}catch(exception){somestatements;}
- }finally{
- somestatements;}

错误处理

try...catch的嵌套

- 如果在catch区域发生异常，可以在catch区域中再使用一组try...catch语句，即嵌套使用try...catch语句。
- 语法:
- try{
- somestatements;
- }
- catch(exception){
- try{somestatements;}catch(exception){somestatements;}
- }finally{
- somestatements;}

错误处理

try...catch的嵌套

- 案例8-02-1:
- <script language="javascript">
- try{
- document.forms.input.length;
- }catch(exception){
- alert("try区域有异常发生"); //弹出错误提示信息
- try{
- document.forms.input.length;
- }catch(exception2){
- alert("catch区域有异常发生");
- }
- }finally{
- alert("结束try...catch...finally语句"); //最终程序调用执行的语句
- }
- </script>

错误处理

throw

- 在程序中使用throw语句可以有目的的抛出异常。
- 语法：<script>
- throw new Error('somestatements');
- </script>

错误处理

throw

- 案例3:
- <body>
- <script language="javascript">
- try{
- var num=1/0;
- if(num=="Infinity"){
- throw new Error("除数不能为0");
- }
- }catch(exception){
- alert(exception.message);
- }
- </script>
- </body>

调试技术

使用write（ ）

- 异常是每个程序员都会遇到的问题，调试对任何程序设计来说都是一个关键的技术。
- 有时程序员希望将所有的调试信息以列表的方式显示在页面中，这时可以使用write()方法进行调试。
- 语法：

```
<script>
document.write();
</script>
```

- 例如：<script>
- function alertTest(){
- document.write(“程序开始”);
- var a=10；var b=20；
- document.write(“程序执行”);
- document.write(a+b);
- document.write(“程序结束”);
- }
- </script>

调试技术

使用alert（ ）

- 当程序开发者不能定位程序发生错误引发的异常时，可以采用代码跟踪方式查找错误，这时可以将alert()语句放在程序的不同位置，用它来显示程序中的变量、函数返回值等。
- 语法：
- 例如：<script>
- function alertTest(){
- alert(“程序开始”);
- var a=10；var b=20；
- alert(“程序执行”);
- alert(a+b);
- alert(“程序结束”)；}
- </script>

```
<script>
alert();
</script>
```

调试技术

- 抛出错误也是调试代码的好办法。如果错误消息很具体，基本上就可以把它当作确定错误来源的依据。但这种错误消息必须能够明确给出导致错误的原因。
- 语法：
- try{
- throw（somestatements）;}
- catch(exception){
- alert(exception.message);
- }

调试技术

抛出错误

- 例如：function divide(num1,num2){
- return num1/num2;}
- 以上函数计算两个数的除法，如果一个参数不是数值，它会返回NaN。计算结果也会返回NaN，就会在Web中导致问题。先检测每个参数是否都是数值。
- function divide(num1,num2){
- if(typeof num1 !=”number” || typeof num2 !=”number”){
- throw new Error(“divide():both arguments must be numbers.”);
- }
- return num1/num2;
- }

调试技术

抛出错误

案例4：<script>

```
            var width = 12;                                    // width variable
var height = 'test';                                  // height variable
function calculateArea(width, height) {
  try { var area = width * height;                      // Try to calculate area
    if (!isNaN(area)) {                               // If it is a number
      return area;                                    // Return the area
    } else {                                          // Otherwise throw an error
      throw new Error('calculateArea() received invalid number');  }
  } catch(e) {                                       // If there was an error
    console.log(e.name + ' ' + e.message);            // Show error in console
    return 'We were unable to calculate the area.'; // Show users a message
  }
}
document.getElementById('area').innerHTML = calculateArea(width, height);
        </script>
```

调试技术

- 案例5：将异常提示信息显示在弹出的提示框中
- <script language="javascript">
- try{
- document.forms.input.length;
- }catch(exception){
- alert("错误信息为："+exception.message+"\n错误类型字符串为："+exception.name);
- }finally{
- alert("结束try...catch...finally语句");
- }
- </script>

调试技术

- 案例6：判断参数的个数和除数是否为0

```
<script language="javascript">
function test(num1,num2){
        try{
                if(arguments.length<2){
                        throw new Error("参数个数不够");          //抛出异常
                }
                if(num1/num2=="Infinity"){
                        throw new Error("除数不能为0") //抛出异常
                }

        }catch(exception){
                alert(exception.message);
        }
}
</script>
```

- 无效字符
- 根据语法，JavaScript文件必须只包含特定的字符。在JavaScript文件中存在无效字符时，IE会抛出无效字符（invalid character）错误。firefox会抛出非法字符（illegal character）错误；Safari会报告发生了语法错误，而Opera则会报告发生了ReferenceError（引用错误），因为它会将无效字符解释为未定义的标识符。所谓无效字符，就是JavaScript语法中未定义的字符。

调试技术

- 未找到成员
- 如果对象被销毁后，又给该对象赋值，就会导致未找到成员错误。发生这个错误最常见情形是使用event对象的时候。IE中的event对象是window的属性，该对象在事件发生时创建，在最后一个事件处理程序执行完毕后销毁。如下面的例子：

```
document.onclick=function(){
    var event=window.event;
    setTimeout(function(){
     event.returnValue=false;},1000);    //未找到成员错误
}
```

调试技术

- 未知运行时错误
- 当使用innerHTML或outerHTML以下列方式指定HTML时，就会发生未知运行错误（Unknow runtime error）：一是把块元素插入到行内元素时，二是访问表格任意部分（<table><tbody>等）的任意属性时。例如，从技术角度说，标签不能包含 之类的块级元素，因此下列代码就会出现未知运行时错误：
- span.innerHTML=” Hi ”;
- 在遇到把块级元素插入到不恰当位置的情况时，其他浏览器会尝试纠正并隐藏错误，而IE就会提示错误。

调试技术

- 语法错误
- 只要IE报告发生了语法错误（syntax error），都可以很快找到原因。这时候，原因可能是代码中少了一个分号，或者花括号前后不对应。
- 如果你引用了外部的JavaScript文件，而该文件最终没有返回JavaScript代码，IE也会抛出语法错误。语法错误的位置通常在脚本第一行的第一个字符处。Opera和Safari也会报告语法错误，但它们会给出导致问题的外部文件的信息；IE就不会给出这个信息。Firefox会忽略那些被当作JavaScript内容嵌入到文档中的非JavaScript文件中的解析错误。

调试技术

- 系统无法找到指定资源
- 系统无法找到指定资源：在使用JavaScript请求某个资源URL，而该URL的长度超过了IE对URL最长不能超过2083个字符的限制时，就会发生这个错误。IE不仅限制了JavaScript中使用URL的长度，而且也限制了用户在浏览器自身中使用URL长度（其他浏览器对URL的限制没有这么严格）。例如：

```
function createLongUrl(url){
    var s=”?”;
    for(var i=0,len=2500;i<len;i++){
      s+=”a”;
}
return url+s;
}
var x=new XMLHttpRequest();
x.open(“get”,createLongUrl(“http://www.somedomain.com”),true);
x.send(null);
```

调试技术

常见的错误

- 1、回归基础

JS大小写是敏感的，所以请检查大小写情况。不要在变量名中使用保留字和横线；检查并确保单双引号是成对的；检查并确保变量值中的引号已经转义。检查并确保HTML中的id属性值是唯一的。

- 2、缺少或多余字符
- 每条语句都应该以分号结尾；检查并确保没有缺少右花括号和右括号。检查并确保调用函数时没有缺少参数。
- 3、数据类型错误
- 使用=并不能判断值是否匹配，而是会对变量赋值，应当使用==。一旦switch语句找到匹配的case，就会执行该case后面的所有代码，直到遇到break或return为止。

想一想

- 首先定义两个变量a和b，然后使用try...catch...finally语句处理异常，在程序中调用不存在的变量c，弹出在catch区域中设置的异常提示信息“没有定义变量c”，并且最终弹出finally提示信息“结束try...catch...finally语句”。