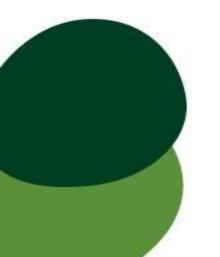


# JavaScript进阶

---Error及异常处理





#### 内容提纲

- **▶** JS异常处理
- > Error对象及其子对象



- ·JS中异常处理的作用(处理程序运行时出现的意异常)
- 异常处理语法(try catch finally)

```
try{
   //try_statements 可能出现错误部分
   console.log("try_statements");
   throw "Some Error";//可以抛出异常
catch(e){ //catch和finally至少有一个
   //catch_statements 捕获处理异常
   console.log("catch_statements",e);
finally{ //catch和finally至少有一个
   //finally statements 最终处理
   console.log("finally_statements");
```

无论是否捕获到异常, finally都会执行



## ·JS中异常处理嵌套的情况

```
try {
     try {
         throw "oops";
     catch (ex) {
         console.error("inner", ex);
     finally {
         console.log("finally");
 catch (ex) {
     console.error("outer", ex);
```

```
throw "oops";
    catch (ex) {
        console.warn("inner", ex);
        throw ex;
    finally {
        console.log("finally");
catch (ex) {
    console.warn("outer", ex);
```

## ·JS中的异步回调函数捕获异常的问题(思考下述代码)

```
try{
try{
    function abc(x,cb){
                                      function abc(x,cb){
        console.log(x);
                                          console.log(x);
                                          cb();
        cb();
    abc("xx",function(){
                                  catch(e){
        var arr = new Array(-1);
                                                         此处是否能
    });
                                      console.log(e);
                                                         捕获到异常?
                                  abc("xx",function(){
catch(e){
                    此处是否能
    console.log(e);
                                      var arr = new Array(-1);
                    捕获到异常?
                                  });
```

## \_ -

## •JS中异常处理的案例

```
window.onload = function () {
   window.Foo = function () {
       var inputValue = document.getElementById("inputID").value;
       try{
           var n = parseInt(inputValue);
           var a= new Array(n);//定义一个数组 传3试试、再传-5试试
           for(var i=0;i<n;i++){a[i] = i;}
        }catch(e){
           alert(e.name+e.message);
        finally {
           document.getElementById("labelID").innerHTML = a;
```

#### 内容提纲

- ➤ JS异常处理
- > Error对象及其子对象



## JS中的错误以及Error对象

## •JS中的错误概述

- 当 JavaScript 引擎执行 JavaScript 代码时,可能会发生各种错误
- 可能是语法错误、或是由于浏览器差异产生的错误、或是来自服务器或用户导致的错误
- 有些错误是可以控制和避免的,有些是不可控的(比如来自用户输入等第三方的操作)

## •JS中对错误的处理

- 优化代码避免可控错误,对不可控错误需要使用异常处理来进行处理,避免程序直接崩溃

## •Error对象

- 当运行时错误产生时,会抛出一个错误对象,可以对此对象进行捕获和处理
- 也可以通过Error的构造器new一个错误对象,当检测到异常时或不满足逻辑时,手动抛出错误对象
- 所有错误对象的基础原型是Error.prototype,默认的name属性为 "Error", message属性为 ""







### Error对象

## • Error的子类

- ReferenceError 引用错误、RangeError 范围错误、TypeError 类型错误
- URIError 资源定位错误、EvalError 与eval()有关的错误、其他错误

```
//Part 33333333333 类型错误 TypeError
try{
    var a;a.aa();
    //var a= new 123; //在chrome中测试
}catch(e){
    console.log(e.name,e.message);
}
finally {
    console.log("finally");//有无异常该句都会执行
}
```



