try-catch 能否监听多线程中的错误?

对于 try-catch 这组异常处理函数，同学们可用接触的并不多，也不是很了解，今天咱们就来说说它，但是这还没完，顺便看看 try-catch 对于多线程的处理机制是什么样的。一说到多线程，同学们也会蒙，不都说 JavaScript 是单线程的，这个多线程是怎么回事？别着急，接下来我们一起来探究。

try-catch 函数

基本使用

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，通常是程序员造成的编码错误或错别字。可能是拼写错误或语言中缺少的功能（可能由于浏览器差异）。可能是由于来自服务器或用户的错误输出而导致的错误。当然，也可能是由于许多其他不可预知的因素。当错误发生时，当事情出问题时，JavaScript 引擎通常会停止，并生成一个错误消息。这种情况被称为：JavaScript 抛出异常。对于可能出现的问题的代码我们需要预判，这就涉及到异常的测试与捕获。

基本语法：

```
try {
    //在这里运行代码
}
catch (err) {
    //在这里处理错误
}
```

try 语句允许我们定义在执行时进行错误测试的代码块。

catch 语句允许我们定义当 try 代码块发生错误时，所执行的代码块。

JavaScript 语句 try 和 catch 是成对出现的。

有一个运行代码的函数 runCode

```
function runCode(code) {
    return eval(code);
}
```

当我们执行 `runCode('1+1')` 可以得到 2，这里利用 `eval` 函数特性，可以执行字符串。`eval` 并不是我们这篇文章的主要内容。对于 `code` 是用户输入，当用户输入的内容是一段错误代码，我们的函数，就会报错，进而影响之后的逻辑执行。这是我们需要利用 `try-catch` 进行捕获。代码改写如下：

```
function runCode(code) {
    try {
        // 接下来执行的 eval(code)可能会出错
        return eval(code);
    } catch (error) {
        // 上面的部分报错，会对外输出错误信息，并不会终止程序
        console.log('代码存在错误')
    }
}
```

当我执行 runCode('aaaaa'), 会给我们提示信息： 代码存在错误，这样我们的错误就成功捕获，并不影响后面的逻辑执行。

使用 try-catch 的建议

try catch 的使用，永远应该放在你的控制范围之内，而不应该防范未知的错误。也就是说你很清楚知道这里是有可能"出错"的，而且你很清楚知道什么前提下会出错，你就是要故意利用报错信息来区分错误，后续的程序会解决所有的出错，让程序继续执行。如果让用户先发现你根本没预料到的错误，而不是你先发现错误，你是失职的。

try-catch 与多线程

JavaScript 中的"多线程"

一说到 JavaScript 是多线程，同学们是不认可的，别急。听我慢慢道来。

浏览器中的 JavaScript 确实是以单线程的方式执行的，也就是说 JavaScript 执行使用一个主线程，但是 JavaScript 提供了异步操作，比如定时器(setTimeout、setInterval)事件、Ajax 请求、Promise, I/O 等。它们将会被放入浏览器的事件任务队列（event loop）中去，等到 JavaScript 运行时执行线程空闲时候，事件队列才会按照先进先出的原则被一一执行。但是对于以上的异步操作过程中，能进行的计时，发送请求，I/O 操作都是其他的线程在做的事情，所以说是"多线程"。

这里也要说一下，**多线程不等于异步**，异步和多线程并不是一个同等关系,异步是最终目的,多线程只是我们实现异步的一种手段。异步是当一个调用请求发送给被调用者,而调用者不用等待其结果的返回而可以做其它的事情。实现异步可以采用多线程技术或则交给另外的进程来处理。

那么对于这些所谓的异步操作，try-catch 能否监听到错误，我们来一探究竟。

1. setTimeout 函数

代码如下：

```
try {
    setTimeout(function () {
        console.log(a.b)
    }, 1000)
} catch (error) {
    console.log('有错误')
}
```

setTimeout 中的 a.b 会报错 a is not defined，并未被捕获。得出结论 try-catch 无法监听 setTimeout 函数中的错误。

2. Promise 函数

代码如下：

```
try {
  let p = new Promise((resolve, reject) => {
      a.b
  })
} catch (error) {
    console.log('有错误')
}
```

Promise 中的 a.b 会报错 a is not defined，并未被捕获。得出结论 try-catch 无法监听 Promise 函数中的错误。

3. Ajax 函数

```
function ajax(url) {
    var xhr = window.XMLHttpRequest ? new XMLHttpRequest() :
ActiveXObject("microsoft.XMLHttp")
    xhr.open("get", url, true);
    xhr.send();
    xhr.onreadysattechange = () => {
        if (xhr.readystate == 4) {
            if (xhr.status == 200) {
                var data = xhr.responseTEXT;
```

```
                return data;
            }
        }
    }
}

try {
   ajax('someurl')
} catch (error) {
    console.log('有错误')
}
```

Ajax 中的请求出错，并未被捕获。得出结论 try-catch 无法监听 Ajax 函数中的错误。

总结

对于以上几种异步操作，我们看的出来 try-catch 并未帮我们监听的到里面的错误。原因是：**javaScript 引擎对异步方法进行 try/catch 操作只能捕获当次事件循环内的异常，对 call back 执行时抛出的异常将无能为力。**

但是对于异步操作，只要是代码逻辑没有问题，我们在适当的问题出口把问题暴露出去就可以了，比如 Promise 的 then，Ajax 的状态判断等等。