查找并解决 JavsScript 代码的错误

# 目标

|  |  |
| --- | --- |
| **预备知识：** | 计算机基础知识，初步理解HTML和CSS，了解JavaScript。 |
| **目标：** | 获得独立修复简单问题的能力和信心。 |

## 错误类型

* **语法错误**：代码中存在拼写错误，将导致程序完全或部分不能运行，通常会收到一些出错信息。只要熟悉语言并了解出错信息的含义，就能够顺利修复它们。
* **逻辑错误**：有些代码语法虽正确，但执行结果和预期相悖，这里便存在着逻辑错误。这意味着程序虽能运行，但会给出错误的结果。由于一般不会收到来自这些错误的提示，它们通常比语法错误更难修复。

## 一个出错的示例

重回猜数字游戏，故意引入一些错误。首先查看开发者控制台，看是否存在语法错误，然后尝试修复。详见下文。

# 修复语法错误

语法错误：第一轮

打开 number-game-errors.html 所在的标签页，然后打开 JavaScript 控制台。将看到以下出错信息：

|  |
| --- |
|  |

（1）红色1表示页面有1个错误。

（2）一条出错信息，表示问题出在哪儿：“TypeError：**guessSubmit**.addeventListener is not a function”（类型错误：**guessSubmit**.addeventListener 不是函数）

（3）提示出错的行，以及导致错误的首个字符号。这里错误来自 44 行，点击链接可到达源，观察错误行

|  |
| --- |
|  |

（4）出错信息显示“guessSubmit.addeventListener 不是一个函数”，说明这里可能存在拼写错误。如果你不确定某语法的拼写是否正确，可以到 MDN 、菜鸟网、w3cchool上去查找

（5）因此这里错误显然是我们把函数名写错造成的。请记住，JavaScript 区分大小写，所以任何轻微的不同或大小写问题都会导致出错。将 addeventListener 改为 addEventListener 便可解决。

语法错误：第二轮

保存页面并刷新，可以看到出错信息不见了。现在，果尝试输入一个数字并按确定按钮，你会看到...另一个错误！

|  |
| --- |
|  |
|  |

1. 此次出错信息为“TypeError：lowOrHi is null”（“类型错误：lowOrHi为null”），在第 36 行。

**注**：[Null](https://developer.mozilla.org/en-US/docs/Glossary/Null)是一个特殊值，意思是“什么也没有”，或者“没有值”。这表示 lowOrHi 已声明并初始化，但没有任何有意义的值，可以说：它没有类型没有值。

**注**：这条错误没有在页面加载时立即发生，是因为它发生在函数内部（checkGuess() { ... }块中）。函数内部的代码运行于一个外部代码相互独立的域内.只有当代码调用 checkGuess() 函数运行至36行并，代码才会抛出出错信息。

1. 请观察第 36 行代码：

lowOrHi.textContent = '你猜高了！';

1. 该行试图将 lowOrHi 变量中的 textContent 属性设置为一个字符串，但是失败了，这是因为 lowOrHi 并不包含预期的内容。为了一探究竟，你可以在代码中查找一下该变量的的其他实例(双击该变量，高亮显示)。lowOrHi 最早出现于第 6 行：

var lowOrHi = document.querySelector('lowOrHi');

1. 此处，让该变量包含一个指向文档 HTML 中特定元素的引用。我们来检查一下在该行代码执行后变量的值是否为 null。在第 7行添加以下代码：

console.log(lowOrHi);

**注**：[console.log()](https://developer.mozilla.org/en-US/docs/Web/API/Console/log)是一个非常实用的调试功能，它可以把值打印到控制台。因此我们将其置于代码第7行时，它会将 lowOrHi 的值打印至控制台。

1. 保存并刷新，你将在控制台看到 console.log() 的执行结果：显然，此处 lowOrHi 的值为 null，所以第 6 行肯定有问题。
2. 我们来思考问题有哪些可能。第6行使用 [document.querySelector()](https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector) 方法和一个 CSS 选择器来取得一个元素的引用。进一步查看我们的文件，我们可以找到有问题的段落：

<p class="lowOrHi"></p>

1. 这里我们需要一个类选择器，它应以一个点开头（.），但被传递到第 6行的querySelector()方法中的选择器没有点。这可能是问题所在！尝试将第 6 行中的lowOrHi 改成 .lowOrHi。
2. 再次保存并刷新，此时 console.log() 语句应该返回我们想要的 <p> 元素。终于把错误搞定了！此时你可以把 console.log()  一行删除，或保留它以便随后参考。选择权在你。

**注**：此错误的更多详细信息请参阅：<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Unexpected_type>

语法错误：第三轮

1. 现在，如果你再次试玩，你离成功更进了一步。游戏过程按部就班，直到猜测正确或机会用完，游戏结束。
2. 此时如果点击“开始新游戏”，游戏将再次出错，抛出与开始时同样的错误——“TypeError：resetButton.addeventListener is not a function”！这次它来自第 52 行。
3. 查看第 52 行，很容易看到我们犯了同样的错误。我们只需要再次将 addeventListener改为 .addEventListener。现在就改吧。

# 逻辑错误

此时，游戏应该可以顺利进行了。但经过几次试玩后你一定会注意到要猜的**随机数不是 0 就是 1**。游戏的逻辑肯定是哪里出现了问题，因为游戏并没有返回错误，只是不能正确运行。

1. 寻找 randomNumber 变量和首次设定随机数的代码。保存着游戏开始时玩家要猜的随机数的实例大约在 2 行：

var randomNumber = Math.floor(Math.random()) + 1;

重新开始游戏产生随机数的设定语句大约在 71行：

randomNumber = Math.floor(Math.random()) + 1;

1. 为了检查问题是否来自这两行，在两行代码之后各插入下面的代码：

console.log(randomNumber);

1. 保存并刷新，然后试玩，你会看到在控制台显示的随机数总是等于1。

修正逻辑错误

为了解决这个问题，让我们来思考这行代码如何工作。首先，我们调用 [Math.random()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random),它生成一个在0和1之间的十进制随机数，例如 0.5675493843。

Math.random()

接下来，我们把调用 Math.random() 的结果作为参数传递给 [Math.floor()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/floor)，它会舍弃小数部分返回与之最接近的整数。然后我们给这个结果加上1：

Math.floor(Math.random()) + 1

由于将一个 0 和 1 之间的随机小数的小数部分舍弃，返回的整数一定为 0，因此在此基础上加 1 之后返回值一定为 1。要在舍弃小数部分之前将它乘以100。便可得到 0 到 99 之间的随机数：

Math.floor(Math.random() \* 100);

然后再加 1，便可得到一个 100 以内随机的自然数:

Math.floor(Math.random() \* 100) + 1;

将上述两行内容替换为此，然后保存刷新，游戏终于如期运行了！

# 其它常见错误

代码中还会遇到其他常见错误。

SyntaxError: missing ; before statement  
（语法错误：语句缺少分号）

这个错误通常意味着你漏写了一行代码最后的分号，但是此类错误有时候会更加隐蔽。

例如如果我们把 checkkGuess() 函数中的这一行 :

var userGuess = Number(guessField.value);

改成

var userGuess === Number(guessField.value);

将抛出一个错误。因为系统认为你在其他事情。请不要把赋值运算符（=，为一个变量赋值）和严格等于运算符（===，比较两个值是否相等，返回 true/false）弄混淆。

不管输入什么程序总说“你猜对了！”

这是混淆赋值和严格等于运算符的又一症状。例如我们把 checkGuess() 里的：

if (userGuess === randomNumber) {

改成

if (userGuess = randomNumber) {

因为条件永远返回 true，使得程序报告你猜对了。

SyntaxError: missing ) after argument list  
（语法错误：参数表末尾缺少括号）

这个很简单。通常意味着函数/方法调用后的结束括号忘写了。

**注：**有关此错误的更多详细信息请参考： [SyntaxError: missing ) after argument list](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Missing_parenthesis_after_argument_list)。

SyntaxError: missing : after property id  
（语法错误：属性ID后缺少冒号）

JavaScript 对象的形式有错时通常会导致此类错误，如果把

function checkGuess() {

写成了

function checkGuess( {

浏览器会认为我们试图将函数的内容当作参数传回函数。写圆括号时要小心！

SystaxError: missing } after function body  
（语法错误：函数体末尾缺少花括号）

这个简单。通常意味着函数或条件结构中丢失了一个花括号。如果我们将 checkGuess() 函数末尾的花括号删除，就会得到这个错误。

SyntaxError: expected expression, got '*string*'（语法错误：得到一个 '*string*' 而非表达式）  
或者  
SyntaxError: unterminated string literal（语法错误：字符串字面量未正常结束）

这个错误通常意味着字符串两端的引号漏写了一个。如果你漏写了字符串开始的引号，将得到第一条出错信息，这里的 '*string'*将被替换为浏览器发现的意外字符。如果漏写了末尾的引号将得到第二条。

对于所有的这些错误，想想我们在实例中是如何逐步解决的。错误出现时，转到错误所在的行观察是否能发现问题所在。记住，错误不一定在那一行，甚至可能是由某个不相关的问题造成的！

**注：** 有关这些错误的更多详细信息请参考：[SyntaxError: Unexpected token](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Unexpected_token) 以及 [SyntaxError: unterminated string literal](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Unterminated_string_literal)。