错误

Ref Javascript

Reading error messages

当 JavaScript 无法继续执行时，程序会停止并显示错误提示。提示的第一个词表示错误类型，其余部分说明出错原因及位置。阅读该信息后，请定位到所指出的代码行进行检查。

```
SyntaxError: missing ) after argument list
ReferenceError: totl is not defined
TypeError: Cannot read properties of undefined (reading 'length')
```

错误类型	含义	常见原因
`SyntaxError`	代码书写有误，导致程序无法运行	缺少括号或引号
`ReferenceError`	引用了未定义的标识符	拼写错误：例如将 `totl` 误写为 `total`
`TypeError`	值的类型与预期操作不符	在 `.length` 上使用了 `undefined` 操作

不同浏览器的具体报错措辞可能略有差异，但错误名称保持一致。以下程序捕获每个异常，以便打印其名称：

```
try {
  const total = 5;
  console.log(totl);
} catch (e) {
  console.log(e.name);           // ReferenceError
}

try {
  const pupil = undefined;
  console.log(pupil.length);
} catch (e) {
  console.log(e.name);           // TypeError
}
```

某些错误不会触发报错，仅产生错误结果，这称为逻辑错误。排查此类问题可在流程中插入 `console.log` 打印中间值，并人工核对。

try, catch & throw

可能发生异常的代码应置于 try 块中。若其中发生错误，JavaScript 将跳转至 catch 而非终止程序，且 e 变量将保存该错误对象：

```
try {
  JSON.parse("{not valid}");
  console.log("parsed");         // skipped
} catch (e) {
  console.log("Could not read the data");
}
console.log("The program carries on");
```

可通过 throw 主动抛出错误。这使得函数能够明确拒绝无效输入，而非返回错误结果：

```
function average(list) {
  if (list.length === 0) {
    throw new Error("the list is empty");
  }
  return list.reduce((s, x) => s + x, 0) / list.length;
}

console.log(average([4, 6]));    // 5
try {
  average([]);
} catch (e) {
  console.log(`Error: ${e.message}`);   // Error: the list is empty
}
```

- new Error("…") 会创建一个异常，其 message 属性值为自定义文本。
- 仅在代码可能真正出错（如读取数据）时使用 try。不要用它来掩盖自己代码中的错误。

常见错误

- 跳过该消息。其 **第**一行指出了错误类型及具体原因；请从此处开始排查。
- 不执行任何操作的 catch（即 catch (e) {}）会掩盖问题：至少应打印 e.message。
- throw "text" 可以工作，但 throw new Error("text") 还会记录错误发生的位置。