#### ES2019提案预览

##### 之空值处理

摘要

TypeScript语言特性——空位合并（Nullish Coalescing）

@link <https://github.com/tc39/proposal-nullish-coalescing>

TypeScript语言特性——可选链（Optional Chaining）

@link <https://github.com/tc39/proposal-optional-chaining>

正文

什么是空值？为什么要处理空值？

在TypeScript中，空值包括null、undefined、空字符串、空数组等。在项目中，为了兼容特性不完备的后端语言、框架，需要在前端做空值处理。有时为了确保代码的容错能力，不得不写一些垃圾代码，比如：

if (response == null) {

throw ...

return;

}

if (response.data == null) {

throw ...

return;

}

let pShowDetails = response.data.showDetails;

什么是可选链？

当寻找树状结构深处的属性值时，通常需要检查中间节点是否存在，而且许多API返回对象或null / undefined，并且可能只想从结果中提取属性 当它不为null时。可选的链接运算符允许开发人员处理许多情况，而无需重复自己和/或在临时变量中分配中间结果。

上例中，可以用short-circuit evaluation简化代码为：

let pShowDetails = response && response.data && response.data.showDetails;

而用可选链可以将代码进一步简化：

let pShowDetails = response?.data?.showDetails;

什么是空位合并？

执行属性访问时，通常需要提供默认值，如果该属性访问的结果为null或未定义。 当前，在JavaScript中表达此意图的典型方法是使用||。 这在空值和未定义值的常见情况下效果很好，但是存在一些虚假值可能会产生令人惊讶的结果。空值合并运算符旨在更好地处理这些情况，并用作对空值的相等检查（ null或未定义）。

有什么体会？

编程语言一大抄，抄来抄去有提高。

□