

# 可推导的this类型

## ■ this是JavaScript中一个比较难以理解和把握的知识点：

□ 我在公众号也有一篇文章专门讲解this: [https://mp.weixin.qq.com/s/hYm0JgBI25grNG\\_2sCRITA](https://mp.weixin.qq.com/s/hYm0JgBI25grNG_2sCRITA);

## ■ 当然在目前的Vue3和React开发中你不一定会使用到this：

□ Vue3的Composition API中很少见到this，React的Hooks开发中也很少见到this了；

## ■ 但是我们还是简单掌握一些TypeScript中的this，TypeScript是如何处理this呢？我们先来看两个例子：

```
const obj = {  
  name: "obj",  
  foo: function() {  
    console.log(this.name)  
  }  
}  
  
obj.foo()
```

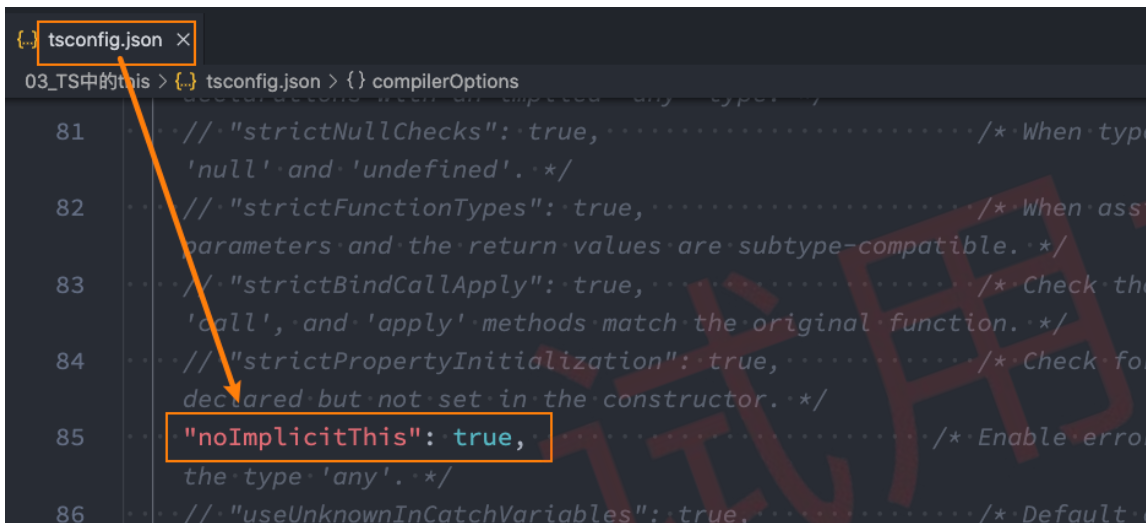
```
function foo1() {  
  console.log(this)  
}  
  
foo1()
```

## ■ 上面的代码默认情况下是可以正常运行的，也就是TypeScript在编译时，认为我们的this是可以正确去使用的：

□ 这是因为在没有指定this的情况，this默认情况下是any类型的；

# this的编译选项

- VSCode在检测我们的TypeScript代码时，默认情况下运行不确定的this按照any类型去使用。
  - 但是我们可以创建一个tsconfig.json文件，并且在其中告知VSCode this必须明确执行（不能是隐式的）；



```
{
  "compilerOptions": {
    "strictNullChecks": true,
    "strictFunctionTypes": true,
    "strictBindCallApply": true,
    "strictPropertyInitialization": true,
    "noImplicitThis": true,
    "useUnknownInCatchVariables": true
  }
}
```

- 在设置了`noImplicitThis`为true时，TypeScript会根据上下文推导this，但是在不能正确推导时，就会报错，需要我们明确的指定this。

```
function foo1() {
  console.log(this)
}

foo1()
```

any

'this' implicitly has type 'any' because it does not have a type annotation. ts(2683)

# 指定this的类型

■ 在开启noImplicitThis的情况下，我们必须指定this的类型。

■ 如何指定呢？函数的第一个参数类型：

- 函数的第一个参数我们可以根据该函数之后被调用的情况，用于声明this的类型（名词必须叫this）；
- 在后续调用函数传入参数时，从第二个参数开始传递的，this参数会在编译后被抹除；

```
function foo1(this: { name: string }) {  
  console.log(this)  
}  
  
foo1.call({ name: "why" })
```

# this相关的内置工具

■ Typescript 提供了一些工具类型来辅助进行常见的类型转换，这些类型全局可用。

■ ThisParameterType:

- 用于提取一个函数类型Type的this (opens new window)参数类型;
- 如果这个函数类型没有this参数返回unknown;

```
function foo(this: { name: string }) {  
  console.log(this.name)  
}  
  
// 获取一个函数中this的类型  
type ThisType = ThisParameterType<typeof foo>
```

■ OmitThisParameter:

- 用于移除一个函数类型Type的this参数类型, 并且返回当前的函数类型

```
// 用于移除一个函数类型Type的this参数类型, 并且返回当前的函数类型  
type FnType = OmitThisParameter<typeof foo>
```

# this相关的内置工具 - ThisType

- 这个类型不返回一个转换过的类型，它被用作标记一个上下文的this类型。（官方文档）

- 事实上官方文档的不管是解释，还是案例都没有说明出来ThisType类型的作用；

- 我这里用另外一个例子来给大家进行说明：

```
interface IState {  
  name: string  
  age: number  
}  
  
interface IData {  
  state: IState  
  running: () => void  
  eating: () => void  
}
```

```
const info: IData & ThisType<IState> = {  
  state: { name: "why", age: 18 },  
  running: function() {  
    console.log(this.name)  
  },  
  eating: function() {  
  }  
}  
  
info.running.call(info.state)
```