**Scope and this keyword**

Ключевое слово **this** позволяет получить контекст текущего объекта.

**Контекст** - это тело родительского объекта.

У браузера глобальный контекст - объект **window**, у NodeJS - объект **globals**.  
  
Плохой практикой является объявление переменных в глобальном контексте, например, **window.foo = 1**. Такие переменные перегружают глобальный контекст, чего стоит избегать.  
  
Рассмотрим this на примере:

const test = {

prop: 42,

func() {

return this.prop

},

funcArrow: () => {

return this.prop

}

}

console.log(test.func()) //в консоли отобразится 42

console.log(test.funcArrow()) //в консоли отобразится undefined

В данном случае, this будет указывать на родительский объект, который является контекстом вписанных в него функций.

**Важно!**  
Как вы заметили, при вызове функции **funcArrow()** мы получим undefined. Это происходит потому, что у стрелочной функции **нет** своего контекста и она всегда смотрит на контекст выше.  
  
Самое интересное то, что для функции можно менять контекст. Для этого используется функция **bind()**

function foo() {

console.log(this)

}

foo() //в консоли отобразит объект Window

let foo2 = foo.bind({ context: 'newContext' })

foo2() //в консоли отобразит объект { context: 'newContext' }

Как мы видим, у foo2() изменился контекст.