this

this 有很多应用场景：

- 函数中
- call、apply、bind
- class
- 对象中
- 箭头函数中

> - 不论如何，this 取什么值，是在执行的时候决定的，不是在定义的时候。
> - this始终返回一个对象.
> - this就是属性或方法“当前”所在的对象。

```
function fn1() {
    console.log(this);
}
fn1();  // window

fn1.call({x: 100}); // {x: 100} , call 一调用就会执行，fn1 函数的 this 就指向 call 方法第一个参数
fn1.apply({x: 100}); // {x: 100} apply 一调用就会执行，fn1 函数的 this 就指向 apply 方法第一个参数
const fn2 = fn1.bind({x: 200});  // bind 会返回一个新的函数，这个新函数的 this 就指向 bind 第一个参数
fn2();  // {x: 200}
```

```
// 对象中
const zhangsan = {
    name: "zhangsan",
    sayHi() {
        console.log(this);
    },
    wait() {
        setTimeout(function() {
            console.log(this);
        })
    }
};

zhangsan.sayHi();  // 当前 zhangsan 对象
zhangsan.wait(); // window，setTimeout 里面的函数执行，是 setTimeout 本身触发的执行，并非是当前对象触发
```

```
// 箭头函数
const lisi = {
    name: "李四",
    sayHi() {
        // 当前对象
        console.log(this);
    },
    wait() {
        setTimeout(() => {
            // this 即当前对象，箭头函数中 this 指向上级作用域的 this，和 sayHi 中
this 一样
            console.log(this);
        })
    }
};

lisi.sayHi();  // lisi 这个对象
lisi.wait();  // lisi 这个对象
```

```
// class 中
class People {
    constructor(name) {
        this.name = name;
        this.age = 20;
    }
    sayHi() {
        console.log(this);
    }
}
const zhangsan = new People("张三");
zhangsan.sayHi();  // 张三这个对象 {name: "张三", age: 20}
```