两句话理解js中的this

2017 年 11 月 17 日

this

前言: 一直都搞不清javascript中this的指向,你不知道的javascript(上卷)这本书中有3章都是在讲解this,去年第一次看完还是觉得似懂非懂的,一深入的问还是不清楚,现在在看一遍,真心觉得这本书里讲的是真好,想深入了解一下的,这本书是一个不错的选择.

下面我就简单的说一下我的理解,用两句话记住了javascrpt中this的指向:

this的指向

1, 普通函数指向函数的调用者:有个简便的方法就是看函数前面有没有点,如果有点,那么就指向点前面的那个值;

2, 箭头函数指向函数所在的所用域: 注意理解作用域,只有函数的{}构成作用域,对象的{}以及 if(){}都不构成作用域;

```
const obj = {
    name: 'objName',
    say() {
        console.log(this.name);
    },
    read: () => {
        console.log(this.name);
```

```
    }
}
obj.say(); // objName
obj.read(); // undefined
```

- 普通函数,调用者是obj,所以结果是 objname;也是理解say()是普通函数,前面有点,所以this指向obj;
- 箭头函数,this指向函数所在的作用域,当前的作用域为全局环境,所以this.name为undefined,
- 举下面的例子更清楚的了解一下箭头函数this的指向,箭头函数所在的作用域是普通函数say,say()的调用者是obj

```
const obj = {
  say: function () {
      setTimeout(() => {
          console.log(this)
      });
  }
}
obj.say(); // obj,此时this指的是定义他的obj
```

补充知识点

- 浏览器默认的this为window

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

- node.js中全局环境默认this为{},普通函数中默认this为global

```
console.log(this); // {}
function test() {
  console.log(this);
}
test(); //global
```

来两道题检查你是否掌握了

example1

```
var length = 10;
function fn() {
 console.log(this.length);
}

const  obj = {
    length: 5,
    method: function(fn) {
    fn();
    arguments[0]();
    }
};

obj.method(fn, 1);
```

输出 10, 2

刚开始看到这道题我也是蒙蒙的,现在也终于理解了,
method这个函数传入了两个参数,一个参数为`fn()`,`fn()`为普通函数,this指向函数的调用者,此时指向全局(也可以看这个函数前面没有点),所以运行结果为10,arguments是函数的所有参数,是一个类数组的对象,arguments[0] (),可以看成是arguments.0(),调用这个函数的是`arguments`,此时this就是指`arguments`,this.length就是`angument.length`,就是传入的参数的总个数2

注:

- 上面例子在node环境中的运行结果为 undefined 2, `var length = 10`改成`global.length = 10;`是因为node环境下定义在全局的变量不会绑定到global,浏览器也会自动绑定到全局环境window
- 之前写的有点小问题,感谢大家的指出:用const来定义length,然后打印0和2; var在全局定义的变量会自动定义到window,但是const和let在全局定义的变量不会自动定义到window,改成const打印window.length为0,是因为window.length表示当前页面有几个iframe,可以看一下MDN上关于window.length的定义

改成下面这样结果又是什么呢?

```
var length = 10;

function fn() {
```

```
    console.log(this.length);
}

const obj = {
    length: 5,
    method: function(fn) {
        fn();
        const fun = arguments[0];
        fun();
    }
};

obj.method(fn, 1);
```

10, 10

example 2

```
window.val = 1;
var obj = {
    val: 2,
    dbl: function() {
        this.val *= 2;
        val *= 2;
        console.log(val);
        console.log(this.val);
    }
}
obj.dbl(); // 2 4
var func = obj.dbl;
func(); // 8 8
```

这个就是有点绕了,不过一步步来分析就很容易理解了:

obj.dbl();执行这行代码时，this指的是obj，所以this.val === obj.val*=2,最后结果为4,val*=2 === window.val *= 2，最后结果是2

func()，执行这行代码时，func()没有任何前缀，this指的是window.func();所以此时this值得是window，this.val === window.val *= 2,此时window.val 为4，val*=2 === window.val *2,最后结果为8，最后console.log(this.val),与console.log(val),指的都是window.val，最后结果都是8

上述是我自己的理解,如果有什么问题,欢迎指正~

About

github

blog