# 第九章 代理与反射

代理是目标对象的抽象

代理是使用 Proxy 构造函数创建的。这个构造函数接收两个参数:目标对象和处理程序对象。缺少其中任何一个参数都会抛出 TypeError。

// 创建空代理  
  
const handler = {};  
const obj = {name:'laomao',list:[1,2,3]};  
const proxy = new Proxy(obj,handler);  
console.log(proxy)  
  
// 因为操作的目标都是目标对象obj  
console.log(proxy.name); //'laomao'  
console.log(proxy.list); //[1,2,3]  
console.log(proxy.list === obj.list); //true  
console.log(proxy.hasOwnProperty('name')); //true  
  
  
console.log(proxy instanceof Object); //true  
  
// Proxy的prototype为undefined  
console.log(Proxy.prototype) //undefined  
console.log(proxy instanceof Proxy)  
//Uncaught TypeError: Function has non-object prototype 'undefined' in instanceof check  
  
  
console.log(proxy == obj); // false  
console.log(proxy === obj); // false

## 捕获器

const target = {  
 foo: 'bar'  
 };  
  
const handler = {  
 // 捕获器在处理程序对象中以方法名为键 get() {  
 return 'handler override';  
 }  
};  
const proxy = new Proxy(target, handler);  
console.log(target.foo);  
console.log(proxy.foo);  
console.log(target['foo']);  
console.log(proxy['foo']);  
// bar  
// handler override  
// bar  
// handler override  
console.log(Object.create(target)['foo']); // bar console.log(Object.create(proxy)['foo']); // handler override

## 捕获器不变式

如果目标对象有一个不可配置且不可写的数据属性，那么在捕获器返回一个与该属性不同的值时，会抛出 TypeError

const target = {};  
  
Object.defineProperty(target, 'foo', {  
 configurable: false,  
 writable: false,   
 value: 'bar'  
});  
 const handler = {  
 get() {  
 return 'qux';  
 }  
 };  
 const proxy = new Proxy(target, handler);  
 console.log(proxy.foo);  
 // TypeError

revocable()方法，这个方法支持撤销代理对象与目标对象的关联。

撤销代理的操作是不可逆的。而且，撤销函数(revoke())是幂等的，调用多少次的结果都一样。撤销代理之后 再调用代理会抛出 TypeError。

const target = {  
 foo: 'bar'  
};  
const handler = {  
 get() {  
 return 'intercepted';  
 }  
};  
//撤销函数和代理对象是在实例化时同时生成的  
const { proxy, revoke } = Proxy.revocable(target, handler);  
console.log(proxy.foo); // intercepted  
console.log(target.foo); // bar  
revoke();  
console.log(proxy.foo); // TypeError

## 代理可以捕获 13 种不同的基本操作。

这些操作有各自不同的反射 API 方法、参数、关联 ECMAScript 操作和不变式。

1、get()

2、set()

3、has()

4、defineProperty()

5、getOwnPropertyDescriptor()

6、deleteProperty()

7、ownKeys()

8、getPrototypeOf()

9、setPrototypeOf()

10、isExtensible()

11、preventExtensions()

12、apply()

13、construct()

##### 利用代理实现数据绑定与可观察对象

const userList = [];  
class User {  
 constructor(name) {  
 this.name\_ = name;  
 }  
 }  
 const proxy = new Proxy(User, {  
 construct() {  
 const newUser = Reflect.construct(...arguments);  
 userList.push(newUser); 12 return newUser;  
 }   
});  
new proxy('John');  
new proxy('Jacob');  
new proxy('Jingleheimerschmidt');  
console.log(userList); // [User {}, User {}, User{}]