Proxy 与 Reflect

2020年1月9日

14:44

**参考内容 : [http://es6.ruanyifeng.com/?search=Proxy&x=0&y=0#docs/proxy](http://es6.ruanyifeng.com/?search=Proxy&x=0&y=0" \l "docs/proxy)**

1. **什么是proxy**

Proxy即是对某些操作的行为进行修改,可以理解为在目标对象之前架设一层拦截

1. **Proxy的语法**

var proxy = new Proxy(target, handler);

proxy即对目标对象的代理,其中,handler定义了代理的内容

**例子:**

var obj = new Proxy({}, {

get: function (target, propKey, receiver) {

//target为目标对象,propKey为作用的属性名,receiver为当前proxy对象

console.log(`getting ${propKey}!`);

return Reflect.get(target, propKey, receiver);

},

//value为赋予的属性,当前属性值可以通过target[propKey]获取

set: function (target, propKey, value, receiver) {

console.log(`setting ${propKey}!`);

return Reflect.set(target, propKey, value, receiver);

}

});

obj.count = 1

// setting count!

++obj.count

// getting count!

// setting count!

// 2

1. 其他的拦截操作 :

get(target, propKey, receiver)：拦截对象属性的读取，比如proxy.foo和proxy['foo']。

set(target, propKey, value, receiver)：拦截对象属性的设置，比如proxy.foo = v或proxy['foo'] = v，返回一个布尔值。

has(target, propKey)：拦截propKey in proxy的操作，返回一个布尔值。

deleteProperty(target, propKey)：拦截delete proxy[propKey]的操作，返回一个布尔值。

ownKeys(target)：拦截Object.getOwnPropertyNames(proxy)、Object.getOwnPropertySymbols(proxy)、Object.keys(proxy)、for...in循环，返回一个数组。该方法返回目标对象所有自身的属性的属性名，而Object.keys()的返回结果仅包括目标对象自身的可遍历属性。

getOwnPropertyDescriptor(target, propKey)：拦截Object.getOwnPropertyDescriptor(proxy, propKey)，返回属性的描述对象。

defineProperty(target, propKey, propDesc)：拦截Object.defineProperty(proxy, propKey, propDesc）、Object.defineProperties(proxy, propDescs)，返回一个布尔值。

preventExtensions(target)：拦截Object.preventExtensions(proxy)，返回一个布尔值。

getPrototypeOf(target)：拦截Object.getPrototypeOf(proxy)，返回一个对象。

isExtensible(target)：拦截Object.isExtensible(proxy)，返回一个布尔值。

setPrototypeOf(target, proto)：拦截Object.setPrototypeOf(proxy, proto)，返回一个布尔值。如果目标对象是函数，那么还有两种额外操作可以拦截。

apply(target, object, args)：拦截 Proxy 实例作为函数调用的操作，比如proxy(...args)、proxy.call(object, ...args)、proxy.apply(...)。

construct(target, args)：拦截 Proxy 实例作为构造函数调用的操作，比如new proxy(...args)。

1. **什么是Reflect?**

Reflect和Proxy一样,是为了操作对象而提供的API

1. **Reflect的静态方法**

Reflect.apply(target, thisArg, args)

Reflect.construct(target, args)

Reflect.get(target, name, receiver)

Reflect.set(target, name, value, receiver)

Reflect.defineProperty(target, name, desc)

Reflect.deleteProperty(target, name)

Reflect.has(target, name)

Reflect.ownKeys(target)

Reflect.isExtensible(target)

Reflect.preventExtensions(target)

Reflect.getOwnPropertyDescriptor(target, name)

Reflect.getPrototypeOf(target)

Reflect.setPrototypeOf(target, prototype)