深入探究ES6中的Proxy

jCodeLife
0.366 2020.09.18 17:22:35 字数 1,238 阅读 46

编辑文章

`Proxy`字面上是代理的意思
通过`typeof`来检测一下

```
console.log(typeof Proxy)
```

得知`Proxy`是定义在`window`上的全局变量，类型为`function`
并且首字母大写，我们可以猜出它是一个构造函数或者说是一个类
那本身可以执行吗

```
let res = Proxy()
```

以上代码会报错：`Uncaught TypeError: Constructor Proxy requires 'new'`
原来使用`Proxy()`执行，需要配合`new`
好吧，那我们来`new`一个`Proxy`子类对象

```
let p = new Proxy()
```

以上代码执行也会抛出错误：`Uncaught TypeError: Cannot create proxy with a non-object as target or handler`,意思是创建`proxy`对象时，不能使用不是对象的东西作为`target`或者`handler`传入
这里体现两个重要信息：

1. `Proxy`在构造对象时接受两个参数：`target`和`handler`
2. 两个参数的类型必须是`object`

那问题来了，这两个参数`target`和`handler`分别表示什么呢
在最开始，我说过`Proxy`的本意是代理意思，表示由它来“代理”某些操作；网上还有另外一个更恰当的理解，就是：

> 可以将`Proxy`理解成“拦截”，在目标对象之前架设一层“拦截”，当外界对该对象的访问，都必须先通过这层拦截，正因为有了一种拦截机制，当外界的访问我们可以对进行一些操作（过滤或改写）

所以我们可以很好的理解，`target`表示的就是要拦截（代理）的目标对象；而`handler`是用来定制拦截行为
target很容易理解，关键就在`handler`里头到底可以填什么呢？分别用于拦截什么操作呢？
要想明白这点，就得回顾我们之前怎么操作对象
例如：

```
let obj = {
    name: 'alice',
    showName() {
        console.log(`my name is ${this.name}`)
```

推荐阅读

5个技巧让你更好的编写JavaScript(ES6) 中条件语句
阅读 106

Es6基础语法
阅读 1,255

JS面试题汇总（未更新完）
阅读 379

vue项目实现登录携带token
阅读 374

浅谈JS原型链
阅读 176

```
console.log(obj.name)//alice
```

2. 给对象添加属性

```
obj.age = 12;
```

3. 判断属性是否在对象中

```
console.log('age' in obj)//true
```

4. 删除对象属性

```
delete obj.age
```

5. 通过各种方法遍历对象的所有属性

```
console.log(Object.getOwnPropertyNames(obj));//["name", "showName"]
console.log(Object.getOwnPropertySymbols(obj));//[]
console.log(Object.keys(obj))//["name", "showName"]
for (let key in obj){
    console.log(key)
}//分别打印name showName
```

6. 获取对象的某个属性的描述对象

```
let d = Object.getOwnPropertyDescriptor(obj,'name')
console.log(d)
//{value: "alice", writable: true, enumerable: true, configurable: true}
```

7. 使用Object身上的方法，为某个对象添加一个或多个属性

```
Object.defineProperty(obj,'age',{
    value:12,
    writable:true,
    enumerable:true,
    configurable:true
})
Object.defineProperties(obj,{
    showAge:{
        value:function(){console.log(`我今年${this.age}岁了`)},
        writable:true,
        enumerable:true,
        configurable:true,
    },
    showInfo:{
        value:function(){console.log(`我叫${this.name},我今年${this.age}岁了`)},
        writable:true,
        enumerable:true,
        configurable:true,
    }
})
```

8. 获取一个对象的原型对象

推荐阅读

5个技巧让你更好的编写JavaScript(ES6) 中条件语句
阅读 106

Es6基础语法
阅读 1,255

JS面试题汇总（未更新完）
阅读 379

vue项目实现登录携带token
阅读 374

浅谈JS原型链
阅读 176

```
Object.setPrototypeOf(obj,null);
//表示设置对象的原型为null，也可以传入其他对象作为其原型
```

10. 让一个对象变得不可扩展，即不能添加新的属性

```
Object.preventExtensions(obj)
```

11. 查看一个对象是不是可扩展的

```
console.log(Object.isExtensible(obj));//false，因为上面设置了该对象为不可扩展对象
```

12. 如果对象为function类型，function类型的对象可以执行被执行符号()以及.call()和.apply()执行

```
function fn(...args){
    console.log(this,args)
}
fn(1,2,3);
fn.call(obj,1,2,3);
fn.apply(obj,[1,2,3]);
```

13. 如果对象时作为构造函数时，则该对象可以用new生成出新的对象

```
function Person(){}
let p1 = new Person();
```

以上都是对对象的一些操作！
那回到我们得`new Proxy(target,handler)`中的`handler`，我们之前说了，`handler`是用于设置拦截行为的，其实拦截的内容就是上面这一系列的对象操作，当对象执行某个操作时，就会触发`handler`里面定义的东西，
通过`Proxy`可以拦截到被代理的对象执行的相关操作；当被代理对象执行某个操作时，那么它会执行该操作所对应的handler里面的函数，有点像我代理你去做事情，这也是被叫做`Proxy`的本意

下面来看具体`handler`对象可以设置哪些参数，分别拦截被代理对象的哪些操作

1. `get(target, propKey, receiver)`

get方法用于拦截某个属性的读取操作，比如`proxy.foo`和`proxy['foo']`

```
var person = {
  name: "Alice"
};
var proxy = new Proxy(person, {
  get: function(target, propKey) {
    if (propKey in target) {
      return target[propKey];
    } else {
      throw new ReferenceError(`Prop name ${propKey} does not exist.`);
    }
  }
});
proxy.name // "Alice"
```

推荐阅读

5个技巧让你更好的编写JavaScript(ES6) 中条件语句
阅读 106

Es6基础语法
阅读 1,255

JS面试题汇总（未更新完）
阅读 379

vue项目实现登录携带token
阅读 374

浅谈JS原型链
阅读 176

2. `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。

3. `has(target, propKey)`

   拦截`propKey in proxy`的操作，返回一个布尔值。

4. `deleteProperty(target, propKey)`

   拦截delete proxy[propKey]的操作，返回一个布尔值。

5. `ownKeys(target)`

   拦截`Object.getOwnPropertyNames(proxy)`、`Object.getOwnPropertySymbols(proxy)`、`Object.keys(proxy)`、`for...in`循环，返回一个数组。

6. `getOwnPropertyDescriptor(target, propKey)`

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

7. `defineProperty(target, propKey, propDesc)`

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

8. `preventExtensions(target)`

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

9. `getPrototypeOf(target)`

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

10. `isExtensible(target)`

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

11. `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。

12. `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(...)`。

13. `construct(target, args)`

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

待续...

jCodeLife 书山有路勤为径，学海无涯苦作舟
总资产75 (约5.76元) 共写了13.6W字 获得342个赞 共30个粉丝

推荐阅读

5个技巧让你更好的编写JavaScript(ES6) 中条件语句
阅读 106

Es6基础语法
阅读 1,255

JS面试题汇总（未更新完）
阅读 379

vue项目实现登录携带token
阅读 374

浅谈JS原型链
阅读 176

全部评论 0 只看作者 关闭评论 按时间倒序 按时间正序

被以下专题收入，发现更多相似内容 投稿管理

+收入我的专题 前段开发那些事儿

推荐阅读

5个技巧让你更好的编写JavaScript(ES6) 中条件语句
阅读 106

Es6基础语法
阅读 1,255

JS面试题汇总（未更新完）
阅读 379

vue项目实现登录携带token
阅读 374

浅谈JS原型链
阅读 176