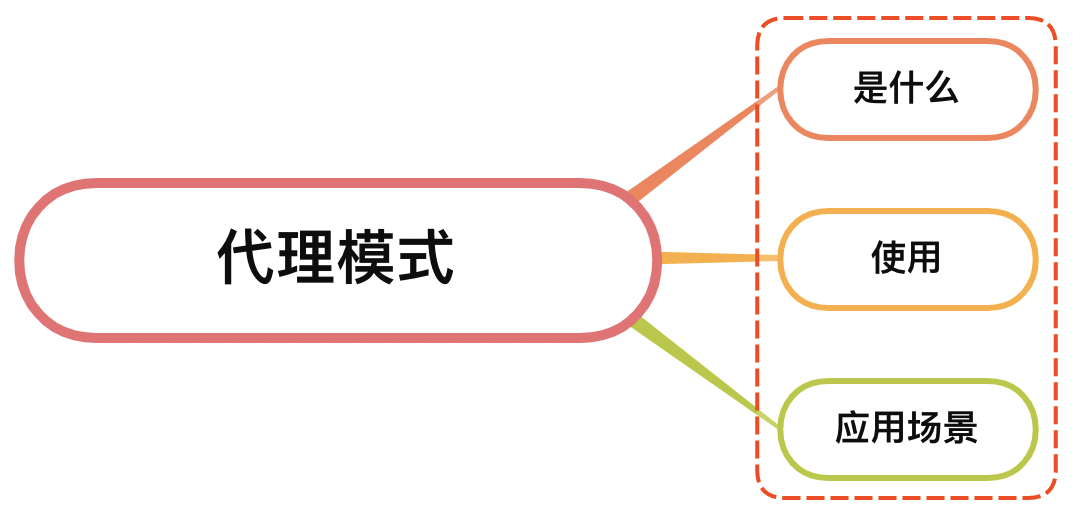
# 面试官：说说你对代理模式的理解？应用场景？



## 一、是什么

代理模式（Proxy Pattern）是为一个对象提供一个代用品或占位符，以便控制对它的访问

代理模式的关键是，当客户不方便直接访问一个对象或者不满足需要时，提供一个替身对象来控制这个对象的访问，客户实际上访问的是替身对象



在生活中，代理模式的场景是十分常见的，例如我们现在如果有租房、买房的需求，更多的是去找链家等房屋中介机构，而不是直接寻找想卖房或出租房的人谈。此时，链家起到的作用就是代理的作用

## 二、使用

在ES6中，存在proxy构建函数能够让我们轻松使用代理模式：

const proxy = new Proxy(target, handler);

关于Proxy的使用可以翻看以前的文章

而按照功能来划分，javascript代理模式常用的有：

* 缓存代理
* 虚拟代理

### 缓存代理

缓存代理可以为一些开销大的运算结果提供暂时的存储，在下次运算时，如果传递进来的参数跟之前一致，则可以直接返回前面存储的运算结果

如实现一个求积乘的函数，如下：

var muti = function () {  
 console.log("开始计算乘积");  
 var a = 1;  
 for (var i = 0, l = arguments.length; i < l; i++) {  
 a = a \* arguments[i];  
 }  
 return a;  
};

现在加入缓存代理，如下：

var proxyMult = (function () {  
 var cache = {};  
 return function () {  
 var args = Array.prototype.join.call(arguments, ",");  
 if (args in cache) {  
 return cache[args];  
 }  
 return (cache[args] = mult.apply(this, arguments));  
 };  
})();  
  
proxyMult(1, 2, 3, 4); // 输出:24  
proxyMult(1, 2, 3, 4); // 输出:24

当第二次调用 proxyMult(1, 2, 3, 4) 时，本体 mult 函数并没有被计算，proxyMult 直接返回了之前缓存好的计算结果

### 虚拟代理

虚拟代理把一些开销很大的对象，延迟到真正需要它的时候才去创建

常见的就是图片预加载功能：

未使用代理模式如下：

let MyImage = (function(){  
 let imgNode = document.createElement( 'img' );  
 document.body.appendChild( imgNode );  
 // 创建一个Image对象，用于加载需要设置的图片  
 let img = new Image;  
  
 img.onload = function(){  
 // 监听到图片加载完成后，设置src为加载完成后的图片  
 imgNode.src = img.src;  
 };  
  
 return {  
 setSrc: function( src ){  
 // 设置图片的时候，设置为默认的loading图  
 imgNode.src = 'https://img.zcool.cn/community/01deed576019060000018c1bd2352d.gif';  
 // 把真正需要设置的图片传给Image对象的src属性  
 img.src = src;  
 }  
 }  
})();  
  
MyImage.setSrc( 'https://xxx.jpg' );

MyImage对象除了负责给img节点设置src外，还要负责预加载图片，违反了面向对象设计的原则——单一职责原则

上述过程loding则是耦合进MyImage对象里的，如果以后某个时候，我们不需要预加载显示loading这个功能了，就只能在MyImage对象里面改动代码

使用代理模式，代码则如下：

// 图片本地对象，负责往页面中创建一个img标签，并且提供一个对外的setSrc接口  
let myImage = (function(){  
 let imgNode = document.createElement( 'img' );  
 document.body.appendChild( imgNode );  
  
 return {  
 //setSrc接口，外界调用这个接口，便可以给该img标签设置src属性  
 setSrc: function( src ){  
 imgNode.src = src;  
 }  
 }  
})();  
// 代理对象，负责图片预加载功能  
let proxyImage = (function(){  
 // 创建一个Image对象，用于加载需要设置的图片  
 let img = new Image;  
 img.onload = function(){  
 // 监听到图片加载完成后，给被代理的图片本地对象设置src为加载完成后的图片  
 myImage.setSrc( this.src );  
 }  
 return {  
 setSrc: function( src ){  
 // 设置图片时，在图片未被真正加载好时，以这张图作为loading，提示用户图片正在加载  
 myImage.setSrc( 'https://img.zcool.cn/community/01deed576019060000018c1bd2352d.gif' );  
 img.src = src;  
 }  
 }  
})();  
  
proxyImage.setSrc( 'https://xxx.jpg' );

使用代理模式后，图片本地对象负责往页面中创建一个img标签，并且提供一个对外的setSrc接口；

代理对象负责在图片未加载完成之前，引入预加载的loading图，负责了图片预加载的功能

上述并没有改变或者增加MyImage的接口，但是通过代理对象，实际上给系统添加了新的行为

并且上述代理模式可以发现，代理和本体接口的一致性，如果有一天不需要预加载，那么就不需要代理对象，可以选择直接请求本体。其中关键是代理对象和本体都对外提供了 setSrc 方法

‘

## 三、应用场景

现在的很多前端框架或者状态管理框架都使用代理模式，用与监听变量的变化

使用代理模式代理对象的访问的方式，一般又被称为拦截器，比如我们在项目中经常使用 Axios 的实例来进行 HTTP 的请求，使用拦截器 interceptor 可以提前对 请求前的数据 服务器返回的数据进行一些预处理

以及上述应用到的缓存代理和虚拟代理

## 参考文献

* https://juejin.cn/post/6844903555036364814#heading-2
* https://juejin.cn/post/6992510837403418654#heading-7
* https://sothx.com/2021/06/26/proxy/