```
自动转换请求和响应数据
                                                                                                                                                                      对比ajax,axios
                                                                                             const controller = new AbortController()
                                                                                            axios.get('/foo/bar', {
                                                                                              signal: controller.signal
                                                                                            }).then(function(response) {
                                                                                                                                 —— 取消请求
                                                                                                                                                  优势 —— ajax
                                                                                            // 取消请求
                                                                                            controller.abort()
                                                                                                                          客户端支持防止CSRF/XSRF
                                                                                axios既提供了并发的封装,也没有fetch的各种问题,而且体积也较小,当之无愧现在最应该
                                                                                选用的请求的方式。
                                                                               拦截<mark>timeout</mark>错误
                                                                                 在Axios响应拦截器中,处理<mark>timeout</mark>错误,代码如下:
                                                                                  if(error.message.includes('<mark>timeout</mark>')){ // 判断请求异常信息中是否含有超对<mark>timeout</mark>字符号
console.log("错误回调", error);
alert("网络超时");
                                                                                                                                     无论请求成功与否,它都返回一个 Promise 对象
                                                                                                                                   请求数据放在body中,并需要手动将对象转为字符串
                                                                                                                                          如果想流畅的使用fetch,需要手动进行封装
                                                                                                                                                                                        fetch
                                                                                             fetch("http://example.com/")
                                                                                              .then((response) => response.json()) // 需要对响应数据进行转换
                                                                                               .then((data) => {
                                                                                                console.log(data);
                                                                                               .catch((error) => console.error(error));
                                                                                                                                                  需要手动对响应数据进行转换
                                                                                                                            arrayBuffer()
                                                                                                                                                                                                                                            constructor

    blob()

                                                                                                                                                                                                                                             · 继承 —— extends和super
                                            使用Fetch时你需要清楚请求后的数据类型是什么,然后再用对应的方法将它进行转
                                                                                                                            json()
                                                                                                                            text()
                                                                                                                            formData()
                                                                                                                                                                                                                                               属于原始值,而不是引用类型
                                                                                                                                                             无法取消请求
                                                                                                                                                                                                                                              作为对象属性的唯一标识符
                                                                                                                                                           上传进度无法获取
                                                                                                                                                                                                                                              - 对象中Symbol()属性不能被for...in遍历 —— 只能通过Object.getOwnPropertySymbols(含有symbol的对象)来读取
                                                                                           对于非200状态码,不会进入catch,而是在resolve中的OK值为false
                                                                                                                                               - 无法捕获到400,500状态码错误
                                                                                                                                                                                                                                              - Symbol不能作为构造函数使用,就是说,不能用new
                                                                                                     仅当网络故障时或请求被阻止时,才会标记为 reject。
                                                                                                                                                                                                                              Symbol
                                                                                                                                                                                                                                                               Symbol作为key未被包含在对象自身的属性名集合(property names)之中。所以,利用该特
                                                                                                                              要发送需要加上credentials —— 不会发送跨域 cookies
                                                                                                                                                                                                                                                              性,我们可以把一些不需要对外操作和访问的属性使用Symbol来定义。
                                                                                                                                                             超时处理麻烦
                                                                                                                                                                                                                                                              定义类的私有属性和方法
                                                                                                                                                               有兼容问题
                                                                                                                                                                                                                                               参考链接 —— https://juejin.cn/post/7028830157230047263
                                                                                                                                                                 原生API
                                                                                                                                                  更低层,可以自行按需求封装 一 优势
                                                                                                                                                                                                                                                                                                  所有的集合对象(数组、Set集合及Map集合)和字符串都是可迭代对象,这样就需要
                                                                                                                        符合关注分离,没有将输入、输出和用事件来跟踪的状态混杂在一个对象里
                                                                                                                                                                                                                                                                                                   一种统一的接口机制,来处理所有不同的数据结构。Iterator就是这种机制.任何数据接  —— 也可以理解为 Iterator 接口主要为 for of 服务的,供for...of进行消费。
                                                                                                                                                                                                                                                                                                  口只要部署Iterator接口,就可以使用for...of。
                                                                                                                                                                                                                                                                                                    迭代器是一个拥有next()方法的特殊对象,每次调用next()都返回一个结果对象,结果对
                                                                                                       用于创建一个对象的代理,从而实现基本操作的拦截和自定义(如属性查找、赋值、枚
                                                                                                                                                                                                                                                                                                    象是:
                                                                                                      举、函数调用等)
                                                                                                                                                                                                                                                                                                                                                                       value是可迭代对象的元素,done表示是否可以继续迭代
                                                                      target表示所要拦截的目标对象(任何类型的对象,包括原生数组,函数,甚至另一个
                                                                                                                                                                                                                                                                                                      value: item,
                                                                                                                                                                                                                                                                                                                                                                         迭代到最后时, value:undefined,done:true
                                                                      代理))
                                                                                                                                                                                                                                                                                                      done: true/false
                      拦截对象取值 _____ get(target,propKey,receiver)
                                                                                                                                                                                                                                                                                                           ES6规定,默认的Iterator的接口部署在数据结构的Symbol,iterator属性,或者说,一个数据
                                                                                                                                         var proxy = new Proxy(target, handler) —— 用法 🥄
                拦截对象设置值 —— set(target,propKey,value,receiver)
                                                                                                                                                                                                                                                                                                           接口只要部署看Symbol.iterator属性,就可以使用for...of。
                                                                                                                                                                                        Proxy
                                                                                                                                                                                                                                                                                                           扩展运算符:只要数据结构部署了Iterator就可以使用扩张运算符
                         拦截对象中是否有值,并返回布尔值 —— has
                                                                   可代理的方法包括 —— handler对象,key为可代理的方法名,value为可代理方法的函数
                                                 deleteProperty
                                                                                                                                                                                                                                                                                                      return() — 遍历器遍历过程中,提前退出,会调用遍历器对象的return()。(必须返回一个对象)
拦截Object.keys(proxy)、for...in等循环,返回一个数组 —— ownKeys(target)
                                                                                                                                                                                                                                                                                                                   一 配合Generator函数使用
                                                                                                                                                                          使用场景
                                                                                                                                                                                                                                                                                                                       function getIterator(list) {
                                                                                                                                                                                                                                                                                    Iterator
                                                                                                                                                                                                                                                                                                                         var i = 0;
return {
                                                                                                                                                                                                                                                                                                                          next: function() {
                                                                                                                                                                                                                                                                                                                           var done = (i >= list.length);
var value = !done ? list[i++] : undefined;
                                                                                                                     一种类似Object的新的数据结构,一种叫做字典的数据结构,Map可以理解为是Object
                                                                                                                                                                                                                                                                                                                              done: done,
                                                                                                                     的超集,打破了以传统键值对形式定义对象,对象的key不再局限于字符串,也可以是
                                                                                                                                                                                                                                                                                                fakelterator
                                                                                                                     Object。可以更加全面的描述对象的属性。
                                                                                                                              Map 对象保存键值对。任何值(对象或者原始值) 都可以作为一个键或一个值
                                                                                                                                                                                                                                                                                                                       var it = getIterator(['a', 'b', 'c']);
console.log(it.next());// {value: "a", done: false}
                                                                                          Object遵循普通的字典规则,键必须是单一类型,并且只能是整数、字符串或是Symbol类型
                                                                                                                                                                                                                                                                                                                       console.log(it.next());// {value: "b", done: false}
                                                                                                                                                                                                                                                                                                                        console.log(it.next());// {value: "c", done: false}
                                                                                                                                                                                                                                                                                                                        console.log(it.next());// "{ value: undefined, done: true }
                                                                                                                 Map中,key可以为任意数据类型(Object, Array等)
                                                                                                                                                                   map和object的区别
                                                                                                                                                                                                                                                                                                                           function isIterable(object) {
                                                                                                          Map会保留所有元素的顺序,而Object并不会保证属性的顺序 —— 元素顺序
                                                                                                                                                                                                                                                                                                                               return typeof object[Symbol.iterator] === "function";
                                                                                                                                                                                                                                                                                                 判断是否是可迭代对象 ——
                                                                                                               ● Map 在涉及频繁增删键值对的场景下会有些性能优势。 —— 性能
                                                                                                                                                    key简单
                                                                                                                                                                                                                                                                                                                           console.log(isIterable('abcdefg')); // true
                                                                                                                                               value值是函数 —
                                                                                                                                                                                                                                                                                                                     可以使用for...of..
                                                                                                                                                  转为JSON
                                                                                                                                                                                                                                                                                                                    - 可以使用解构赋值
                                                                                                                                                                                                                                                                                                 lterator对象作用:
                                                                                                                                             需要了解size的时候
                                                                                                                                                                                                                                                                                                                     可以使用扩展运算符..
                                                                                                                                             有频繁增删操作时候
                                                                                                                                                                                                                                                                                                  生成器是一种返回迭代器的函数
                                                                                                                                                                                                                                 b代器(Iterator)与生成器(Generator)
                                                                                                                                            需要多种类型的key时
                                                                                                                                                                                                                                                                                                   特点:
                                                                                              ● 只接受对象作为键名(null 除外),不接受其他类型的值作为键名 ── WeakMap 只能将对象作为键名
                                                                                                                                                                                                                                                                                                               ·函数体内部使用yield表达式,定义不同的内部状态
                                           const e1 = document.getElementById('foo');
                                                                                                                                                                                                                                                                                                    Generator 函数是分段执行的,调用next方法函数内部逻辑开始执行,遇到yield表达
                                          const e2 = document.getElementById('bar');
                                                                                                                                                                                                                                                                                                    式停止,返回{value: yield后的表达式结果/undefined, done: false/true},再次调用
                                          const arr = [
                                                                                                                                                                                                                                                                                                    next方法会从上一次停止时的yield处开始,直到最后。
                                            [e1, 'foo 元素'],
                                            [e2, 'bar 元素'],
                                                                                                                                                                                                                                                                                                                          ● 调用生成器函数后,函数并不立即执行,返回的不是函数运行结果,而是一个迭
                                                                                                                                                                                                                                                                                                                            代器对象。
                                                                                                                         —— 强引用
                                                                                                                                                                                                                                                                                                   与普通函数的区别
                                              在上面的代码中,e1和e2是两个对象,通过arr数组对这两个对象添加一些文字说
                                                                                                                                                                                                                                                                                                                          • 通过迭代器对象next方法分段执行函数,返回yield后面表达式的值
                                              明。但是这样就形成了arr对e1和e2的引用,而这种引用又是强引用。它的区别就体
                                                                                                                                                                                                                                                                                                                                             const r = foo(0);
                                                                                                                                                                                                                                                                                                                     function* foo(x) {
                                              现在。当我们不再需要这两个对象时,我们必须手动的删除这个引用,解除arr都两个
                                                                                                                                                                                                                                                                                                                                             r.next(); // { value: 1, done: false }
                                                                                                                                                                                                                                                                                                                      yield x + 1;
                                              对象的引用关系,否则垃圾回收机制不会释放e1和e2占用的内存。因为,arr仍然存
                                                                                                                                                                                                                                                                                                                                             r.next();// { value: 2, done: false }
                                                                                                                                                                                                                                                                                                                      yield x + 2;
                                                                                                                                                                                                                                                                                                  使用方式:
                                              在着对对象的引用! 当忘记了手动删除引用, 就会造成内存泄漏
                                                                                                                                                                                                                                                                                                                                             r.next();// { value: 3, done: true }
                                                                                                                                          WeakMap 的键名引用的对象是弱引用
                                                                                                                                                                                                                                                                                                                      return x + 3;
                                                                                                                                                                                                                                                                                                                                             r.next();// { value: undefined, done: true } 被return终结了
                                                                                                                                                                                                                                                                                                                      yield x + 4;
                                                                                  let obj = new WeakObject(
                                                                                                                                                                                                                                                                                                                                             r.next();// { value: undefined, done: true }
                                                                                                                                                                                                                                                                                                                                     function* foo () {
                                                                   我们就可以静静的等待垃圾车来把它拖走了,obj所引用的对象就会被回收
                                                                                                                                                                                                                                                                                                                                      yield 1;
                                                                                                                                                                                                                                                                                                                                      yield 2;
                                                                      弱引用关系,数据的引用计数为 0 ,程序垃圾回收机制在执行时会将引用对象回收。
                                                                                                                                                                          - weakMap
                                                                                                                                                                                                                                                                                                                                      return 3;
                                                                      而如果时强引用关系则引用计数为 1 ,不会被垃圾回收机制清除。
                                                                                                                                                                                                                                                                                                  · 注意: —— 使用for...of... ——
                                                                                                                                                                                                                                                                                                                                     for (const v of foo()) {
                                                                                           正因为WeakMap对键名所引用的对象是弱引用关系,因此WeakMap内部成员是会却决于垃圾
                                                                                                                                                                                                                                                                                                                                      console.log(v);
                                                                                                                                                                                                                                                                                    Generator
                                                                                          回收机制有没有执行,运行前后成员个数很可能是不一样的,而垃圾回收机制的执行又是不可 不可遍历
                                                                                                                                                                                                                                                                                                                                           for...of...不执行 return 后的语句
                                                                                          预测的,因此不可遍历
                                                                                 const wm = new WeakMap();
                                                                                  const loginButton = document.querySelector('#login')
                                                                                                                                          DOM 节点元数据
                                                                                 wm.set(loginButton, {disabled: true});
                                                                                                                                                                                                                                                                                                                                                                                            const i = g();
i.next();
                                                                                                                                   注册监听事件的 listener 对象
                                                                                                                                                                                                                                                                                                                  Generator函数返回的遍历器对象都有一个throw方法,可以在函数体外抛出错误,然后在
                                                                                                                                                                                                                                                                                                                  Generator函数体内捕获。
                                                                                                                                                                                                                                                                                                                                                                                              i.throw('a');// 在函数体外始出错误, 在函数体内捕获错误
i.throw('b');// 在函数体外始出错误, 在函数体外捕获错误
                                                                                                                                                                                                                                                                                                                                                                                            } catch (e) {
    console.log('外部捕获', e);
                                                                           利用弱映射,将内部属性设置为实例的弱引用对象,当实例删除时,私有属性也会随之消失,
                                                                                                                                        部署私有属性 应用场景
                                                                           因此不会内存泄漏
                                                                                                                                                                                                                                                                                                                                                  function* iterEntries(obj) {
                                                                                                                                                                                                                                                                                                                                                   let keys = Object.keys(obj);
                                                                                                                                                                                                                                                                                                                                                   for (let i=0; i < keys.length; i++) {
                                                                                                                                                                                                                                                                                                                                                    let key = keys[i];
                                                                                                                                                                                                                                                                                                                                                    yield [key, obj[key]];
                                         onst cache = new WeakMap()
                                           nction countOwnKeys(obj) {
                                                                                                                                                                                                                                                                                                                ~ 在对象上部署Iterator接口 ——
                                           if (cache.has(obj)) {
                                                                                                                                                                                                                                                                                                                                                  let myObj = { foo: 3, bar: 7 };
                                             return cache.get(obj);
                                                                          当我们需要在不修改原有对象的情况下储存某些属性等,而又不想管理这些数据时,可以使用
                                            } else {
                                                                                                                                                                                                                                                                                                  应用场景
                                                                                                                                            —— 数据缓存
                                             const count = Object.keys(obj).len
                                                                                                                                                                                                                                                                                                                                                   console.log(key, value);
                                              return count;
                                                                                                                                                                                                                                                                                                                                                  // foo 3
                                                                                                                                                                                                                                                                                                                                                  // bar 7
                                                                                                                                                                                                                                                                                                                                           function* doStuff() {
                                                                                                                       https://juejin.cn/post/6993101968545677319#heading-17 —— 参考资料
                                                                                                                                                                                                                                                                                                                                            yield fs.readFile.bind(null, 'hello.txt');
                                                                                                                                                                                                                                                                                                                                            yield fs.readFile.bind(null, 'world.txt');
                                                                                                                                                                                                                                                                                                                                            yield fs.readFile.bind(null, 'and-such.txt');
                                                                                                   及时清除引用非常重要。但是,你不可能记得那么多,有时候一疏忽就忘了,所以才有那么多
                                                                                                                                                                                                                                                                                                                 对文件的批量处理 ——
                                                                                                    内存泄漏。ES6 考虑到这点,推出了两种新的数据结构: weakset 和 weakmap 。
                                                                                                                                                                                                                                                                                                                                          for (task of doStuff()) {
                                                                                                    他们对值的引用都是不计入垃圾回收机制的,也就是说,如果其他对象都不再引用该对象,那
                                                                                                                                                                                                                                                                                                                                            // task是一个函数,可以像回调函数那样使用它
                                                                                                    么垃圾回收机制会自动回收该对象所占用的内存。
                                                                                                                                                                        weakset和weakmap
                                                                                                     注册监听事件的 listener 对象很适合用 WeakMap 来实现。
                                                                                                     ele.addEventListener('click', handler, false)
                                                                                                                                                                                                                                                                            给每一个参数都在后面添加.catch来return error     这样参数抛的错误会走进promise.all的.then中
                                                                                                                                                    一 应用场景:
                                                                                                     const listener = new WeakMap()
listener.set(ele, handler)
                                                                                                                                                                                                                                                                             或者在参数中吧错误resolve出来而不是reject出来
                                                                                                     ele.addEventListener('click', listener.get(ele), false)
                                                                                                                                                                                                                                              — promise.all —— 报错处理
                                                                                                                                                                                                                                                                                                                Primise.allSettled使用方法跟Primise.all一样,不过不同之处在于无论参数实例
                                                                                                    代码 2 比起代码 1 的好处是:由于监听函数是放在 WeakMap 里面,一旦 dom 对象 ele 消失,与它绑定的
                                                                                                                                                                                                                                                                                                                resolve 还是 reject , Promise.allSettled 都会执行 then 方法的第一个回调函数
                                                                                                     监听函数 handler 也会自动消失
                                                                                                                                                                                                                                                                                                                 (意思就是不会 catch 到参数实例的 reject 状态)
                                                                                                                                                                                                                                                                            使用Primise.allSettled代理promise.all
                                                                                                                                                                                                                                                                                                                其回调函数的参数返回的数组的每一项是一个包含 status 和 value 或者 reason 的一
                                                                                                                                                                            无序且唯一
                                                                                                                       类似Array的新的数据结构,Set是一种叫做集合的数据结构,类似于数组,但没有重复
                                                                                                                       的值
                                                                                                                                                                                                                                                    let,const
                                                                                                                                                 Set 本身是一个构造函数,用来生成 Set 数据结构
                                                                                                                                                                                                                                                    箭头函数
                                                                                                                             size:返回集合所包含元素的数量,相当于数组的length —— Set的属性
                                                                                                                                                                                                                                                    解构赋值
                                                                                                                                            * add(value): 向集合添加一个新的项
                                                                                                                                                                                                                                                   扩展运算符
                                                                                                                               * has(value): 如果值在集合中存在,返回true,否则false
                                                                                                                                                                                                                                                    Promise
                                                                                                                                            * delete(value): 从集合中移除一个值
                                                                                                                                                                                                                              ES6常用语法
                                                                                                                                                 * clear(): 移除集合里所有的项
                                                                                                                                                                                                                                                              reduce
                                                                                                                                                                                                                                                    数组:
                                                                                                                                                                                                                                                               filter
                                                                                          let set = new Set(['a', 'b', 'c'])
                                                                                          console.log('keys',set.values())
                                                                                                                                                                                                                                                               forEach
                                                                                          console.log('values',set.values())
                                                                                                                                 * keys(): 返回一个包含集合中所有键的遍历器
                                                                                                                                                                                                                                                                模板字符串: `${name}是我`
                                                                                          keys ▶ SetIterator {'a', 'b', 'c'}
                                                                                                                                                                                                                                                    字符串
                                                                                                                                                                                                                                                                               let str='git://www.baidu.com/23448';
                                                                                          values ▶ SetIterator {'a', 'b', 'c'}
                                                                                                                                                                                                                                                                               if(str.startsWith('http://')){
                                                                                                                                                                                                                                                               startWith ____
                                                                                                                                                                             遍历方法
                                                                                                                                                                                                                                                                                         alert('普通地址');
                                                                                                                                     * values(): 返回一个包含集合中所有值的遍历器
                                                                                                                  * entries:返回一个包含集合中所有键值对的数组(感觉没什么用就不实现了)
                                                                                                                               * forEach(): 用于对集合成员执行某种操作,没有返回值
                                                                                                                      const items = new Set([1, 2, 3, 2])
                                                                                                                      const array = Array.from(items) —— Array.from 方法可以将 Set 结构转为数组
                                                                                                                      console.log(array) // [1, 2, 3]
                                                                                                        let set1 = new Set([1, 2, 3])
                                                                                                        let set2 = new Set([4, 3, 2])
                                                                                                        let intersect = new Set([...set1].filter(value => set2.has(value)))
                                                                                                        let union = new Set([...set1, ...set2])
                                                                                                                                                  交集、并集、差集
                                                                                                        let difference = new Set([...set1].filter(value => !set2.has(value)))
                                                                                                        console.log(intersect) // Set {2, 3}
                                                                                                                                                                           应用场景:
                                                                                                        console.log(union) // Set {1, 2, 3, 4}
                                                                                                        console.log(difference) // Set {1}
                                                                                                                                       [...new Set([1, 2, 2, 3])] —— 数组去重
                                                                                                                  集合 是以 [value, value]的形式储存元素,字典 是以 [key, value] 的形式储存 —— 与Map的区别
                                                                                                                                              WeakSet 只能储存对象引用
                                                                                                                                                   weakSet无法遍历
                                                                                                                                                                      set和weakSet区别
                                                                ● 即垃圾回收机制不考虑 WeakSet 对该对象的应用,如果没有其他的变量或属性引用这
                                                                  个对象值,则这个对象将会被垃圾回收掉(不考虑该对象还存在于 WeakSet 中),所
                                                                  以,WeakSet 对象里有多少个成员元素,取决于垃圾回收机制有没有运行,运行前后
                                                                                                                             —— WeakSet 对象中储存的对象值都是被弱引用的
                                                                  成员个数可能不一致,遍历结束之后,有的成员可能取不到了(被垃圾回收了),
```

用Promise吧XHR封装起来 —— axios

Axios的一大卖点就是它提供了拦截器,可以统一对请求或响应进行一些处理

WeakSet 对象是无法被遍历的(ES6 规定 WeakSet 不可遍历),也没有办法拿到它

包含的所有元素

可以为请求附加token、为请求增加时间戳防止请求缓存,以及拦截响应,一旦状态码

不符合预期则直接将响应消息通过弹框的形式展示在界面上