promise使用

构造一个Promise实例需要给Promise构造函数传入一个函数。 传入的函数需要有两个形参，两个形参都是function类型的参数。 第一个形参运行后会让Promise实例处于resolve状态，所以我们一般给第一个形参命名为resolve,使 Promise 对象的状态改变成成功，同时传递一个参数用于后续成功后的操作 第一个形参运行后会让Promise实例处于reject状态，所以我们一般给第一个形参命名为reject,将 Promise 对象的状态改变为失败，同时将错误的信息传递到后续错误处理的操作

```
let promise = new Promise((resolve, reject) => {
    setTimeout(() => {
        if(Math.random()>0.5)
            resolve('This is resolve!');
        else
            reject('This is reject!');
    }, 1000);
});
promise.then(Fulfilled,Rejected)
```

promise模拟

```
//es5
function Promise(fn) {
    fn((data)=> {
        this.success(data);
    }, (error)=> {
        this.error();
    });
}

Promise.prototype.resolve = function (data) {
    this.success(data);
}

Promise.prototype.reject = function (error) {
    this.error(error);
}

Promise.prototype.then = function (success, error) {
    this.success = success;
    this.error = error;
```

```
}
//es6
class Promise {
    constructor(fn) {
        fn((data)=> {
            this.success(data);
        }, (error)=> {
            this.error();
        });
    }

    resolve(data) {
        this.success(data);
    }

    reject(error) {
        this.error(error);
    }

    then(success, error) {
        this.success = success;
        this.error = error;
    }
}
```

目前的调用中，多以promise实例作为返回值

```
function ajaxPromise (queryUrl) {
  return new Promise((resolve, reject) => {
    let xhr = new XMLHttpRequest();
    xhr.open('GET', queryUrl, true);
    xhr.send(null);
    xhr.onreadystatechange = () => {
      if (xhr.readyState === 4) {
        if (xhr.status === 200) {
          resolve(xhr.responseText);
        } else {
          reject(xhr.responseText);
        }
      }
    }
  });
}

ajaxPromise('http://www.baidu.com')
  .then((value) => {
```

```
    console.log(value);
  })
  .catch((err) => {
    console.error(err);
  });
```

链式调用

```
readFile('1.txt').then(function (data) {
    console.log(data);
    return data;
}).then(function (data) {
    console.log(data);
    return readFile(data);
}).then(function (data) {
    console.log(data);
}).catch(function(err){
 console.log(err);
});
```

第三方插件

bluebird，直接给异步方法封装一层promise

```
var Promise = require('./bluebird');

var readFile = Promise.promisify(require("fs").readFile);
readFile("1.txt", "utf8").then(function(contents) {
    console.log(contents);
})

var fs = Promise.promisifyAll(require("fs"));

fs.readFileAsync("1.txt", "utf8").then(function (contents) {
    console.log(contents);
})
```

案例

promise小球动画