promise

2019年12月10日

16:22

|  |  |
| --- | --- |
|  | **异步处理之.Promise**  doSomething()  .then(doSomethingElse(value))  .then(doThirdThing(newResult))  .then(finalResult => console.log(`Got the final result: ${finalResult}`))  .catch(failureCallback);  注释:  doSomething,doSomething()等异步函数通过返回一个Promise对象,从而附加回调函数来使用.    Promise传入一个函数作为回调函数,回调函数具有两个形参(resolve,reject);Promise对象调用.then或者.catch方法  then() 或者catch()函数会返回一个全新的 Promise对象.    new Promise((resolve, reject) => {  console.log('初始化');  resolve(11);  })  .then((result) => {  console.log(result);  new Promise((resolve,reject) => {reject(12)})  })  .catch((result) => {  console.log(result);  })  .then(() => {  console.log('执行「这个」，无论前面发生了什么');  });    ((result) => {return new Promise((resolve,reject) => {console.log(result),resolve(result+1)})})(result)  .then((newResult => {new Promise((resolve,reject) => {console.log(newResult),resolve(newResult+1)})})) |

**异步处理之async,await**

const delay = ms => new Promise(resolve => setTimeout(resolve, ms));

或者

function timeout(ms) {

return new Promise((resolve) => {

setTimeout(resolve, ms);

});

}

async function asyncPrint(value, ms) {

await timeout(ms);

console.log(value);

}

asyncPrint('hello world', 50);

**通过函数只调用实现异步的处理 :**

function f1(ms,t) {

let times = t -1;

console.log(times);

if(times>0){

setTimeout(() => {

f1(ms,times)

},ms)

}else {

return 0;

}

}