**Promise**

第一章：基礎知識準備：

1. 物件的分類 (陸：對象)：

1. 函數物件：將函數作為物件使用時，簡稱為函數物件。

2. 實例物件：new 函數產生的物件，簡稱為物件。

* 例子：

function Fn(){ // Fn是函數，()左邊就是函數

}

const fn = new Fn(); // Fn是構造函數，fn是實例對象(簡稱為對象)

console.log(Fn.prototype); // Fn是函數對象 => .

Fn.call({}); // Fn是函數對象

$('#test') // jQuery函數

$.get('/test') // jQuery函數對象

1. 回調函數的分類：

1.同步回調：

理解：立即執行，完全執行完了才結束，不會放入回調隊列中。

例子：陣列遍歷相關的回調函數 / Promise的excutor函數。

2.異步回調：

理解：不會立即執行，會放入回調陣列中將來執行。

例子：定時器回調 / ajax回調 / Promise的成功|失敗的回調。

* 例子：

1.同步回調函數：

const arr = [1,2,3];

// 遍歷回調，同步回調函數不會放入隊列，一上來就要執行完

arr.forEach( item => {

console.log(item);

})

console.log('forEach()之後');

// Ans： 1 2 3 'forEach()之後'

2.異步回調函數：

setTimeout(()=>{ // 異步回調函數，會放入隊列中將來執行

console.log('timeout callback()');

},0)

console.log('setTimeout()之後');

// Ans：'setTimeout()之後' 'timeout callback()'

1. Error分類：

目標：進一步理解JS中的錯誤(Error)和錯誤處理。

(一)、錯誤的類型：

1.Error：所有錯誤的父類型。

2.ReferenceError：引用的變量不存在。

3.TypeError：數據類型不正確的錯誤。

4.RangeError：數據值不在其所允許的範圍內。

5.SyntaxError：語法錯誤。

(二)、錯誤處理：

1.補獲錯誤：try....catch

2.拋出錯誤：throw error

(三)、錯誤物件：

1.message屬性：錯誤相關訊息。

2.stack屬性：函數調用棧紀錄信息。

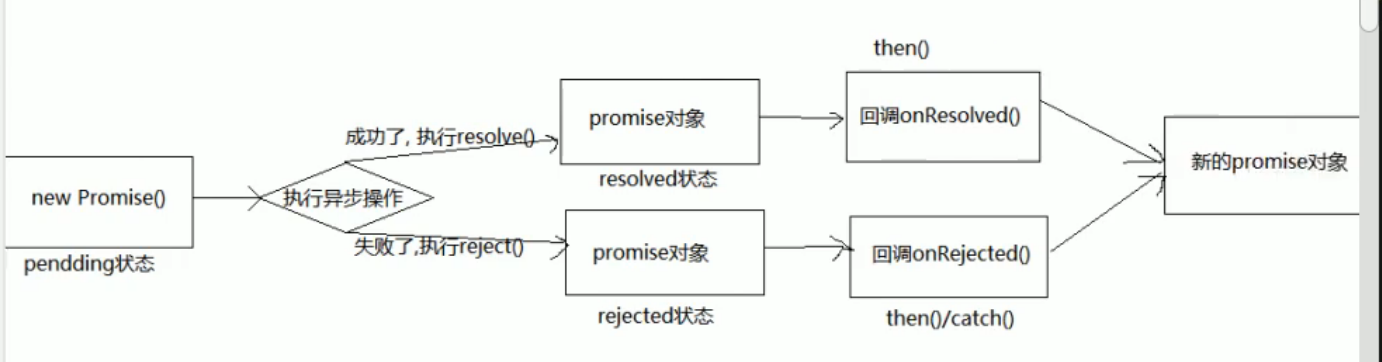
第二章：Promise的理解和使用：

1. Promise是什麼？
2. 抽象表達：

Promise是JS中進行**異步編程**的新的解決方案(舊的是誰?)

1. 具體表達：
   1. 從語法上來說：Promise是一個構造函數。
   2. 從功能上來說：Promise物件用來封裝一個異步操作並可以獲取其結果。
2. Promise的狀態改變：
   1. pending (等待) —變為→ resolved (成功/解決)
   2. pending (等待) —變為→ rejected (失敗/拒絕)
      * 說明：只有這 2 種狀態，且一個Promise物件只能改變一次，無論變為成功還是失敗，都會有一個結果數據。

成功的結果數據一般稱value，失敗的結果數據一般稱reason (原因)。

1. Promise的基本流程：
2. Promise的基本使用方式：

第一步：創建一個 new Promise()

const p = new Promise((resove,reject) => {

第二步：執行異步操作

setTimeout(()=>{

第三之一步：如果成功了，調用resolve(value)

If(true){

resolve(value)

}else{

第三之二步：如果失敗了，調用reject(reason)

reject(reason);

}

},1000)

})

第四步：接收成功/失敗的結果數據

p.then(

第四之一：接收得到成功的value數據

value => {

console.log(value);

},

第四之二：接收得到失敗的reason數據

reason => {

console.log(reason);

}

)

1. 為什麼要使用Promise？

1.指定回調函數的方式更加靈活：

舊的方式：必須在啟動異步任務前指定回調函數。

Promise：啟動異步任務 => 返回promise物件 => 給promise物件綁

定回調函數(甚至可以在異步任務結束後指定)。

2.支持鏈式調用，可以解決回調地獄問題：

舊的方式：

2-1.什麼是回調地獄？

Ans：回調函數嵌套調用，外部回調函數異步執行的結果是該

嵌套回調函數的執行條件。

2-2.回調地獄的缺點？

Ans：不便於閱讀 / 不便於異常處理。

解決方案？

Ans：Promise鏈式調用，但仍有回調函數的使用。

終極解決方案？

Ans：async / await，不會使用回調函數來執行。

1. Promise的API ：

● API：大致意義上可以分為 語法 或 前後台的接口。

● Promise的API (語法)：

1.Promise構造函數：Promise(excutor){}

excutor函數：同步執行 (resolve,reject) => {}

resolve函數：內部定義成功時我們調用的函數 value => {}

reject函數：內部定義失敗時我們調用的函數 reason => {}

※ 說明：excutor(執行器函數)會在Promise內部立即同步回調，

異步操作在執行器中執行。

1. Promise.prototype.then方法 (實例物件的方法)：

(onResolved,onRejected) => {}

onResolved函數：成功的回調函數 (value) => {}

onRejected函數：失敗的回調函數 (reason) => {}

※ 說明：指定用於得到成功value的成功回調和用於得到失敗

reason的失敗回調，並返回一個新的Promise物件。

3.Promise.prototype.catch方法(實例物件的方法)：(onRejected) => {}

onRejected函數：失敗的回調函數 (reason) => {}

※ 說明：then()的語法糖，相當於：then(undefined, onRejected)

4.Promise.resolve方法 (函數物件的方法)： (value) => {}

value：成功的數據或Promise物件。

※ 說明：返回一個成功/失敗的Promise物件。

5.Promise.reject方法 (函數物件的方法)： (reason) => {}

reason：失敗的原因。

※ 說明：返回一個失敗的Promise物件。

6.Promise.all方法 (函數物件的方法)： (promises) => {}

promises：包含N個Promise的陣列。

※ 說明：返回一個新的Promise，只有所有的Promise都成功才

成功，只要有一個失敗了就直接失敗，其順序是依照陣

列的順序，非執行時先返回的值為準(與race相反)。

7.Promise.race方法 (函數物件的方法)： (promise) => {}

promises：包含N個Promise的陣列。

※ 說明：返回一個新的Promise，第一個返回的Promise的結果

狀態就是最終的結果狀態，其結果是依最先執行完成的

為準，非依照陣列順序(與all相反)。

race：比賽、競賽的意思。

1. Promise的幾個關鍵問題：

● Promise的關鍵字：串連、傳透、中斷、誰來決定新Promise的狀態。

● Promise的幾個關鍵問題(7個)：

1. 如何改變Promise的狀態(resolved、rejected)？

1-1.resolve(value)：如果當前是pending就會變為resolved。

1-2.rejected(reason)：如果當前是pending就會變成rejected。

1-3.拋出異常(throw)：如果當前是pending就會變為rejected。

1. 一個Promise指定多個成功/失敗的回調函數，都會調用嗎？

Ans：當Promise改變為對應狀態時都會調用。

1. 改變Promise狀態和指定回調函數時，誰先誰後？

3-1.都有可能，正常情況下是先指定回調函數再改變狀態，但也可以先

改變狀態再指定回調函數。

3-2.如何先改狀態再指定回調函數？

Ans：

3-2-1.在執行器中直接調用 resolve() / reject()

3-2-2.延遲更長時間才調用then() (計時器)

3-3.什麼時候才能得到數據？

Ans：

3-3-1.如果先指定回調函數，當狀態發生改變時，回調函數就

會調用，得到數據。

3-3-2.如果先改變狀態，那當指定回調函數時，回調函數就會

調用，得到數據。

1. Promise.then()返回的新Promise的結果狀態由什麼決定？

4-1.簡單表達：由then()指定的回調函數所執行的結果來決定。

4-2.詳細表達：

4-2-1.如果是拋出異常，新Promise變為rejected，reason為 拋出

的異常值。

4-2-2.如果返回的是非Promise的任意值，新Promise變為

resolved，value為返回的值。

4-2-3.如果返回的是另一個新Promise，此Promise的結果會成為

新Promise的狀態結果。

1. Promise如和串連多個操作任務？

5-1.Promise的then()返回一個新的Promise，可以開成then()的鏈式調

用。

5-2.通過then()的鏈式調用串連多個同步/異步任務。

※ 異步操作都要包在new Promise裡！！

1. Promise異常傳透？

6-1.當使用Promise的then()鏈式調用時，可以在最後指定失敗的回調

函數。

6-2.當前面任何操作出了異常，都會傳到最後失敗的回調函數中處理。

※ 相當於前面未寫的失敗回調函數都默認為 reason => {throw reason}

1. 中斷Promise鏈？

7-1.當使用Promise的then()鏈式調用時，再中間中斷，不再調用後面

的回調函數。

Ans：再回調函數中返回一個pending狀態的Promise物件即可。