```
 1 const _ = window._;
 2 const WorkerClass = {};
 3
 4 WorkerClass.GModules = {};
 5
 6 export { WorkerClass };
 7
 8 let WORKER_UID = 0;
 9
10 // 抽象類
11 class WorkerProxy {
12     // employment: 是否是僱員
13     constructor(manager, employment) {
14
15         this._;
16
17         this.manager = manager;
18
19         this.sourceScriptPath;
20
21         this.extension1Path;
22
23         this.idleTime;
24
25         this.workerUrl;
26
27         // 需要被 worker 導入的 scripts
28         this.importScriptList = [];
29         //------------------
30         this.id = WORKER_UID++;
31
32         // 當前任務
33         this.job;
34
35         // worker
36         this.worker;
37
38         this.timeHandle;
39         //------------------
40         this.flag_busy = false;
41
42         // 旗標
43         this.flag_inited = false;
44
45         // 是否是正職者
46         this.employment = !!employment;
47
48         // this.flag_waitCount = 0;
49
50         //------------------
51         this.event_end;
52         this.event_error;
53         //------------------
54
55         this._init();
56     }
57     //-----------------------------------------------------------
58     _init() {
59         this._getSettings();
60
61         this.initWorker();
62     }
```

```
 63     //-----------------------------------------------------------
 64     _getSettings() {
 65         // debugger;
 66 
 67         this._ = this.manager._;
 68 
 69         if (this._.$extension1 == null) {
 70             throw new Error("need import _extension1");
 71         }
 72 
 73         const info = _.$extension1.info;
 74         const settings = this.manager.settings;
 75 
 76         // sourceScriptPath
 77         this.sourceScriptPath = settings.sourceScriptPath;
 78 
 79         // extensionPath1
 80         this.extension1Path = settings.extension1ScriptPath;
 81 
 82         this.importScriptList = settings.importScriptList.slice();
 83 
 84         if (this.extension1Path) {
 85             this.importScriptList.unshift(this.extension1Path);
 86         } else {
 87             throw new TypeError('no _extension1 path');
 88         }
 89 
 90         if (this.sourceScriptPath) {
 91             this.importScriptList.unshift(this.sourceScriptPath);
 92         } else {
 93             throw new TypeError('no set (lodash|underscore) path');
 94         }
 95     }
 96     //-----------------------------------------------------------
 97     initWorker() {
 98         console.log('manager employment new worker(%s), employment: %s', this.id,
this.employment);
 99 
100         this.manager.addWorker(this);
101         this._initWorker();
102     }
103 
104     // @override
105     _initWorker(){
106         throw new Error('need override _initWorker');
107     }
108 
109     //-----------------------------------------------------------
110     // @override
111     _event_getEndEvent() {
112         throw new Error('need override _event_getEndEvent');
113     }
114     //-----------------------------------------------------------
115     // @override
116     _event_getErrorEvent() {
117         throw new Error('need override _event_getErrorEvent');
118     }
119     //-----------------------------------------------------------
120     // API
121     // CEO 請他接下一個任務
122     takeJob_callByManager(job) {
123         debugger;
```

```
124
125         // reset
126         // this.flag_waitCount = 0;
127
128         if (this.timeHandle) {
129             // 臨時僱員，正在閒置中
130
131             console.log('worker(%s)正在 idle，被 manager 叫來接工作', this.id);
132
133             clearTimeout(this.timeHandle);
134             this.timeHandle = undefined;
135         }else{
136             // 正職員工
137             console.log('mananger 指派 worker(%s)接工作', this.id);
138         }
139
140         // 非同步
141         // 執行任務
142         this._doJob(job);
143
144         // 是否還有多的工作
145         // 若有再看能否多請人來接
146         // this.manager.noticeWorkers_checkJob(this);
147
148     };
149     //-----------------------------------------------------------------
150     // API
151     takeJob_callBySelf() {
152         debugger;
153
154         console.log('worker(%s)自己檢查是否有工作可拿', this.id);
155
156         // 檢查是否有任務
157         let job = this.manager.getJob_callByWorker();
158
159         if (job == null) {
160
161             if (this.employment) {
162                 console.log('沒工作可拿，正職 worker(%s)，等待', this.id);
163             } else {
164                 console.log('沒工作可拿，兼職 worker(%s)沒工作，進入 idle', this.id)
165                 this._idle();
166             }
167         } else {
168
169             // 非同步
170             // 執行任務
171             this._doJob(job);
172
173             // 是否還有多的工作
174             // 若有再看能否多請人來接
175             // this.manager.noticeWorkers_checkJob(this);
176         }
177     }
178     //-----------------------------------------------------------------
179     // 執行任務
180     _doJob(job) {
181         this.flag_busy = true;
182
183         this.job = job;
184         // debugger;
185
```

```
186         let command = this.job.getCommand();
187         command.id = this.id;
188 
189         console.log('worker(%s)接任務(%s)', this.id, this.job.id);
190 
191         // 請 worker 工作
192         this.worker.postMessage(command);
193     };
194     //-----------------------------------------------------------------
195     // 進入 idle 狀態
196     // 若已在 idle 狀態中，就不動作
197     _idle() {
198         // this.waitDeadth = true;
199         debugger;
200 
201         const idleTime = this.manager.settings.idleTime;
202 
203         console.log('worker(%s)進入 idle(%sms)', this.id, idleTime);
204 
205         if (this.timeHandle != null) {
206             clearTimeout(this.timeHandle);
207             this.timeHandle = undefined;
208             console.log('worker(%s)問題點', this.id);
209         }
210         let $this = this;
211 
212         this.timeHandle = setTimeout(function () {
213             $this.timeHandle = undefined;
214             $this.terminate();
215         }, idleTime);
216     }
217     //-----------------------------------------------------------------
218     terminate() {
219         console.log('worker(%s) will terminate', this.id);
220 
221         let w_info = this.manager.getAllworkersInfo();
222 
223         console.log(JSON.stringify(w_info));
224 
225         let all = w_info.all;
226         let idle = w_info.idle;
227 
228         if (all.length >= 2 && idel.length == 1) {
229             // 大家都在忙，只剩我一個人有空
230             // 不要終結自己，等待工作
231             // 當事情很多時
232             // 多出一個閒置的 worker 在等待接工作
233             console.log('worker(%s) 大家都在忙，有空的只剩我一個人，在加班一下，再進入 idle
等工作', this.id);
234             this._idle();
235             return;
236         }
237         //-----------------------
238         console.log("worker(%s) terminate", this.id);
239         this.manager.removeWorker(this);
240 
241         this.closeWorker();
242     }
243     //-----------------------------------------------------------------
244     // @override
245     closeWorker() {
246         throw new Error('need override workerProxy.closeWorker()');
```

```
247     }
248     //-------------------------------------------------------------------
249     isReady2TakeJob() {
250         return (this.flag_inited && !this.flag_busy);
251     }
252     //-------------------------------------------------------------------
253     isBusy() {
254         return (!this.flag_inited || this.flag_busy);
255     }
256 }
257 //=============================================================================
258 // browser 環境用下用的 worker
259 class WebWorkerProxy extends WorkerProxy {
260     constructor(manager, employment) {
261         super(manager, employment);
262     }
263     //-----------------------------------------------------------
264     // @override
265     closeWorker() {
266         this.worker.removeEventListener('message', this._event_getEndEvent());
267         this.worker.removeEventListener('error', this._event_getErrorEvent());
268 
269         this.worker.terminate();
270         this.worker = undefined;
271     }
272     //-----------------------------------------------------------
273     _getWorkerUrl() {
274         // debugger;
275 
276         let workerContent;
277 
278         if (WorkerProxy.workerContent == null) {
279 
280             let fn = this.getWorkerFnContent();
281 
282             workerContent = fn.toString();
283 
284             // console.log(workerContent);
285 
286             let reg = /^[^{]+\{([\s\S]*)\}[^}]*$/;
287             let res = reg.exec(workerContent);
288             workerContent = res[1];
289 
290             WorkerProxy.workerContent = workerContent;
291         } else {
292             workerContent = WorkerProxy.workerContent;
293         }
294 
295         // 注入 importScriptList
296 
297         let scriptList = JSON.stringify(this.importScriptList);
298 
299         workerContent = 'const scriptList = ${scriptList};
300         ${workerContent}';
301 
302         // console.log(workerContent);
303 
304         let bb = new Blob([workerContent]);
305 
306         return URL.createObjectURL(bb);
307     }
308     //-----------------------------------------------------------
```

```
309     // @override
310     _initWorker() {
311
312         // debugger;
313
314         console.log('新進員工準備中');
315         this.workerUrl = this._getWorkerUrl();
316
317         this.worker = new Worker(this.workerUrl);
318
319         this.worker.addEventListener('error', this._event_getErrorEvent());
320         this.worker.addEventListener('message', this._event_getEndEvent());
321
322     }
323     //---------------------------------------------------------
324     // @override
325     _event_getEndEvent() {
326
327         if (this.event_end == null) {
328             // worker 工作完會呼叫此
329             this.event_end = (function (e) {
330                 // debugger;
331
332                 this.flag_busy = false;
333                 //----------------------------
334                 let data = e.data || {};
335                 let res;
336
337                 if (this.flag_inited) {
338                     // worker 已經初始化過
339
340                     if (this.job) {
341                         // 等待 worker 的工作完成
342                         console.log('worker(%s) job finished', this.id);
343
344                         let job = this.job;
345                         this.job = undefined;
346                         job.resolve(data.res);
347                     }
348
349                 } else {
350                     // worker 尚在初始化中
351                     if (this.job) {
352                         throw new Error('worker(${this.id}) get job but not initialize yet');
353                     }
354
355                     if (data.initialized != null && data.initialized) {
356                         // worker 傳來初始化的消息
357                         console.log('新員工(%s)準備好了', this.id);
358                         this.flag_inited = true;
359                     }
360                 }
361
362                 this.takeJob_callBySelf();
363             }).bind(this);
364         }
365
366         return this.event_end;
367     }
368     //---------------------------------------------------------
369     // @override
370     _event_getErrorEvent() {
```

```
371
372         if (this.event_error == null) {
373             // worker error 完會呼叫此
374             this.event_error = (function (e) {
375                 let job = this.job;
376                 this.job = undefined;
377                 this.flag_busy = false;
378
379                 if (job) {
380                     // 等待 worker 的工作錯誤
381                     job.reject(e);
382                 }
383
384                 console.log('worker(%s) error', this.id);
385
386                 // fix
387                 // 如何處理發生錯誤
388                 this.takeJob_callBySelf();
389
390             }).bind(this);
391         }
392         return this.event_error;
393     }
394     //-----------------------------------------------------------
395     // worker 的內文
396     getWorkerFnContent() {
397         return function () {
398             ////////////////////////////////////////////////////////////////////
399             //
400             // _.worker() 本體
401             //
402             ////////////////////////////////////////////////////////////////////
403             debugger;
404
405             // console.log('i am worker');
406
407             // console.log('href=(%s)', location.href);
408
409             // here
410             // let scriptList = "@@_scriptList_@@";
411
412
413             scriptList.forEach(function (scriptPath) {
414                 try {
415                     importScripts(scriptPath);
416                 } catch (error) {
417                     throw new Error('script(' + scriptPath + ')  load error');
418                 }
419             });
420
421             if (self._ == null) {
422                 throw new Error('(lodash|underscore) load error');
423             }
424
425             const $_ = self._;
426             //================================================
427             self.addEventListener('message', function (e) {
428                 // debugger;
429
430                 let data = e.data || {};
431                 //----------------------------
432                 // 命令
```

```
433                 let command = data['command'] || '';
434
435                 // 參數
436                 let args = data.args || [];
437
438                 let id = data.id;
439                 let jobID = data.jobID;
440
441                 //----------------------------
442                 // console.log('*********** in worker env >> worker(%s) start job(%s)', id,
jobID);
443
444                 if ($_ == null) {
445                     throw new Error('(lodash|underscore) load error');
446                 } else {
447                     // worker 接運算任務
448                     // debugger;
449
450                     if (!command && typeof $_[command] != 'function') {
451                         throw new TypeError('_ no this function');
452                     }
453                     // debugger;
454                     // _ 的運算
455                     let res = $_[command].apply($_, args);
456                     //----------------------------
457                     forTest(res, true);
458                 }
459
460                 function forTest(res, test) {
461
462                     if (test) {
463                         setTimeout(function () {
464                             // console.log('*********** in worker env >> worker(%s) finish
job(%s)', id, jobID);
465                             console.log('---------------');
466                             self.postMessage({
467                                 res: res
468                             });
469                         }, 1000);
470                     } else {
471                         // console.log('*********** in worker env >> worker(%s) finish
job(%s)', id, jobID);
472                         console.log('---------------');
473                         self.postMessage({
474                             res: res
475                         });
476                     }
477                 }
478             });
479             //================================================
480
481             // 通知已初始化完畢
482             self.postMessage({
483                 initialized: true
484             });
485         }
486     }
487 }
488
489 WorkerClass['WebWorkerProxy'] = WebWorkerProxy;
490 //============================================================================
491 // node.js 環境下用的 worker
```

```
492 class NodeJsWorkerProxy extends WorkerProxy {
493     constructor(manager, employment) {
494         super(manager, employment);
495     }
496     //-----------------------------------------------------------
497     _getWorkerUrl() {
498         let $extension1_path = this._.$extension1.info.extension1ScriptPath;
499 
500         // 返回 worker 的 realPath
501         // 必須在 extension 目錄下 /worker/worker 放置 nodeJsWorker.js 檔
502         this.workerUrl = $extension1_path + '/worker/worker/nodeJsWorker.js';
503     }
504     //-----------------------------------------------------------
505     _initWorker() {
506         super._initWorker();
507 
508         const importScriptList = this.importScriptList;
509 
510         this.worker = new Worker(this.workerUrl, {
511             workerData: {
512                 importScriptList: importScriptList
513             }
514         });
515 
516         this.worker.on('error', this._event_getErrorEvent());
517         this.worker.on('message', this._event_getEndEvent());
518 
519     }
520     //-----------------------------------------------------------
521     _event_getEndEvent() {
522     }
523     //-----------------------------------------------------------
524     _event_getErrorEvent() {
525 
526     }
527 }
528 WorkerClass['NodeJsWorkerProxy'] = NodeJsWorkerProxy;
```