關於 Javascript 非同步的那些事兒

陳小風 @ ModernWeb 2017

自我介紹

- 陳鋒逸（陳小風）
- 經歷
  - 微軟最有價值專家（MVP）
  - SkillTree兼任講師
  - 社群研討會講師
    - @TechPodcastNight
    - twMVC
    - AgileCommunity.tw
    - Javascript.tw

粉絲團: 愛流浪的小風

Agenda

- Javascript 與非同步
- callback vs promise
- genertor 的出現及設計精神
- 使用 async/ await 讓非同步程式更簡潔

Javascript 與非同步

Javascript

- 單執行緒
- 非阻塞式
- 非同步
- Run Everywhere

單執行緒

- 瀏覽器導向
- 避免複雜性
- 核心特徵
- Html 5 Web Worker 

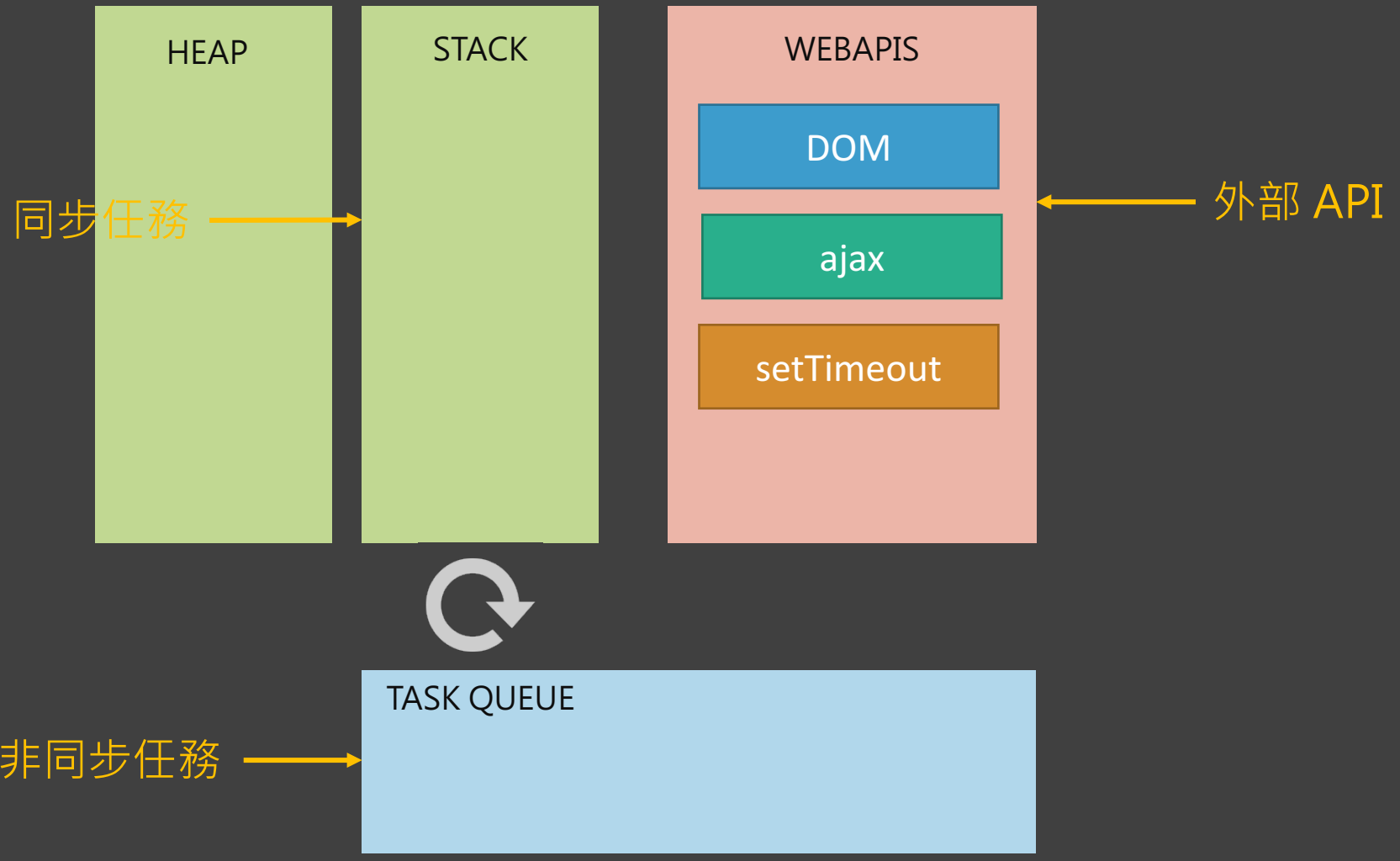
阻塞...

```
while(true) {
  console.log('Looping...')
}

console.log('Finished!')
```

永遠不會執行

https://codepen.io/kirkchen/pen/eEzVBe?editors=1010#0

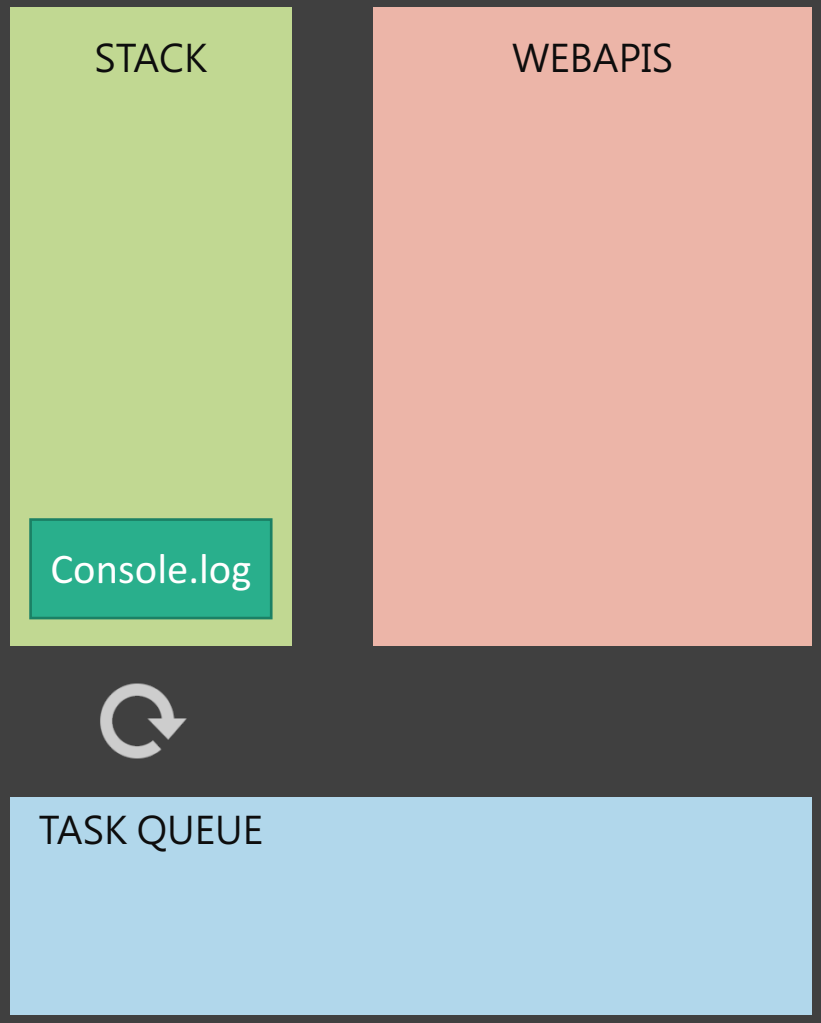
Event Loop


HEAP
STACK
WEBAPIS
DOM
ajax
setTimeout
同步任務
外部 API
TASK QUEUE
非同步任務

Event Loop

```
console.log('Hello');

setTimeout(function cb(){
  console.log('2017');
}, 2000);

console.log('ModernWeb')
```

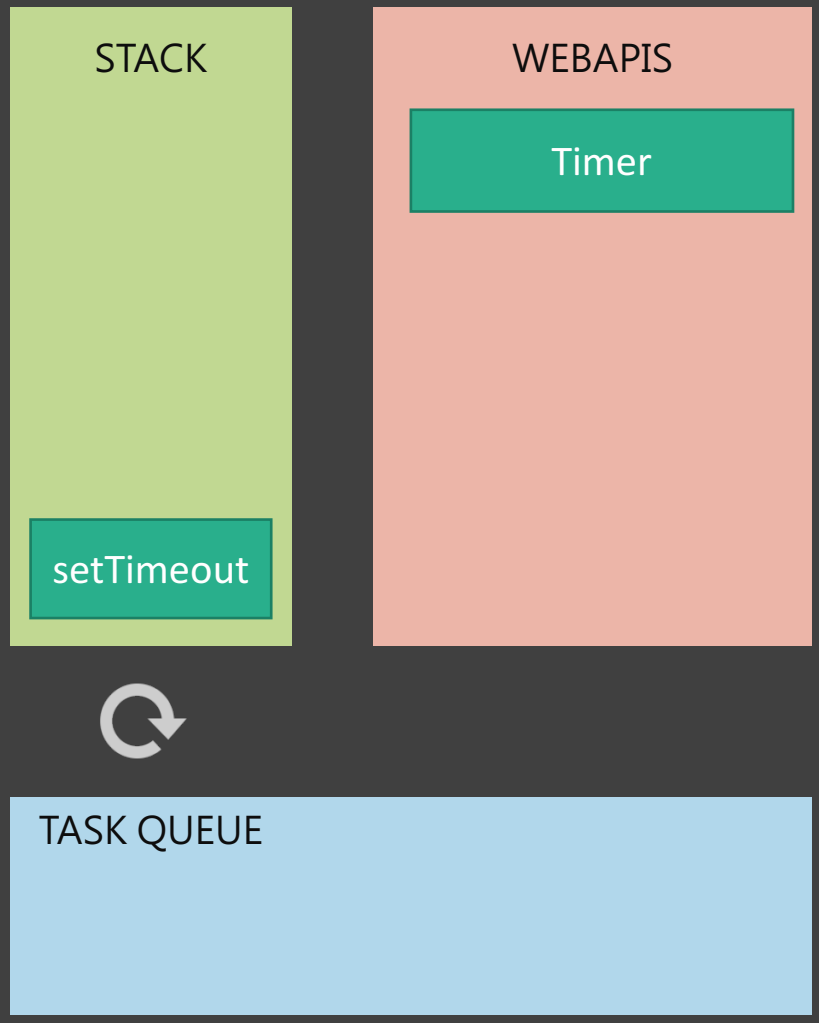
Hello



Event Loop

```
console.log('Hello');

setTimeout(function cb(){
  console.log('2017');
}, 2000);

console.log('ModernWeb')
```

Hello

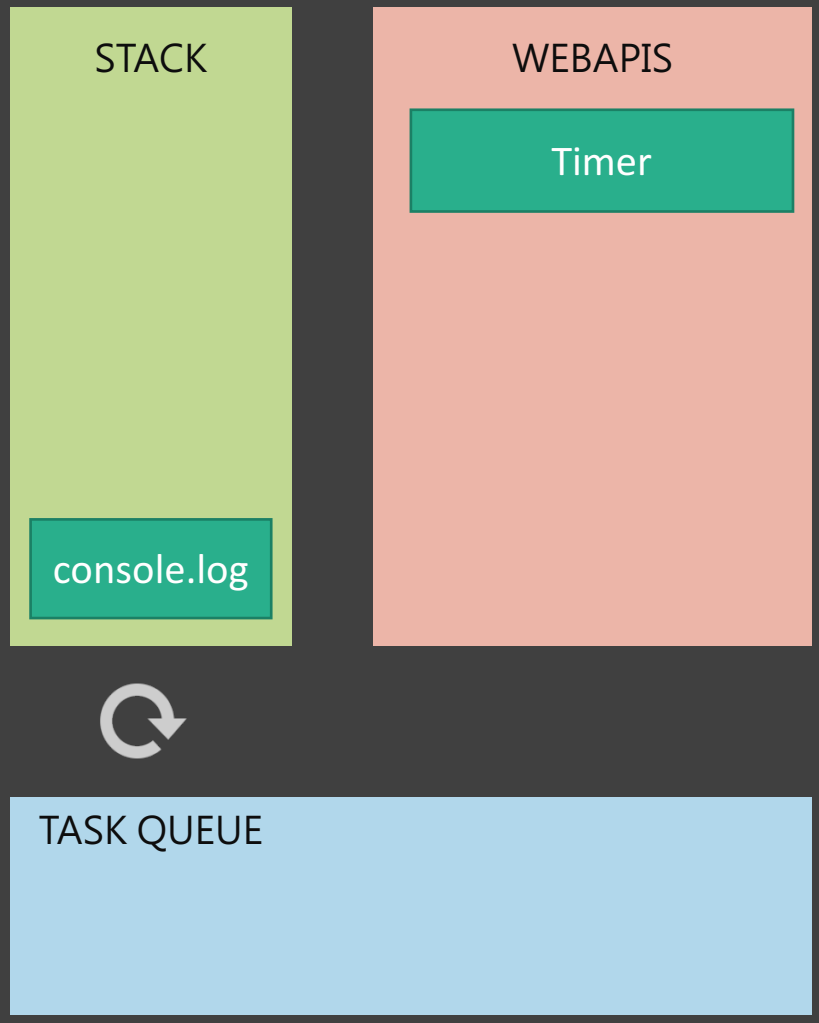


Event Loop

```
console.log('Hello');

setTimeout(function cb(){
  console.log('2017');
}, 2000);

console.log('ModernWeb')
```

Hello

ModernWeb



Event Loop

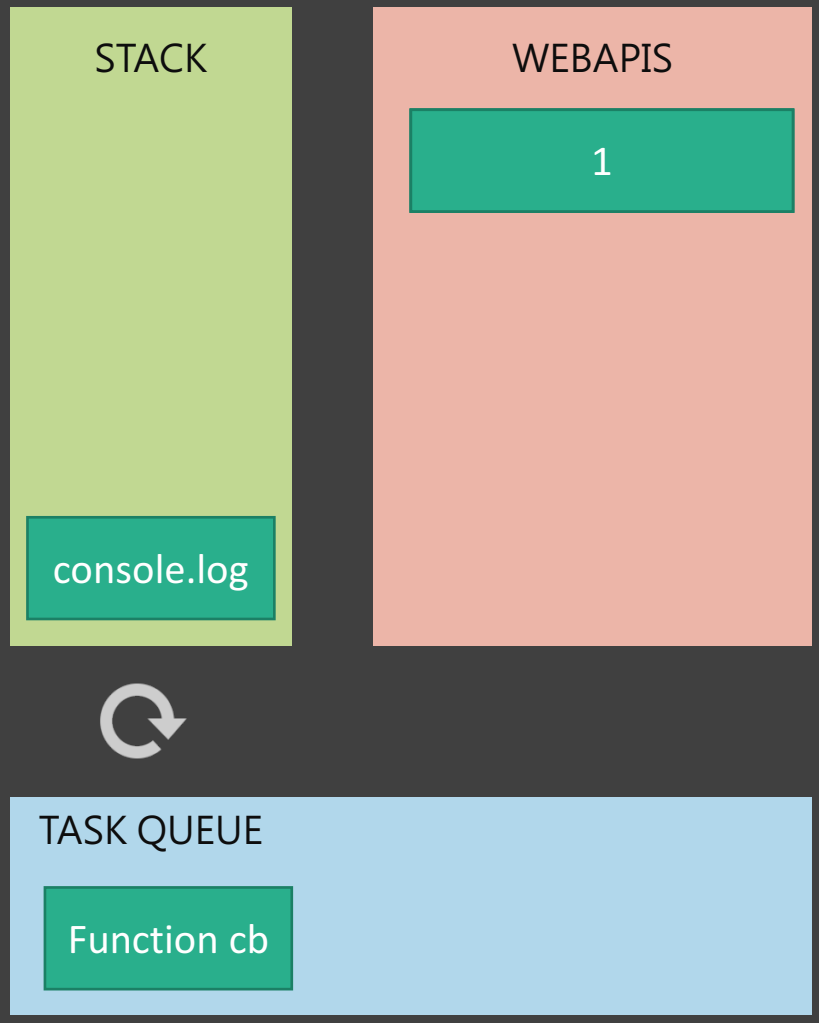
```
console.log('Hello');

setTimeout(function cb(){
  console.log('2017');
}, 2000);

console.log('ModernWeb')
```

Hello

ModernWeb

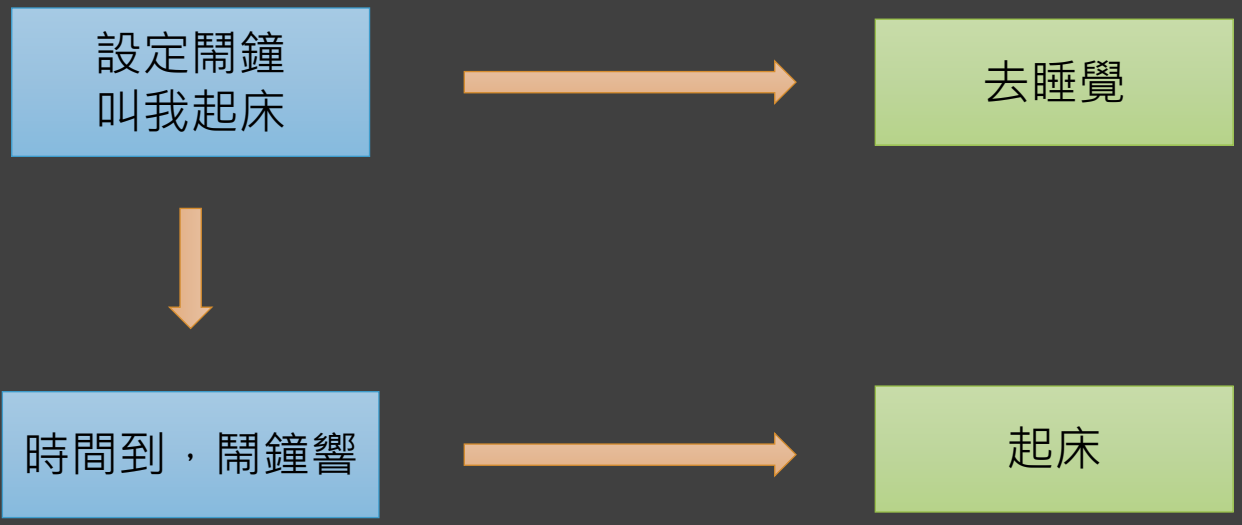
2017

http://latentflip.com/loupe/



callback vs promise

What is callback?

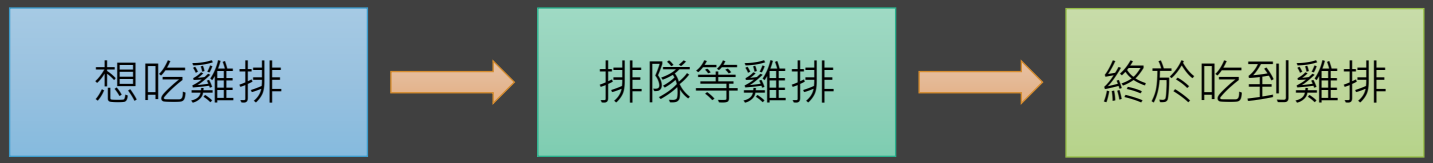


Callback

- 非同步
- 滿足情境時觸發
- 複雜邏輯不好維護
- 較難處理錯誤

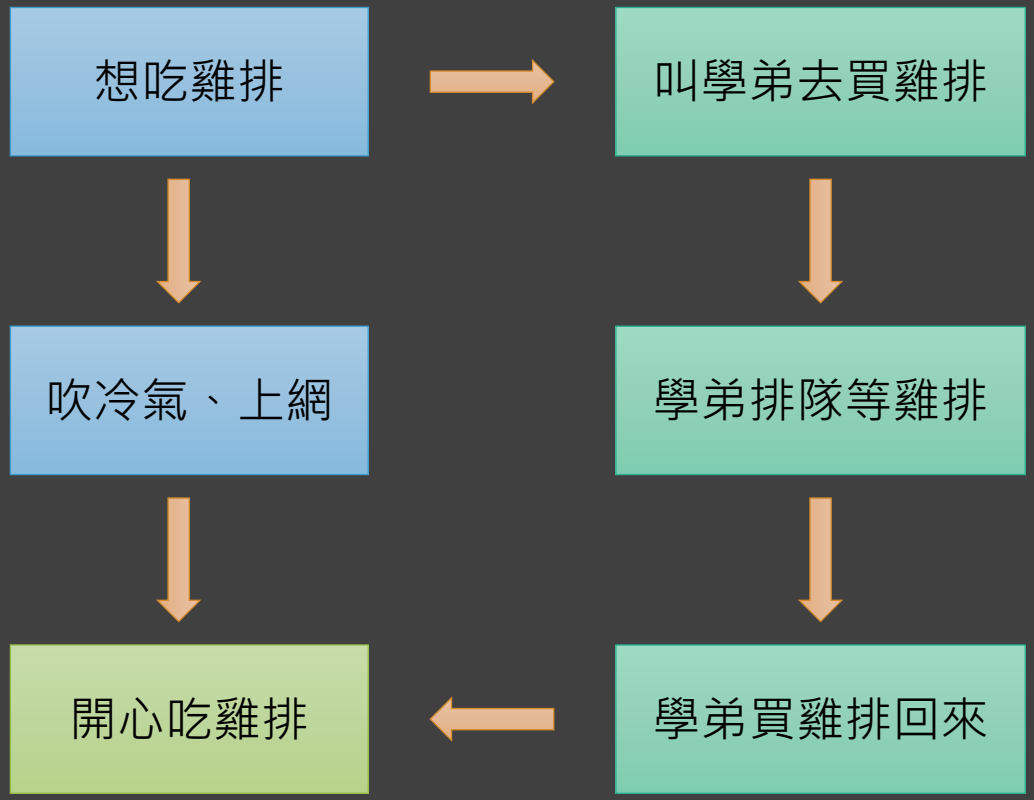
```
setTimeout(
  function cb(){
    console.log('2017');
  },
  2000
);
```

What is promise?

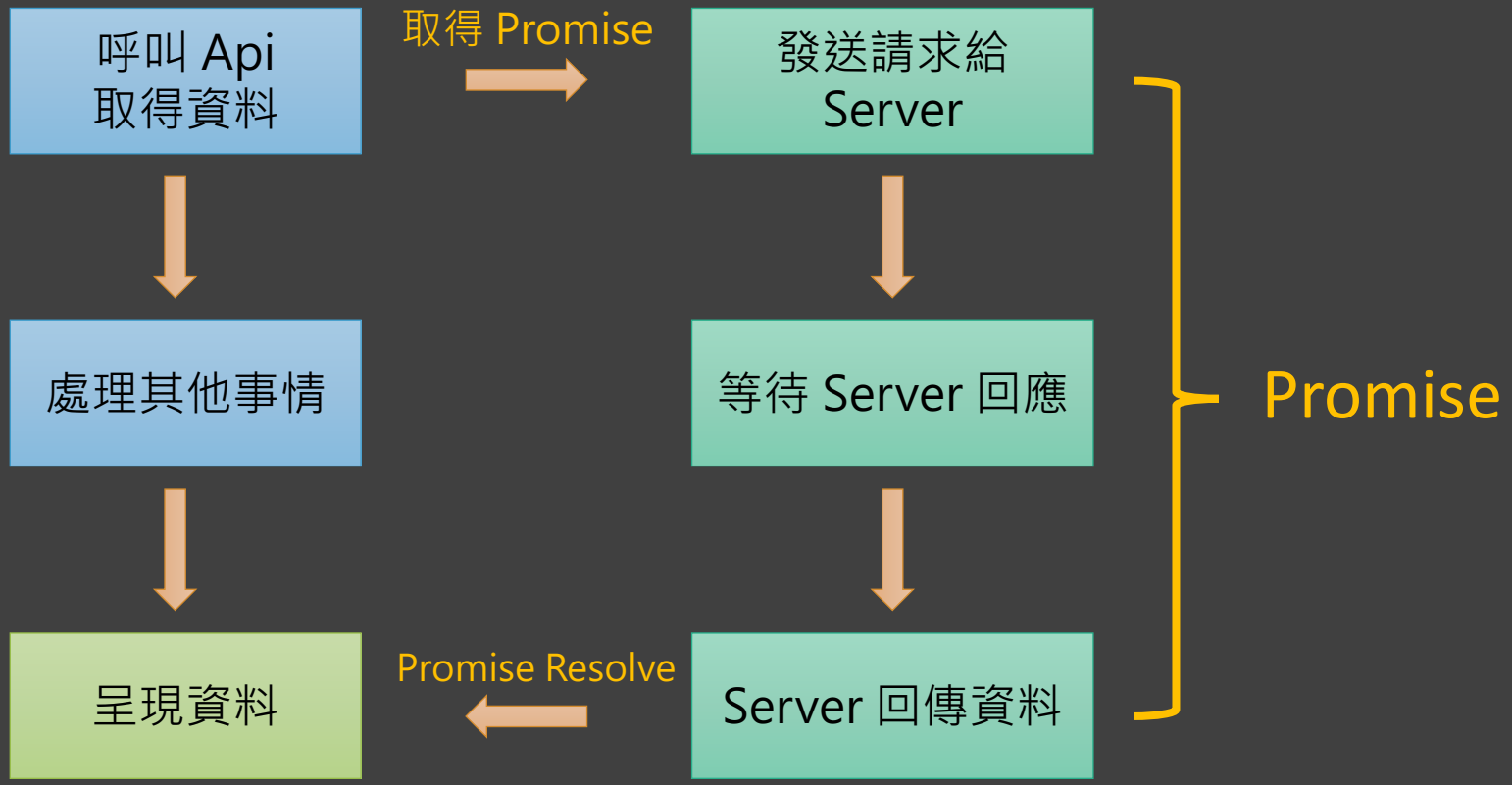


等了 30 分鐘，又熱又累...

What is promise?



What is promise?



Promise

- 兼容性
- 狀態明確
- 可以連續執行
- 更好閱讀

```
var promise = $.get('http://api.getdata');

promise.then(
  function success(data) {
    console.log(data);
  },
  function(err){
    console.log(err)
  }
);
```

Success

Failure

genertor 的出現及設計精神

Iterator

```
function GetDataList() {
  var dataList = [];
  var i = 0;

  while (i < 10000) {
    dataList.push(i);
    i++;
  }

  return dataList
}

var dataList = GetDataList();
for(var data of dataList){
  console.log(data)
}
```

- 佔記憶體空間
- 記憶體讀寫花時間
- 處理 10000 個物件

Generator

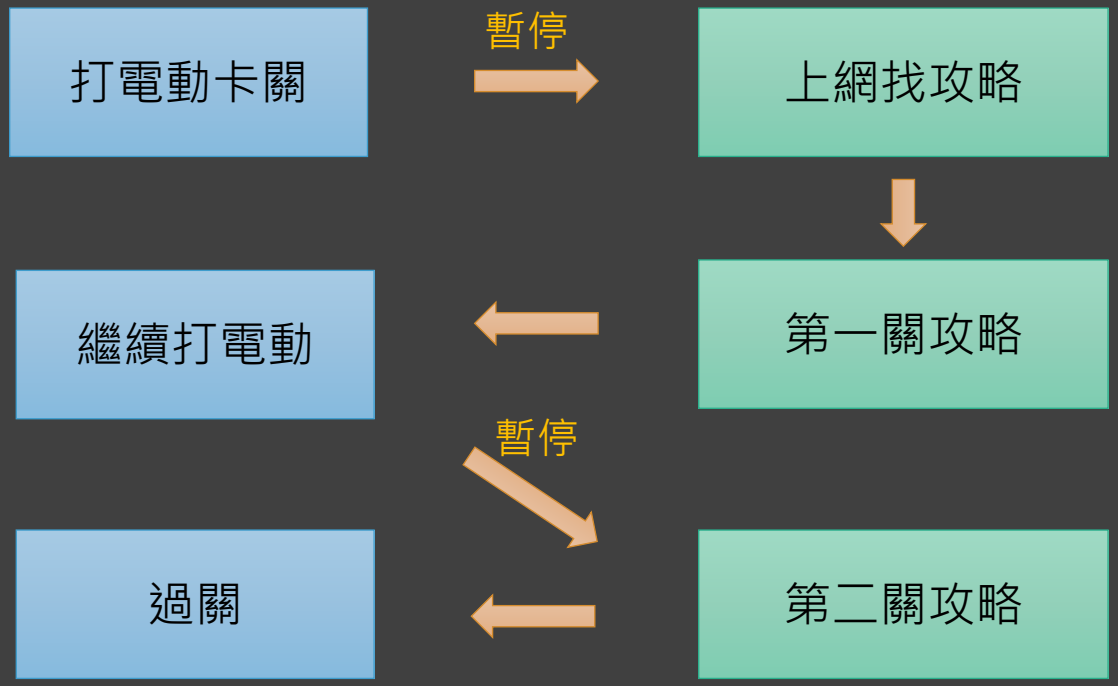
```
function* GetDataList() {
  var i = 0;

  while (i < 100) {
    yield i;
    i++;
  }
}
```

- 一次一個
- 停在上次執行之處
- 節省記憶體

```
var dataList = GetDataList();
console.log(dataList.next().value);
console.log(dataList.next().value);
console.log(dataList.next().value)
```

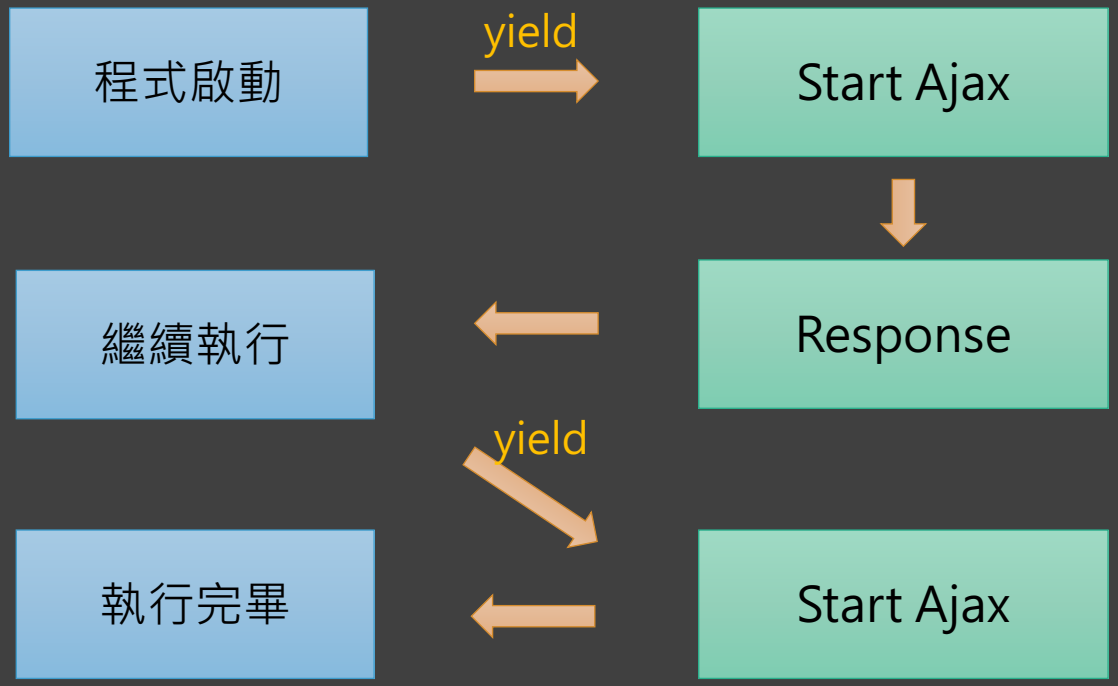
Generator

- 打電動，看攻略



Generator

- 呼叫 Api



Generator

```
co(function* (){
  const a = yield request('api.com/a');
  const b = yield request('api.com/b');

  console.log(a);
  console.log(b);
})
```

- 暫時交出執行權
- 封裝非同步任務，處理 Promise
- 邏輯清楚好理解

Generator

```
co(function* (){
  const a = yield request('api.com/a');
  const b = yield request('api.com/b');

  console.log(a);
  console.log(b);
})
```

- 讓 Generator 自動執行
- 回傳為 Promise

使用 async/ await
讓非同步程式更簡潔

async / await

- 結合 Promise 與 Generator
- 不需要 第三方套件
- 非同步程式同步化
- 直覺化的邏輯思考
- ES7, TypeScript 支援

async / await

```
co(function* (){
   const a = yield request('api.com/a');
   const b = yield request('api.com/b');

   console.log(a);
   console.log(b);
})
```

async / await

```
async function Main(){
   const a = await request('api.com/a');
   const b = await request('api.com/b');

   console.log(a);
   console.log(b);
})
```

async / await

- async – 標住在 function 前面
- await – 標住在 promise 前面
- 一定要用 function 包裝
- 回傳為 Promise 類型

回顧

callback

```
var a, b;
request('api.com/a', function(err, result){
  if(err) {
    // 錯誤處理
  }
  a = result;

  request('api.com/b', function(err, result){
    if(err){
      // 錯誤處理
    }
    b = result;

    console.log(a);
    console.log(b);

  })
}
```

promise

```
var a, b;
request('api.com/a')
.then(function(result){
  a = result;
})
.then(function(){
  return request('api.com/b');
})
.then(function(result){
  b = result;

  console.log(a);
  console.log(b);
})
```

generator

```
co(function* (){
  const a = yield request('api.com/a');
  const b = yield request('api.com/b');

  console.log(a);
  console.log(b);
})
```

async / await

```
async function Main(){
  const a = await request('api.com/a');
  const b = await request('api.com/b');

  console.log(a);
  console.log(b);
})
```

總結

- 善用非同步
- 學習新寫法
- 理解運作機制
- 節省開發時間

Reference

- Philip Roberts: Help, I’ m stuck in an event-loop.
- JavaScript 運行機制詳解：再談Event Loop
- Going Async With ES6 Generators
- JavaScript 非同步程式革命 - async、await 與 TypeScript 2.1

We're Hiring

jobs.kktix.cc

謝 謝 大 家