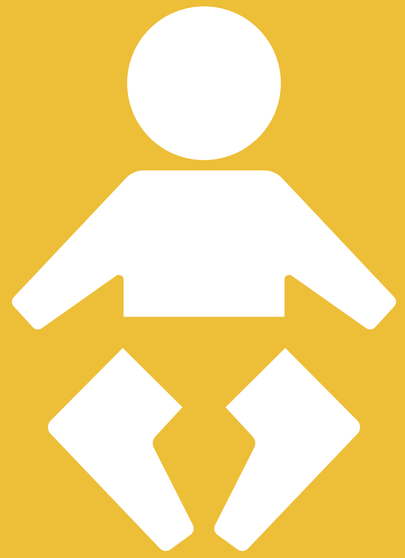
Javascript asynchronous 1

(callback, promise, async/await)

ASYNCHRONOUS MAIN METHOD

Main method :

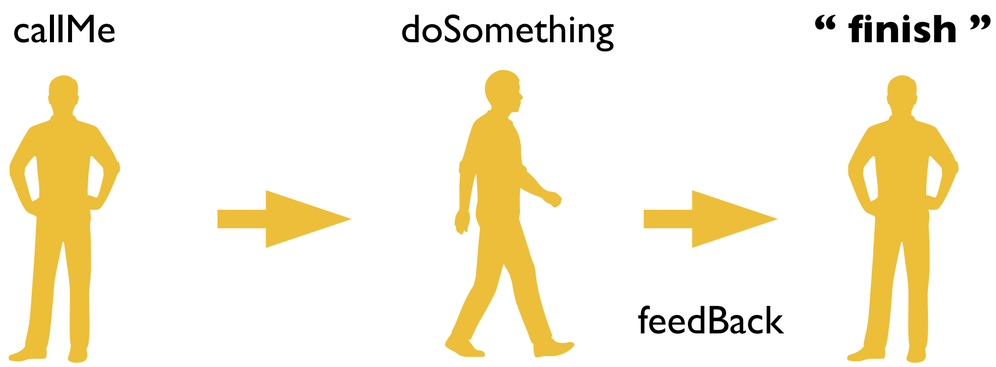
- CallBack
- Promise
- Async / Await (ES7)



Simple Callback

最早被提出，屬於最單純最直接的非同步回呼方式

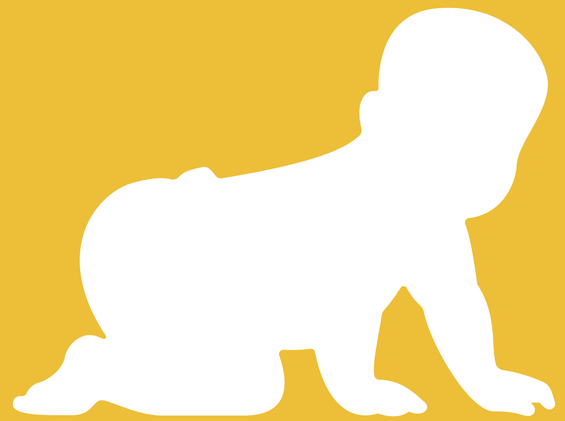
```
Function callMe(do, feedBack) {
    doSomething.done(finish(){
        feedBack("finish");
    })
}

callMe("eat Dinner", function(feedBack){
    console.dir(feedBack);
});
```



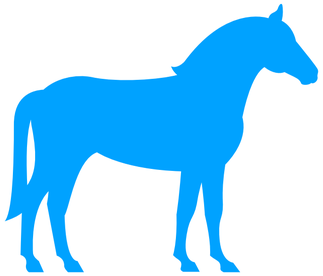
CallBack 最容易遇到的問題 (Callback Hell)

Ex:
```
step1 => function(value1) =>
step2 => function(value2) =>
step3 => function(value3)
```

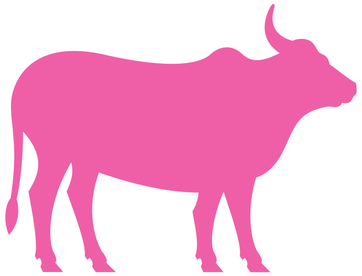
```
step1(x, function(value1){
  //do something...
  step2(y, function(value2){
    //do something...
    step3(z, function(value3){
        //do something...
    })
  })
})
```



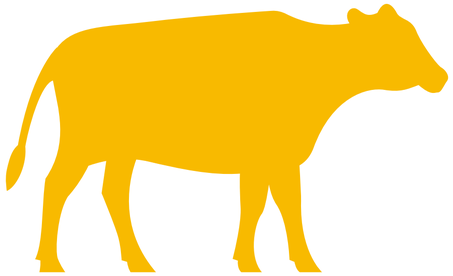
Walking Promise



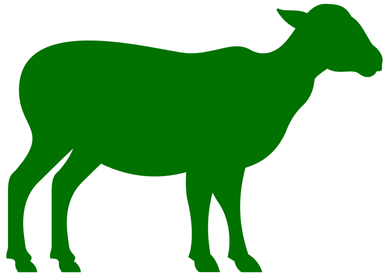
jQuery



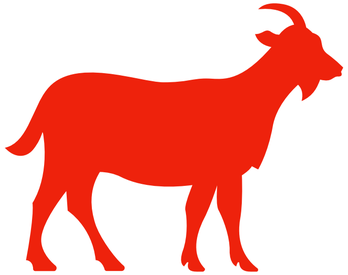
YUI



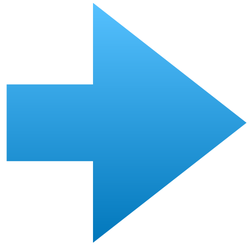
Ember



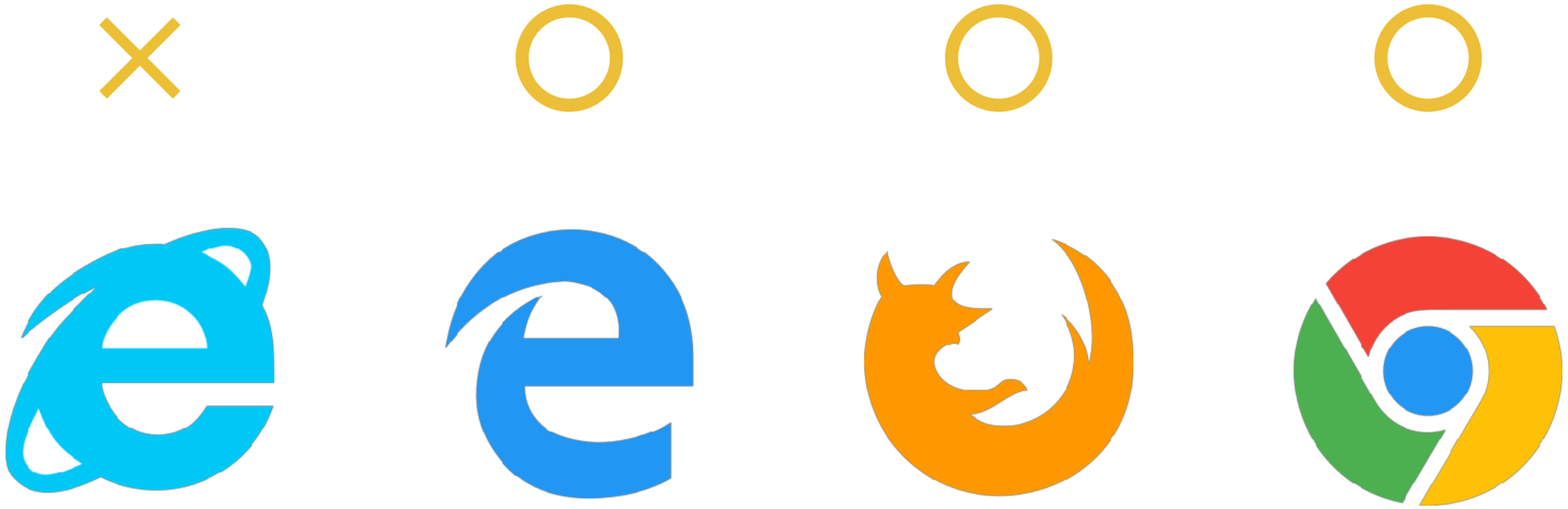
Angular



WinJS



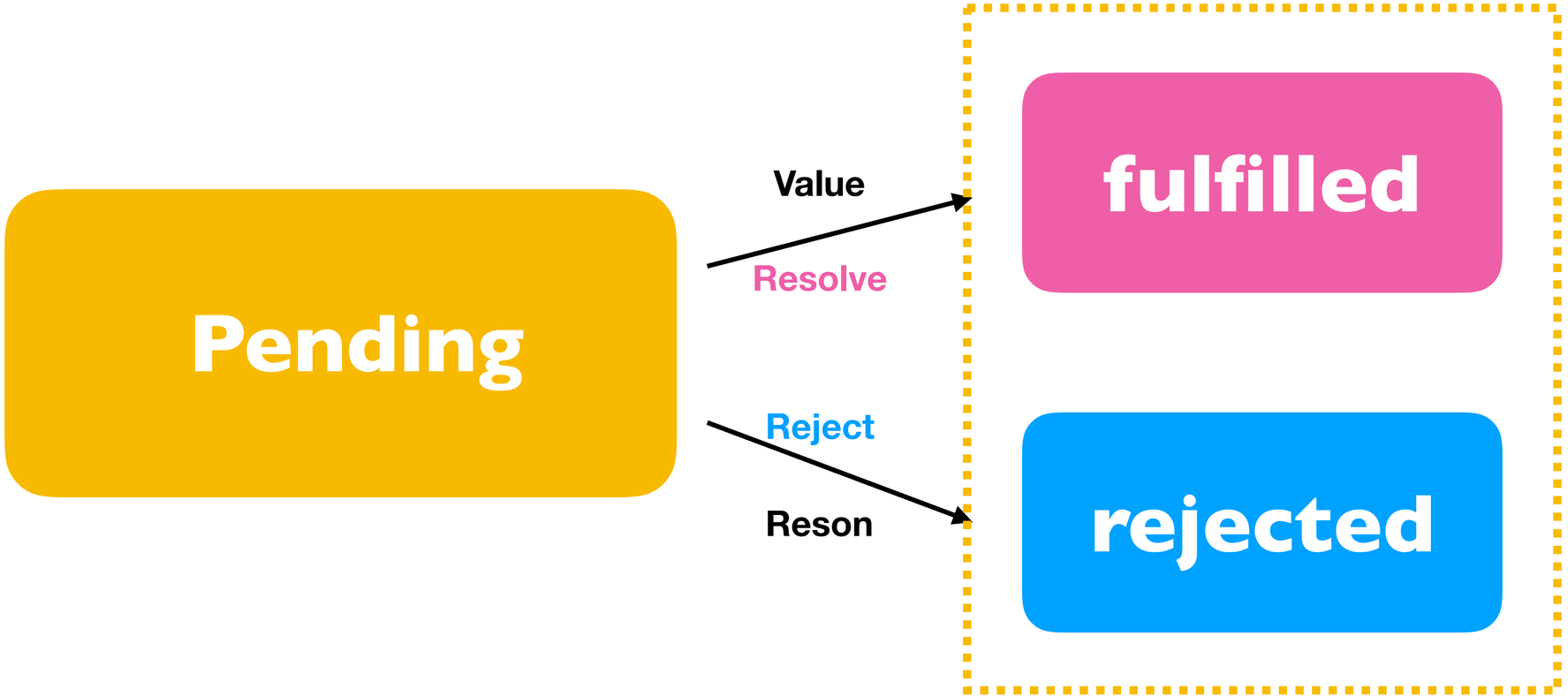
ES6 Promise


IE
Edge
FireFox
Chrome

Promise main method：

- resolve
- reject
- then
- catch
- all

Promise 流程：



Promise.then

Promise.then(onFulfilled, onRejected)

- **onFulfilled** => 當Promise (resolve) 成功後 onFulfilled 會被調用
- **onRejected** => 當Promise (reject) 失敗後 onRejected 會被調用

Resolve solution ex:

```
Promise.then(function(error){
    console.error(error);
}, null);
```

Reject solution ex:

```
Promise.then(null, function(error){
    console.error(error);
});
```

Promise.catch

Promise.catch(onRejected)

- **onRejected** => 當Promise (reject) 失敗後 onRejected 會被調用，並回傳一個新的 Promise 物件

Reject 實作 :

```
Promise.catch(function(reason) {})
```

=

```
Promise.then(undefined, function(reason) {})
```

Reject solution ex:

```
Promise.catch(function(reason) {
   // onRejected to do
})
```

Promise.all

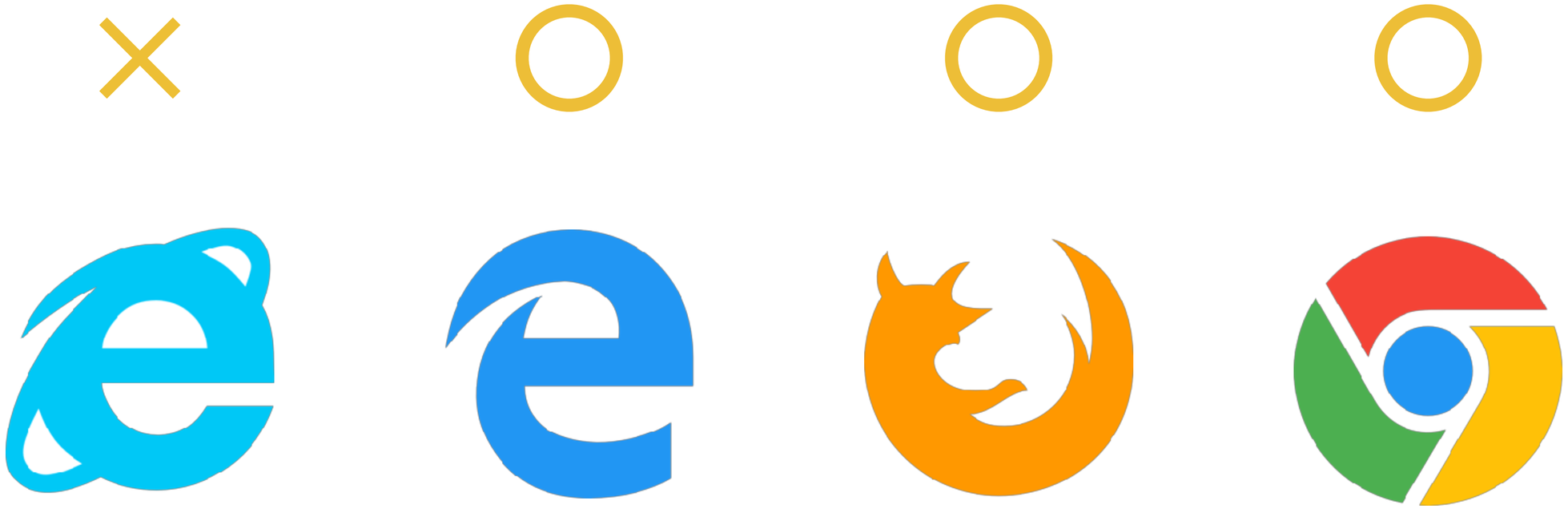
Promise.all(iterable)

- **iterable** => 為一個物件 (Array) 或者為一個字串 (String)

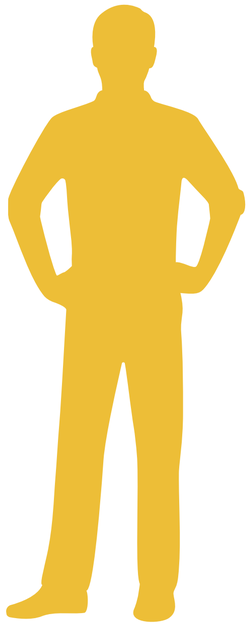
Promise.all 特點：

- 當有任意的一個錯誤或者全部完成後回傳
- 當有任意的錯誤，不管在哪個階段則會強制跳出
- 可傳入字串，字串會被解析

Async & Await


IE
Edge
FireFox
Chrome

Different ?



Mr. Async

Mr. Await

Async

async function name(... param) { // statements }

特點：

- 當 Async 被呼叫時則回傳一個Promise
- Async 為一個 new Promise 的語法糖
- 可以更快將 Function 定義為非同步的async function
- Promise 及 Async 可混用

Async vs Promise

Async vs **Promise**

Await

let {variable} = **await** expression;

特點：

- 當出現 Await 時則會等待該事件完成解析才會回傳值
- Await 必須執行於 Async 環境中
- Await 錯誤會自動拋出錯誤
- Promise 及 Await 可混用

Async Await & Promise Different

區別：

- 使用的時間點不同
- Async 及 Await 其實作皆來自 Promise
- 兩者皆可混用
- Async 可解省程式碼數量
- ES6 及 ES7 版本區別
- 瀏覽器目前支援度不同

THANK YOU