Web Programming Spring 2021

#14

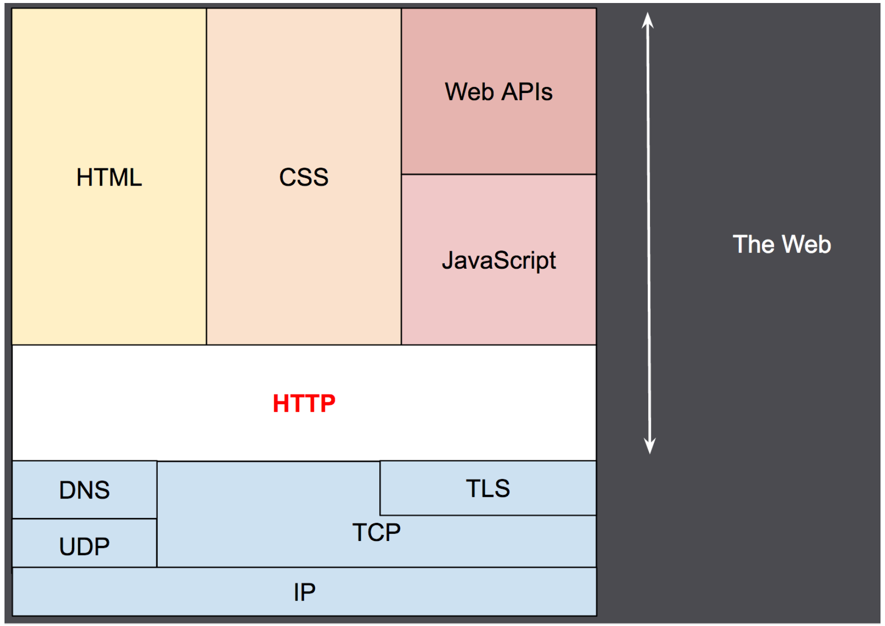
Chi-Jen Wu

Topics

- The concepts of Web Services
- Web data protocols
  - HTTP, WebSocket, WebRTC
  - HTML, CSS
- **Web JavaScript programming**
- Cookies and sessions
- Web Frontend frameworks
- Web Backend frameworks
- RESTful API design

Web data protocols

- HTTP, HTTPS
- Web APIs
- HTML, HTML5
- CSS, CSS3
- **JavaScript**
- Conclusion



JavaScript

- Introduction
- Basics
- Document Object Model
- Browser Object Model
- jQuery & AJAX
- JavaScript ES6

Ajax

- Introduction
- XMLHttpRequest
- Request
- Response
- <u>Promises</u>
- <u>Async/Await</u>
- <u>AJAX Examples</u>

AJAX
Asynchronous Javascript And XML

Promises

- Introduction
- Asynchronous JavaScript
- Promise Object

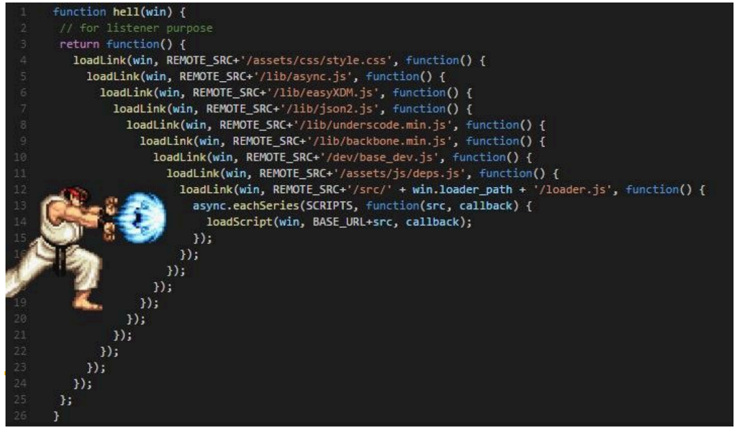
Promises Introduction

- JavaScript Promise Object
- 執行非同步（API request，等待使用者點擊）的流程時，因為**不知道什麼時候會完成**，通常會接受一個**callback function**作為參數，完成會呼叫此callback function以執行下一步。

Event <—> callback

多個非同步工作要做時

```
<script>
asyncA(function(dataA) {
  asyncB(dataA, function(dataB) {
    asyncC(dataB, function() {
      ...
    })
  })
})
</script>
```



```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

Promises 可以

```
<script>
asyncA()
   .then(asyncB)
   .then(asyncC)
   .catch() // Error Handling
</script>
```

有沒有覺得很好看！！

```
<script>
asyncA()
   .then(asyncB)
   .then(asyncC)
   .catch()
</script>
```

```
<script>
var data = $.getJSON(dataUrl);

data.done( function( msg ) {
// just do it
});

data.fail( function( msg ) {
// just do it
});
</script>
```

有沒有覺得和jQuery $.get()有點像
之後 react 也差不多是這樣

Promises History

- 並行程式語言中同步程式執行的**構造**
- 1977年 提出概念
- 源自於 函式語言程式設計 functional programming
- 她發明的 (1988) $\longrightarrow$

芭芭拉·利斯科夫, MIT

jQuery 1.5 之後

- 實作了 Promises
- 就是說 也有別人實作 Promises
- 所以 大家實作的 Promises 有可能不一樣
- 但是先不要管他們不一樣
- 因為等你發現他們不一樣
- 你已經是高手 高高手 現在先會用他來寫

Asynchronous JavaScript

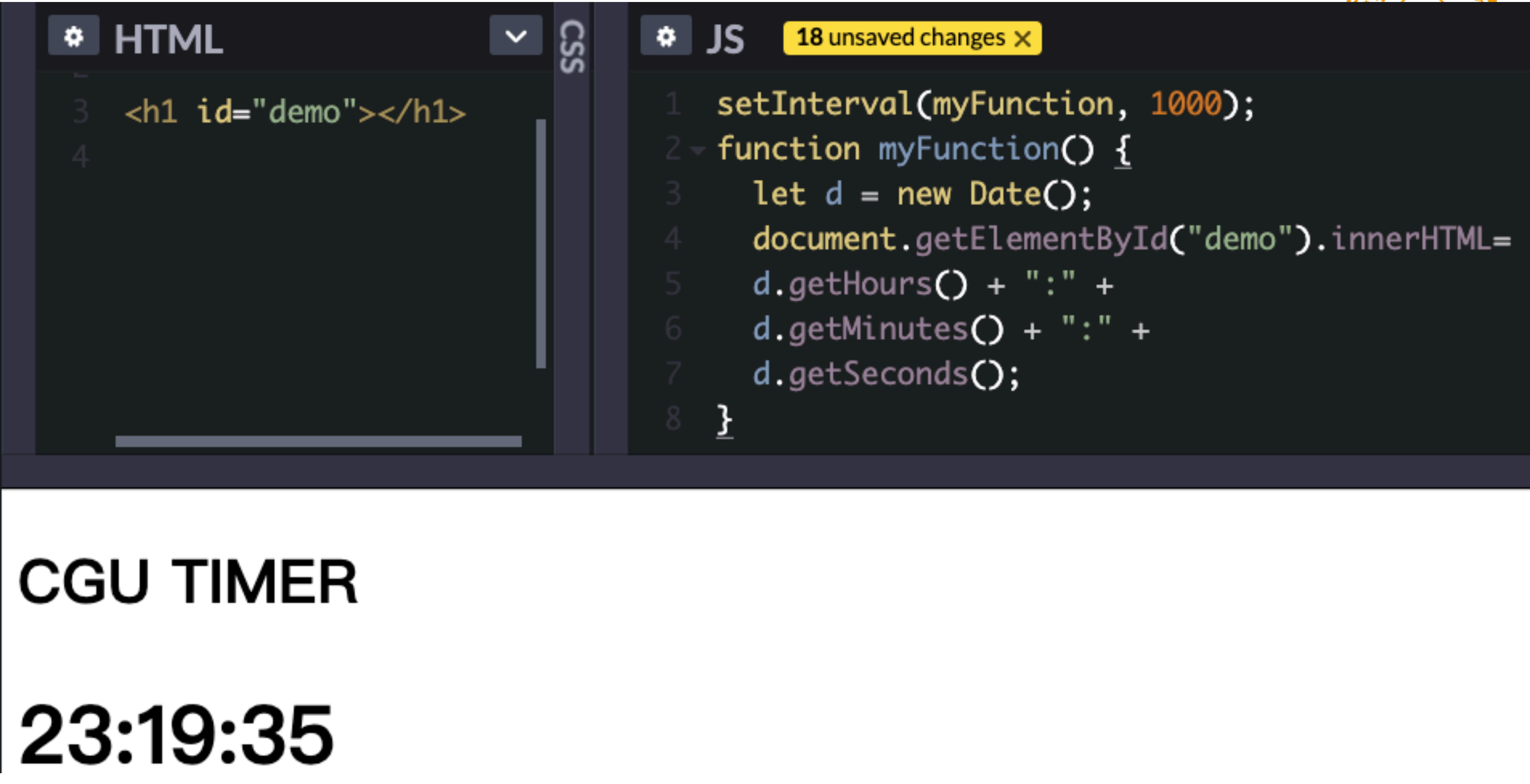
- JavaScript是synchronous在執行的
- JavaScript如何執行非同步事件
  - asynchronous callback
  - event queue

OS kernel裡面也一樣

- EventListener
- setInterval

Asynchronous JavaScript

- EventListener
  - 等待IO Events
- setInterval
  - 沒有event 要自己觸發時
  - 例如 寫一個小時鐘



```html
<h1 id="demo"></h1>
```

```javascript
setInterval(myFunction, 1000);
function myFunction() {
  let d = new Date();
  document.getElementById("demo").innerHTML=
  d.getHours() + ":" +
  d.getMinutes() + ":" +
  d.getSeconds();
}
```

CGU TIMER

23:19:35

JavaScript Scheduling

- setTimeout
- setInterval
- clearInterval

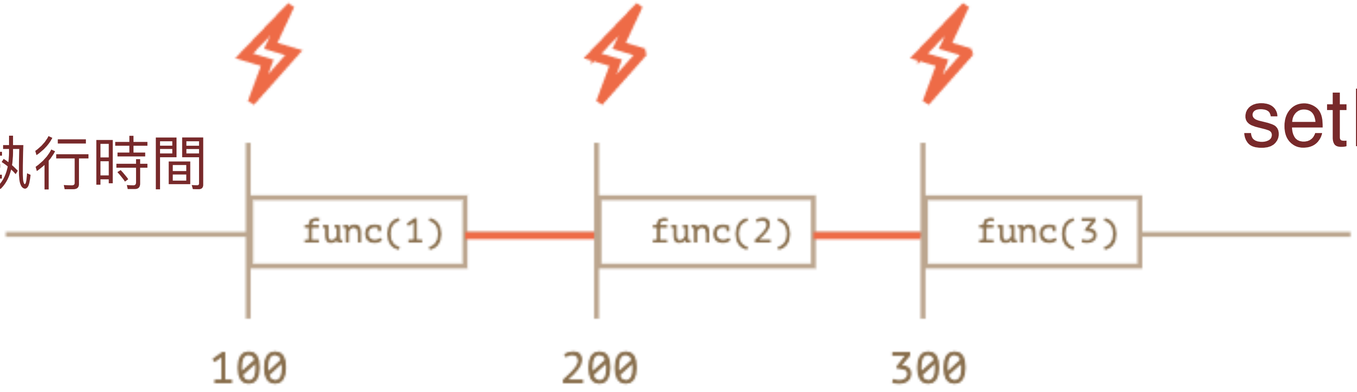
```
<script>
function sayHi() {
  alert('Hello');
}

setTimeout(sayHi, 1000);
</script>
```

setTimeout VS setInterval

```
<script>
let i = 1;
setTimeout(function
run() {
   func(i++);
   setTimeout(run, 100);
}, 100);
</script>
```

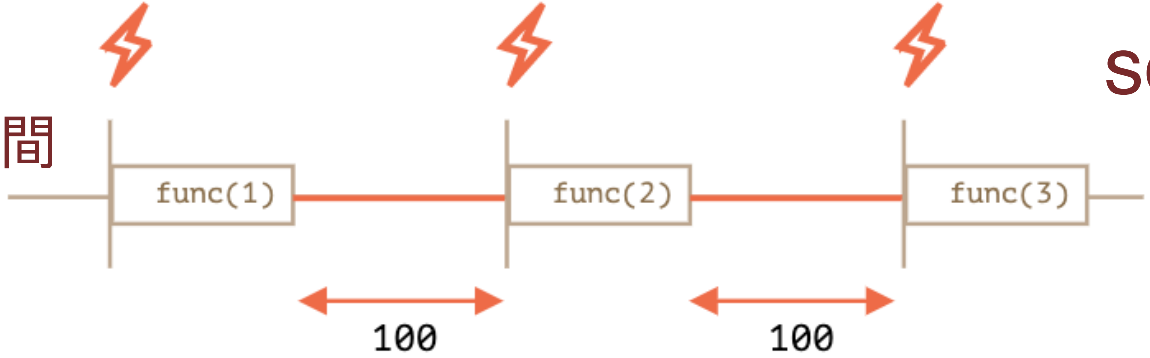
```
<script>
let i = 1;
setInterval(function() {
  func(i++);
}, 100);
</script>
```

setTimeout VS setInterval



setInterval

func 執行時間



setTimeout

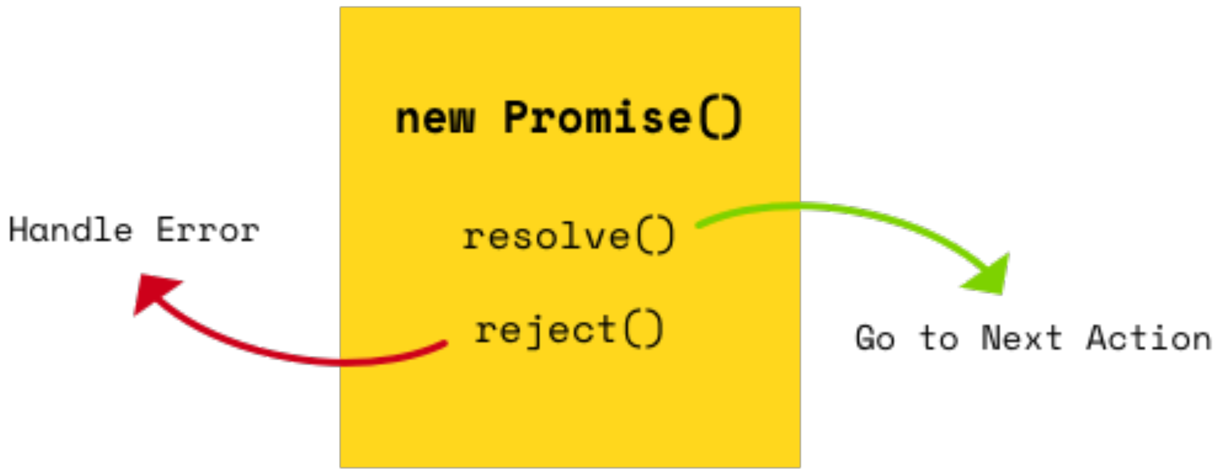
func 執行時間

那是啥？？

Promise Object

```
<script>
let myPromise = new Promise(function(myResolve, myReject) {
// "Producing Code" (May take some time)

  myResolve(); // when successful
  myReject();  // when error
});

// "Consuming Code" (Must wait for a fulfilled Promise)
myPromise.then(
  function(value) { /* code if successful */ },
  function(error) { /* code if some error */ }
);
</script>
```

Promise Object



Promise vs callback

`setTimeout`

```
<script>
setTimeout(function() { myFunction("I love You !!!");
}, 3000);

function myFunction(value) {
  document.getElementById("demo").innerHTML = value;
}
</script>
```

Promise vs callback

```
Promise

<script>
let myPromise = new Promise(function(myResolve, myReject) {
  setTimeout(function() { myResolve("I love You !!"); }, 3000);
});

myPromise.then(function(value) {
  document.getElementById("demo").innerHTML = value;
});
</script>
```

Promise 缺點

- 無法取消Promise
- 不太好debug
- 沒寫好 function(myResolve, myReject)
  - 不知道那裡出錯
  - 所以要非常小心
- 真正 Promise 會比剛剛說得還要複雜很多
  - 牽扯到很多函式語言程式觀念 (高階應用)
  - 在此不多介紹 有這個概念就好

Conclusion

- Promise
- Introduction
- Asynchronous JavaScript
- Promise Object

ECMAScript 6

Promise

Thanks!

Open for any questions

CJ Wu

cjwu@mail.cgu.edu.tw