Web Programming Spring 2021 #11

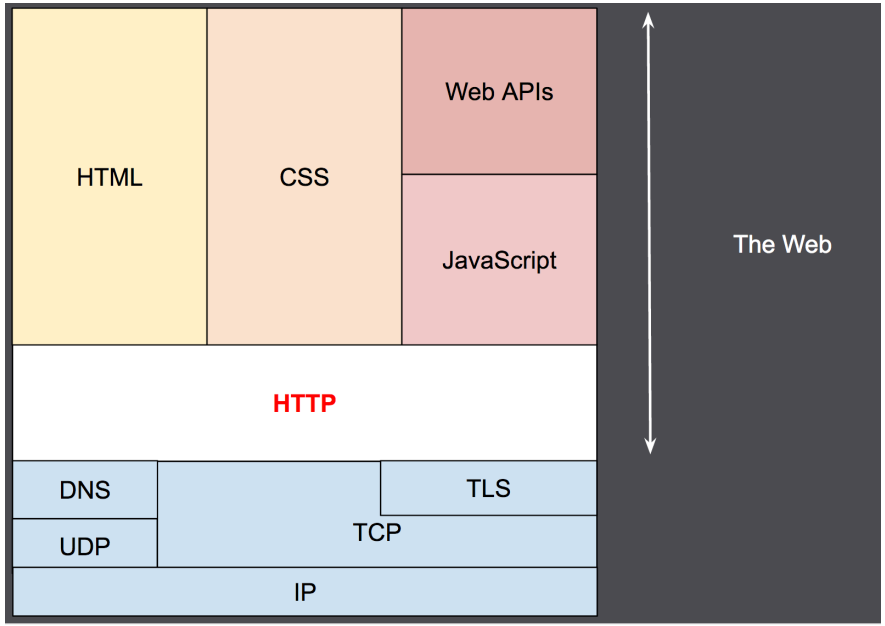
Chi-Jen Wu

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
- Promises
- Async/Await
- AJAX Examples

Ajax Introduction

- Asynchronous JavaScript and XML
- (非同步的 JavaScript 與 XML 技術)
- Google Map 2005年 開始使用
- 其實1999年微軟就發明了類似ajax的技術 (ActiveX)
- 網頁不用重新整理，就能即時地透過瀏覽器去跟伺服器溝通，撈出資料

原本HTML和Server 溝通方式

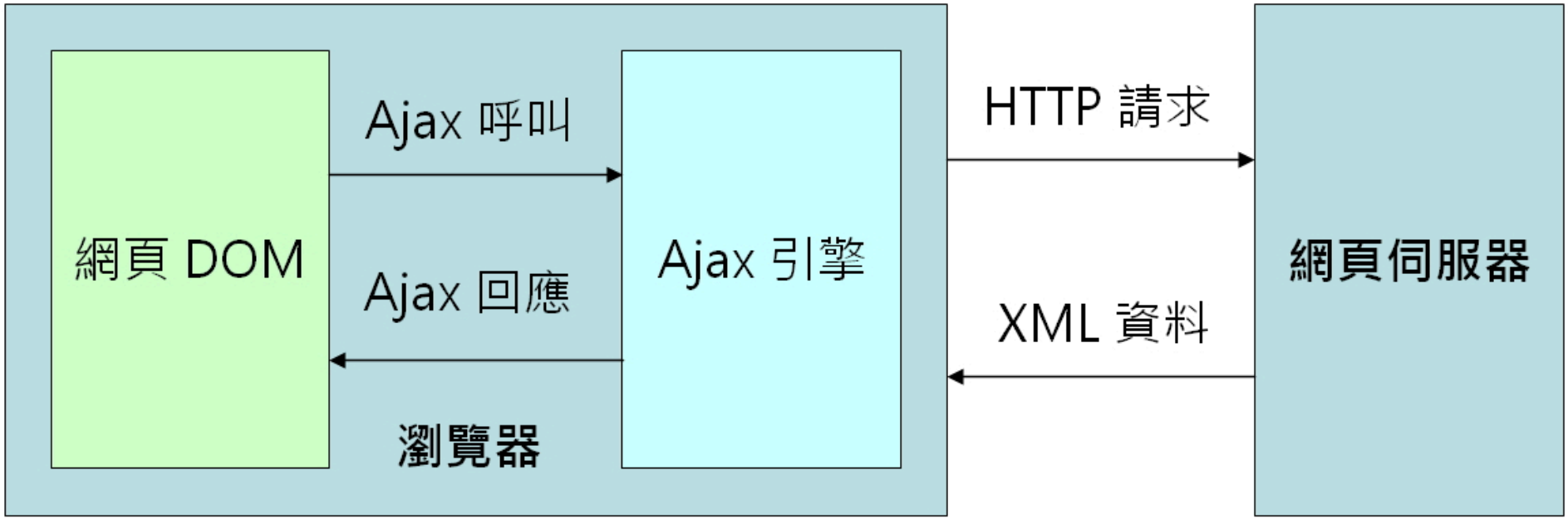
- <From> 發送get, post 請求 刷新頁面或是新開頁面
- <a> 發送 get 請求刷新頁面或是新開頁面
- <img> get 請求 image
- <link> get 請求 css, favicon
- <script> get 請求javascript
- **Ajax 可以發送 GET, POST, PUT, DELETE (HTTP methods)**

HTTP 請求方法

Ajax Why

- 使用者體驗大幅度提升
  - 因為非同步存取
  - 不用下載全部檔案到Browser
  - 轉嫁Server工作量到Browser端
- 桌面軟體過度到網頁軟體
  - 桌遊 vs 頁遊
  - Outlook vs gmail

Ajax 請求流程



Ajax 整合

- CSS和HTML呈現網頁
- DOM 存取網頁元素
- XMLHttpRequest和Server 異步通訊
- Javascript 控制協調
- **目前網站設計都是用Ajax的思考方式**

Ajax 缺點

- Back (上一頁)機制被破壞
  - 但是有解法
- 安全問題
- 行動裝置支援
  - Different browsers implement the Ajax API differently
- Cross-domain problem
- 和原本URL 定位資源的想法衝突

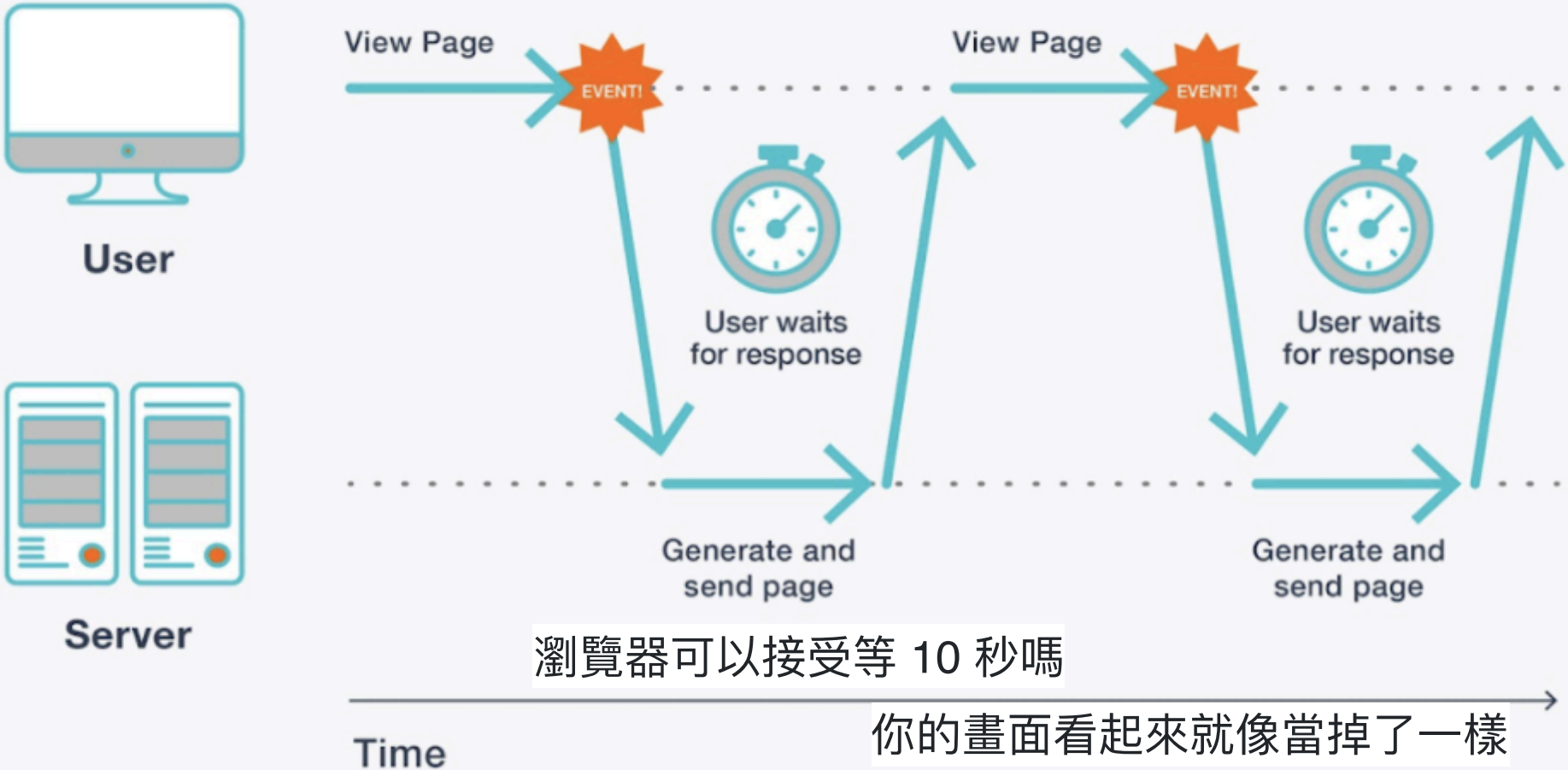
XMLHttpRequest (XHR)

- JavaScript HTTP Client
- 利用XMLHttpRequest發出請求
- Server 返回 XML格式字串 (目前大都返回JSON, protocolbuffer, gRPC)
- JavaScript 解析XML
- JavaScript 更新局部頁面
- 以非同步方式
- 非同步/異步 是指執行請求不會卡住
  - 等待 Server 回應，阻斷 (block) 執行

同步請求 v.s. 非同步請求

- 同步請求 (Synchronous request)：
  - 客戶端 (client) 對伺服器端 (server) 送出 request ，並且在收到伺服器端的 response 之後才會繼續下一步的動作，等待的期間無法處理其他事情。這個作法並不理想，因為通常伺服器端的運算速度比本地電腦慢上好幾倍。
- 非同步請求 (Asynchronous request)：
  - 客戶端 (client) 對伺服器端 (server) 送出 request 之後，不需要等待結果，仍可以持續處理其他事情，甚至繼續送出其他 request。Responese 傳回之後，就被融合進當下頁面或應用中。

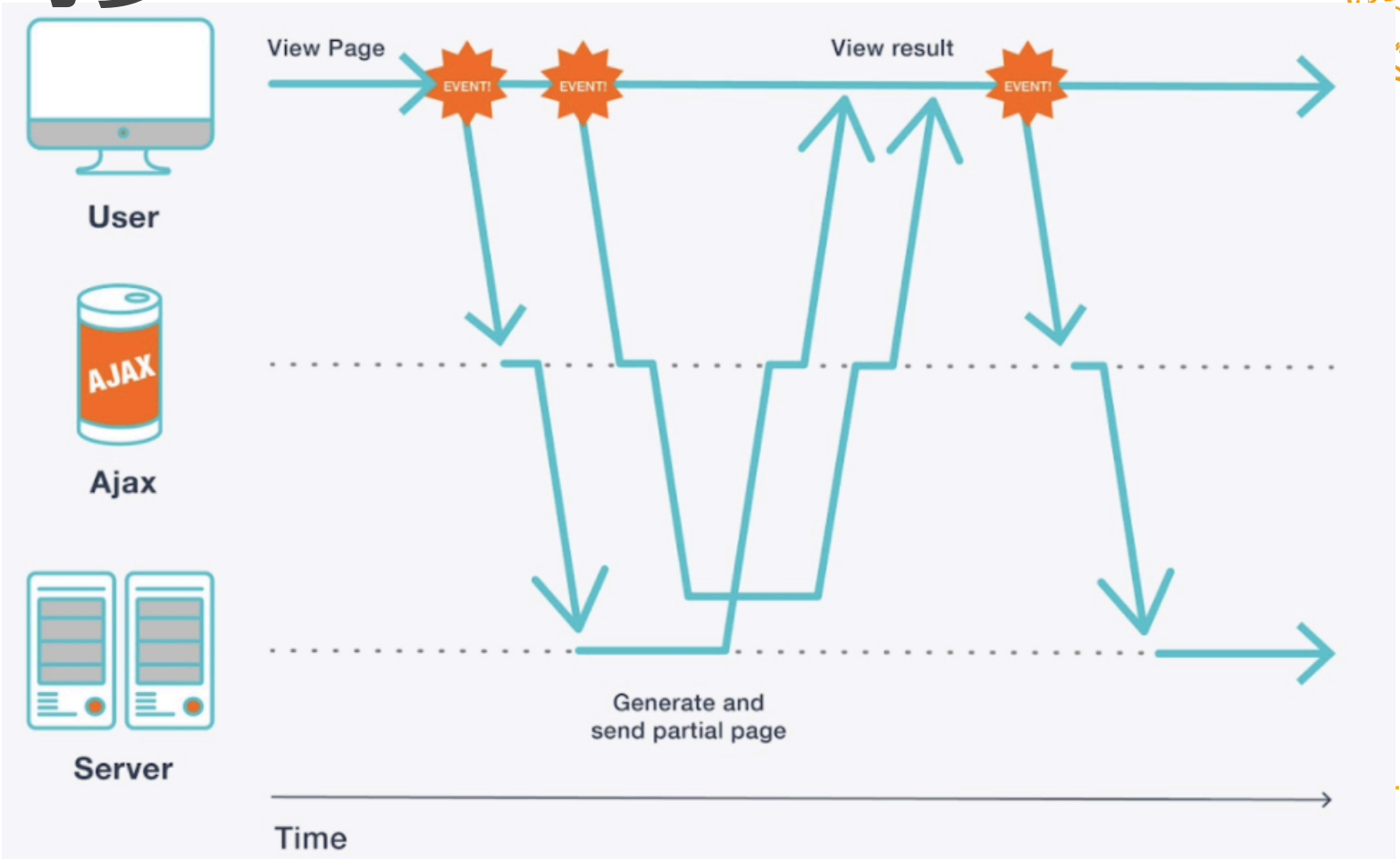
同步

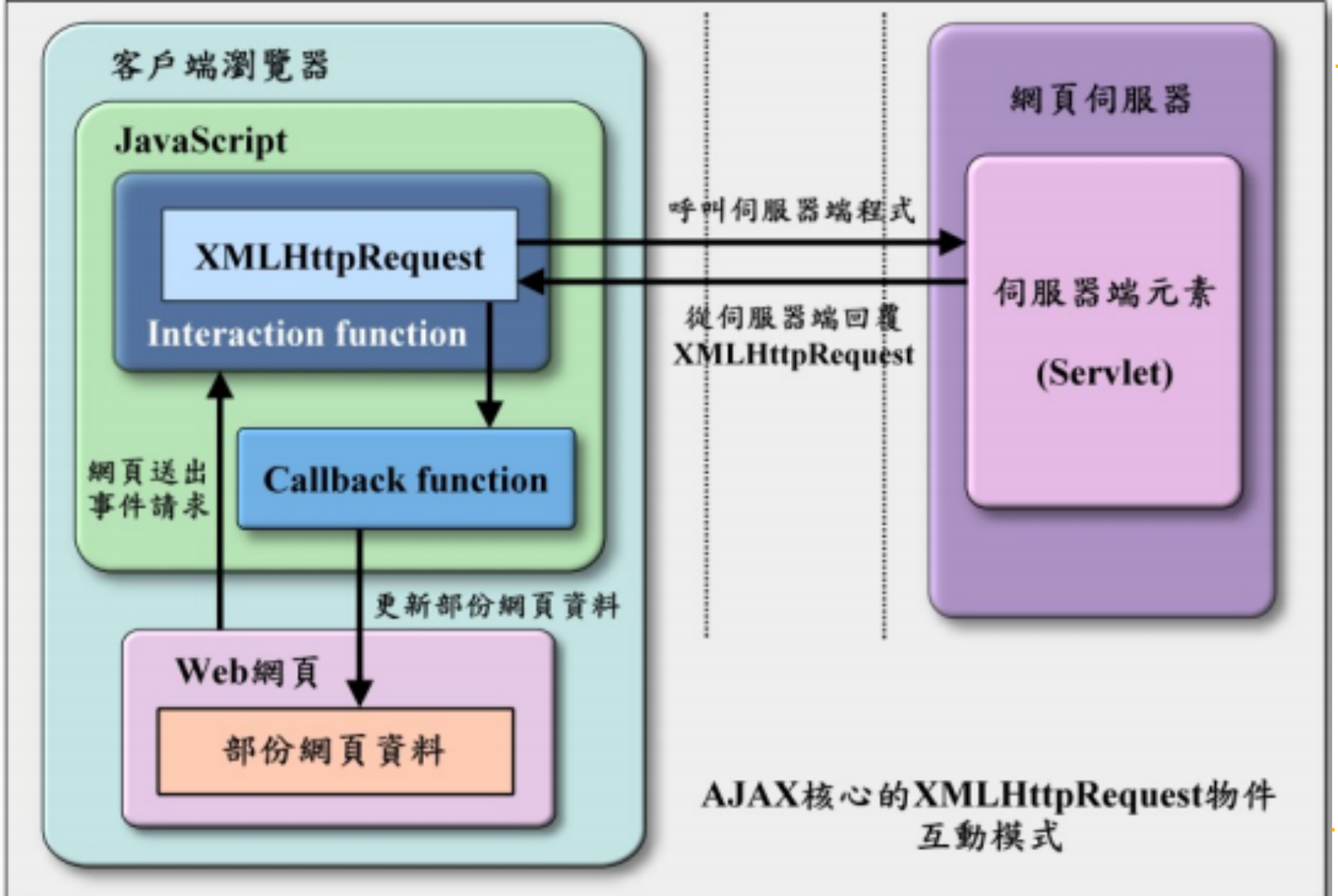


瀏覽器可以接受等 10 秒嗎

你的畫面看起來就像當掉了一樣

非同步


View Page
EVENT!
EVENT!
View result
EVENT!
User
AJAX
Ajax
Server
Generate and
send partial page
Time


客戶端瀏覽器
JavaScript
XMLHttpRequest
Interaction function
Callback function
網頁送出
事件請求
更新部份網頁資料
Web網頁
部份網頁資料
呼叫伺服器端程式
從伺服器端回覆
XMLHttpRequest
網頁伺服器
伺服器端元素
(Servlet)
AJAX核心的XMLHttpRequest物件
互動模式

XMLHttpRequest 缺點

- XHR 已經出現十幾年了
- 容易寫出回調函式地獄 (Callback Hell)
- 結構設計，已無法應對現今複雜的 Web 環境
- 不建議使用在中大型專案
- 但是要了解
- 之後的都是基於他改良
- side project 還是很好用

```
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);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

XMLHttpRequest API

- open()
  - 開啟對伺服端的連結；method 為 HTTP 請求方式（'GET'、'POST'、'HEAD' 等）
- setRequestHeader()
  - 為 HTTP 請求設定一個標頭值
- send(content)
  - 傳送請求，open 的 method 為 'GET' 時，content 設為 null，'POST' 時 content 可放字串、XML、JSON 格式的內容，會放在 POST 本體中發送。

XMLHttpRequest API

- abort()
  - 中斷請求
- getAllResponseHeaders()
  - 傳回一個字串，其中包含 HTTP 請求的所有回應標頭
- getResponseHeader
  - 傳回一個字串，其中包含指定的回應標頭值
- onreadystatechange()
  - 狀態變化，則會呼叫所設置的處理器函式

request.readyState

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

XMLHttpRequest 範例

```
<script>
let request = new XMLHttpRequest()
request.onreadystatechange = () =>{
  if(request.readyState === 4){
    if(request.status >= 200 && request.status <= 300){
      console.log('成功')
    }else if(request.status >= 400){
      console.log('失敗')
    }
  }
}
request.open('GET', '/xxx')
request.send()
</script>
```

實際體驗

HTML 25 unsaved changes

```
<div id="demo">
<h2>按按看</h2>
<button type="button"
onclick="loadDoc()">來喔</button>
</div>
```

CSS

JS

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "https://cjwu.github.io", true);
  xhttp.send();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
}
```

按按看

來喔

實際剪貼一下

```
<div id="demo">
<h2>按按看</h2>
<button type="button" onclick="loadDoc()">來喔</button>
</div>
```

https://codepen.io/pen/

實際剪貼一下

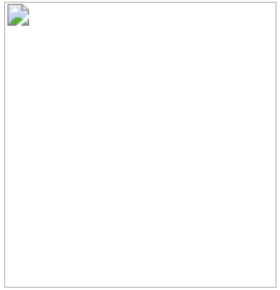
```
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "https://cjwu.github.io", true);
  xhttp.send();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
}
</script>
```

HTML 25 unsaved changes

```html
<div id="demo">
<h2>按按看</h2>
<button type="button"
onclick="loadDoc()">來喔</button>
</div>
```

JS

```js
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "https://cjwu.github.io", true);
  xhttp.send();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
}
```



Chi-Jen Wu (吳齊人)
Assistant Professor

Cross-Origin Resource Sharing (CORS) problem

用callback 的方式

```
<script>
function loadDoc(callbackfunc) {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "https://cjwu.github.io", true);
  xhttp.send();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      callbackfunc(this);
    }
  };
}
function myFunction(xhttp) {
  document.getElementById("demo").innerHTML = xhttp.responseText;
}
</script>
```

修改一下
用callback 的方式
跑跑看

Cross-domain problem

- 跨來源寫(Cross-origin writes)通常被允許
- 跨來源嵌入(Cross-origin embedding)通常被允許
- 跨來源讀取(Cross-origin read) 通常不被允許

- CORS（Cross-Origin Resource Sharing，跨來源資源共享）
  - Server 那邊加上一些 response header
    - Access-Control-Allow-Origin
  - JSONP (JSON with Padding）jQuery 目前有支援 $.getJSON()
    - 但是也是要Server支援喔
  - 還有很多奇奇怪怪的方式

與http://store.company.com/dir/page.html屬於同源的

URL	Outcome	Reason
http://store.company.com/dir2/other.html	同源	
http://store.company.com/dir/inner/another.html	同源	
https://store.company.com/secure.html	不同源	協定不同
http://store.company.com:81/dir/etc.html	不同源	埠號不同
http://news.company.com/dir/other.html	不同源	主機位置不同

Ajax + jQuery

- $.ajax(url [, settings])
  - jQuery 最底層的 Ajax 物件
- $.load(url [, data] [, complete])
  - 動態載入 HTML 文件並把它插入 DOM 中
- $.get(url [, data] [, success] [, dataType])
  - HTTP GET 不同步請求
- $.post(url [, data] [, success] [, dataType])
  - HTTP POST 不同步請求函式
- $.getScript(url [, success])
  - 載入 JavaScript 檔案
- $.getJSON()
  - 取得 JSON 資料

Ajax + jQuery

第三層	$.getJSON(), $.getScript()
第二層	load(), $.get(), $.post()
底層	$.ajax(), $.ajaxSetup()

Ajax + jQuery


```
<script>
var jqxhr = $.ajax('https://cjwu.github.io')
  .done(function() {
    alert('成功');
  })
  .fail(function() {
    alert('失敗');
  })
  .always(function() {
    alert('結束');
  });
</script>
```

Ajax + jQuery


```
<script>
$.ajax({
  url:, // 要請求資料的網址
  method:, // 請求資料的方式(Ex:POST / GET / PUT...等)
  dataType:, // 請求資料的類型(Ex:xml, json, script, html等)
  data: // 要傳送資料

  success:function(res){console.log(res)},
  error:function(err){console.log(err)},
});
</script>
```

.load(url [, data] [, complete])

load 函式用來動態載入 HTML 文件並把它插入 DOM 中。此函式預設是以 GET 的方式來發送請求，但是如果有設參數 data 則會自動轉為 POST。

參數	型別	說明
url	String	指定要進行呼叫的位址
data	Map	要傳給server的Key/value值對
complete	Function	Ajax 請求完成時 (不需要是 success) 呼叫的 callback

```
<script>
$('#result').load('ajax/test.html',
function(responseText, textStatus, jqXHR) {
   // this - 指向 #result DOM 元素
   // responseText - 請求的文件內容
   // textStatus - 請求狀態 (success, error)
   // jqXHR - XMLHttpRequest Object
});
</script>
```

jQuery.get(url [, data] [, success] [, dataType])

`$.get()` 一個簡單的 HTTP GET 不同步請求，如果你想在出錯時 (error) 能執行一些函式，那你得使用 $.ajax()。

參數	型別	說明
url	String	指定要進行呼叫的位址
data	Object	要傳給 server 的 Key/value 值對
success	Function	Ajax 請求完成時 (必需是 success) 呼叫的 callback
dataType	String	返回的資料類型 - xml, html, script, json, jsonp, text。不設定的話 jQuery 會幫你猜返回的內容格式是什麼。

```
<script>
$.get('ajax/test.html', function(data) {
  $('.result').html(data.title);
});
</script>

<script>
// 發出 HTTP 請求 test.php?name=John&time=2pm
$.get('test.php', {name: "John", time: "2pm"});
</script>
```

Ajax & jQuery 重點

- Request 處理
  - headers
  - data
  - dataType
- Response 狀態處理
  - success
  - error

```
<script>
 $.ajax({
   type: "post",
   url: “http:///cjwu.cgu.edu.tw/getinfo”,
   headers: {
     Accept: "application/json; charset=utf-8",
     token: "" + token
   },
   data:JSON.stringify(json_data),
   contentType: "application/json",
   dataType: "json",
   success: function(data){
     console.log('ok');
   },
   error:function(){
     console.log(‘error');
  }
})
</script>
```

JSON Web Token (JWT)

```
<script>
 $.ajax({
   type: "post",
   url: “http:///cjwu.cgu.edu.tw/getinfo”,
   headers: {
     Accept: "application/json; charset=utf-8",
     token: "" + token
   },
   data:JSON.stringify(json_data),
   contentType: "application/json",
   dataType: "json",
   success: function(data){
     console.log('ok');
   },
   error:function(){
     console.log(‘error');
  }
})
</script>
```

JSON Web Token (JWT)

線上練習 quiz#2

https://codepen.io/pen/

1.正妹牆實作練習

2. Commit **quiz2.html, quiz2.js** and push to quiz repository

HTML

```
<div id="contain"></div>
<h1>cgu 正妹牆</h1>
<div id="contain"></div>
<script type="text/javascript"
src="https://cdnjs.cloudflare.com/
ajax/libs/jquery/3.1.0/jquery.min.
js">
```

JS 100+ unsaved changes

```
var tags = "李知恩";
var dataUrl = "https://api.flickr.com/services/feeds/photos\_public.gne?tags=" + tags +"&tagmode=any&format=json&per_page=400&jsoncallback=?";
var data = $.getJSON(dataUrl);



data.done( function( msg ) {
```

```
<html>
<div id="contain"></div>
<h1>cgu 正妹牆</h1>
<div id="contain"></div>
<script type="text/javascript"
src="https://cdnjs.cloudflare.com/
ajax/libs/jquery/3.1.0/jquery.min.js">
</html>
```

```
<script>
var tags = "李知恩";
var dataUrl = "https://api.flickr.com/services/feeds/
photos\_public.gne?tags=" + tags
+"&tagmode=any&format=json&per_page=400&jsoncallback=?";

var data = $.getJSON(dataUrl);

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

Thanks!

Open for any questions

CJ Wu

cjwu@mail.cgu.edu.tw