壹、課程說明

單元名稱	**Google Apps Script 入門**
單元摘要	實作一個小型應用程式
設計者	江其勳 教師 (國立高雄師範大學附屬高級中學)
學習目標	能實作出 1 個小型應用程式
課綱範圍	電腦應用 - 電腦軟體
教學節數	一、Google App Script 簡介（時間：20 分鐘） 二、認識 Apps Script（時間：20 分鐘） 三、建立 Apps Script 的開發環境（時間：20 分鐘） 四、開始牛刀小試（時間：50 分鐘） 五、頁面的轉發（時間：50 分鐘） 六、實作：BMI 計算（時間：50 分鐘） 七、結語（時間：20 分鐘）
先備知識	Google 線上文件應用能力 基本網路概念 HTML 語法基礎能力
評量方法	實作作品
分享方式	
參考資源	https://developers.google.com/apps-script/

貳、教學活動計畫

教學活動名稱：

一、Google App Script 簡介（時間：20 分鐘）
二、認識 Apps Script（時間：20 分鐘）
三、建立 Apps Script 的開發環境（時間：20 分鐘）
四、開始牛刀小試（時間：50 分鐘）
五、頁面的轉發（時間：50 分鐘）
六、實作：BMI 計算（時間：50 分鐘）
七、結語（時間：20 分鐘）

教學活動說明：

一、Google App Script 簡介

任何一種技術的產生都必定有其必要性及需求，但看這個必要性與需求是否確實擊中自己的需求。因此要學習一個新的應用之前，我們有必要先了解為什麼要用這個東西？有什麼好處？以前的舊方法做得到同樣的事情嗎？清楚的理解了這幾個問題之後，才會有動機來繼續研究這項新技術。

我們這次要來介紹 Google Apps Script 這項工具，這個工具根據 Google 官網的簡單說明為「Google Apps Script 是一種 JavaScript 雲端腳本語言用來擴展 Google Apps 以及建立 Web 應用程式」。簡單的幾句話，卻透露出巨大的應用潛力。
首先，這是一個雲端腳本語言，長得有點像 JavaScript，其實，是不是 JavaScript 完全不是個重點，因為它也只有語法像是 JavaScript 其功能卻有相當大的差異。當然好處就是學過 JavaScript 的人能夠比較快的上手。

第二，可以用來擴展 Google Apps 的能力。Google Apps 是一個相當受歡迎的服務，整合了許多超強服務在上面，有大家熟知的雲端硬碟、線上文件、協作平台、等等。什麼叫擴展？舉一個簡單的例子來說，Office 的 Excel 再加上巨集可以做到許多自動化的工作(巨集就是讓我們可以在 Excel 裡面寫程式來進行各種額外的操作，比如說我們可以把一連串的操作錄製成巨集，下次只要一個動作就可以讓他一次執行完畢)。同樣的在 Spreadsheed 裡面 Apps Script 就扮演了類似巨集的角色，但因為天生就是網路服務，因此

可以存取及執行超級豐富的 Google APIs ，可說是超級強大的工具，巨集完全無法相提並論。

第三，可以建立 Web 應用程式。這種應用程式可以有兩種運作模式，一、附加於協作平台上，二、獨立運作。這個可能為我們的網路世界帶來一個相當大的改變。傳統上我們是這麼建立一個 Web 應用程式的，找一台電腦，安裝好作業系統，安裝伺服器軟體，撰寫伺服端的應用程式，放上伺服器，然後公開上線。期間，必須做好備份，以防機器忽然出錯而導致資料流失。然而 Google Apps Script 卻能夠直接建立 Web 應用程式，讓我們完全不需要自己準備機器，直接利用 Google 這個便利、普及、不用擔心故障，就算故障了也有人會幫我們修好的機器來架設自己的 Web 應用程式。

二、認識 Apps Script（時間：50 分鐘）

所有重要的訊息均可在以下網址查到。

https://developers.google.com/apps-script/

Apps Script 有以下幾種基本形態：

1. 獨立執行：可直接在以網頁為基礎的 Script Edit 撰寫完畢並執行。

2. 嵌入執行：依附在某一個應用程式裡執行。這個部分有點類似巨集之於 Excel 的關係。在 Google Docs, Spreadsheet, Forms 都可以嵌入指令碼。

3. Web 應用程式：透過 HTML Service 來建構使用者界面，Apps Script 也可以直接成為一個獨立的 Web 應用程式。

4. 協作平台小工具：Goolge Sites Gadgets 。可以透過 Apps Script 創作一個協作平台小工具，讓協作平台不只有呈現資訊的能力，還可以加入應用小程式。

綜合以上幾個重大功能，Apps Scripts 可說已成了 Google 上的準伺服端語言。 從下圖即可看出一些端倪。

上面這個圖標示出 Google Apps Script 在所有 Google 服務當中的位置，也就是它可以存取諸如 Google 文件、雲端硬碟、日曆、郵件、協作平台...等等重要的服務，就像是一個可以讀取背後這個 Google 龐大資料庫的一個前端程式語言。所有的存取都可以直接透過它來作，所以雖然是前端語言，但卻擁有伺服端的強大存取能力，這是它很重要的一個特色。

三、建立 Apps Script 的開發環境

Apps Script 跟其它的程式寫作有一個很大的不同，就是 Apps Script 程式寫作也是直接在網頁上完成，不需要安裝任何本機端的程式，並且功能也不差，程式碼著色、函數提示、Debug 等重要的 IDE 功能都具備了。好處就是任何可以上網的地方都可以用瀏覽器直接撰寫程式，壞處當然就是直接在網頁上撰寫程式還是無法比許多專業的單機 IDE 程式比美，比如 Eclipse.

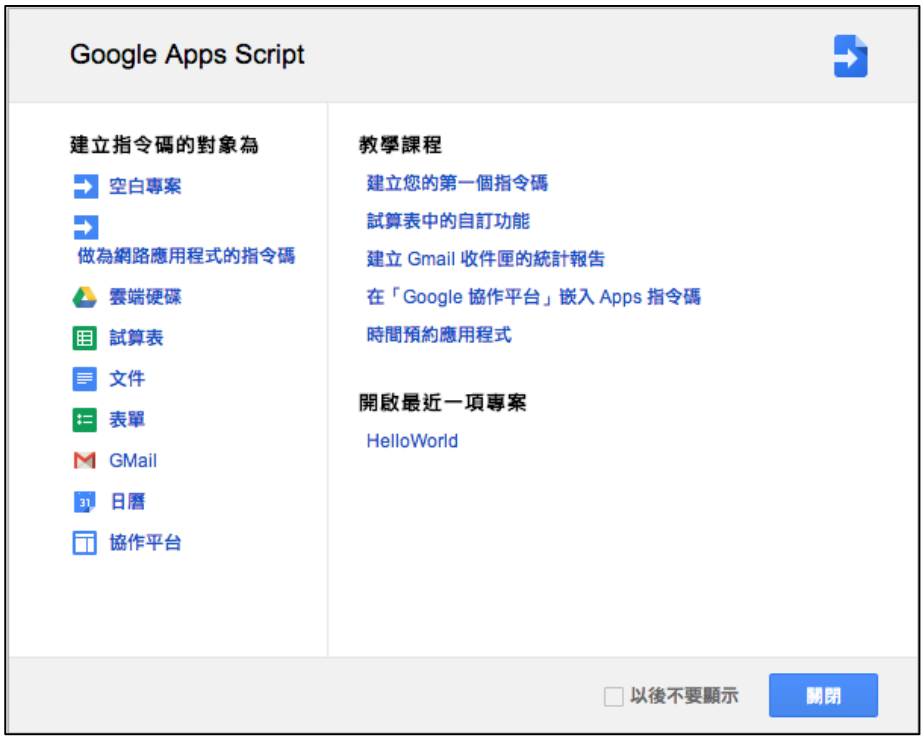
四、開始牛刀小試

開始建立一個最簡單的頁面吧！

我們計劃直接顯示一個簡單的網頁，並且在網頁上面顯示幾個簡單的文字，如 Hello world 之類的。

首先我們登入到 gmail 信箱，進入到“雲端硬碟”選項裡。點選“建立” -> “指令碼”

然後會出現底下的畫面：



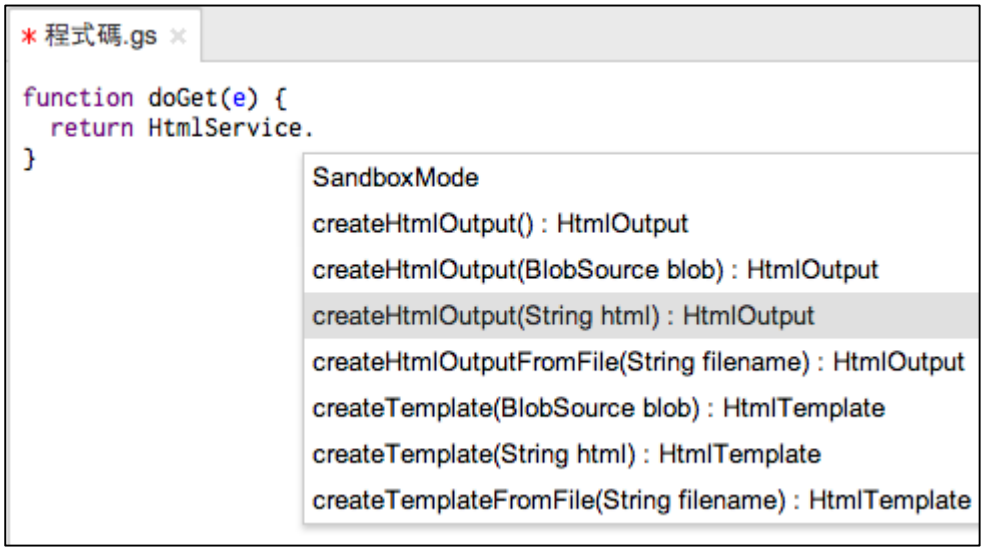
我們先選擇一個“空白專案”。

會看到如上圖的程式編輯畫面。程式碼檔案為“程式碼.gs”，預先幫我們建立好了一個 myFunction() 的函式。

我們現在要建立一個顯示幾個簡單文字的網頁，必須要有一個進入點，也就是說這個網頁要執行的時候第一個會被執行的 function 是哪一個？不太懂的話，可以想像一下以前做網頁的時候，同一個資料夾底下有一大堆 .html 網頁檔，首頁就是進入點，哪一個才是首頁？規定就叫做 index.html。相同的觀念，在這個 .gs 的程式檔裡面，也會有很多個 function, 但第一個被執行的 function 就是進入點。這個進入點就規定就叫做 doGet。

因此我們把 myFunction() 改為 doGet(e)。

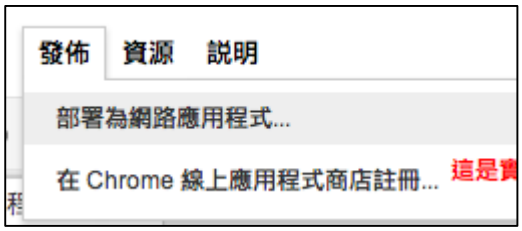
因為我們的程式碼並不是要直接呈現在網頁頁面上面的內容，因此我們要透過一個轉發的過程將頁面或 UI 轉發個一個真正的 HTML 檔案。這時我們就要用到 HtmlService。我們在程式碼當中打入 HtmlService 就會出現提示功能囉，相當方便，不再需要查詢一大堆文件來找到可用的功能函式。



```
function doGet(e) {
  return HtmlService.createHtmlOutput("<h1>我是一個網頁！</h1>");
}
```

初步寫好了之後該怎樣看到結果呢？

這裡必須要用到“發佈”-> 部署為網路應用程式



底下這個對話框非常重要，一定要仔細填寫，因為它關乎了這個應用程式具備那種權限，比如，我們如果寫了一個自動建立日曆的程式，如果我們發佈出去是要讓使用者們能夠簡單的建立他們自己的日曆的話，權限就必須為“使用者存取”，但如果選成“我”的話，所有可以使用這個程式的使用者都可以大大方方的在“我”的日曆裡面建立新的日曆了。這可是天大的錯誤！因此要非常注意執行者到底要有什麼權限。

成功部署後，直接點選“最新的程式碼”測試網路應用程式。

就完成啦～

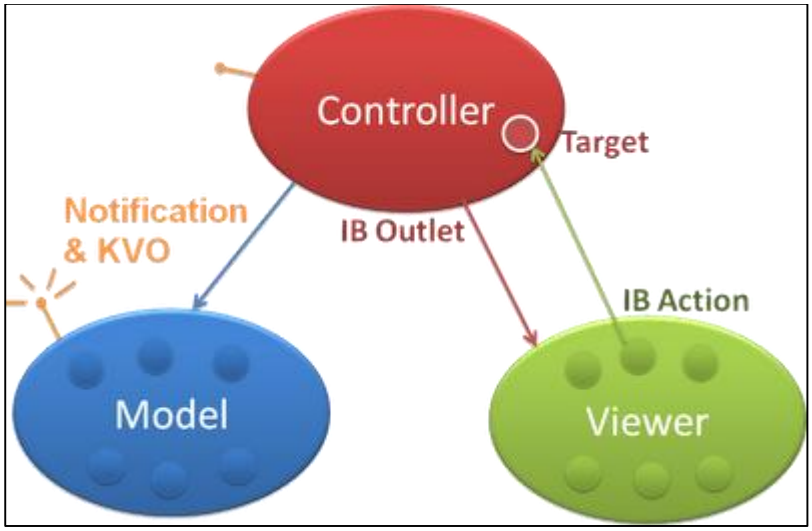
五、頁面的轉發

從上節“牛刀小試”的例子裡我們可以看到，有一個地方會出大問題。未來我們如果需要一個稍微複雜的界面的時候，恐怕會非常難以撰寫，一大堆 html 原始碼擠在一起，怎麼樣看都是一個不對勁的事情。

return HtmlService.createHtmlOutput("<h1>我是一個網頁！</h1>");

當然，Google 也會注意到這一點，所以才會有轉發的機制。其實也就是程式設計當中行之有年的 MVC 架構的實踐而已。也就是業務邏輯不應該跟展示層混雜在一起。

這邊稍微說明一下 MVC 到底是什麼？



Model, Controller, Viewer 是一種軟體架構的方法，它存在於各種軟體設計當中，比如一個單機版的踩地雷遊戲，程式當中使用者最直接看到的就是 View，也就是最前端的介面呈現的樣子。使用者開始玩遊戲之後，會開始點選其中一些格子，格子點下去之後就會產生一些『事件』，如果這個事件是點到一個空地，這個事件就會觸發『把這個空地附近的所有空地都點開』的這個動作，如果點到一個地雷，就會跑出 Game Over. Controller 就是負責轉發這些事件到適當的函式當中，也就是負責控制整個程式的流程的地方。最後一個就是 Model，專業術語就叫做放置『業務邏輯』的地方，但是這個太抽象，我們還是回到踩地雷遊戲，Model 負責什麼呢？當我們點到一個空地， Controller 會把程式轉發給一個負責『找出這個空地附近的所有空地』的函式或演算法來執行，如果點到的是地雷，就會轉給一個『開始統計還有多少地雷沒有被找出來，並且結束遊戲』的函式來執行，這些地方的程式碼因為是實際在執行這個軟體的核心業務，因此叫做業務邏輯，就被統稱為 Model。

為什麼要這樣把這些程式碼都分開呢？想想看如果我們在設計『找出這個空地附近的所有空地』這個演算法的時候，我們當然只希望專注在用最簡單的座標來計算出這個座標附近的所有空的座標點，如果這時候還要去考慮這個空地的顏色，或者界面上顯示的大小要有幾個像素，這樣就非常雜亂無章，完全不具有可維護性。

同樣的道理在網頁應用程式的設計也是套用同樣的概念，因此我們將我們的程式改進如下，將一個 html 頁面單獨建立出來，讓所有界面的設計都放在這裡，後面的程式碼就不要去擔心字型、顏色、大小等等的雜事。

我們依照網頁設計慣例，建立一個 Index.html 的檔案。

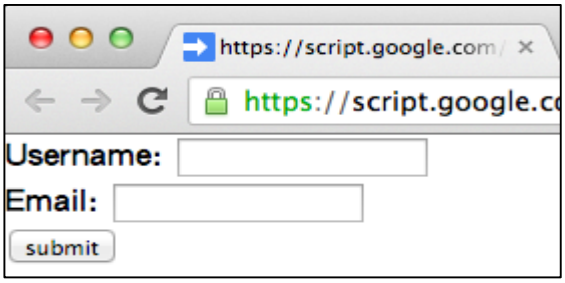
貼上我們想要的原始碼

```
<html>
  <body>
    <form id="regForm">
      <div>Username: <input type="text" id="username"/></div>
      <div>Email: <input type="text" id="mail"/></div>
      <input type="submit" value="submit"/>
    </form>
  </body>
</html>
```

然後回到 程式碼.gs

改成如下

```
function doGet(e) {
  return HtmlService.createHtmlOutputFromFile("Index");
}
```



成功轉發到一個純 Html 檔案去囉。對使用者來說，他看到的就是一個不折不扣的網頁形態。

六、實作：BMI 計算

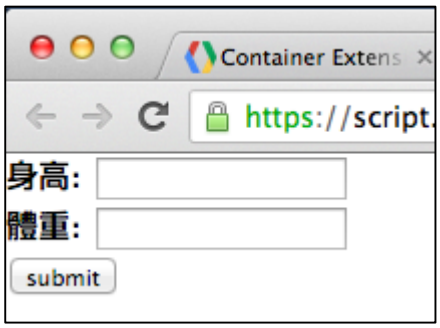
雖然一個精美設計的 HTML 頁面也可以相當漂亮，但在這個 Web 2.0 都已經過時的時代，靜態網頁實在是非常不吸引人。而且我們要作為一個“應用程式”第一個要素就是要能夠取得使用者給定的資料，才能夠把取得的資料拿來重整、加工。表單送出一切由此開始。讓我們開始來做一個表單送出的範例吧。

範例：BMI 計算

首先，我們已經能夠做到將程式碼轉發給 Index.html 這個檔案，所以第一步我們就把 Index.html 改成我們想要的樣子。

```
<html>
  <body>
    <form id="bmiForm" action="<?= serviceUrl ?>" method="post">
      <div>身高: <input type="text" name="h"/></div>
      <div>體重: <input type="text" name="w"/></div>
      <input type="submit" value="submit"/>
    </form>
  </body>
</html>
```

執行後我們可以得到一個看起來非常熟悉的標準網頁形態的應用程式樣子。



從這邊開始，程式碼開始變得相當不同了，因為，仔細看看 Index.html 的原始碼，action="<?= serviceUrl ?>" 這裡用了一個很奇怪的東西，看起來也不是 Html 標準原始碼。

所以，這個 Index.html 已經不是一個單純的 Html 檔案了，在 Google Apps Script 裡面稱

他為 Template(模板)。簡單來說就是在一個 Html 檔案裡面放入一些變數，讓這些變數可以隨著程式碼的運算而得到不同的結果。

既然已經不是一個單純的 Html 檔案，那轉發的方式也必須要不同。轉發方法如下：

```
HtmlService.createTemplateFromFile('Index').evaluate();
```

接下來，這一個 serviceUrl 該從何而來？

到目前為止，還沒有人告訴前端的網頁檔這個變數到底是多少？我們必須從程式碼端給他一個值，這個值就是要取得這個程式碼的 url 網址。

因此我們將程式碼改寫一下：

```
 var t = HtmlService.createTemplateFromFile('Index');
 t.serviceUrl = ScriptApp.getService().getUrl();
 return t.evaluate();
```

我們創建了一個 Template（模板） 之後，給他一個 serviceUrl 變數一個值，好讓他帶著這個變數移動到前端的網頁裡面。

嘗試執行看看是否會得到如上圖的身高、體重輸入欄位以及一個送出鈕。

接下來測試一下填完資料後按送出，保證錯誤。因為我們的程式碼根本沒有接收的機制，也就是去哪裡接收剛剛在表單裡面填進來的身高、體重的參數？

規定的接收機制就是 doPost(e).

程式碼如下：

```
function doPost(e){
 var h = e.parameter.h;
 var w = e.parameter.w;
 return HtmlService.createHtmlOutput("bmi="+w/h/h);
}
```

首先，後端程式接受到從前端以 POST 方式送出來的表單，必須要先知道要如何讀取 Form 裡面所帶的值。我們前端包含兩個欄位，一個是身高，一個是體重。他們的欄位名稱分別是 h, w。後端的接收方式就是　e.parameter.h 及 e.parameter.w;

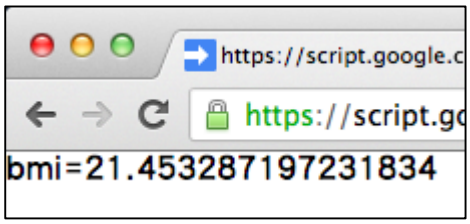
讀取進來之後，就可以分配給兩個變數用來存放。如下：

```
var h = e.parameter.h;
var w = e.parameter.w;
```

一旦拿到這兩個變數，後面的事情就簡單多了。就把變數拿出來運算，最後回傳到前端即可。

```
HtmlService.createHtmlOutput("bmi="+w/h/h)
```

最後結果：



當然，根據我們前面提供的轉發機制，也可以將這個結果轉發給另一個 html 檔案，以提供更漂亮的 UI ，也就更接近一個正常的網頁應用程式。這裡我們就不再贅述。

大功告成！我們成功的利用 Apps Script 撰寫了一個不需要主機、伺服器軟體就可以執行的 Web 應用程式。

七、結語

我們用一個簡單的實例來實作讓 Apps Script 成為一個獨立的 Web app。但，Apps Scirpt 不只可以成為一個獨立的 Web app. 還能夠從 Google docs 裡面讀取資料，比如：可以從試算表裡面讀取一群資料，重新整理顯示在網頁上，或者也可以從日曆中讀取行事曆資料來進行後續的顯示或操作。

Apps Script 也可以很容易的存取所有 Google API, 從而引進了諸如 gmail, 地圖... 等等超級強大的功能，甚至提供連接 JDBC 的能力、URL 呼叫的能力，可說是一個完整的雲端程

式語言。其後的整合運用確實具有無限的想像空間。雖然我們只用了一個小小的簡單實例，但卻可以從中發現 Apps Script 確實擁有著巨大的潛力，可望成為 Google App Engine 之後另一個強大的服務。