程式人 月刊雜誌

Programmer

捐發票愛心條碼

讀書做善事、寫書做公益 – 歡迎程式人認養專欄或捐出您的網誌

參考價： NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體

羅慧夫顱顏基金會 彰化銀行 (009) 帳號： 5234-01-41778-800

程式人雜誌

2015 年 1 月

本期焦點：令人眼花撩亂的 javascript -- 前端技術篇

程式人雜誌

前言

編輯小語

HTML5 技術將網頁帶入了更豐富的應用開發領域，而 node.js 的出現則讓 javascript 統一了前後端的技術，這兩個技術的出現，讓 javascript 得以橫跨「前端與後端」，於是 javascript 就成了雲端技術當中最耀眼的一種語言。

本期的「程式人雜誌」探討的焦點是「javascript 前端技術」，企圖為讀者介紹「javascript 與前端技術的發展現況」，以便讓大家可以在這些令人眼花撩亂的技術框架中，整理出一些頭緒，希望透過這樣的介紹，可以讓大家對 javascript 技術的現況能有更清楚的認識。

----（「程式人雜誌」編輯 - 陳鍾誠）

授權聲明

本期焦點：令人眼花撩亂的 javascript -- 前端技術篇

令人眼花撩亂的 javascript 新世界

如果您尚未學過 HTML/CSS/javascript 等語法的話，建議您可以先看看『少年程式人雜誌』2014年12月號的下列文章，再來看本期的內容，應該會比較順利。

- 少年科技人雜誌 -- 2014 年 12 月號
  - 基於 HTTP/HTML/CSS/JavaScript 的 Web 技術
  - HTML 網頁設計
  - CSS 版面設計
  - JavaScript -- 讓網頁動起來

在我所學過的語言當中，javascript 大概是被誤解最多的一個語言。像是 JavaScript 是 Java 語言的簡化版、JavaScript 語言很難用、JavaScript 語言設計很差勁、javascript 只能用來寫寫小程式等等。

然而，這些誤解其實是我們不瞭解 JavaScript 所造成的。如果您用心的理解 JavaScript，您會發現這是一個「簡單、輕巧又優美」的語言，其原型導向的設計方式，用很簡單的概念達成了物件導向語言的功能，真的很適合做為瀏覽器上的共通語言。

javascript 的歷史還算蠻悠久的，在1995年時就由瀏覽器始祖網景公司 (Netscape) 的布蘭登·艾克 (Brendan Eich) 所設計出來，企圖在瀏覽器中引入程式語言，讓網頁可以擁有更多的互動性。

原本 javascript 稱為 livescript，但網景公司為了和昇陽公司合作推動 java，結果將此以語言改名為 javascript，於是javascript常被誤解為java版本的script，但其實兩者的設計上並沒有太大關係。

不過瀏覽器確實需要一個程式語言，可惜當初昇陽公司主推java語言的applet技術在瀏覽器上表現不佳，瀏覽器內的動畫等領域反而被Macromedia 公司的flash 佔領，而java卻陰錯陽差的成了桌上應用與伺服端的重要語言，這段歷史讓javascript沈寂了一段時間。

但是自從 2009 年 node.js 推出以來，javascript 就開始進入一個『大爆發』時期，各式各樣的『框架、函式庫、套件、平台、應用』層出不窮，而 HTML5 的成熟化更進一步讓 javascript 進入了一個嶄新的時代，為 javascript 創造出了全新的可能性。

現在的javascript技術演進可以說是令人眼花撩亂，從前端的 jQuery、backbone.js、angular.js、react.js 等技術，到後端的 node.js、express.js 、connect.js、mongoose.js，以及橫跨兩端用來通訊的 websocket, ajax 技術與 socket.io 等套件，還有經常搭配 javascript 使用的 HTML5, CSS3 技術，以及 bootstrap 等顯示框架，這些技術的蓬勃發展讓 javascript 再度回春，成為具有強大生命力的一種語言。

除了瀏覽器前後端之外，javascript也開始入侵到一些本地應用與手機APP等領域，像是 Titanium, PhoneGap 可以讓您用javascript/html/css 等技術撰寫橫跨iOS、android、window phone 等平台的APP，而 微軟新一代的 windows App 也是依靠 HTML/CSS/javascript 這樣的技術體系所建立的，另外像 Intel 的王文睿也釋出了 Node-webkit 這種可以用 HTML/CSS/javascript 創建以瀏覽器為何心的視窗程式環境，而 Unity 這個跨平台的遊戲引擎也主要支援 javascript /c# 這兩種語言，這讓 javascript 成為可以用來創建「從命令列程式、網路程式、web程式、視窗程式到遊戲程式」這樣涵蓋度極為廣泛的一種語言，大大的增加了javascript這個語言的吸引力。

未來、javascript 還會被用在什麼領域呢？我們很難預測，因為也有人開始試圖將 javascript 用到嵌入式系統與設計作業系統上了，而 javascript 的語法標準 ECMAScript 6 又為 javascript 添加了包含正規物件導向、 mixin 模組、預設值參數、yield語法等等，讓那些抱怨javascript物件語法不正統、語法不夠強大的人們，也有機會用正規的物件導向方式與強大的語法來撰寫javascript了。

這些 javascript 的新發展，可是讓身為『程式控』的我，感到非常的激動與興奮呢！

HTML5 與前端技術的新進展

HTML5 是網頁技術上的一個重要里程碑，雖然其 W3C 的標準直到2014年才進入正式版的recommandation模式，但是在各家的瀏覽器當中，卻幾乎都已經相當完整的支援了HTML5的大部分功能。

HTML5 裡面有很多受到注目的功能，這些功能很可能改變整個web的設計與使用型態，其中較為人所知的部份如下：

- 2D 繪圖的 canvas 功能
- 3D 繪圖的 WebGL 功能
- GPS 定位的 geo location 功能
- 更多的語意標記 (article, ...)
- 搭配 CSS3 的動畫與更多排版功能
- 區域儲存 （localstorage, indexedDB）與離線應用的功能
- 雙向網路通訊的 websocket 功能
- 可以不阻礙畫面顯示的 webworker 功能

其中有關 2D、3D、GPS等功能比較一目了然，可以想見會造成哪些影響。這些功能讓遊戲動畫與地理資訊類的程式可以搬到HTML5/CSS3/Javascript 的平台上呈現使用，不需要再依賴flash或其他外掛了。

至於語意標記則是W3C在XML無法普及後企圖讓網路稍微具有點語意功能的方法，而CSS3則讓網頁排版可以更加動態與活潑，但是直到最近我才發現，其實最後三項(區域儲存、websocket、webworker) 的影響力或許會比前面那些更大也說不定。

關注web與行動技術的朋友一定都會注意到，1993年網路大爆發之後，web技術大致在javascript提出之後就底定了，中間經過flash動畫的銜接，但web整體的技術並沒有很明顯的改變。反而是2007年開始智慧型手機上網後導致APP興起，於是技術又回到各家廠商主導的平台導向世界。

但是、HTML5 提出了區域儲存的功能之後，就為離線應用鋪好了一條路，這條路有可能直達APP的地盤，讓網頁也能做出類似APP的應用，於我們就有可能用HTML5做出跨平台的APP，而且不需要依靠像Titanium或PhoneGap 這樣的平台。

當然、基於安全性的顧慮，有些系統功能還是無法用HTML5做出來的，例如存取硬碟、呼叫系統函數等等，但是對於那些不需要存取系統函數的程式而言，可以將資料存在localstorage或indexedDB當中，然後用websocket的方式回傳到伺服端，這種方法可以部份的彌補原本網頁程式難以 APP化的問題。

最近、像是 Backbone.js , Angular.js, React.js 等著重在『單頁應用』的框架開始受到大量的關注，我想也是與 HTML5 技術有關的。但是、即使不做成單頁應用，而是做成多頁應用，然後再利用『區域儲存』的localstorage或indexedDB技術達成跨網頁間的資料共用，其實也可以做出類似 APP的效果，我們將在這兩期的『程式人雜誌』當中探討此一議題。

前端框架套件 -- jQuery, backbone.js, angular.js 與 react.js

jQuery, backbone.js, angular.js 與 react.js 等開放原始碼專案都是屬於『前端』的顯示框架，但是初學者要清楚的理解這些框架的差別，與可以適用在哪些情況，其實並不容易，因為太多的javascript框架已經到了令人眼花撩亂的程度了。

因此、雖然筆者只用 jQuery 寫過幾個小程式，看了一本backbone.js的書，沒有寫過angular.js與react.js，但是看了幾篇相關的文章，也斗膽在此分享一下自己對這些框架的理解與看法，讓大家可以在決定要學習哪個框架的時候，能夠不至於迷失在五里霧中。

jQuery

jQuery 其實是用來處理 HTML 的樹狀結構 DOM 的一個函式庫，透過物件導向的 a.b().c.d()..... 這樣的『鏈式語法』讓您可以用簡短的呼叫去巡覽、取得並處理對應的樹狀節點(node)，這種『鏈式語法』可以讓程式碼縮短，用很簡潔的語法完成節點的取得與處理動作。

舉例而言、在w3school網站當中有下列的範例，該範例示範了如何用 jQuery 用很簡潔的語法做出滑動的效果。

- http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_chaining

```
$(document).ready(function()
  {
  $("button").click(function(){
    $("#p1").css("color","red").slideUp(2000).slid
eDown(2000);
  });
});
```

這種『鏈式語法』的設計關鍵其實是在物件的成員函數中，盡可能的用 return this傳回

自身。舉例而言，假如我們定義下列這樣一個物件，並在每個成員函數的最後都傳回this，那麼我們就可以用 obj.a().b().c().a().d().c().....這樣的語法進行鏈式呼叫了。

```
var obj = {
  a:function() {... return this; }
  b:function() {... return this; }
  c:function() {... return this; }
  d:function() {... return this; }
}
```

另外、因為jQuery是集合導向的作法，每次都是處理一大堆節點，搭配上『鏈式語法』之後，就可以用一條鏈式語法對一大堆的節點進行連續的處理動作，這比起每次都要寫迴圈的方式要有效率多了。

另外、jQuery 也支援了一些像 post(), ajax() 與伺服端溝通的函數，讓網頁設計者可以用統一的語法搞定網頁的前端處理工作。

Backbone. js

熟悉『設計模式』的朋友們應該聽說過MVC這個著名的設計模式，也就是『Model-View-Controller』，該模式將一個系統分成背後的『模型』(Model)、『顯示』(View)與連接的『控制』(Controller)等三大部份，這樣就能將模型與顯示兩者分開，然後再用controller將兩者結合在一起。

傳統上這種方法會用在『網路程式或視窗程式』上。對於web而言，我們通常會將model放在後端的伺服器，而將view放在前端的瀏覽器當中。

但是、由於javascript+HTML5這些技術讓瀏覽器可以承受的工作越來越多，因此前端就越來越複雜了，於是前端本身有時就包含了複雜的模型與控制部份，這時候就有人想到要在前端實現完整的MVC架構，以便處理這些複雜的工作，於是像backbone.js這樣的 前端MVC框架就被實現出來了。

如果您對backbone.js想有個初步的認識，可以參考下列文章，

- Javascript 前端工具 Backbone.js Framework 初學介紹

在 backbone.js 當中，實現的並不是 Model-View-Controller 這樣的模式，而是Model-Collection-View 這三類物件，以及利用event進行串連的方式，除此之外，javascript語言本身就扮演了某種程度的controller角色，因此您也可以輕易的將jQuery與backbone.js 搭在一起使用，兩者可以很完美的融合運作，不會有任何違和感。

backbone.js 由於使用了 Underscore.js 這個框架，可以很容易的進行集合的鏈式處理，而 Underscore.js 框架裡又有一個簡易的樣板引擎，可以用來將樣板轉換成HTML區塊輸出，只要用view物件搭配Underscore.js，就可以得到一個相當完整的 MVC 框架了。

在 model 部份，backbone 的模型資料有任何修改時，都會觸發一些資料修改事件，只要在這些事件當中加入對應的程式碼，例如更新畫面或回傳資料到伺服器，就可以成為一個完整的網頁應用程式了。

React.js

React.js 是 facebook 公司所創建的一個開源專案，扮演的角色主要是 MVC 架構中的 View 角色，在 React.js 的官網中有個非常簡單的範例如下：

```
 <!DOCTYPE html>
<html>
  <head>
    <script src="build/react.js"></script>
    <script src="build/JSXTransformer.js"></scrip
t>
  </head>
  <body>
    <div id="example"></div>
    <script type="text/jsx">
      React.render(
        <h1>Hello, world!</h1>,
        document.getElementById('example')
```

```
    );
  </script>
 </body>
</html>
```

這個範例大概可以說是 React.js 版本的 hello world! 您可以看到其中的關鍵部份是 React.render 這一段。

```
    React.render(
      <h1>Hello, world!</h1>,
      document.getElementById('example')
    );
```

React 的特別之處是在 javascript 裡面放入了 HTML/XML 的內容，以下是 react官網中的一段關鍵描述，說明了react為何要搭配 JSXTransformer 去將 HTML/XML 語法轉換成 javascript 的原因，這樣就可以將　HTML/XML 完全的 javascript 化了。

> The XML syntax inside of JavaScript is called JSX; check out the JSX syntax to learn more about it. In order to translate it to vanilla JavaScript we use <script type="text/jsx"> and include JSXTransformer.js to actually perform the transformation in the browser.

看清楚上述範例之後，您可以試著看看 5 Practical Examples For Learning The React Framework 這篇文章，該文章用 jsFiddle 提供了五個活生生的線上範例，讓您可以直接感受到 React.js 的功能，這五個範例如下：

- Timer -- http://jsfiddle.net/martinaglv/3fZT2/light/
- Navigation menu -- http://jsfiddle.net/martinaglv/sY6nX/light/
- Real-time search -- http://jsfiddle.net/martinaglv/3N6D3/light/
- Order form -- http://jsfiddle.net/martinaglv/mr7gY/light/
- Image app with AJAX -- http://jsfiddle.net/martinaglv/Bnhe8/light/

看完上述的範例，我對 react.js 的認知大概是，一種將 XML/HTML 包在 javascript 裡撰

寫的技術，用一套物件架構將網頁呈現過程包在物件內部，將 HTML/XML 的呈現動作完全變成 javascript 的物件。

Angular.js

雖然最近 React.js 的勢頭似乎有點蓋過 Angular.js 了，但是瞭解一下 Angular.js 也是不錯的，您可以看看 Learn AngularJS With These 5 Practical Examples 這篇文章與其中的範例。

我發現 Angular.js 的想法似乎與 React.js 完全相反, Angular 感覺是以 HTML 的 ng-* 自訂屬性為主，所發展出來的一套顯示框架，像是 ng-app, ng-controller, ng-model, ng-show 等屬性，然後將 ng-* 屬性、嵌入的 {{*}} 變數以及 $ 為字首的 javascript 變數或函數關聯起來，形成一套 MVC 架構。

所以我認為 Angular.js 應該是一套完整的 MVC 架構, 不需要再和其他框架結合起來就能運作了。相反的， facebook 則建議 React.js 應該和 Flux 這個架構搭配 , 才能形成更完整的架構。（當然您也可以將 React.js 拿來和 backbone.js 搭配）

結語

對於這些前端框架的用途，筆者已經困惑好久了，寫完這篇文章，腦袋裏終於有些似懂非懂的概念了，希望我的理解沒有錯誤阿！

網站設計的一種新模式 - 基於 localstorage 與 websocket 的可離線設計方法

雖然最近的 Backbone.js , Angular.js, React.js 等框架似乎都提供了單頁式應用 （Single Page Application) 更好的發展工具，但是網頁做成單頁式的畢竟還是會喪失原本的優勢之一，也就是每頁分開的獨立性較高，分工較容易的的特點。

舉例而言、假如我想作一個翻譯系統，那麼除了『翻譯功能』之外，還會有字典管理、帳號管理、登入模組等功能，如果做成單頁應用，那麼這些功能就必須整合在一頁當中，雖然可以分開放在不同的javascript模組，但是會納入到一個系統當中，而不能個別獨立的運作了。

於是我開始思考，有沒有可能做成很多個獨立的頁面，但是這些頁面合起來還是可以形成一個完整的系統呢？

在看到了localstorage與 websocket 這些HTML5的新技術之後，我開始覺得這樣的設計應該是可行的。

舉例而言，當我們設計了一個網頁專門作『翻譯功能』的時候，我們可能會需要用到字典，於是我們可以用一個預設的字典放在程式裡，但是讓使用者可以透過 localstorage 取得『字典管理』頁面所記錄下來的『其他字典』，這樣就可以讓『翻譯功能』與『字典管理』共用字典了。

然後、當網頁偵測到可以連上伺服器時，就可以利用websocket的模組將資料傳回伺服器儲存，但是在連不上伺服器時就採取離線的方式運作，這樣就可以讓一個網頁再沒有伺服器時也能運作，但是在有伺服器時就可以取得伺服端的資料並於修改後存回伺服器。

在本期雜誌的後半部與下期的雜誌中，我們將會用實際的範例，說明這種『多網頁可離線的合作模式』是如何進行的，讓大家能透過程式碼來理解這種模式的實際運作方式。

我們將展示一個包含『逐字翻譯、英文單字測驗、字典管理、帳號登入、多國語言』等幾個單獨網頁的系統，而這些系統透過上述模式進行資料分享與溝通，並與伺服端結合的範例，展示這種『多網頁可離線的合作模式』的設計原理。

Web 可離線應用 1 -- 逐字英翻中系統

本系統已經上傳到 github 上，您可以點選下列連結試用一下這個「英翻中系統」系統，然後在開始閱讀本文：

- http://programmermagazine.github.io/201501/code/mt.html

為了要說明如何用 HTML5/CSS/JavaScript 建構出可單頁獨立運作，又可透過 localstorage 記住使用者資訊的網頁應用，我們建立了一個「逐字英翻中系統」，讓使用者可以透過 localstorage 儲存使用者紀錄的一些資訊。

首先我們在 spa.js (Single Page Application 的簡寫) 當中創建了一個稱為 DB 的物件，該物件可以用來協助「載入或儲存」資訊到 localstorage 當中，如此就可以用 load(), save() 函數記住這些資訊，必須注意的是 localstorage 當中只能儲存字串資訊，因此我們必須用 JSON.stringify() 等函數將 JSON 物件轉換成字串後才能儲存，而在取出後再用 JSON.parse() 將字串轉回 JSON 物件。

檔案: spa.js

```
...
var DB = {};

DB.forget = function DB_forget(name) {
  window.localStorage.removeItem(name);
}

DB.load = function DB_load(name) {
   if (window.localStorage[name] !== undefined)
     return JSON.parse(window.localStorage[name])
```

```
;
    else
      return undefined;
}

DB.save = function DB_save(name, obj) {
  window.localStorage[name] = JSON.stringify(obj);
}
...
```

然後、我們撰寫了 mt.html 這個網頁程式，以下是該網頁的一個執行畫面：

圖、逐字英翻中系統

該網頁的完整原始碼如下所示：

檔案：mt.html

```
<!-- firefox 似乎不支援 ruby tag : http://www.w3schools.com/tags/tag_ruby.asp -->
<html>
<head>
<meta charset="utf-8" />
  <link rel="icon" href="favicon.ico">
  <link href="css/bootstrap.min.css" rel="stylesheet">
  <link href="elearn.css" rel="stylesheet">
</head>
<body onload="load()">
    <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation" id="navbar">
    </nav>
    <div id="panelMT" class="container panel">
      <p class="lead">
        <form name="formMT" class="lead">
          <div class="row" style="height:300px">
            <div class="col-md-6">
              <div class="page-header">
                <span data-mt="English">原文</span>
                <input type="text" id="query" equired autofocus data-mt="Query" placeholder="" size=
```

```
"12"/>
                <button class="btn btn-primary" t
ype="button" data-mt="Query">查詢</button>
                <button class="btn btn-primary" t
ype="button" data-mt="Translation" onclick="doMT()"
  style="float:right">翻譯</button>
            </div>
            <textarea id="ebox" class="form-con
trol" style="height:100%">
The snow glows white on the mountain tonight.
Not a footprint to be seen.
A kingdom of isolation, and it looks like I'm the
Queen.
The wind is howling like this swirling storm insid
e.
Couldn't keep it in; Heaven knows I've tried.

Don't let them in, don't let them see.
Be the good girl you always have to be.
Conceal, don't feel, don't let them know.
Well now they know.

Let it go, let it go.
Can't hold it back any more.
Let it go, let it go.
```

```
Turn away and slam the door.
I don't care.
What they're going to say.
Let the storm rage on, the cold never bothered me anyway.
          </textarea>
            </div>
            <div class="col-md-6">
              <div class="page-header">
                <span data-mt="Translation"></span>
                <input type="text" id="queryResult" equired autofocus data-mt="queryResult" placeholder="" size="15"/>
                <button class="btn btn-primary" type="button" data-mt="Save">儲存</button>
                <button class="btn btn-success" type="button" data-mt="Forget" onclick="forget()" style="float:right">忘記</button>
              </div>
              <div id="cbox" style="width:100%; height:100%; border:1px dotted #888; overflow:auto;" class="form-control"></div>
            </div>
            <div></div>
```

```
          </div> <!-- row -->
        </form>
      </p>
    </div>
<script>
var ebox, cbox, dict;

function load() {
  ebox = document.getElementById("ebox");
  cbox = document.getElementById("cbox");
  dict = e2cTV;
  var dbKnowWords = DB.load('knowWords');
  if (dbKnowWords === undefined)
    knowWords = {};
  else
    knowWords = dbKnowWords;
}

function forget() {
  DB.forget('knowWords');
  knowWords = {};
}

function normalize(e) {
  return e.replace("'", '_').toLowerCase();
```

```
}

function mt(str) {
  var re = /([\w']+)/gi;
  var toStr = "";
  var si = 0;
  while (m = re.exec(str)) {
    var eword = m[1], elower=eword.toLowerCase();
    var cword = dict[eword.toLowerCase()];
    toStr += str.substring(si, re.lastIndex-eword.
length);
    if (cword === undefined || knowWords[elower]
!== undefined)
      cword = "";
    toStr += '<ruby class="'+normalize(eword)+'"><
rb>'+eword+'</rb><rt>'+cword+'</rt></ruby>';
    si = re.lastIndex;
  }
  return toStr;
}

function doMT() {
  var cstr = mt(ebox.value);
  cbox.innerHTML = cstr.replace(/\n/g, "<BR/>");
  $('ruby').click(function() {
```

```
    var e = $(this).find('rb').text().toLowerCase
();
    var c = $(this).find('rt').text();
    knowWords[e] = c;
    $('#query').val(e);
    $('#queryResult').val(e+'='+c);
    $('.'+normalize(e)).find('rt').hide();
  });
}

window.onbeforeunload = function(){
  DB.save('knowWords', knowWords);
};
</script>
    <script src="js/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="e2c.js"></script>
    <script src="spa.js"></script>
    <script src="dict.js"></script>
    <script src="navmenu.js"></script>
</body>
</html>
```

在上述程式中，我們透過攔截 window.onbeforeunload 事件，在離開網頁之前儲存想記憶的物件。

```
window.onbeforeunload = function(){
  DB.save('knowWords', knowWords);
};
```

然後在網頁載入時，我們會檢查是否有已經記住的物件，若有則將之載入恢復，這樣就可以透過 localstorage 在網頁中記住較大量的資訊 (記憶量大小各家瀏覽器不同，但大致都在 5MB 以上)，這比傳統的 cookie 大多了，而且 localstorage 不會像 cookie 那樣每次都被放在表頭裡傳回伺服器，因此很適合用來於瀏覽器當中儲存較大量的資訊。

```
function load() {
...
  var dbKnowWords = DB.load('knowWords');
  if (dbKnowWords === undefined)
    knowWords = {};
  else
    knowWords = dbKnowWords;
}
```

那麼、我們到底用 knowWords 物件來記住甚麼資訊呢？ 關於這點，請讀者仔細觀看下列的 doMT() 程式段落，該函數是用來將 ebox 英文區塊透過 mt() 函數逐字翻譯後，用 ruby 標記顯示中英對照在 ebox 當中，於是讀者才能看到中文在上英文在下的對照翻譯。

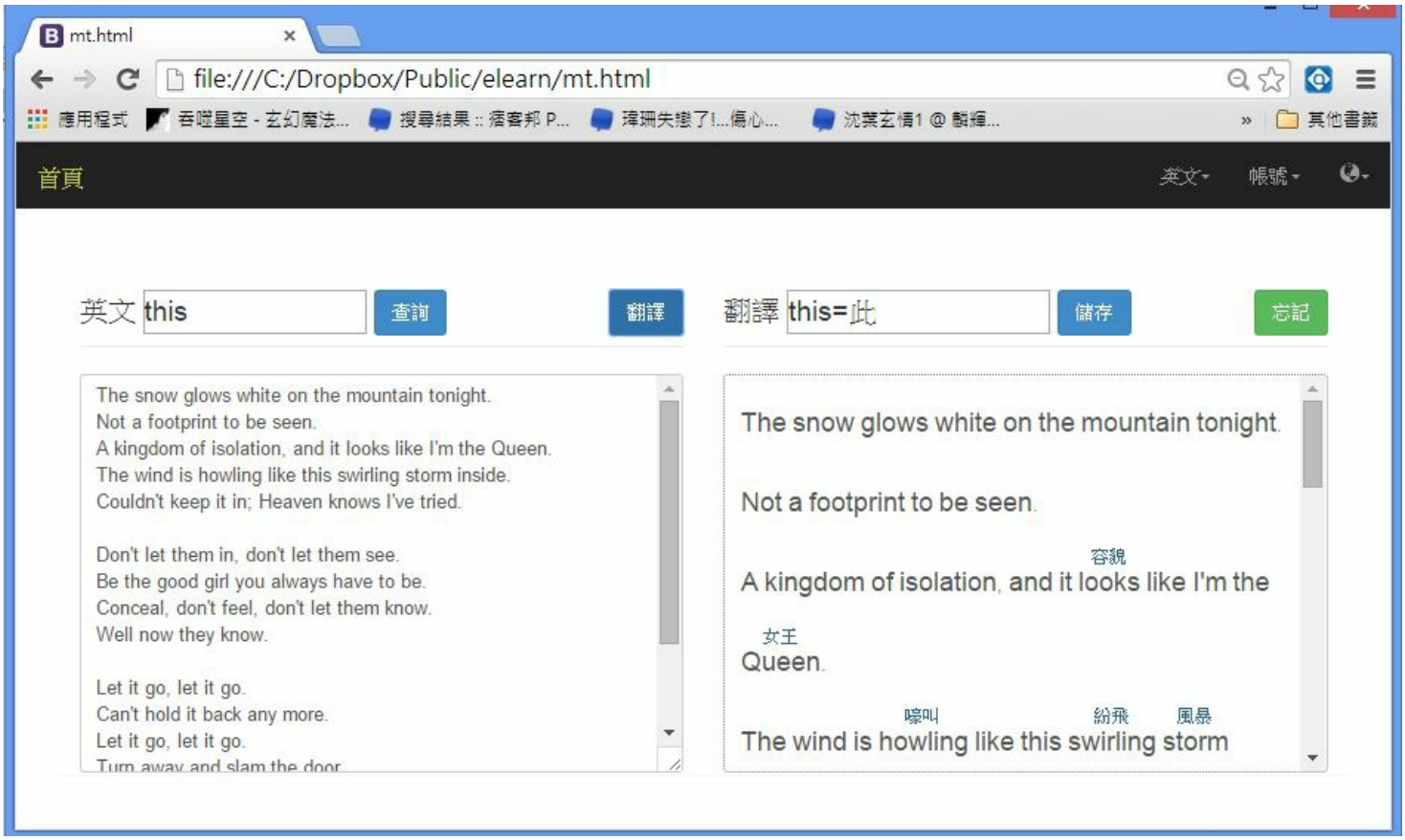
```
function mt(str) {
  var re = /([\w']+)/gi;
  var toStr = "";
  var si = 0;
  while (m = re.exec(str)) {
```

```
    var eword = m[1], elower=eword.toLowerCase();
    var cword = dict[eword.toLowerCase()];
    toStr += str.substring(si, re.lastIndex-eword.
length);
    if (cword === undefined || knowWords[elower]
!== undefined) // 已經認識的字詞就不需要再翻譯了
      cword = "";
    toStr += '<ruby class="'+normalize(eword)+'"><
rb>'+eword+'</rb><rt>'+cword+'</rt></ruby>';
    si = re.lastIndex;
  }
  return toStr;
}

function doMT() {
  var cstr = mt(ebox.value);
  cbox.innerHTML = cstr.replace(/\n/g, "<BR/>");
  $('ruby').click(function() {
    var e = $(this).find('rb').text().toLowerCase
();
    var c = $(this).find('rt').text();
    knowWords[e] = c;
    $('#query').val(e);
    $('#queryResult').val(e+'='+c);
    $('.'+normalize(e)).find('rt').hide(); // 將使
```

```
用者點掉的字詞之翻譯隱藏起來。
    });
}
```

當使用者點選某個「中英對照」字詞的時候，代表該使用者已經認識該字詞了，所以我們會將該字詞記錄在 knowWords 這個字典物件當中，並且隱藏該字詞的翻譯，如此當使用者認識的字越來越多，被翻譯的字詞也就會越來越少，於是隨著使用者的進步就可以逐漸完全讀懂整篇原文，而不需要依賴系統的翻譯功能了 (這也是本系統與一般翻譯系統最大的不同點，本系統是幫助學習英文，而不是企圖做出一個很厲害的機器翻譯系統)。



圖、將已經學會的單字點掉後，離開網頁再回來時的情況

上述程式裡的幾個關鍵部分，我們已經用了中文註解進行說明，讀者應該很容易可以看出這幾個關鍵程式碼的功能才對！

透過這種方式，我們可以讓網頁變成一種類似 APP 的應用，而且不需要伺服端的配合。

(當然、如果加上伺服端之後，還可以將這些 localstorage 中的資訊傳回到伺服器永久

記住，這樣即使換了一台電腦，也不會忘記這些資訊了)。

演算法統治世界（AUTOMATE THIS）（作者：研發養成所 Bridan）

> 演算法統治世界 Automate This: How algorithms came to rule the world 行人文化實驗室發行，ISBN 978-986-9028790 克里斯多夫・史坦能（Christopher Steiner）著，陳正芬譯

我把這本書當作小說來看，內容是真實的並且正在進行中，只是有些事還沒那麼顯著，過些年當你發現一些工作被電腦或機器人取代時，不要太訝異，因為這些都是遲早會發生的事。

故事從 1960 年代華爾街開始，一位匈牙利裔的美國人以駭客手法切入股市交易，這就是電腦程式交易的始主，台灣的股市散戶要小心，如果你還是以直覺或聽信消息以手動交易，那你很容易小贏大輸，因為越來越多人採用電腦自動交易，平時研究好某種交易策略，然後就讓程式機器人自動跑，隨時抓取線上交易資料，程式判定狀況後，立即送單交易，就像百米賽跑一樣，別人一聽到槍聲就全力衝刺，而你是看到別人開跑後才起步，已經輸人一截。

演算法就是教電腦如何執行命令的方法，電腦語言就像每個國家的語言，每個物件動作有自己的名詞與動詞，雖然說法不同，但同指相同的事物，因此演算法才是精隨所在。通常精通數學的人在這方面比較吃香，想自動化工作，需要知道事物之間的關係，數學的學習就是這方面訓練的基礎，例如計算矩形面積，就是長方形長寬相乘與面積的關係。書中也簡述演算法與數學的歷史，值得探尋。

在二十一世紀初，已經有一些演算法被實質應用於特殊場合，例如 電影公司掃描未上映電影腳本預測票房結果，音樂公司掃描歌曲預測是否會流行，利用隨機演算法創作動人音樂，想要巴哈風格的古典樂也可以。未來如果有文辭優美的詩詞創作來自電腦產生不要太驚訝。

現在一些眼光先進的投資者，想利用速度取勝，因此投資電腦硬體或相關設備以取得或保持優勢，像美國東西部之間的光纖電纜就是這樣建出來的。

許多人應該知道 1997 IBM 深藍 (Deep Blue) 打敗世界西洋棋王卡斯帕洛夫 (Garry Kasparov)，2011 IBM 另一新作華生 (Watson) 在猜謎電視節目「危機重重!」(Jeopardy!)

擊敗所有人類對手，現在不同的演算法分別侵入撲克牌賽、體育賽事、挑選潛力選手、當中情局分析師、交友配對等。

美國器官捐贈配對、電腦斷層掃描分析也漸漸轉成電腦處理，未來如果遇到電腦取代醫生問診，不要太訝異，因為電腦的誤診率比人類醫生還低。美國 NASA 在登月計畫能後來居上贏過蘇聯，其中一樣就是精準分類人格特質 (有六類：感情驅動、思考基礎、行動導向、反思導向、意見導向、反應型)，讓個性相合的人在一起工作，避免因成員衝突導致任務失敗，現在已經有公司依據客戶性格將客服電話轉接到適合的人員處理。

在 2008 年前，許多一流的演算法人才被華爾街企業蒐羅，但在 2008 秋天雷曼兄弟因次級房貸崩潰後，人才開始往矽谷移動，其中臉書就內藏許多演算法，隨時記錄你使用臉書的習慣，從中擷取精華提供合適的廣告以獲取利益。想想看全球超過十億人口隨時使用臉書，大家隨便 PO 個文貼個圖點個讚，那資料要怎麼存怎麼用，這沒有厲害的數學高手寫程式是沒辦法處理海量資料的。

未來是屬於演算法創作者的世界，只要願意花點時間思考訓練自己，想想可以利用電腦程式幫你做些事情，這樣你才能控制程式，而不是被機器人取代。

(本文來自「研發養成所」 Bridan 的網誌，原文網址為 http://4rdp.blogspot.tw/2014/12/automate-this.html ，由陳鍾誠編輯後納入程式人雜誌)

雜誌訊息

讀者訂閱

程式人雜誌是一個結合「開放原始碼與公益捐款活動」的雜誌，簡稱「開放公益雜誌」。開放公益雜誌本著「讀書做善事、寫書做公益」的精神，我們非常歡迎程式人認養專欄、或者捐出您的網誌，如果您願意成為本雜誌的專欄作家，請加入 程式人雜誌社團 一同共襄盛舉。

我們透過發行這本雜誌，希望讓大家可以讀到想讀的書，學到想學的技術，同時也讓寫作的朋友的作品能產生良好價值 － 那就是讓讀者根據雜誌的價值捐款給慈善團體。 讀雜誌做公益也不需要有壓力，您不需要每讀一本就急著去捐款，您可以讀了十本再捐，或者使用固定的月捐款方式，當成是雜誌訂閱費，或者是季捐款、一年捐一次等都 OK！甚至是單純當個讀者我們也都很歡迎！

本雜誌每期參考價：NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體。例如可捐贈給「羅慧夫顱顏基金會 彰化銀行(009) 帳號：5234-01-41778-800」。(若匯款要加註可用「程式人雜誌」五個字)

投稿須知

給專欄寫作者： 做公益不需要有壓力。如果您願意撰寫專欄，您可以輕鬆的寫，如果當月的稿件出不來，我們會安排其他稿件上場。

給網誌捐贈者： 如果您沒時間寫專欄或投稿，沒關係，只要將您的網誌以 [創作共用的「姓名標示、非商業性、相同方式分享」授權] 並通知我們，我們會自動從中選取需要的文章進行編輯，放入適當的雜誌當中出刊。

給文章投稿者： 程式人雜誌非常歡迎您加入作者的行列，如果您想撰寫任何文章或投稿，請用 markdown 或 LibreOffice 編輯好您的稿件，並於每個月 25 日前投稿到程式人雜誌社團 的檔案區，我們會盡可能將稿件編入隔月1號出版程式人雜誌當中，也歡迎您到社團中與我們一同討論。

如果您要投稿給程式人雜誌，我們最希望的格式是採用 markdown 的格式撰寫，然後將所有檔按壓縮為 zip 上傳到社團檔案區給我們， 如您想學習 markdown 的撰寫出版

方式，可以參考 看影片學 markdown 編輯出版流程 一文。

如果您無法採用 markdown 的方式撰寫，也可以直接給我們您的稿件，像是 MS. Word 的 doc 檔或 LibreOffice 的 odt 檔都可以，我們 會將這些稿件改寫為 markdown 之後編入雜誌當中。

參與編輯

您也可以擔任程式人雜誌的編輯，甚至創造一個全新的公益雜誌，我們誠摯的邀請您加入「開放公益出版」的行列，如果您想擔任編輯或創造新雜誌，也歡迎到 程式人雜誌社團 來與我們討論相關事宜。

公益資訊

公益團體	聯絡資訊	服務對象	捐款帳號
財團法人羅慧夫顱顏基金會	http://www.nncf.org/ lynn@nncf.org 02-27190408分機232	顱顏患者 (如唇顎裂、小耳症或其他罕見顱顏缺陷）	銀行：009彰化銀行民生分行 帳號：5234-01-41778-800
社團法人台灣省兒童少年成長協會	http://www.cyga.org/ cyga99@gmail.com 04-23058005	單親、隔代教養.弱勢及一般家庭之兒童青少年	銀行：新光銀行 戶名：台灣省兒童少年成長協會 帳號：103-0912-10-000212-0