

讓使用者全盤托出

不曉得小生這副溫文儒雅、謙謙君子的派頭，能否拿到這位美人的電話號碼……可能需要一點身家調查，你知道吧？

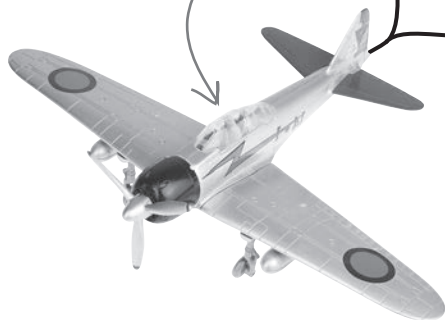


有了 JavaScript，你不用風度翩翩或鬼鬼祟祟，一樣能成功取得使用者的資訊。但你必須千萬小心。人類很容易犯錯，也就是說，線上表單提供的資料，不見得全都準確或合格。JavaScript 能幫上一點忙。把輸入的表單資料傳給 JavaScript 碼，可讓網路應用程式更可靠，也能為伺服器減少一些負擔。珍貴的頻寬應讓給重要事物，例如精彩的影片或可愛寵物的照片。

Bannerocity : 飛翔的廣告

Howard 是名熱愛特技表演的飛行員，他最近的熱忱轉向空中標語事業，Bannerocity。Howard 希望為「橫幅廣告」(banner ad)帶來全新意義——他要在網上接受空中標語的訂單。除了開啟新事業，Howard 希望線上訂單系統能幫他省下待在地上的時間，盡情於天際遨翔。

Howard 擁有一架二戰年代的老式飛機。



Bannerocity...banner ads in the sky!

Howard 現有的訂單格式，已經具備所有空中標語的必須資訊。

Message: Duncan's Donuts... only the best!

ZIP code: 74129

Fly date: December 14th, 2008

Name: Duncan Glutenberg

Phone #: 408-555-5309

Bannerocity 的線上訂單系統，必須捕捉所有和空中標語相關的訂購資訊，這很重要。Howard 認為，線上訂單應該包含現有紙本訂單的每個欄位，另外加上電子郵件欄，因為客戶將在網路上填寫訂單。

Message (標語訊息)

客戶想用空中標語呈現的訊息。

Fly date (飛行日期)

載著標語飛上天的日期。

Name (姓名)

客戶姓名。

ZIP code (郵遞區號)

呈現訊息的地理區域。Howard 會於飛過指定郵遞區號地域時亮出標語。

Phone number (電話號碼)

客戶的電話號碼。

Email (電子郵件)

客戶的電子郵件。

Bannerocity 的 HTML 表單

向 HTML 尋求了一點協助，Howard 踏上 Bannerocity 線上訂購的第一步，看來還不錯。

Bannerocity - Personalized Online Sky Banners

Enter the banner message:

Enter ZIP code of the location:

Enter the date for the message to be shown:

Enter your name:

Enter your phone number:

Enter your email address:

Done

哇！這份線上訂單很有模有樣嘛！

全新的 Bannerocity 訂購表單具有一切必要欄位，也在沒有 JavaScript 的情況下準備好接受訂單。有什麼不對勁嗎？

Howard 目前為私人機構服務，不過他就是很想念以前那套制服。



請至 <http://www.headfirstlabs.com/books/hfjs/> 下載，準備上陣吧！

磨筆上陣



用 Howard 的 HTML 表單填寫一份訂單吧。別擔心，我們不會跟你索取空中廣告的費用啦！

Enter the banner message:

Enter ZIP code of the location:

Enter the date for the message to be shown:

Enter your name:

Enter your phone number:

Enter your email address:

磨筆上陣 解答

你不知道 Howard 的空中標語只裝得下 32 個字元，這一欄太長了。

用 Howard 的 HTML 表單填寫一份訂單吧。別擔心，我們不會跟你索取空中廣告的費用啦！

郵遞區號太長了——應該只有五位數。

Enter the banner message: Mandango... the movie seat picker for tough guys!

Enter ZIP code of the location: 100012

Enter the date for the message to be shown: March 11, 2009

Enter your name: Seth Tinselman

Enter your phone number: (212) 555-5339

Enter your email address: setht@mandango

還好，姓名沒問題。

這是個無效的電子郵件位址——缺少 .biz 這類的網域名稱。

電話號碼的格式應該是 ###-###-####，不採用括號。

日期並未採用表單預期的 MM/DD/YYYY 格式。

當只有 HTML 還不夠的時候

Howard 發現線上表單只能提供資料輸入的管道，他還需要 JavaScript 協助維持表單資料的可靠性。他也需要清楚告知使用者，什麼樣的資料，才構成了「好資料」。舉例來說，Bannerocity 的頁面上需要一些提示，使用者才會知道空中標語需限定在 32 個字元以內，或是日期應該採用 MM/DD/YYYY 格式等等。

不好意思，標語文字僅限 32 個字元。

Enter the banner message: Mandango... the movie seat picker for tough guys!

借用一點 JavaScript 的協助，可阻擋不合格的資料。

HTML 表單通常不會講話吧！

但有個小問題。如果 Howard 不知道如何讓 JavaScript 存取表單資料，JavaScript 資料操縱碼再聰明伶俐也無用武之地……

存取表單資料

為了取用輸入表單的資料，首先需要區分表單中每個欄位。這點可利用 HTML 碼的 `id` 或 `name` 屬性（或兩者並用）處理。

`id` 屬性可獨一無二地識別網頁組件。

`name` 屬性可獨一無二地識別表單中的欄位。

```
<input id="zipcode" name="zipcode" type="text" size="5" />
```

以上兩個屬性均成為 `input` 欄位的識別字。

Enter ZIP code of the location:

表單欄位具有兩種識別方式的原因，均與取用表單單位的途徑有關。第一種途徑使用 `getElementById()` —— 可取用網頁上任何組件的函式。這個方式沒問題，但還有更簡單、更針對表單設計的途徑。

每個表單欄位都有一個 `form` 物件，可被傳給任何驗證表單資料的函式。

```
<input id="zipcode" name="zipcode" type="text" size="5" onclick="showIt(this.form)" />
```

`form` 物件厲害的地方，在於它也是個陣列，負責儲存表單中所有欄位。但它的陣列元素並非利用數值索引儲存；而是使用欄位獨有、於 `name` 屬性設定的識別字。假設 `form` 物件以引數 `theForm` 為名、傳給某個函式，則輸入郵遞區號欄位（ZIP code）的值將以下列方式存取：

`form` 物件的引數名稱。

```
function showIt(theForm) {
  alert(theForm["zipcode"].value);
}
```

表單欄位的獨有名稱，如 `<input>` 標籤的 `name` 屬性所設定。

我們希望儲存欄位值，而不是欄位本身。

呈現出郵遞區號欄位值。

100012

OK

利用 `name` 屬性的途徑，與使用 `getElementById()` 函式並無高下之分，只差在它形成的原始碼比較簡短易讀。既然 `form` 物件提供了捷徑，我們就不要客氣，好好利用吧。



問：為什麼每個表單欄位都能取用 form 物件？

答：有時候不行，但請記得，表單欄位能用於呼叫「需要取用其他表單欄位」的驗證函式。此時，「每個表單欄位裡都能取得 form 物件」成為方便取用其他表單欄位的關鍵。這個物件通常傳入驗證函式，讓函式快速抓出需要的欄位。我們的 Bannerocity 範例將繼續重用 form 物件，從訂購表單中取用欄位。

問：value 是表單欄位的一個特性（property）嗎？這表示每個表單欄位其實都是一個物件嗎？

答：沒錯，正是如此。對 JavaScript 碼，每個表單欄位均表現為一個物件，form 物件則提供了快速而輕鬆地取用這些物件的方式（以取得表單中的任何欄位）。一旦你用 form["objectname"] 拿到表單欄位物件，再加上 value 特性就能取用欄位值。第 9 章和第 10 章還會進一步討論物件。

聽起來，如何取用表單資料，對於在 JavaScript 裡確認資料正常與否是很重要的。但是，該怎麼知道何時檢查資料呢？

檢查表單資料的時機，取決於選擇處理正確的使用者輸入事件。

對於「何時」驗證資料的答案，與事件有關；另外，還要知道讓你察覺「使用者何時輸入資料至欄位」的事件。換句話說，挑戰在於回應「於資料輸入後立刻觸發」的事件。但問題還沒有答案……究竟是哪個事件？

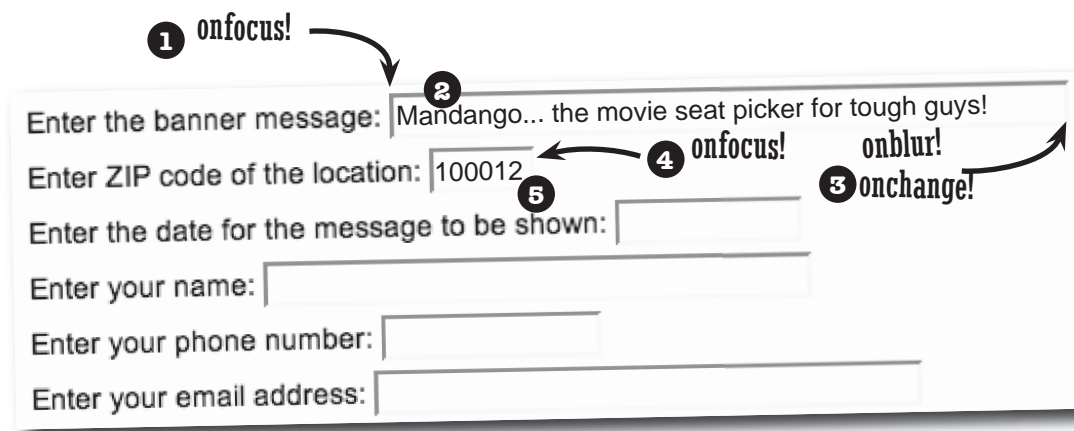


表單欄位帶來一連串事件

當資料輸入表單時，產生了一連串事件。你可以根據欄位基礎，使用這些事件做為驗證欄位資料的進入點。我們先仔細觀察一般輸入資料時的順序，並理解發生了**哪些事件**…與**何時**發生。

- ❶ 選擇輸入的欄位 (onfocus)。
- ❷ 在欄位裡輸入資料
- ❸ 離開該欄位，移向下個目標 (onblur/onchange)。
- ❹ 選擇下個輸入欄位 (onfocus)。
- ❺ 在欄位輸入資料…… (下略)

輸入資料至表單，
點燃一系列有趣的
JavaScript 事件。



一開始選擇輸入的欄位時，發出 `onfocus` 事件；當欄位不再被選擇輸入時，則發出 `onblur`。`onchange` 事件與 `onblur` 有點相似，但它只在某個欄位不再被選擇而且輸入內容被改變時發生。



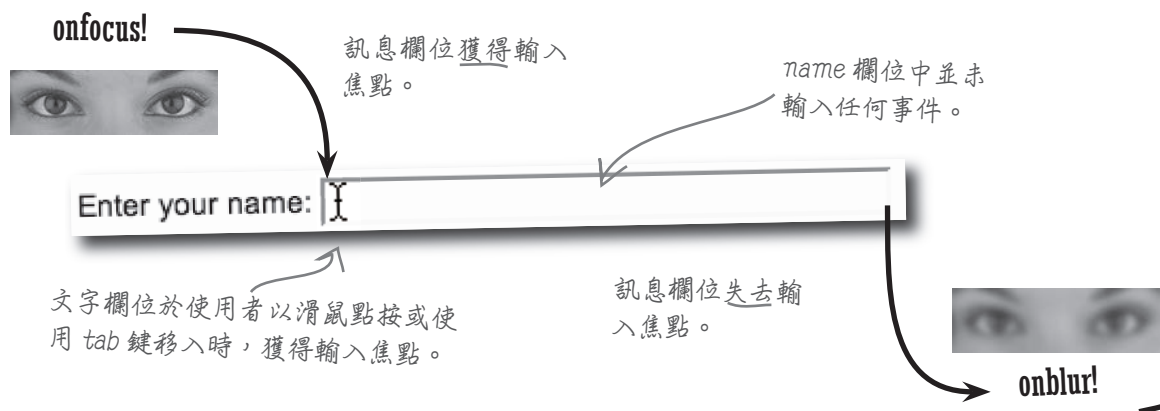
重動重動腦

哪個事件最適合用於驗證表單欄位的資料？

不再是焦點：onblur

雖然關於使用 `onchange` 事件於資料驗證尚有爭論，其中尤以「`onchange` 不可用於驗證空欄位」特別值得注意。表單初次載入時，完全沒有任何欄位資料，就算使用者瀏覽了每個空欄位，但因為表單資料並未改變，`onchange` 將無法觸發。`onblue` 事件能解決這個問題，每次輸入選擇（`input selection`，或稱 **focus**、焦點）的離開，都會觸發這個事件。

onblur 事件極度適合觸發資料驗證。



不像 `onchange`，只要欄位不再是焦點，即觸發 `onblur`，就算並未觸及資料。也就是說，`onblur` 非常強大，但也表示你必須小心處理資料驗證問題的告知方式與時機。例如說……`alert` 方塊，即可能是個容易、但有風險的驗證告知備案。



問：在使用者實際輸入資料的時候，不會產生某些事件嗎？

答：當然會。有許多回應鍵盤行為的事件，像是 `onkeypress`、`onkeyup`、`onkeydown`…等等。雖說有些狀況適合回應這些事件，但通常不適合驗證，因為這些事件觸發的時間點，使用者多半還在輸入資訊。利用鍵盤相關行為於驗證資料，將顯得有點操心過頭，像是一下子提醒使用者打錯了字、一下子提醒資訊輸入不完整等等。最好等到使用者離開欄位；離開欄位表示資料的輸入已經完畢，這項行為則由 `onblur` 事件負責回應。

問：`onblur`，這個事件名稱好奇怪。它是什麼意思？

答：`onblur` 其實是 `onfocus` 的相反事件。如果 `onfocus` 在組件或表單欄位取得輸入焦點時觸發，`onblur` 則於欄位失去焦點時觸發。雖然「`focus`」在此並非意指視線的聚焦，但「`blur`」還是代表失去焦點。這是 JavaScript 的文字遊戲，後來變得有點讓人搞不清楚。總之，請記得 `onblur` 在欄位失去焦點時觸發。

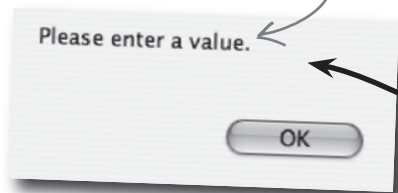
可使用 alert 方塊呈現驗證訊息

alert 方塊實在很適合對使用者快速地呈現資訊，也正好是最簡單、讓使用者知道表單資料有問題的通知形式。如果在處理 onblur 事件時，偵測到有問題的表單資料，呼叫 alert() 函式就對啦！

檢查表單欄位是否空白。

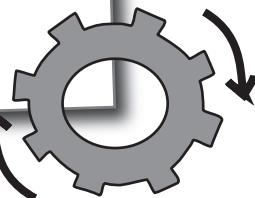
呼叫驗證函式，檢查姓名資料。

指示使用者修正問題資料。



```
function validateNonEmpty(inputField) {
  // See if the input value contains any text
  if (inputField.value.length == 0) {
    // The data is invalid, so notify the user
    alert("Please enter a value.");
    return false;
  }
  return true;
}
```

既然 name 欄位空白，應呈現 alert 方塊。



磨筆上陣



依照下圖的輸入順序，將產生多少個 onblur 事件？多少個 onchange 事件？先別煩惱 onfocus 的問題。

Enter your name:

Enter your phone number:

Enter your email address:

onblur 事件的數量：

onchange 事件的數量：

磨筆上陣 解答



依照下圖的輸入順序，將產生多少個 `onblur` 事件？多少個 `onchange` 事件？先別煩惱 `onfocus` 的問題。

Enter your name: **onblur!**
onchange!

Enter your phone number: **onblur!**

Enter your email address: **onblur!**
onchange!

`onblur` 事件的數量：.....³.....

`onchange` 事件的數量：.....².....

麻辣夜話



今晚主題：`onblur` 與 `onchange` 討論處理壞表單資料的時機

onblur:

最近，指令稿好像總在擔心使用者的行為。我們兩個大概就被叫來解決問題的吧！

我就是想跟你討論這個問題。據說世面上有些空資料欄位四處流竄，很多人都把這件事的矛頭指向你。

確實如此。沒人懷疑你在資料改變時的可靠性。問題在於，如果表單剛開始是空白資料，而且後來一直沒有改變呢？

onchange:

沒錯。我們實在是很忠實的好朋友，總是隨時待命，讓有需要的人知道組件或表單欄位不再是焦點，或某些資料已被改變……還可以兩件事都通知哦！

這項指控真的有點嚇到小人了。你也知道，凡是資料有改變的狀況，我一定會通知指令稿，就算天塌下來我也會盡忠職守。

onblur:

你說得對，一點都不合理，某些使用者也覺得不合理，但他們就是把矛頭指向你身上了。

冷靜一點，放輕鬆，沒事的。這不是你的錯。聽著，資料沒被改變的問題，不是你的責任。記住，你的名字是 **onchange**。

我們先不要想得那麼遠。就像我剛講的，事情的責任不在你身上。如果指令稿驗證資料時還要擔心欄位是否空白，實在不該用你觸發驗證碼。

振作一點啊！反應不要那麼激烈。雖然你可能不適合觸發驗證碼，但也不表示指令稿對資料的改變與否完全沒興趣。想想看允許使用者編輯資料、並另外儲存的表單？你在這種情況下，就很適合當個把關者，只儲存真正有改變的資料。

當然啦！不要再一直打擊你自己了。

別客氣。好了，我還想再多聊一點，不過我得去驗證資料了……回頭見！

onchange:

你說使用者居然可以送出包含空白欄位的表單？根本不合理嘛！

好吧……現在情況是有一個剛開始為空白欄位的表單，但使用者跳過某些資料不輸入，而送出仍然有空白欄位的表單……天哪、天哪！噢、我的心臟！我的心臟！

可是我們剛剛不是才討論了一個很可怕的情況嗎？我對空白資料的管理讓指令稿失望了，結果整個宇宙因為這個隙縫開始崩壞了……

呼……聽起來好多了，就算表示我再也沒有用處了。等一下，我好像又開始緊張了……

嗯，對耶。所以我還是有用囉？

謝謝。我現在安心多了。

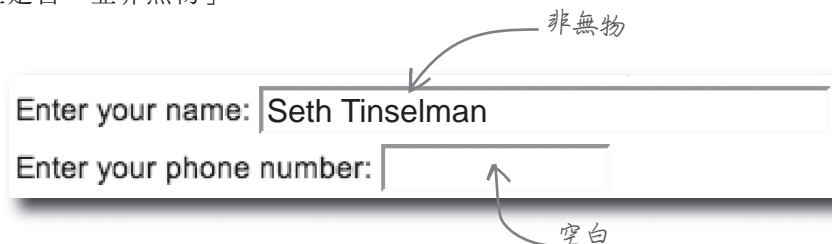
走吧！無物可看

檢查……某些東西

回到 Bannerocity 的生意上。Howard 曉得，他的 Bannerocity 表單，至少需要驗證所有欄位都有資料。但從 JavaScript 的角度來看，這件事要從一個很奇怪的觀點入手。說得更清楚一點，我們並非檢查欄位是否具有某物（something），而是確認欄位是否**並非無物**。換句話說，「有某物」等於「非無物」。

有某物 = 非無物

如此反直覺思考的原因，就在於檢查「空白」的表單欄位，比檢查「填滿」的表單欄位更簡單。所以，資料驗證防禦網的第一行，就是檢查欄位是否「並非無物」：



Howard 的驗證函式，必須回應每個表單欄位的 `onblur` 事件，以執行非無物驗證。如下所示：

表單欄位均有獨一無二的辨識字，因此能從指令稿的其他部分取用欄位。

```
<input id="phone" name="phone" type="text" size="12"
      onblur="validateNonEmpty(this);" />
```

呼叫 `validateNonEmpty()` 函式，回應 `onblur` 事件，以檢查欄位是否並非無物。

表單欄位物件使用關鍵字 `this` 傳至函式。

上例原始碼使用了關鍵字 `this`，參考表單欄位本身。把表單欄位當成物件傳送給驗證函式，提供函式取用表單欄位值，還有取用（儲存所有表單欄位的）`form` 物件的機會，有時候相當好用。

確認欄位「並非無物」的驗證

每個表單欄位均有連接 `onblur` 事件與 `validateNonEmpty()` 函式的相似程式碼。藉由聯繫每個欄位的 `onblur` 事件與函式，表單的所有資料才可接受驗證。

```
<input id="name" name="name" type="text" size="32"
  onblur="validateNonEmpty(this);"/>
```

表單欄位 `phone` 呼叫驗證函式，確認是否為電話號碼。

表單欄位 `name` 呼叫驗證函式，確認是否為姓名。

```
function validateNonEmpty(inputField) {
  // See if the input value contains any text
  if (inputField.value.length == 0) {
    // The data is invalid, so notify the user
    alert("Please enter a value.");
    return false;
  }
  return true;
}
```

`length` 特性以字串格式公布字元數量。

函式找不到電話號碼，故呈現 `alert` 方塊並回傳 `false`。

因為有姓名，所以函式回傳 `true`...

Please enter a value.

OK

就本例而言，`validateNonEmpty()` 函式的回傳值並未使用。其目的在於回頭對呼叫函式的程式碼溝通結果：如果資料沒問題，回傳 `true`；如果有問題，回傳 `false`。稍後，我們將看到這些回傳值如何用於確認表單資料沒有問題，而後才送出表單，交給伺服器處理。

非無物的驗證函式，
確認表單欄位並非
放著空白不管。



動動腦

你能想到任何使用 `alert` 方法提醒使用者「表單資料有誤」的缺點嗎？

不使用煩人 alert 方塊的驗證

Howard 很快就發現，alert 方塊並非通知使用者「資料不合格」的理想方式。他接到很多來自線上填寫 Bannerocity 訂單的使用者的抱怨電話，大家紛紛抗議一直彈出來的 alert 方塊。當彈出的 alert 方塊干擾了線上體驗時，我們多少都會因此覺得火大，資料驗證也不例外，雖然此時的 alert 方塊是為了幫助使用者而出現。



Howard 解決方式，就是採用「被動輔助系統」，它不會使用 alert 方塊，因此不會干擾資料輸入的動線。然而，被動的使用者通知方式，需要新增幾個新的 HTML 表單組件。

新的 HTML 組件，提供了呈現驗證輔助訊息的地方。

Enter your phone number: Please enter a value.

新加入的 HTML 輔助組件，呈現優於 alert 方塊的大幅改善，它們不會阻擋使用者，但仍能傳遞相同的訊息。就結構而言，我們只需加入 HTML 的 `` 組件，且其名稱與代表的表單欄位相同。在網頁的表單原始碼中，新加入的 `` 標籤，就在輸入欄位的下面。

傳給 `validateNonEmpty()` 的第二個引數，現在隨著輔助組件一同傳送。

```
<input id="phone" name="phone" type="text" size="12"
      onblur="validateNonEmpty(this, document.getElementById('phone_help'));" />
<span id="phone_help" class="help"></span>
```

`` 標籤原為空白，但它具有 ID，可與電話號碼的表單欄位相聯繫。

樣式類別 (class) 用於設定說明文字為「紅色斜體字」(不過從我們的印刷上看不出紅色)。

這兩個 ID 必須一致，輔助訊息才能根據輸入欄位而呈現。

有了 `span` 組件管理輔助說明，只剩下實際呈現輔助訊息的原始碼還沒出現。根據上例出現的 `validateNonEmpty()` 函式的第二個引數，這個函式極可能負責確認使用者收到了輔助訊息。

更精密的非無物驗證

Howard 製作的智慧被動輔助方案，出現在全新改良過的 `validateNonEmpty()` 函式裡，新函式現在也為表單欄位設定和清除輔助訊息。

`helpText` 物件做為傳入函式的第二個引數。

```
function validateNonEmpty(inputField, helpText) {
  // See if the input value contains any text
  if (inputField.value.length == 0) {
    // The data is invalid, so set the help message
    if (helpText != null)
      helpText.innerHTML = "Please enter a value.";
    return false;
  }
  else {
    // The data is OK, so clear the help message
    if (helpText != null)
      helpText.innerHTML = "";
    return true;
  }
}
```

首先確認 `helpText` 組件的
存在 (`helpText != null`)，
再把輔助訊息指定給
`innerHTML` 物件。

在使用者完成表單欄位
輸入後，清除輔助訊息
也是重要的一環。

好多了……沒有 `alert` 方
塊……比較不會打擾使用者。

Bannerocity 的資料驗證已有大幅改善，都要感謝新加入的被動式驗證法，它依舊使用有益的 JavaScript 處方，但乾淨許多，至少在使用者體驗的流程上較為乾淨。

Enter ZIP code of the location: Please enter a value.
 Enter the date for the message to be shown: Please enter a value.
 Enter your name:
 Enter your phone number: Please enter a value.
 Enter your email address: Please enter a value.

缺少資料時，Bannerocity
現在改為呈現被動式輔助
訊息。

有了姓名資料，
所以它不會出
現輔助訊息。



一切都合適嗎？

太多也是個問題

結果，非無物的驗證運作確實良好，但太多資料與太少資料都可能帶來問題。請看 Howard 最新收到的標語訂單，它突顯出了 Bannerocity 訂單的新問題。



尺寸很重要…

Bannerocity 遇到的困難，在於 Howard 的空中標語只能容納 32 個字元，但訂單上的訊息欄位卻沒有加上限制。當然，使用者能夠收到「尚未填入標語訊息」的警告是件好事，但太長的訊息仍然可以通過驗證，不受阻擋。這對 Howard 卻是天大的問題！

使用者輸入太多文字，但 Bannerocity 不會提示訊息有問題。

Enter the banner message: Mandango... the movie seat picker for tough guys!

Mandango... the movie seat picker fo

因為文字超過標語能容納的數量，廣告標語就被切斷了……糟糕！

想在有限的空間呈現無限的訊息，不可能成功，而且只會讓客戶不高興……就像現在的 Seth 與 Jason。對訊息欄位加上最大長度的限制，才是解決之道。最好也針對新的驗證機制，特別製作一段輔助說明，確保使用者知道訊息長度的限制。

標語文字需在 32 個字元以內。

Enter the banner message: Mandango... macho movie tickets!

加上尺寸限制後，文字剛好能填入空中標語裡。

Mandango... macho movie tickets!



磨筆上陣

請設計一段虛擬碼，呈現新的 Bannerocity 長度驗證函式如何運作，請確認最小和最大長度都獲得驗證。

.....

.....

.....

.....

磨筆上陣 解答

Bannerocity 標語訊息的函式引數 `minLength` 與 `maxLength`，分別設定為 1 與 32。

請設計一段虛擬碼，呈現新的 Bannerocity 長度驗證函式如何運作，請確認最小和最大長度都獲得驗證。

```
If (fieldValue is shorter than minLength OR fieldValue is longer than maxLength) ...
  Show the help text
Else
  Clear the help text
```

驗證資料長度

新打造的 `validateLength()` 函式，用於檢查並確認表單資料值是否遵守特定最大長度與最小長度。在 Bannerocity 的案例中，這個函式主要限制標語訊息欄位的長度，不過它也同時限制最小長度為 1 個字元。Howard 大概不會遇到只想看到孤零零的「L」在天上飛的客戶吧！不過，重點在於確認該欄位不會大於 32 個字元，不會小於 1 個字元。

除了以 `validateLength()` 強制規定最小與最大長度，函式也需要驗證另外兩個引數，即為輸入欄位 (`inputField`) 與呈現輔助說明的文字訊息。所以，這個函式共需四個引數。



```
validateLength(minLength,
  maxLength, inputField,
  helpText);
```

maxLength

輸入欄位允許的最大長度。

Enter the banner message: Please enter a value 1 to 32 characters in length.

minLength

輸入欄位允許的最小長度。

inputField

需要驗證長度的輸入欄位。

helpText

呈現輔助訊息的組件。

```
<input id="message" name="message" type="text" size="32"
  onblur="validateLength(1, 32, this, document.getElementById('message_help'))" />
<span id="message_help" class="help"></span>
```

標語訊息輸入欄位的物件。

`validateLength()` 函式接受 `inputField` 引數值，而後檢查該欄位的長度至少與 `minLength` 相同，但小於 `maxLength`。如果欄位長度值太短或太長，則以 `helpText` 組件呈現輔助訊息。



複習要點

- 每個表單欄位均可做為 JavaScript 物件而取用。
- 在表單欄位物件裡，有個 `form` 特性，使用陣列表示了**整份表單**的欄位。
- `onblur` 事件，於輸入焦點離開某個欄位時發生，它是觸發資料驗證函式的絕佳方式。
- `alert` 方塊是種很煩人的驗證問題通知方式。
- 被動式驗證輔助比較直覺，也比較不會騷擾使用者。
- 字串 `length` 特性可顯示字串包含的字元數量。



磨筆上陣

請完成 `validateLength()` 函式碼，請務必仔細注意傳入函式的引數。

```
function validateLength(minLength, maxLength, inputField, helpText) {
    // See if the input value contains at least minLength but no more than maxLength characters
    .....

    // The data is invalid, so set the help message
    .....

    // The data is OK, so clear the help message
    .....

}
```

磨筆上陣 解答



請完成 `validateLength()` 函式碼，請務必仔細注意傳入函式的引數。

```
function validateLength(minLength, maxLength, inputField, helpText) {
    // See if the input value contains at least minLength but no more than maxLength characters
    if (inputField.value.length < minLength || inputField.value.length > maxLength) {
        // The data is invalid, so set the help message
        if (helpText != null)
            helpText.innerHTML = "Please enter a value " + minLength + " to " + maxLength +
                " characters in length.";
        return false;
    }
    else {
        // The data is OK, so clear the help message
        if (helpText != null)
            helpText.innerHTML = "";
        return true;
    }
}
```

檢查表單欄位值的最小與最大長度。

設定輔助訊息，反映欄位長度的問題。

如果欄位長度沒問題，則清除輔助訊息。

訊息問題解決了

解決了標語長度的問題，Howard 總算鬆了口氣。既然買不到更長的布條，他也沒有更好的解決方式，所以在 JavaScript 層級處理標語長度問題，變成很好的方式。至少使用者下單前，就知道 Bannerocity 的標語長度有限了。

標語訊息若超過限制，現在會呼叫輔助訊息。

Enter the banner message: Please enter a value 1 to 32 characters in length.



問：利用 alert 方塊的驗證到底錯在哪裡？大多數人不是都能分辨 alert 方塊與彈出式廣告（popup ad）嗎？

答：或許大多數人都知道 JavaScript 的 alert 方塊不是彈出式廣告，但無損於 alert 方塊被認為高度干擾的事實。任何需要使用者停下手邊工作，並按下其他視窗上某物的設計，都是分裂性的。雖然 alert 方塊在 JavaScript 程式設計中佔有一席之地，但資料驗證不是它出場的地方。

問：關於在 onblur 的部分使用 this 代表欄位，我還是覺得一頭霧水。表單欄位就是個物件嗎？或者表單本身才是物件？

答：兩個答案都對。在 HTML 組件的情境中，關鍵字 this 指向代表該組件的物件。若在表單欄位物件的情境中，有個 form 特性可把整份表單當成物件存取。所以，當你看到 this.form 出現在表單欄位的 onblur 碼時，其實它用於參考表單（物件）本身。

Bannerocity 裡 this.form 的用途，在於取得（與某個表單相關的）輔助訊息組件。還記得 this.form 是對 form 物件的參考嗎？它也是個包含所有表單欄位的關聯式陣列（associative array）。所以，假設欄位名稱為 my_field 時，你可以把陣列加入取用欄位的原始碼中，如 this.form["my_field"]，即可快速存取。你也可以使用 getElementById()，但利用表單的方式比較簡捷。

問：當輔助訊息與輸入欄位相關時，它們擁有的 name 與 id 屬性分別是什麼呢？

答：輔助訊息組件的 id 屬性，需根據相關輸入欄位的 id/name 而定，但又不完全一樣。更精確地說，輔助訊息的 ID 會使用輸入欄位的 ID，只是在後面加上 _help。這種命名慣例，是為了在輸入欄位與呈現該欄位輔助訊息的組件間，建立清楚且一致的連結。事實上，你可以隨意命名輔助說明組件的 ID，只要名稱獨一無二，而且能傳入驗證函式就好了。



問：在驗證函式中，為什麼於資料驗證無誤後，清除說明文字是必要行為呢？

答：請記得輔助訊息的重點，是在有問題時協助使用者。如果輸入表單的資料通過檢查，也就是沒有問題，因此不需要呈現輔助訊息。既然輔助訊息可能已經因為前一次的驗證而出現，最好每次通過欄位的驗證時，都清除輔助訊息。

問：如果「輔助訊息」並未當成引數傳給驗證函式，會發生什麼事嗎？

答：指令稿一而再、再而三地搜尋失落的組件，網頁過熱超載，瀏覽器爆炸，剩下一團灰燼……開玩笑的啦！根據設計，如果輔助訊息的引數並未用於驗證函式，Bannerocity 的被動式輔助系統就會靜靜退場。所以輸入欄位的輔助訊息只是不會出現而已。這個方式的好處，就是能隨我們的意願盡量出現或盡量不出現；即使就各個欄位的角度而言，你也不需強制為表單中的每個欄位加上輔助訊息。

檢查 htmlText 引數是否 non-null 的驗證碼，即允許了輔助訊息組件的可選擇性。如果輔助訊息組件不為 null，表示組件存在，能呈現輔助訊息；否則表示沒有組件，什麼事都不做。

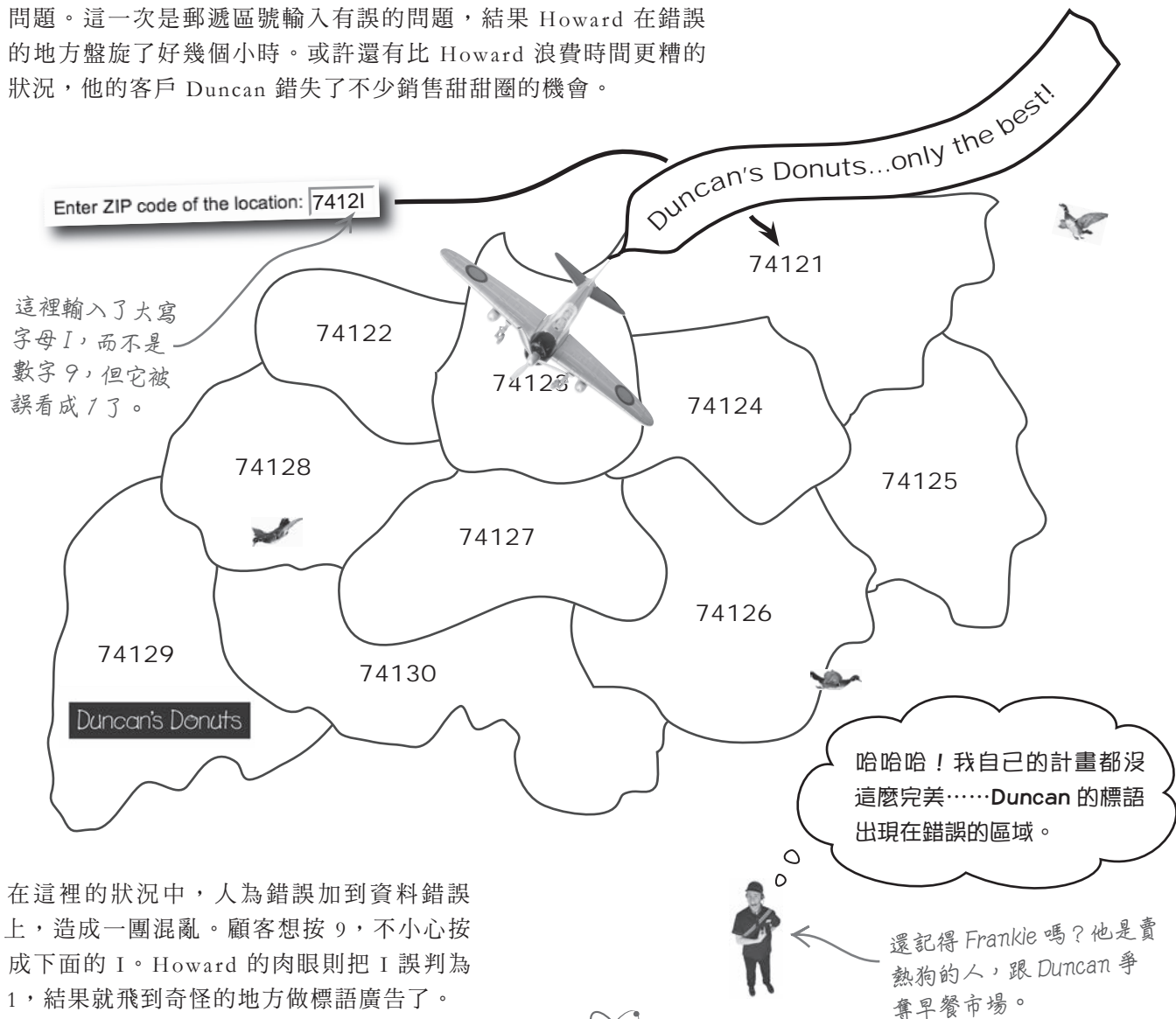
問：HTML 的表單欄位不是有個 size 屬性，難道它沒有限制欄位長度嗎？

答：HTML 的 size 屬性只限制表單欄位在網頁上的實體大小——與輸入資料量的限制沒有關係。舉例來說，Bannerocity 的郵遞區號欄位的 size 屬性設為 5，表示這個欄位在網頁上的尺寸大小，剛好裝得下 5 個字元。使用 HTML 屬性 maxlength，的確可以限制文字的實際長度，但它沒有相對的 minlength 屬性。驗證函式對於控制輸入欄位的字元長度，提供了最大彈性，雖然郵遞區號應該不只確認資料為 5 個字元，還應確認資料是 5 個數字。也許 Howard 應該考慮為 Bannerocity 再加上一些功能……

你在哪裡？

對的標語，錯的地方

雖然 Howard 盡量做到最好的驗證，他的線上表單還是繼續帶來問題。這一次是郵遞區號輸入有誤的問題，結果 Howard 在錯誤的地方盤旋了好幾個小時。或許還有比 Howard 浪費時間更糟的狀況，他的客戶 Duncan 錯失了許多銷售甜甜圈的機會。



在這裡的狀況中，人為錯誤加到資料錯誤上，造成一團混亂。顧客想按 9，不小心按成下面的 I。Howard 的肉眼則把 I 誤判為 1，結果就飛到奇怪的地方做標語廣告了。



你會如何驗證郵遞區號？

驗證郵遞區號

Howard 遇到的問題，與郵遞區號的輸入不正確有關。美國郵遞區號的最簡形式，由 5 個數字構成。所以，驗證郵遞區號可以只是確認使用者輸入了 5 個數字……不多也不少。

 剛好五個
 數字



請完成下列驗證郵遞區號的 `validateZIPCode()` 函式，確保輸入的郵遞區號都是 5 個字元長，而且都是數字。

```
function validateZIPCode(inputField, helpText) {
  // First see if the input value length is anything other than 5
  if ( ..... ) {
    // The data is invalid, so set the help message
    if (helpText != null)
      helpText.innerHTML = "Please enter exactly five digits.";
    .....
  }
  // Then see if the input value is a number
  else if ( ..... ) {
    // The data is invalid, so set the help message
    if (helpText != null)
      helpText.innerHTML = "Please enter a number.";
    .....
  }
  else {
    // The data is OK, so clear the help message
    if (helpText != null)
      helpText.innerHTML = "";
    .....
  }
}
```

磨筆上陣 解答



確認郵遞區號字串的長度是
否不等於 5 (個字元)。

```
function validateZIPCode(inputField, helpText) {
  // First see if the input value length is anything other than 5
  if ( .....inputField.value.length != 5 ..... ) {
    // The data is invalid, so set the help message
    if (helpText != null)
      helpText.innerHTML = "Please enter exactly five digits.";
    .....return false;.....
  }
  // Then see if the input value is a number
  else if ( .....isNaN(inputField.value)..... ) {
    // The data is invalid, so set the help message
    if (helpText != null)
      helpText.innerHTML = "Please enter a number.";
    .....return false;.....
  }
  else {
    // The data is OK, so clear the help message
    if (helpText != null)
      helpText.innerHTML = "";
    .....return true;.....
  }
}
```

既然郵遞區號的長度不是 5，
回傳 false。

isNaN() 函式可檢查值是否
「不為數字」。

既然郵遞區號不是數值，
回傳 false。

回傳 true，指出郵遞區號的
驗證沒問題了。



假設郵遞區號均為數字，並非絕對安全。

如果網路表單能夠接受美國以外地址的郵遞區號，純數值的驗證就不是什麼好主意了。因為很多國家的郵遞區號或許包含文數字的混合。另外，完整的美國郵遞區號應該包含 9 位數，格式為 #####-####，此時，連字號 (-) 將使郵遞區號資料不為數字。

Bannerocity 的驗證函式確實靈巧，但如果使用者無視輔助訊息，就算資料有誤還是照樣按下「Order Banner」鈕呢？表單還是會送至伺服器嗎？



壞資料不該抵達伺服器。

哎呀！儘管有一卡車好意的提醒，如果使用者可以按下按鈕、避開驗證而送出表單，做了這麼多資料驗證碼也是枉然哪！Bannerocity 的致命缺點，就是沒在送出表單時要求驗證，所以壞資料目前還是能被送進伺服器。

如果使用者還是可以選擇送出包含壞資料的表單，這些資料驗證只是白費工夫。

一個真正可靠的應用程式，也會驗證伺服器上的資料，以策安全。

Bannerocity 需要其他函式，它的工作就是驗證所有表單欄位，才能送出表單給伺服器處理。我們自製的 `placeOrder()` 函式與 Order Banner 鈕相聯繫，它將於完成訂單前受到呼叫，並最後一次驗證表單。

```
<input type="button" value="Order Banner" onclick="placeOrder(this.form);" />
```



placeOrder() 函式 靠過來

```
function placeOrder(form) {  
  if (validateLength(1, 32, form["message"], form["message_help"]) &&  
      validateZIPCode(form["zipcode"], form["zipcode_help"]) &&  
      validateNonEmpty(form["date"], form["date_help"]) &&  
      validateNonEmpty(form["name"], form["name_help"]) &&  
      validateNonEmpty(form["phone"], form["phone_help"]) &&  
      validateNonEmpty(form["email"], form["email_help"])) {  
    // Submit the order to the server  
    form.submit();  
  } else {  
    alert("I'm sorry but there is something wrong with the order information.");  
  }  
}
```

函式期待 form 物件做為唯一引數傳入，它才能取用各個表單欄位。

各個表單欄位及輔助訊息組件，均使用陣列表示法、透過 form 物件存取。

函式大致上是個很長的 if/else 敘述，其中為每個表單欄位呼叫驗證函式。

如果表單欄位驗證都通過，則呼叫 submit() 方法以送出表單給伺服器。

既然送出訂單時還有驗證問題，已經重大到值得動用 alert 方塊了。



問：placeOrder() 函式如何控制表單是否送往伺服器。

答：首先，函式中的 if/else 敘述架構為驗證表單中的每個欄位，也就是說，如果任何表單資料不合格，則執行 else 子句 (clause)。else 子句只負責呼叫 alert() 函式，所以，函式如果走到 else 子句，不會再發生其他事情。另外一方面，如果資料驗證沒問題，則呼叫 form 物件的 submit() 方法，用於送出表單給伺服器。所以送出表單的動作，由是否呼叫表單的 submit() 方法所控制，這個方法是 JavaScript 版的「submit」按鈕。

問：我還以為 alert 方塊對驗證有害，它為什麼又出現了？

答：很多時候，是這樣沒錯，但這裡真正的問題，則是何時可以干擾頁面流程，呈現彈出式的訊息 (alert)，要求使用者閱讀訊息並按下「OK」。因為只在使用者想要送出訂單時，才會按下 Order Banner 鈕，值得確認使用者知道資料有問題；而本例的問題嚴重到值得使用者 alert 方塊。別忘記了，被動式輔助訊息仍會呈現在網頁上，引導使用者修補問題。

資料驗證…時機就是一切

很可惜，Howard 對郵遞區號與表單送出驗證的修改，只能帶來一時的輕鬆，因為現在又出現全新的問題。因為有了郵遞區號的驗證，他不再飛過錯誤的地方，但他有時候卻在錯誤的日期廣播標語，這點好像更糟糕。關於輸入飛行日期的地方，一定有哪裡出錯了。

Enter the date for the message to be shown: 05/10/2008

日期欄位應該輸入 9，卻打成附近的 0……好像沒有人能正確地輸入 9。

Howard 把 0 當成 0，他在 10 日飛上天做廣告，而不是在顧客真心想要的 19 日。

Go on a Stick Figure Adventure!

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

奇怪，星期一到了，我的標語在哪裡？

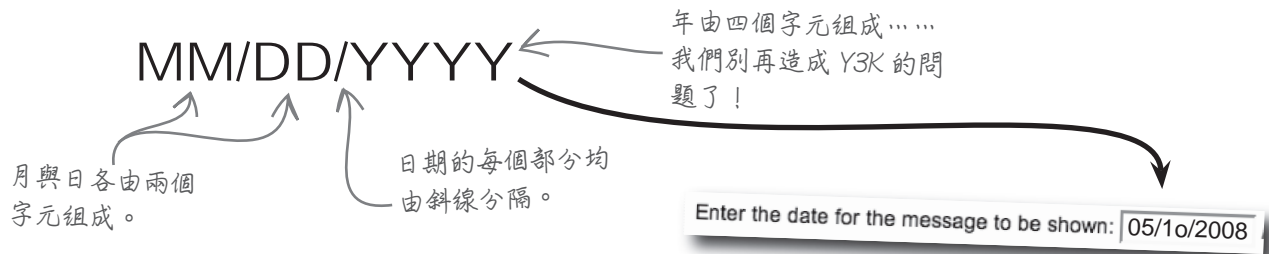
Ellie 一點也不高興。



Howard 該如何驗證表單裡的飛行日期欄位，讓日期符合特定格式（例如 MM/DD/YYYY）呢？

驗證日期

很明顯，Howard 無法只依靠使用者是否輸入日期資料的檢查；他還需要實際確認輸入的日期是否合格。驗證日期的關鍵，取決於指定日期格式，然後強制實行。有種常見的日期格式：先以兩位數表示日，再以兩位數表示月，最後以四位數表示年，其間以斜線區隔。

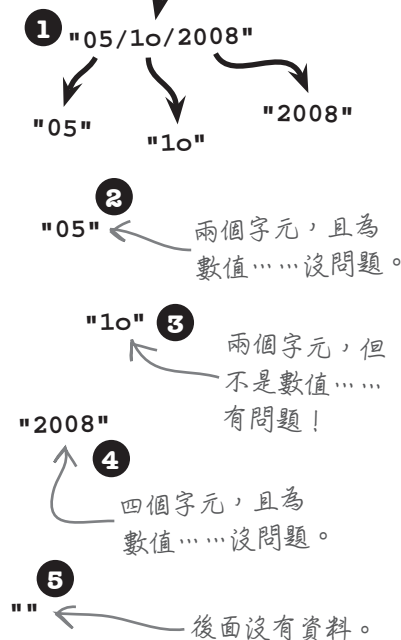


分解日期格式是最簡單的部分……設計出驗證日期是否符合格式的原始碼，才是困難的開始。有些很強大的字串函式，能夠根據特定字元分解字串，例如根據斜線 (/) 分解。但是，分解字串，再確認各個部分均為數值且為指定長度，這項努力實在太複雜了。就像情況最極端的郵遞區號驗證挑戰。

我們逐步分析日期驗證函式的運作方式：

- ❶ 分解表單欄位值，成為許多組子字串 (substring)，以斜線做為分解字串的基礎。
- ❷ 分析「月」子字串，確認剛好只有兩個字元，而且是數值。
- ❸ 分析「日」子字串，確認剛好只有兩個字元，而且是數值。
- ❹ 分析「年」子字串，確認剛好只有四個字元，而且是數值。
- ❺ 忽略其他跟隨在第二個斜線後的資料。

雖然上述步驟還不致於成為編程惡夢，但對於驗證一段簡單的格式而言，好像太複雜了點。



如果有個比分解字串更好的日期驗證方法，該有多好……就讓閨中少女稍微幻想一下吧！



你沒聽過正規運算式嗎？

正規運算式一點都不「正規」

JavaScript 剛好有種非常強大的內建工具 —— 正規運算式 (regular expression, 後文簡稱正規式)，專門設計於比對 (match) 文字樣式 (pattern)。正規式可用於建立樣式，而後套用於文字字串，搜尋符合的部分……就像指認出嫌犯一樣！不過我們的字元合作多了。



比對到了！

樣式 = 高個字、沒戴眼鏡、短髮

正規式用於比對
文字樣式。

樣式包括一個人的
外表屬性。

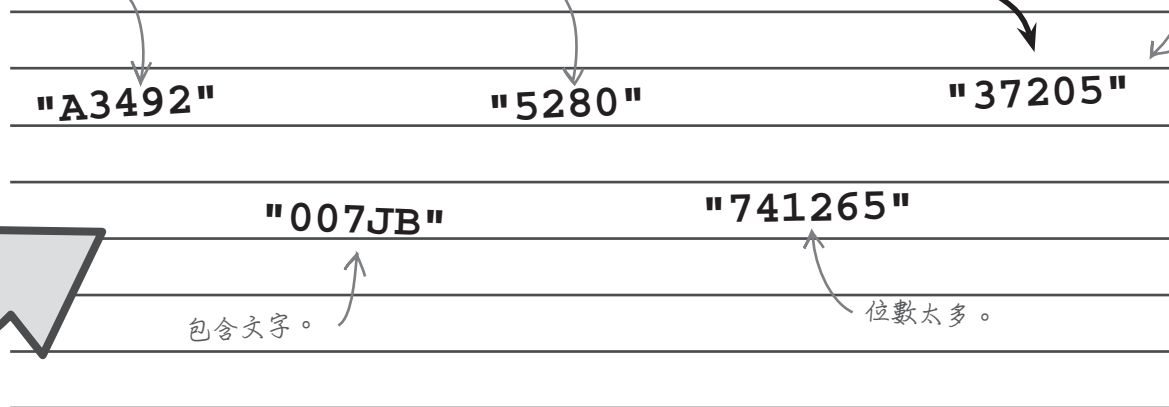
上例的樣式敘述了某個人的外表屬性，然後比對出實際人物。正規式也有相同功用，它能比對字串的樣式。

正規式定義用於比對的樣式

就像警方指認嫌犯時，可能用到身高、髮型或其他外表特徵；文字樣式也有特定的字元順序，例如一系列 5 個數字。等一下，聽起來好耳熟……郵遞區號…嗎？

包含一個文字。

位數不夠。



樣式 = #####

樣式包含一個只有 5 個數字的序列。

比對到了！

很可惜，把 5 位數的郵遞區號轉換為正規式，不太符合直覺。因為正規式在描述文字樣式時，會仰賴一種非常簡潔，但又有點難解的語法。你看右側的正規式範例，很難立刻看出來它用於驗證郵遞區號吧！

字串必須以定義的樣式起始，不可使用數字。

唯一的數字必須重複五次。

樣式 = `/^\d{5}$/`

所有正規式均以斜線圍起。

一個數字。

字串必須以定義的樣式結束。

安靜！我正在辨識樣式。



如果正規式讓你看得頭昏腦脹，別驚慌。

接下來我們還會討論更多驗證實例，你會慢慢瞭解正規式。

揭開正規式的面紗

建立正規式，有點像是建立字串實字 (string literal)，只不過正規式出現在一對斜線 (//) 裡，而不是出現在一對引號裡。



正規式總是以斜線起始和結束。

講到正規式本身，有一組稱為**中介字元** (metacharacter) 的特殊符號，用於連接文字與數字，建立高度描述性的文字樣式。好消息是：建立實用的正規式時，不需要瞭解正規運算式「語言」的每個細微差異。以下是一些常用正規式中介字元：

`.` 對，只是個點號。
比對任何字元，換行字元 (newline) 除外。

`\s` 空格包括空白字元 (space)、tab、換行字元、return/enter。
比對空格。

`\d` 許多中介字元均以反斜線起始……與圍起正規式的斜線非常不一樣。
比對任何數字字元。

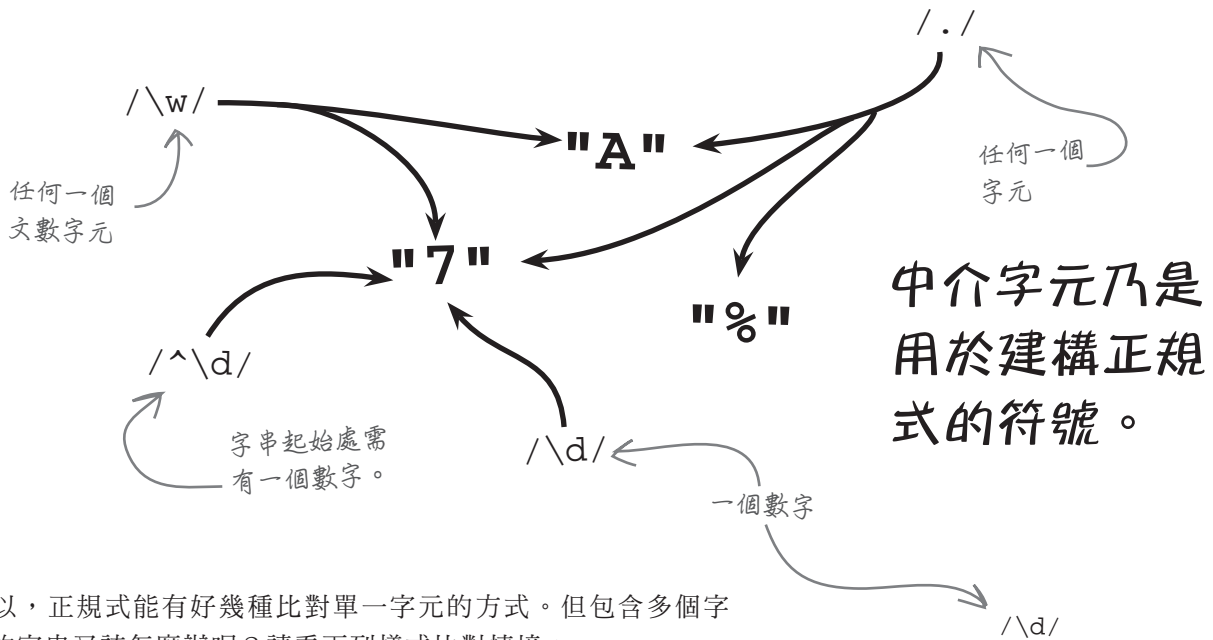
`^` 符合樣式的字串前，不能有其他文字。
字串需以樣式起始。

`\w` 比對任何文數字元。

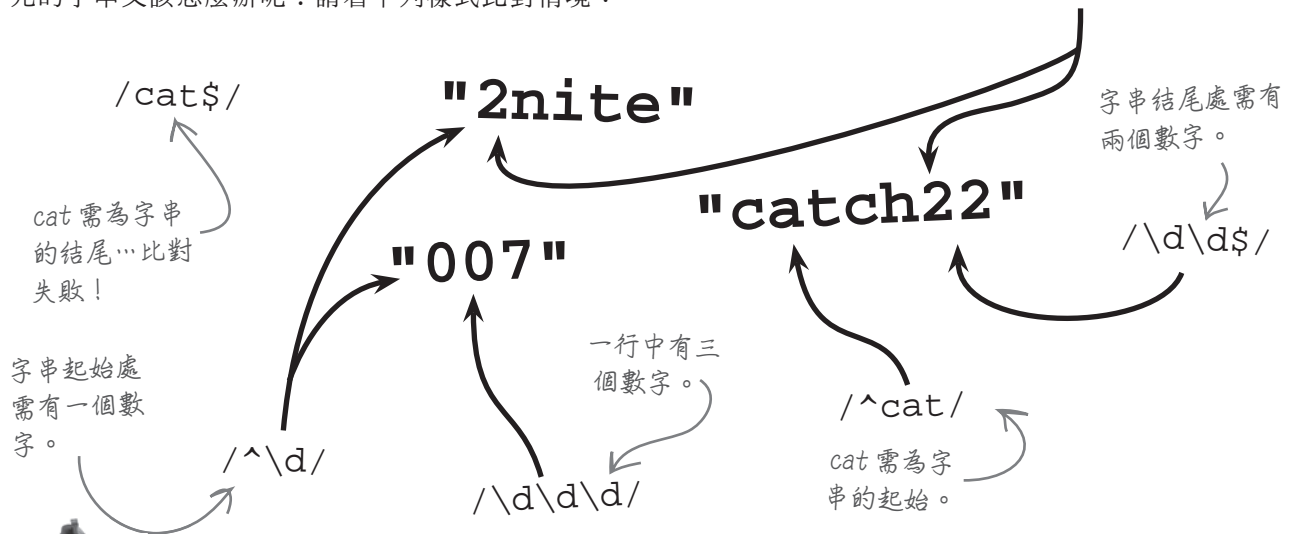
`$` 樣式需為字串的最後一個字元。
字串需以樣式結束。

雖然上述中介字元的描述都很準確，但中介字元放在樣式情境中，其實更容易理解……

中介字元不只表示一個實體字元



所以，正規式能有好幾種比對單一字元的方式。但包含多個字元的字串又該怎麼辦呢？請看下列樣式比對情境：



習題

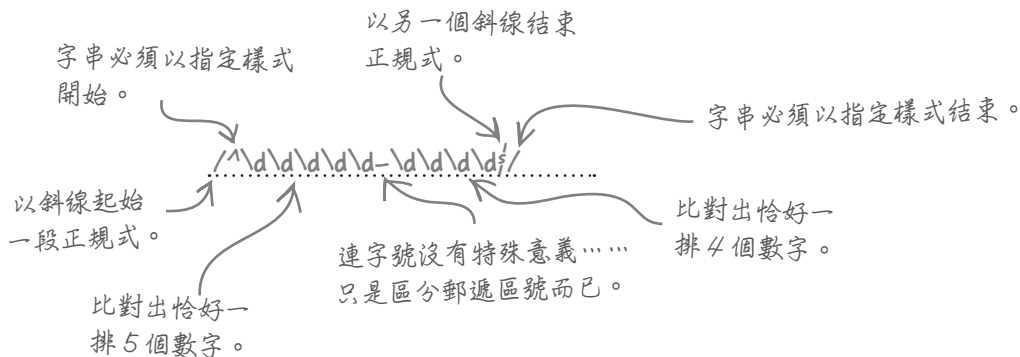
請設計比對完整美國郵遞區號的正規式，格式需為 #####-####，而且只能單獨出現。

.....



習題
解答

請設計比對完整美國郵遞區號的正規式，格式需為 #####-####，而且只能單獨出現。



深入正規式：量詞

不是中介字元的任何文字，將於正規式裡「照樣比對」；也就是說，/howard/ 可找出任何包含「howard」的字串。另外，還有另一種正規式結構，稱為量詞（quantifier），可進一步把樣式調整得更好。量詞前為子樣式（sub-pattern）；量詞即應用在子樣式，並控制子樣式出現在樣式裡的次數。

* ← 子樣式為選用性，可出現任意次數。
量詞前的子樣式必須出現 0 或多次。

{n} ← 控制子樣式可以出現的次數。
量詞前的子樣式必須出現恰好 n 次。

+ ← 子樣式必須出現，可出現任意次數。
量詞前的子樣式必須出現 1 或多次。

? ← 子樣式為選用性，若出現也只能出現一次。
量詞前的子樣式必須出現 0 或 1 次。

雖然括號在技術上不算量詞，但常用於歸類子樣式，就像平常區分數學算式一樣。

() ← 集合字元或 / 和中介字元，成為子樣式。

樣式數量

量詞使得正規式設計比起只有中介字元時更為精確。不再直接重複子樣式，量詞能指定子樣式出現的精確次數。請看使用量詞後的郵遞區號範例：

```
/^\d{5}-\d{4}$/
```

有了 {} 量詞的幫忙，就不用列出每位數字了。

利用中介字元與量詞，可以建立相當強大、幾乎可比對任何字串內容的正規式。

```
/\w*/
```

比對任何文數字元，包括空字串。

```
/.+/
```

任何字元均需出現一次以上……用於比對非無物字串。

```
/(Hot)? ?Donuts/
```

可比對出「Donuts」或「Hot Donuts」。

猜猜我是誰

請把每個正規式中介字元或量詞，與它在樣式中的功用連起來。

.

字串必須以指定樣式結束。

\w

子樣式為**選用性**，且可出現任意次數。

\$

比對任何文數字元。

\d

比對任何字元（newline 除外）。

+

比對任何數字字元。

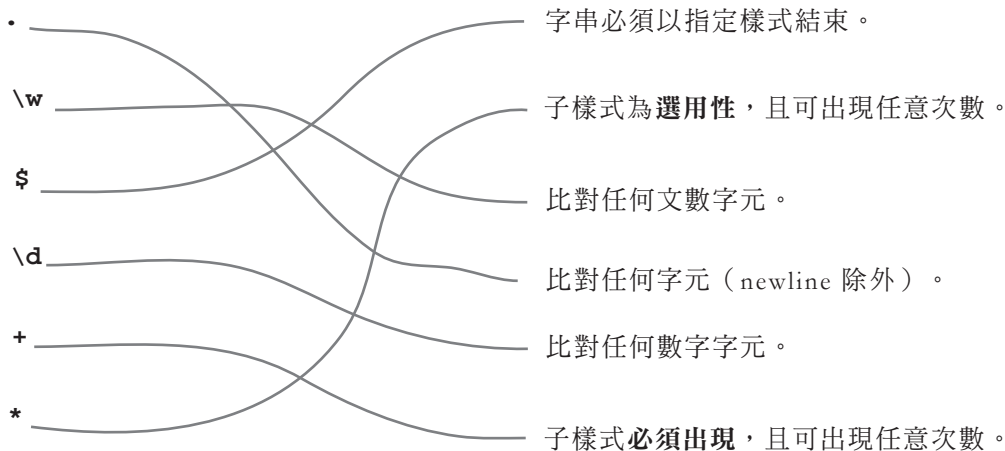
*

子樣式**必須出現**，且可出現任意次數。

量詞控制子樣式出現於正規式裡的次數。

猜猜我是誰

請把每個正規式中元字元或量詞，與它在樣式中的功用連起來。



問：正規式是個字串嗎？

答：不是。你可以把正規式當成「對字串的**描述**」，至少當成對部分字串的描述。正規式與字串緊密相關，且用於比對出現在字串中的文字樣式，但正規式本身不是字串。

問：正規式可以套用在其他類型的資料上嗎？

答：不行，正規式單純設計用於比對文字字串內的字元樣式，所以只能套用在字串上。但這項限制，並未減低正規式處理「只用字串太困難」的複雜文字比對任務時的極度好用程度。

問：如果想比對中介字元，例如 \$ 號，會發生什麼事？

答：與 JavaScript 字串相似，正規式中具有特殊意義的字元，能以反斜線轉義。所以想在正規式裡比對 \$ 時，你必須把原本的 \$ 特別改寫成 \\$。這項規則適用其他在正規式具有特殊意義的字元，例如 ^、*、+ 等等。任何沒有特殊意義的字元，則可直接放入正規式，不需特殊格式。

問：正規式與資料驗證有什麼關係嗎？我們本來不是在驗證 Bannerocity 的日期欄位嗎？

答：耐心一點，年輕的絕地武士。很快就會用到正規式了。是的，這趟小小的正規式岔題之旅，乃是為了尋找驗證複雜格式資料的方式，例如日期或電子郵件位址。Bannerocity 的資料格式前台還需要很多協助，所以有很多套用正規式的機會。請各位耐心地再等候一下。

問：正規式如何用於 JavaScript 裡？

答：我們馬上就要談到了……不騙你！正規式以物件表現於 JavaScript 中，該物件支援數個方法，以利使用正規式比對字串內的樣式。



樣式大師真情指數

本週主題：

神秘難解但魔力強大的正規運算式。

Head First：您……就是轟動萬教、驚動武林、金光閃閃、瑞氣千條，據說可以在字串內做搜尋，也能辨識樣式的神秘客嗎？

正規式：沒錯，我算是某種原始碼中介客，能夠仔細搜尋文字字串，並立刻找出樣式。中情局實在需要我這種人才……不過他們沒有回我的電話。

Head First：欸…你對間諜這一行有興趣哦？

正規式：不是啦！我只是喜歡找出文字間的樣式。事實上，我就是樣式、任何樣式都是我。只需給我一些參數，說明你在尋找的樣式，我就能找出來，至少能讓你知字串中是否包含你要的樣式。

Head First：聽起來很不錯，不過，String 物件的 `indexOf()` 方法不是已經能處理字串搜尋了嗎？

正規式：拜託，你剛才真的提到「那個東西」了嗎……那個業餘的根本不曉得何為樣式。聽好，如果你需要非常簡單、簡單到不行，只是在字串中尋找「濫竽充數」一詞的搜尋功能，你可以用 `indexOf()`。否則，你很快就會發現 `indexOf()` 完全無法嚴格地分析字串。

Head First：字串搜尋不也是樣式比對的一種形式嗎？

正規式：是啊……打開冰箱門也是一種運動，因為有人還沒看過奧運的競技……我的重點是，簡易字串搜尋其實是極度簡化到最簡化的樣式比對——此時的樣式只是個靜態詞彙或片語。你想想看日期或網站 URL 的結構，這些才是真正的樣式；雖然它們遵守嚴格格式，但搜尋的詳細規格卻非靜態。

Head First：我開始懂你的意思了。「樣式」是一組對文字的敘述，這段文字可能出現在字串中，但「樣式」不見得必為文字本身。

正規式：就是這樣啦！就像我請你看到「頭髮短短的、身材高高的、沒戴著眼鏡的人」時通知我一樣。這是一段對外表的敘述，但並非真的是個人。如果有個叫 Alan 的人符合敘述，而且正好走進來，就可以說他符合樣式。但可能也有很多人符合這段敘述。如果沒有樣式，我們就無法根據樣式找人，而必須尋找確定的人物。所以，使用 `indexOf()` 搜尋特定文字，以及使用我比對樣式的差異，就如同直接尋找 Alan，以及尋找一名身材高挑、短髮、沒戴眼鏡的人一樣。

Head First：這樣聽來就清楚多了。但是，樣式比對如何應用在資料驗證上呢？

正規式：這個嘛，驗證資料主要在於確認資料符合「某種預先構想的格式」，也稱為樣式。所以我的工作就是拿到一組樣式，並檢查字串是否符合。如果符合，則資料視為合格 (valid)，否則即為有問題的資料。

Head First：比對不同資料時，也會用到不同的正規式嗎？

正規式：當然啊。在使用我們正規式驗證資料時，找出成功塑造資料格式的正規式，其實才是最費功夫的工作。

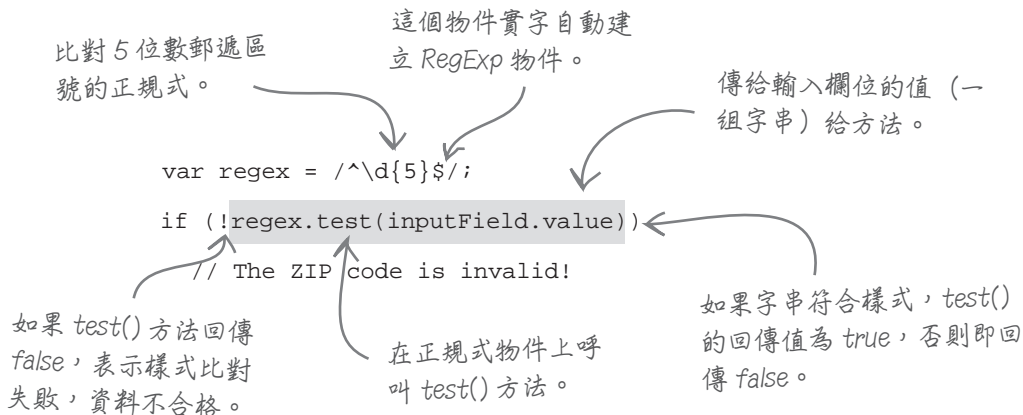
Head First：真是太有趣了。非常感謝你今天為我們說明你在資料驗證中的角色。

正規式：小事一樁。我常常需要自我解釋……這大概是我的行為模式吧！

利用正規式驗證資料

雖然，單純為了找出樣式而建立正規式，好像非常有趣，但我們對正規式另有迫切的需要，我們需要它協助驗證 Bannerocity 的日期欄位，並把 Howard 帶回天空。JavaScript 裡的正規式由 `RegExp` 物件表示，其中包含使用正規式驗證資料的關鍵 —— `test()` 方法，它檢查字串裡是否存在指定的樣式。

RegExp 物件的 `test()` 方法，用於檢查字串是否包含正規式樣式。



雖然也能在每個不同的驗證函式內呼叫 `test()` 方法，但有個機會可建立通用的正規式驗證函式，可供更專化的函式呼叫以執行通用性的驗證工作。下列步驟將由通用的 `validateRegExp()` 函式負責：

- 1 根據當成引數傳入的正規式，對同樣傳入的字串執行檢查。
- 2 如果比對出樣式，把輔助說明指定給（做為引數）傳入的 `helpText`，然後回傳 `false`。
- 3 如果沒有找到樣式，清除說明訊息並回傳 `true`。



```
validateRegExp(regex,
  inputStr, helpText,
  helpMessage);
```

萬事俱備，只欠函式碼。事實上，這個函式碼已有大部分出現在其他驗證函式裡，所以 `validateRegExp()` 其實同時牽涉到原始碼再利用，以及建立全功能的正規式驗證工具。

```
function validateRegEx(regex, inputStr, helpText, helpMessage) {
  // See if the inputStr data validates OK
  if (!regex.test(inputStr)) {
    // The data is invalid, so set the help message and return false
    if (helpText != null)
      helpText.innerHTML = helpMessage;
    return false;
  }
  else {
    // The data is OK, so clear the help message and return true
    if (helpText != null)
      helpText.innerHTML = "";
    return true;
  }
}
```

正規式、輸入的字串、輔助說明的文字、輔助說明組件，均以引數形式傳入。

根據正規式檢驗輸入的字串。

如果檢驗失敗，表示資料不合格，所以要呈現輔助說明。

如果資料檢查沒問題，則清除輔助說明。

磨筆上陣



請撰寫 `validateDate()` 的函式碼，其中呼叫 `validateNonEmpty()` 與 `validateRegEx()` 以驗證 Bannerocity 表單的日期欄位。提示：這個函式接受兩個引數，日期欄位值，以及相關的輔助訊息組件。

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

磨筆上陣 解答

先呼叫 `validateNotEmpty()` 函式，確認欄位並非空白。

請撰寫 `validateDate()` 的函式碼，其中呼叫 `validateNotEmpty()` 與 `validateRegex()` 以驗證 Bannerocity 表單的日期欄位。提示：這個函式接受兩個引數，日期欄位值，以及相關的輔助訊息組件。

```
function validateDate(inputField, helpText) {
    // First see if the input value contains data
    if (!validateNotEmpty(inputField, helpText))
        return false;
    // Then see if the input value is a date
    return validateRegex(/^\\d{2}\\\\d{2}\\\\d{4}$/, inputField.value, helpText,
        "Please enter a date (for example, 01/14/1975).");
}
```

傳遞日期
的正規式給
`validateRegex()`
函式。

因為斜線在正規式裡具有特殊意義，需使用反斜線轉義。

日期正規式使用中介字元與量詞維持 MM/DD/YYYY 的格式。



到了 2100 年，很難說你的指令稿是否還在使用。

能讓使用者只輸入 2 位數的年份，或許也不錯。

Y2100 還早……

既然我們距離下個世紀還很遠，或許使用者只輸入 2 位數年份也沒什麼問題。現實而言，現在寫好的任何 JavaScript 碼不太可能存活九十年，造成世紀交替時的呈現問題。Howard 短暫地考慮了前瞻未來的嚴格驗證版 Bannerocity（維持輸入 4 位數年份），然後決定等到下個世紀再來承擔後果。





問：為什麼 `validateDate()` 需要呼叫 `validateNonEmpty()` 函式？正規式不是也考慮到了空資料嗎？

答：是的，正規式確實天生就會驗證是否為空資料，移除非無物的部分，日期驗證一樣能正常運行。不過，先檢查日期是否輸入，網頁對使用者會顯得比較直覺，因為頁面上提供了各個驗證問題的輔助說明。如果沒有輸入資料，它造成的訊息將不會是輸入不合格資料時的訊息。感覺上，我們的被動式輔助系統，似乎能引導使用者完成輸入表單的過程。這種微妙的可用性強化，或許值得稍微多用一點原始碼。

問：如果我真的想考慮指令稿的未來呢？會是個問題嗎？

答：不會，完全不會。企圖預先考慮未來的需求，並設計能夠適應這些需求的程式碼，很少成為問題。以 `Bannercity` 為例，要求填入 4 位數年份的日期欄位，絕對比 2 位數的版本更能高枕無憂。也請記得，如果你真的想要點手段，也可以讓使用者只輸入 2 位數，但在背後加上代表世紀的前兩位數。所以，表單呈現效果是 2 位數年份，但實際儲存的日期卻是 4 位數年份。

比對最少次數與最多次數

量詞 `{}` 可接受數字，決定子樣式出現在字串裡的次數。這個量詞還有另一個版本，可接受兩個數字，分別指定子樣式出現在字串裡的最少次數 (`min`) 與最多次數 (`max`)。這個版本為子樣式的出現提供良好微調的便利方式。

控制子樣式可以出現的次數範圍。

`{min,max}`

量詞前的子樣式，必須出現至少 `min` 次，但不可多於 `max` 次。

`/^\w{5,8}$/`

有些密碼可選擇 5~8 字元，此時最適合量詞 `{}`。



日期不只 `MM/DD/YYYY` 一種格式。

假設所有使用者均以 `MM/DD/YYYY` 的格式輸入日期，不見得安全。世界上還有很多地方輸入月、日時的格式相反，他們採用的格式是 `DD/MM/YYYY`。

磨筆上陣



重新設計 `validateDate()` 函式使用的正規式，讓它接受 2 位數和 4 位數的年份。

磨筆上陣 解答



加入 `min/max` 的量詞，可設定日期裡允許出現的年份位數。

重新設計 `validateDate()` 函式使用的正規式，讓它接受 2 位數和 4 位數的年份。

```
/^d{2}\d{2}\d{2,4}$/
```

等一下。新的日期驗證碼，好像也能讓 3 位數年份通過耶？沒道理啊……

日期的正規式也讓 3 位數年份通過比對……不太妙！

Enter the date for the message to be shown: 03/01/200

可以對過去十個世紀加上 JavaScript 支援。

既然不支援過去，自然沒有理由允許使用者輸入年份僅到百位數的日期。事實上，有可能的話，根本不需要讓使用者在過去的時間下訂購空中標語廣告。所以，從驗證碼中減去 3 位數年份是項重要的修補，也將協助 Howard 避免被新浮現的 Bannerocity 資料輸入問題攻擊。

複習要點

- 正規式以樣式比對字串裡的文字；正規式需以斜線圍起。
- 除了一般文字，正規式亦由中介字元與量詞建立，可對文字樣式的比對方式提供精細的控制。
- JavaScript 透過內建 `RegExp` 物件而支援正規式，但正規式通常建立為 `literal`，所以很少見到這個物件。
- `RegExp` 物件的 `test()` 方法用於對字串套用正規式樣式的檢測。

削除 3 位數年份，該使用這個「或」那個

另一個在正規式工具箱中非常好用的中介字元，則是選替（alternation），它跟 JavaScript 的 OR 邏輯運算子非常像。但與 JavaScript 的 OR 邏輯運算子不一樣，選替只用到一個 `|`，而且它的功用是讓樣式指定一連串可供選擇的子樣式。換句話說，這些「可供選擇」的子樣式中，只要有一個比對成功，就算樣式比對成功。這一點很像邏輯 OR 運算子，因為它的意思基本上也是「這個、或這個、或這個……」

`this|that`

如果比對出子樣式 `this` 或 `that`，樣式比對就成功了。

選替 (`|`) 提供指定可供選擇的比對樣式的便利方式。

`/(red|blue) pill/`

簡易的二選一，「`red pill`」或「`blue pill`」都符合這個樣式的比對結果。

`/small|medium|large/`

使用多個選替中介字元，即可指定多種比對可能性。

磨筆上陣



再重新設計 `validateDate()` 函式使用的正規式，這次確認年份只可是 2 位數或 4 位數，沒有其他可能。

真的是你家的電話號碼嗎？

磨筆上陣 解答

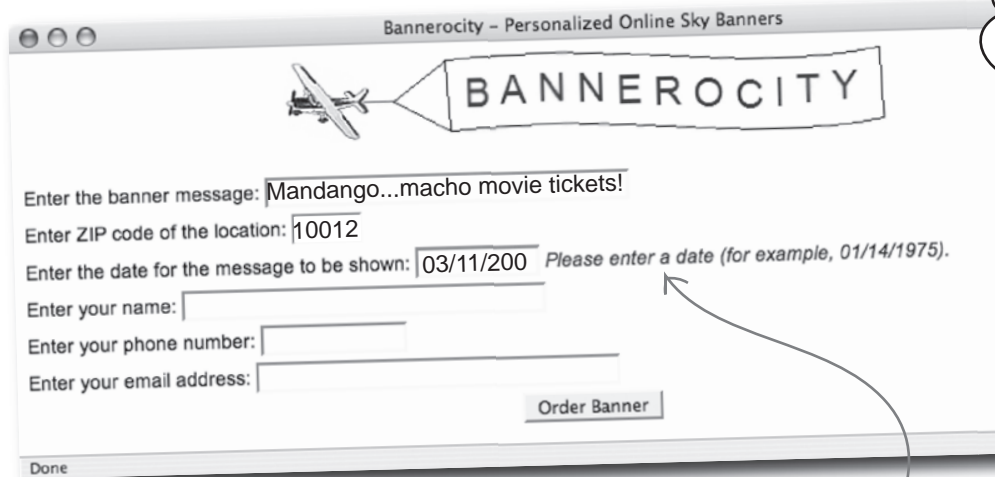
選替中介字元 `()` 讓樣式只接受 2 位數或 4 位數年份。

再重新設計 `validateDate()` 函式使用的正規式，這次確認年份只可是 2 位數或 4 位數，沒有其他可能。


```
/^\\d{2}\\d{2}\\(\\d{2}\\d{4}\\)/
```

不留下任何需要的改變

Howard 真的很喜歡這個使用正規式（以精確比對樣式）、嶄新而牢靠的日期驗證器。事實上，他太喜歡這種驗證方式了，所以他進一步使用正規式，驗證 Bannerocity 剩下的兩個欄位：電話號碼、電子郵件。



Bannerocity - Personalized Online Sky Banners



Enter the banner message:

Enter ZIP code of the location:

Enter the date for the message to be shown: *Please enter a date (for example, 01/14/1975).*

Enter your name:

Enter your phone number:

Enter your email address:

Done

看起來不錯...不過我的要求更高！

表單裡的日期欄位，現在使用正規式驗證，正規式對堅持日期格式非常精確。

Howard 這項驗證 Bannerocity 訂單上電話號碼與電子郵件的想法非常好，但也表示我們將要準備一些新的正規式，才能成功地支配這些資料格式。

你聽到了嗎？電話號碼的驗證

單從驗證角度而言，電話號碼不算太難檢查，它們有固定的格式。當然，沒有正規式的話，還是需要相當程度的字串分解，但正規式使得電話號碼驗證只是小事一樁。美國的電話號碼遵守下列樣式：



樣式 = ###-###-#### ←

既然 Howard 不打算飛離老家太遠，預設採用美國電話號碼格式，應該沒有問題。

把電話號碼樣式裡的連字號 (-) 換成斜線，並稍微調整數字部分的數量，上述電話號碼樣式，顯然與日期樣式非常相似。

日期樣式使用中介字元 \d 與量詞 []，讓日期遵守 MM/DD/YYYY 或 MM/DD/YYYY 格式。

`/^\d{2}\/\d{2}\/\d{2,4}$/`

電話號碼與日期樣式相似，只不過它使用連字號區分不同部分的號碼。

`/^\d{3}-\d{3}-\d{4}$/`

`validatePhone()` 函式變得非常可以預測，都要感謝電話號碼的正規式與 `validateRegExp()` 函式。

```
function validatePhone(inputField, helpText) {
  // First see if the input value contains data
  if (!validateNonEmpty(inputField, helpText))
    return false;

  // Then see if the input value is a phone number
  return validateRegExp(/^\d{3}-\d{3}-\d{4}$/,
    inputField.value, helpText,
    "Please enter a phone number (for example, 123-456-7890).");
}
```

究竟是達康還是打狗？

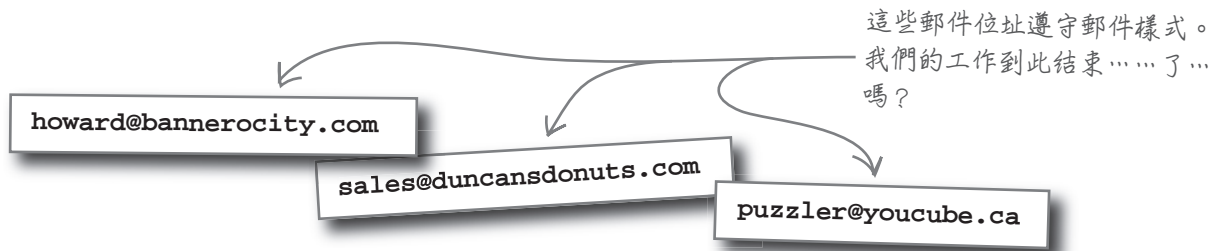
你有一封新郵件：驗證電子郵件

解決了電話號碼驗證的任務，Howard 最後只剩下驗證電子郵件欄位的挑戰。與其他資料一樣，驗證郵件地址的關鍵，在於分解格式，直到成為一以貫之的樣式，能用正規式建立模型。

樣式 = *LocalName@DomainPrefix.DomainSuffix*

2 或 3 個英數字元。

看起來不會很難 —— 電子郵件只是三段英數字元，其中加入 @ 英數字元和點號 (.) 字元。



建立比對上述郵件樣式的正規式非常簡單，畢竟各項組成都很容易預測。

郵件位址必須以一或多個英數字元起始。

點號 (.) 需要轉義，它是正規式裡的特殊字元。

`/^\w+@\w+\.\w{2,3}$/`

郵件位址結尾必須為 2~3 個英數字元。

@ 符號後至少出現一或多個字元。

雖然這個樣式達成了任務，但好像少了什麼。所有郵件位址都會遵守這種可預測的格式嗎？



還有其他可能的郵件樣式嗎？請想想你目前見過、各式各樣的郵件位址。

例外也是規則

郵件位址其實比它的第一眼印象更為複雜。為資料驗證構築可靠的郵件樣式時，其實還要考慮幾種基礎郵件格式的變化版。以下舉出一些完全合格的郵件位址：



cube_lovers@youcube.ca

使用者名稱部分包含底線。

aviator.howard@bannerocity.com

使用者名稱部分包含點號。

rocky@i-rock.mobi

網域名稱的 prefix 部分包含連字號。

網域名稱的 suffix 部分包含四個字元。

i-love-donuts@duncansdonuts.com

使用者名稱部分包含連字號。

我們真的很需要在驗證郵件位址時比對選用字元。

seth+jason@mandango.us

使用者名稱部分包含加號。

ruby@youcube.com.nz

多一個網域 suffix，它只是網域名稱的額外部分。



郵件位址呈現了樣式中需要比對選擇性字元的需求。

結果，我們稍早單純當成英數字處理的部分，有好幾個字元可以點綴在郵件位址的各個部分裡。我們需要合併這些選用字元到樣式裡的方式……

從集中比對選用字元

另一個非常便利、常用，又直接影響郵件樣式的功能，則是字元類組（character class），讓我們能在樣式裡建立受到緊密控制的子樣式。說得更精確一點，字元類組擅長把非常著重選用字元的規則，建立到子樣式中。你可以把字元類組想成一組比對單一字元的規則。

`[CharacterClass]`

字元類組是一組比對單一字元的正規式規則。

字元類組總是以方括號圍起。

在字元類組內，每個列出的字元都是合格的字元比對目標，有點像是中介字元間的選替，能建立可替換的子樣式清單。不過，字元類組的結果，都是對單一字元的比對，除非字元類組後面加上量詞。實際看看範例，比較容易瞭解字元類組的概念。

在正規式的樣式中，字元類組提供控制選用字元的有效率方式。

`/d[iu]g/`

這兩個字串都符合樣式。

"dig"

"dug"

`/\$\d[\d\.]*/`

"\$5"

"\$3.50"

"\$19.95"

這些財務字串都符合樣式。*



別忘記正規式特殊字元需要轉義。

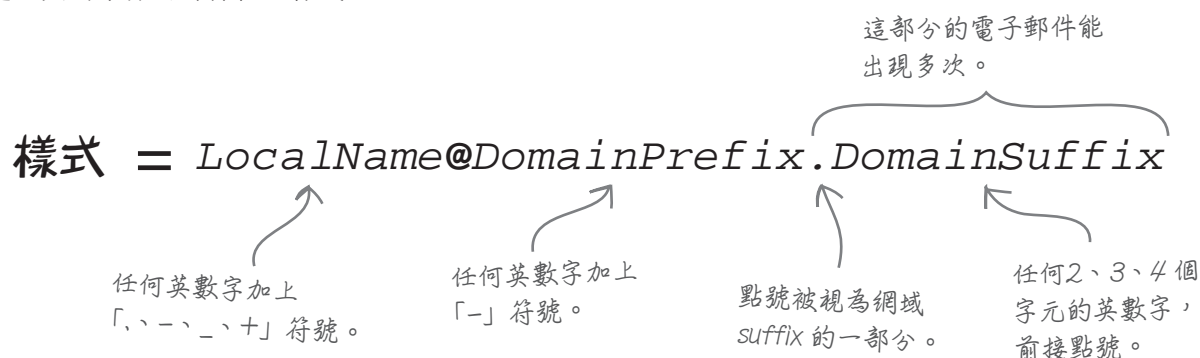
在正規式裡具有特殊意義的字元，需要經過轉義，才能把實際字元放在正規式裡。下列字元前均需加上反斜線（\）做轉義：`[\^$.|?*+()]`。

字元類組正是塑造郵件位址所需的工具，如此一來，也能把郵件驗證加入 Bannerocity……

* 這個樣式並不嚴格，像「\$5...」這類字串也符合它的樣式要求。

建構郵件驗證工具

加入所有可能出現在使用者名稱與網域名稱的選用字元後，終於可以建立更為牢靠的郵件位址樣式。



還有一點要記得，成功建立樣式的方法有很多種，包括郵件位址樣式。想建立一個成功表現某種資料格式的每個微妙差異的樣式，可能比想像中困難許多。我們已經體驗過，通用樣式設計如果行得通，把它轉換成實際的正規式其實相當簡單。



磨筆上陣

請補上 `validateEmail()` 失落的原始碼，這個函式用於驗證 Bannerocity 的電子郵件位址。

```
function validateEmail(inputField, helpText) {
    // First see if the input value contains data
    if (!.....(inputField, helpText))
        return false;

    // Then see if the input value is an email address
    return validateRegEx(.....,
        inputField.value, helpText,
        .....);
}
```

磨筆上陣 解答

請補上 validateEmail() 失落的原始碼，這個函式用於驗證 Bannerocity 的電子郵件位址。

```
function validateEmail(inputField, helpText) {
    // First see if the input value contains data
    if (!.....validateNonEmpty.....(inputField, helpText))
        return false;
    // Then see if the input value is an email address
    return validateRegExp(.....,
        inputField.value, helpText,
        "Please enter an email address (for example, johndoe@acme.com).");
}
```

validateNonEmpty() 函式仍然先被呼叫，以檢查是否缺少資料。

驗證郵件位址的正規式時，用到我們所學過的大部分正規式技巧。

網域名稱 suffix 可使用 2~4 個英數字元做為字串結尾。

使用者名稱可為英數字元，以及、-、_、+等字元，且需位於字串起始處。

如果郵件位址驗證失敗，即呈現輔助訊息，示範正確的輸入格式。

有了萬全防護的 Bannerocity 表單

Bannerocity 對空中標語訂單的資料收集，已經非常完美，一切都要歸功於一些繃緊神經的驗證工作。

Howard 高興到自已下了一張訂單，飛上空中……

Howard 太激動了！終於又能回到他的最愛……飛行啦！



Data validation is a good thing!

Enter the banner message:

Enter ZIP code of the location:

Enter the date for the message to be shown:

Enter your name:

Enter your phone number: Please enter a phone number (for example, 123-456-7890).

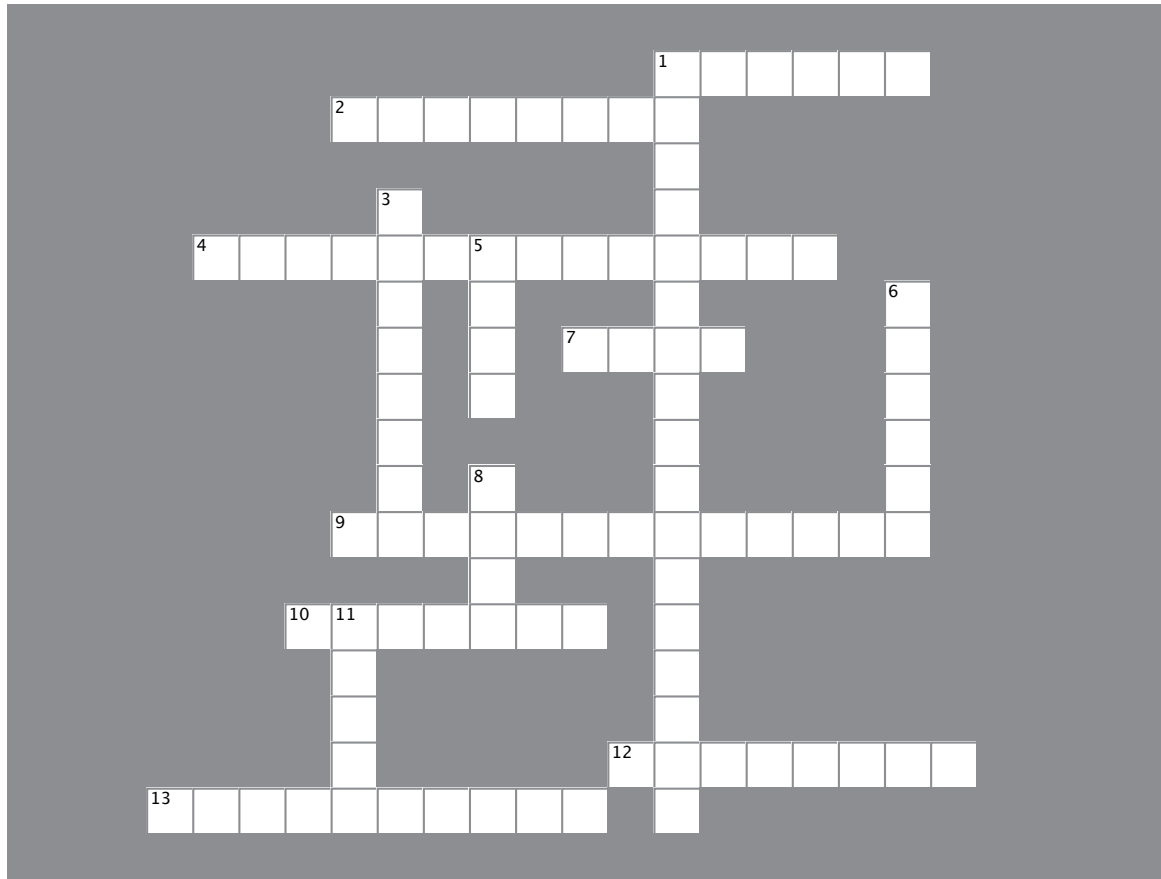
Enter your email address: Please enter an email address (for example, johndoe@acme.com).

電話號碼和郵件位址欄位，現已根據非常嚴格的資料格式接受驗證。



JavaScript 填字遊戲

這個樣式大家應該感到很熟悉……填字遊戲又來了！
現在不用再驗證了，只要填上一些答案。



橫向提示：

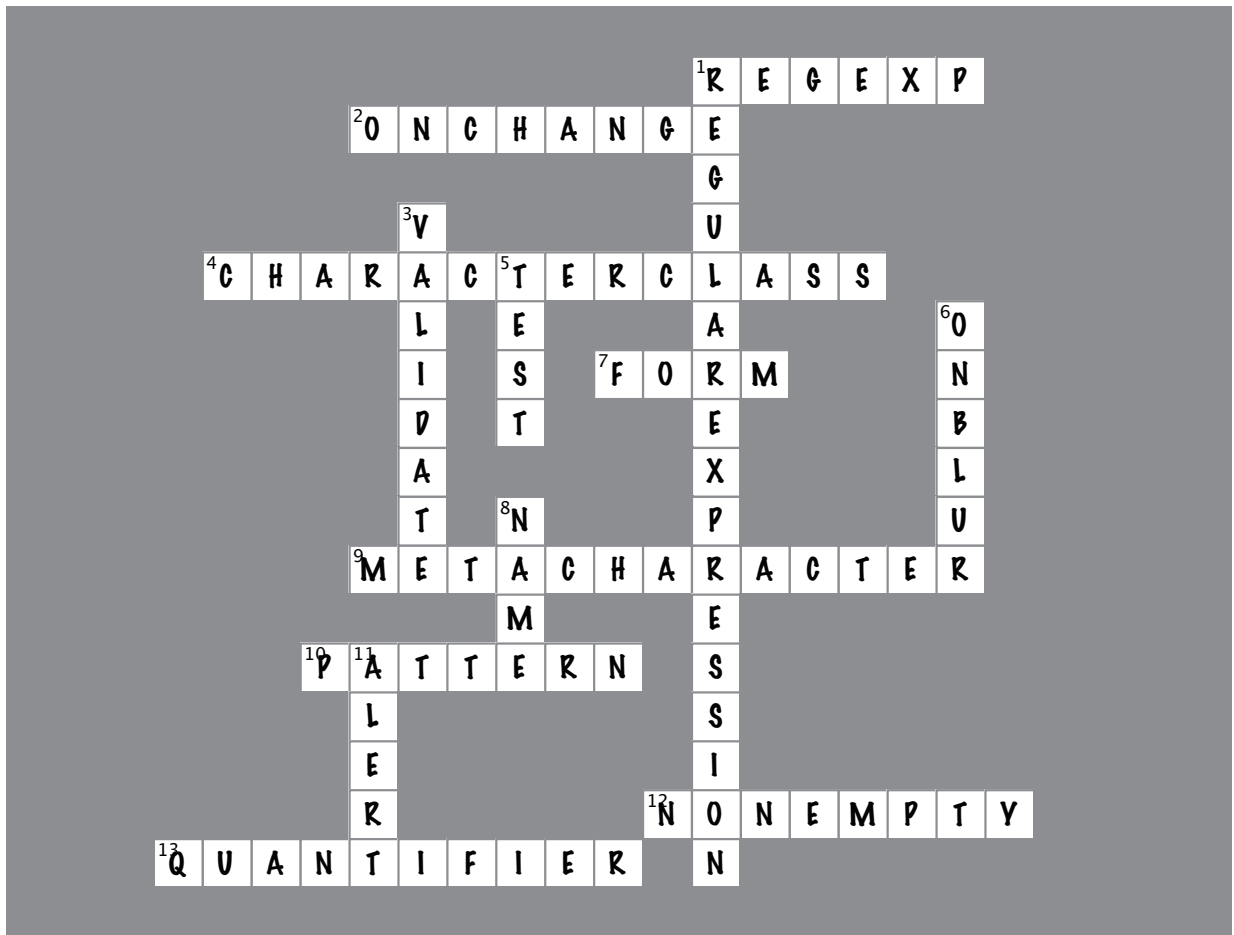
1. 支援正規式的 JavaScript 物件。
2. 當表單欄位的資料改變時，觸發_____事件。
4. 指定正規式中選用字元的便利方式。
7. 這個物件包含表單中各個欄位。
9. 正規式的特殊字元稱為_____。
10. 對資料格式的描述。
12. 確認表單欄位具有資料的驗證是_____。
13. 控制子樣式出現在正規式中次數的方式。

直向提示：

1. 用於比對文字樣式。
3. _____表單資料，確定資料合法。
5. 使用正規式比對字串的方法（method）。
6. 當使用者離開表單欄位時，觸發_____事件。
8. HTML 屬性，能獨一無二地辨識出表單內的欄位。
11. 很多時候都很方便的功能，但通常不是對使用者告知資料不合格的最佳方式。



JavaScript 填字遊戲解答

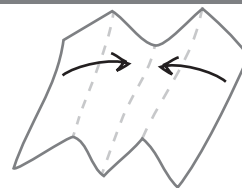


Page Bender

垂直對摺本頁，區分左右 JavaScript 為網路表單帶來什麼貢獻？
腦的範圍，並解決謎題。



心智的秘密會談。



"Mandango...the movie seat picker for tough guys!"

"...macho movie seats!"

105012

100012

03/11/200

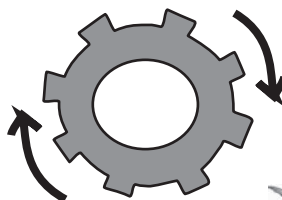
March 11, 2009

212-555-5339

(212) 555-5339

setht@mandango

seth%t@mandango.us



那些資料看起來
七八糟的！！

我覺得還好！感覺
上就像衝浪……



`/^val(ley|ue|krie)/`

`/name|id$/`

JavaScript 對表單有不少貢獻，
所以很難幫任何一項事件
製造合格的引數。
答案幾乎或多或少都牽涉到
資料，但究竟如何牽涉？

