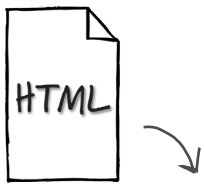
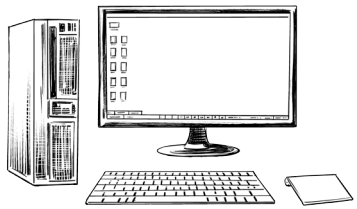
JavaScript 的運作方式

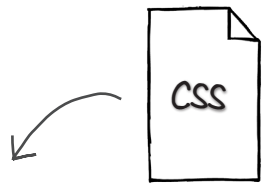
如果你習慣在網頁中建立結構、內容、布局和樣式，那麼是時候也加入一些行為了吧？畢竟，我們不需要只是坐著不動的頁面。有意思的網頁應該是互動（*interactive*）且動態（*dynamic*）的。那正是 JavaScript 發揮作用之處。讓我們先來看看 JavaScript 在網頁生態系（*web page ecosystem*）中的定位：



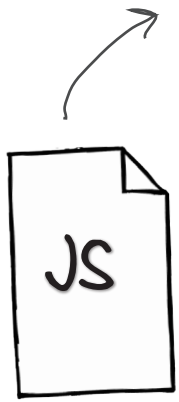
你使用 HTML（Hypertext Markup Language，超文本標記語言）指定網頁的所有**內容（content）**及其**結構（structure）**，例如段落、標題和區段。



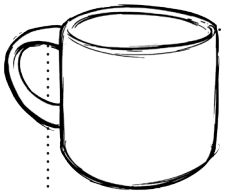
瀏覽器



使用 CSS（Cascading Style Sheets，階層式樣式表）來規範 HTML 呈現的方式，包含頁面上的顏色、字體、邊框、邊距和頁面布局。CSS 為你提供**樣式（style）**，並且透過與頁面結構分離的方式進行。



JavaScript —— HTML 和 CSS 的運算表親——讓你在網頁上建立**行為（behavior）**。需要在使用者點選「30 秒限時特價」按鈕時做出反應嗎？即時檢查使用者的表單輸入？取得來自 NASA 的資料並顯示？還是玩遊戲呢？請把目光轉向 JavaScript。JavaScript 提供一種方式，將程式碼加到你的頁面中，以便進行運算、反應、繪圖、通訊、警告、修改、更新、變動等，我們還可以一直列下去…任何的動態運算都是 JavaScript 在執行。



放輕鬆

放輕鬆。此時，我們不期望你閱讀 JavaScript 時就像從小就接觸它一樣。事實上，我們現在要你做的只是對 JavaScript 長怎樣有個感覺。

話雖如此，你也不能完全置身事外。因為我們得先幫你的腦袋暖暖身，讓它開始運轉起來。還記得上一頁的程式碼嗎？讓我們先大略看過它，感受一下它可能在做什麼：

```
setTimeout(wakeUpUser, 5000);
function wakeUpUser() {
   alert("Are you going to stare at this boring page forever?");
}
```

或許是計算五秒鐘時間的方式？提示：1,000 毫秒 = 1 秒。

建立可重複使用的程式碼，並以稱之為「wakeUpUser」的方式？

顯然是以訊息提醒使用者的一種方式。

問：JavaScript 與 Java 有關係嗎？

答：只有名字像而已。JavaScript 是在 Java 正紅的時候被創造出來的，透過這種 Java 名字，JavaScript 的發明者利用了 Java 的響亮名號。這兩種語言都借用了 C 那類程式語言的一些語法，但除此之外，它們之間的差異相當大。

問：我聽說 JavaScript 是一種功能較弱的語言，是嗎？

答：JavaScript 在早期的確不算厲害，但從那時起，它對 Web 的重要性與日俱增，因此，許多資源（包括來自業界最頂尖人才的腦力）都投入增強 JavaScript 的效能。但是，你知道嗎？即使在 JavaScript 變得超快之前，它就已經是一個出色的語言了。正如你即將看到的，我們會用它來做一些非常強大的事情。

問：JavaScript 是建立動態網頁的最佳方式嗎？

答：JavaScript 是在瀏覽器中進行程式設計的**主要**語言。JavaScript 有許多變體，例如 TypeScript，但你所寫的 TypeScript 程式碼在瀏覽器中執行之前，會先翻譯成 JavaScript。有了現今超快速的 JavaScript 環境和精密的 API，JavaScript 將會繼續存在，並成為瀏覽器中程式設計的標準。

問：我的朋友在音樂應用程式中使用 JavaScript…至少他是這麼說的。這可能嗎？

答：是的！ JavaScript 已經走出瀏覽器，成為許多應用程式的通用指令稿語言（scripting language），從繪圖工具到音樂應用程式，甚至到伺服端程式設計。你在學習 JavaScript 上的投資很可能會在網頁以外的地方得到回報。

問：你說許多其他語言都是經過編譯的。這到底是什麼意思，為什麼 JavaScript 不是？

答：對於 C、C++ 或 Java 等傳統程式語言，你會在執行程式碼前先編譯（compile）程式碼。編譯過程會將你的程式碼用機器能高效處理的形式表達出來，通常會針對執行時期的效能進行最佳化。指令稿語言（scripting languages）通常是**直譯式**（*interpreted*）的，這表示瀏覽器會在遇到每一行 JavaScript 程式碼時直接執行它。指令稿語言並不那麼重視執行時期的效能，而是更偏向於原型製作（prototyping）、互動式程式設計和靈活性等任務。早期的 JavaScript 就是如此，這也是為什麼多年來 JavaScript 的效能都不是很好的原因。然而，還有一個中間地帶存在；直譯式語言可以即時編譯，這也是瀏覽器製造商對現代 JavaScript 所採取的路徑。事實上，有了 JavaScript，你現在就可以享有指令稿語言的便利性，同時享受編譯語言的效能。順便說一下，我們會在本書中使用**直譯**（*interpret*）、**估算**（*evaluate*）和**執行**（*execute*）這幾個詞。它們在不同的情境中有稍微不同的意思，但對於我們的目的而言，它們基本上都代表相同的意思。

看看撰寫 JavaScript 有多麼容易

你還不了解 JavaScript，但我們打賭你可以對 JavaScript 程式碼的運作方式做出一些不錯的猜測。讀一下每一行程式碼，看看你是否能猜出它的作用。請將答案寫在下面。我們已為你先做了一個，讓你可以開始。如果你卡住了，答案就在下一頁。

```
let price = 28.99;
let discount = 10;
let total =
      price - (price * (discount / 100));
if (total > 25) {
   freeShipping();
}

let count = 10;
while (count > 0) {
   juggle();
   count = count - 1;
}

const dog = {name: "Rover", weight: 35};
if (dog.weight > 30) {
    alert("WOOF WOOF");
} else {
    alert("woof woof");
}

let circleRadius = 20;
let circleArea =
   Math.PI * (circleRadius * circleRadius);
```

建立一個名為 price 的變數，並將值 28.99 指定給該變數。

解答在第 8 頁

看看撰寫 JavaScript 有多麼容易解答

題目在第 7 頁

你還不了解 JavaScript，但我們打賭你可以對 JavaScript 程式碼的運作方式做出一些不錯的猜測。讀一下每一行程式碼，看看你是否能猜出它的作用。以下是我們的參考答案。

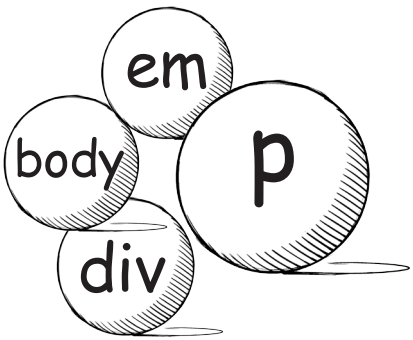
```
let price = 28.99;
let discount = 10;
let total =
      price - (price * (discount / 100));
if (total > 25) {
   freeShipping();
}

let count = 10;
while (count > 0) {
   juggle();
   count = count - 1;
}

const dog = {name: "Rover", weight: 35};
if (dog.weight > 30) {
    alert("WOOF WOOF");
} else {
    alert("woof woof");
}

let circleRadius = 20;
let circleArea =
   Math.PI * (circleRadius * circleRadius);
```

建立一個名為 price 的變數，並將值 28.99 指定給該變數。
建立一個名為 discount 的變數，並將值 10 指定給該變數。
運用折扣（discount）計算新的價格（price），然後將其指定給變數 total。
比較變數 total 的值與 25。如果大於…
…就用 freeShipping 做點什麼。
結束 if 述句。
建立一個名為 count 的變數，並將值 10 指定給該變數。
只要變數 count 大於 0…
…做一些雜耍（juggling），並且…
…每次都將 count 的值減少 1。
結束 while 迴圈。
以名字（name）和體重（weight）建立名為 dog 的常數。
如果 dog 的體重大於 30…
…發出「WOOF WOOF」提醒到瀏覽器的頁面。
否則…
…發出「woof woof」提醒到瀏覽器的網頁。
結束 if/else 述句。
建立變數 circleRadius，並將值 20 指定給該變數。
建立一個名為 circleArea 的變數…
…並將此運算式的結果指定給它（1256.6370614359173）。

我究竟從 DOM 得到了什麼？

用 getElementById 從 DOM 抓取元素時，取得的是一個*元素物件*（*element object*），你可以用它來讀取、更改或替換元素的內容（content）和屬性（attributes）。重點來了：修改元素時，網頁顯示的內容也會跟著改變。

但我們先從頭說起。讓我們再看看剛從 DOM 抓取的元素物件。我們知道這個元素物件代表網頁中 id 為「greenplanet」的 <p> 元素，而該元素中的文字內容是「All is well」。就跟其他 JavaScript 物件一樣，元素物件也有特性和方法。就元素物件而言，我們可以用這些特性和方法來讀取和變更元素。以下是一些你可以用元素物件做的事情：



取得內容（文字或 HTML）。

讀取屬性。

變更內容。

新增屬性。

變更屬性。

移除屬性。

你可以使用元素物件做到的事情。

我們要對 <p> 元素做的是將「All is well」改為「Red Alert: hit by phaser fire!」。我們已經在程式碼中的 planet 變數中儲存了那個元素物件；讓我們藉此來修改它的其中一個特性：innerHTML：

planet 變數含有一個元素物件——就是「greenplanet」<p> 元素的元素物件。

```
let planet = document.getElementById("greenplanet");

planet.innerHTML = "Red Alert: hit by phaser fire!";
```

我們可以使用元素物件的 innerHTML 特性來變更元素的內容！

快速複習

嘿，坐下來，休息一下。你可能會對自己說：「等等，我記得一些關於 id 和類別（class）的東西，但我不記得具體細節，而且它們不是和 CSS 有關係嗎？沒問題；讓我們快速複習一下，了解一些背景知識，然後我們很快就會讓你回到正軌…

在 HTML 中，id 為我們提供了唯一識別（identify）元素的方式。只要一個元素有唯一的 id，我們就能使用 CSS 來選取它以設定樣式。此外，正如你所看到的，我們也可以在 JavaScript 中透過 id 來取得元素。

讓我們看個例子：

```
<div id="menu">
  ...
</div>
```

我們指定給這個 一個唯一的 id，即「menu」。它應該是我們頁面中唯一 id 為「menu」的元素。

我們可以使用 CSS 來選取這個唯一的 id 以設定其樣式，就像這樣：

```
div#menu {
    background-color: #aaa;
}
```

div#menu 是一個 id 選擇器（selector）。

div#menu 會選取 id 為「menu」的 ，如此我們就能套用樣式到該元素，而且只會套用到該元素。

在 JavaScript 中我們也可以透過其 id 來存取這個元素：

```
let myMenu = document.getElementById("menu");
```

別忘了，還有另一種標示元素的方式：使用類別（class）。類別讓我們可以標示一組元素，就像這樣：

```
<h3 class="drink">Strawberry Blast</h3>
<h3 class="drink">Lemon Ice</h3>
```

兩個 <h3> 元素都屬於類別「drink」。類別就像一個群組；同一個群組中可以有多個元素。

在 CSS 和 JavaScript 中，我們也可以透過類別來選擇元素。我們稍後將介紹如何在 JavaScript 中使用這種類別。如果你想要更深入的回顧，請參閱《深入淺出 *HTML & CSS*》的第 7 章，或是你最喜歡的 HTML 與 CSS 參考指南。

圍爐夜話

今夜話題：**相等性和嚴格相等性運算子來告訴我們誰才是老大。**

==（相等）：

哦，看來是那位一本正經的先生來了。

我建議統計一下全世界所有 JavaScript 程式碼中 == 和 === 的使用次數。你絕對會輸得很慘，而且差距懸殊。

我不這麼認為。我提供了有價值的服務。誰不想偶爾比較一下字串形式的使用者輸入和數字呢？

你念小學時是不是每天都得在雪地裡走路上下學，而且還是上下坡呢？你是不是總要用最困難的方式做事？

重點是，我不僅可以做跟你一樣的比較，還能在此基礎上進行型別的優雅轉換，這是我的附加價值。

你寧願就這樣放棄，回傳 false 然後回家？

===（嚴格相等）：

記住，許多重量級的 JavaScript 大師都說開發者應該使用我，而且只用我。他們認為你應該從語言中徹底移除。

你知道嗎，你可能說對了，但人們慢慢開始理解了，這些數字正在改變。

但這伴隨著使用 == 時必須記住的所有規則。讓生活和程式碼簡單點，使用 === 吧，如果你需要把使用者輸入轉換成數字，有專門的方法可以做到。

很好笑。在比較時保持嚴謹和有明確的語義並沒有錯。如果你沒有記住所有規則，可能會發生不好的意外狀況。

每次看到你的規則我都會有點反胃。我是說，把 Boolean 值和任何東西比較就要把 Boolean 值轉換成數字？這對我來說一點道理都沒有。

規劃程式碼…

與其將所有的這些猜測處理程式碼都放入 `processGuess` 方法中，我們不如寫一個小小的輔助函式（畢竟，我們也許會再用到這個函式）。我們將此函式命名為 `parseGuess`。

開始寫程式碼之前，讓我們先逐步了解一下它要如何運作：

1. 以經典的 Battleship 形式獲得玩家的猜測，即單一字母後面跟著一個數字。
2. 檢查輸入是否有效（不是 null 或太長太短）。
3. 將字母轉換為數字：A 轉為 0、B 轉為 1，以此類推。
4. 檢視步驟 3 中的數字是否有效（介於 0 和 6 之間）。
5. 檢查第二個數字是否有效（也在 0 到 6 之間）。
6. 若有任何檢查失敗，就回傳 null。否則，將兩個數字串接為一個字串，並傳回該字串。