金門大學 資訊工程系 101 學年度下學期 網路程式設計 期末考 出題者：陳鍾誠

考生姓名：　　學號：　　得分：

以下題目學號為奇數同學做奇數題，學號為偶數同學作偶數題，共 100 題，每人做 50 題，每題 2 分。

名詞解釋：

JavaScript 相關

1. **JavaScript** // 一種很彈性的程式語言，是內建於瀏覽器的標準語言，通常用來做製作動態網頁。
2. **Node.js** // 讓 **JavaScript** 可以在伺服端執行的一種開發環境，採用 **Google** 的 **V8 JIT** 引擎。
3. **express** 套件 // **Node.js** 當中，用來處理 **get/post** 訊息的一種很方便使用的套件。
4. **regular expression** // 正規表達式，是用來比對、擷取或轉換字串時使用的字串比對樣式。
5. **AJAX** // **Asynchronous JavaScript and XML,** 網頁中的 **JavaScript** 可以透過 **AJAX** 動態的存取伺服端的文件。
6. **eval** 函數 // 可以用來把字串當作程式執行的 **JavaScript** 函數
7. **jQuery** // 一種很好用的 **JavaScript** 的框架，可以查詢選取操作網頁畫面，並且有很多延伸套件，像是 **jQuery.UI** 可以做互動式介面。
8. **npm** // **Node.js Package Manager – Node.js** 的套件安裝管理工具，可用來安裝所指定的套件，例如 **npm install express** 可以用來安裝 **express** 套件。
9. **module.exports** // **Node.js** 的模組匯出物件，當您用 **<name> = require(<package>)** 時就可引用該套件。
10. **callback** // 回呼函數、這種函數在某件事情完成後就會被呼叫，這在 **javascript** 與 **node.js** 當中都很常用。
11. **closure** // 閉包、**Javascript** 採用 **lexical scope**，也就是根據定義時的上下文決定變數領域的方式 **(**而非執行時的上下文**)**，因此好像有點將變數定義地點的上下文帶著走的感覺，這種領域決定方式稱為 **closure**。
12. **JSON** // 用 **JavaScript** 語法 **(**代替 **XML)** 定義物件的格式，這種格式可以用做網頁間的資料傳遞格式。

HTML/CSS 相關

13. **DHTML** // 動態網頁：可與使用者進行互動的網頁，像是功能表、**Google Map** 等都是 **DHTML** 的範例。
14. **DOM** // **Document Object Model , HTML** 在瀏覽器中會被建立為一種階層物件的架構，稱為 **DOM**，可用來讓 **JavaScript** 存取並使用。
15. **innerText** // **DOM** 中某個元素的內在文字，稱為 **innerText**，可以用這個屬性來取得或設定該元素的文字內容。
16. **innerHTML** // **DOM** 中某個元素的內在 **HTML**，稱為 **innerText**，可以用這個屬性來取得或設定該元素的 **HTML** 內容。
17. **Form** // 表單：**HTML** 中用來框住輸入物件的標記，並且可用 **action** 指定輸入完成後資料要送到哪裡。
18. **form** 中的 **method** 屬性 // **form** 的 **method** 可能為 **POST** 或 **GET**，這個參數決定表單輸入完成後要用哪種格式 **(GET** 或 **POST)** 的方式送出。
19. **form** 中的 **action** 屬性 // **form** 中用來指定輸入資料完成後要送到哪裡的欄位，例如 **<form action= "login" >** 的話就會送到 網站的 **/login** 路徑中，給對應的程式處理。
20. **form** 中的 **submit** 之用途 // **form** 中的 **submit** 是一種按鈕，當按下後會將資料傳遞到 **action** 所指定之處。
21. **POST** // **POST** 是 **HTML** 網頁用來傳遞輸入資料的一種方式，其輸入資料是附加在 **HTTP** 表頭的最後傳出的。
22. **GET** // **POST** 是 **HTML** 網頁用來傳遞輸入資料的一種方式，其輸入資料是附加在 **HTTP** 表頭的網指部分傳出的，所以這些資料會顯示在瀏覽器的網址上。
23. **CSS** // **Cascading Style Sheet ,** 階層式樣式表，是用來指定 **HTML** 當中特定區塊之顯示樣式的一種文件格式，例如我們可以用 **a { color:red }** 讓超連結變成紅色的。
24. **style** // 當您想將 **CSS** 屬性直接寫在 **HTML** 某標記內時，就可以用 **style** 屬性，例如 ** ** 就可以讓其內部的文字以藍色顯示。
25. **class** // **HTML** 標記的 **class** 屬性，可以用來讓 **CSS** 格式方便的指定服合這些類別的標記，例如 **.header** 就可以用來指定 ** … ** 這種標示 **header** 類別的標記。
26. **id** // **HTML** 標記內的 **id** 屬性，可以讓 **JavaScript** 透過 **getElementById** 的方式取得 **DOM** 中的該物件，例如： ** ... ** 就可以用 **getElementById("front_table")** 來取得。

程式解釋 1：

```
1. <script type="text/javascript">  // HTML 中 JavaScript 的起始標記
2. function obj() { // 定義一個稱為 obj() 的函數 (這個函數其實是物件 obj 的建構函數)
3.   this.x = 3;  // 設定該物件的 x 欄位為 3
4.   this["y"] = 5; // 設定該物件的 y 欄位為 5
5.   this.sum = function() { return this.x + this.y; } // 定義 sum 函數，傳回 x+y
}

6. var o = new obj(); // 呼叫 obj() 建構函數，建立  obj 物件，並傳給 o 變數。

7. document.write("o['x']="+o['x']+" y="+o.y+" sum()="+o.sum()); // 印出物件 o 當中的 x, y 欄位，並呼叫
sum 函數。
</script>
```

考生姓名：　　學號：

程式解釋 2：

```
<script type="text/javascript">
8. var george = { // 以 JSON 的語法建立 george 物件
9.   "name": "George", // 加入欄位 name, 其內容為字串 'george'
10.   "age": 25, // 加入欄位 age, 其內容為數字 25
11.   "friends": [ // 加入欄位 friends ，其內容為一個陣列
12.     {"name": "John", "age": 22 }, // 陣列的第一個成員為 (name= “John”, age=22) 的物件
13.     {"name": "Mary", "age": 28 } // 陣列的第二個成員為 (name= “Mary”, age=28) 的物件
    ]
};

14. document.write("george.age="+george.age+"<br/>"); // 印出 george.age 的內容
15. document.write("george.friends:\n<ol>"); // 印出 george.friends，並開始用 <ol> 顯示 HTML 項目列表：
16. var friends = george.friends; // 取得 george 物件的 friends 欄位
17. for (i in friends) // 對於 friends 陣列中的每個成員
18.   document.write("<li>"+friends[i].name+" is "+friends[i].age+"years old!</li>"); // 都輸出姓名年齡
19. document.write("</ol>"); // 輸出 </ol> 結束項目列表。
</script>
```

程式解釋 3：

```
<script type="text/javascript">
var a=3, b=5;
20. var result = eval('a+b'); // 利用 eval 函數計算 'a+b' 的值
21. alert('a+b='+result); // 顯示 a+b=8 的訊息對話框。
</script>
```

程式解釋 4：

```
 <html><body>
  <div id ="hi"><b>你好!</b></div>
22.   <input type="button" value="hi.innerText"
       onclick="alert(document.getElementById('hi').innerText)");
```

```
// 一個按鈕、一開始顯示 hi.innerText，當按下後會顯示 你好！

23.  <input type="button" value="hi.innerHTML"
       onclick="alert(document.getElementById('hi').innerHTML)");
// 一個按鈕、一開始顯示 hi.innerHTML，當按下後會顯示 <b>你好</b>！
</body></html>
```

程式解釋 5：

```
<html><body>
  <div id ="hi"><b>你好!</b></div>
24.   <input type="button" value="hi.show()"
onclick="document.getElementById('hi').style.visibility='visible'");
// 一個按鈕、一開始顯示 hi.show() ，當按下後會讓上述 <div id ="hi"><b>你好!</b></div> 這個段落顯示出來。
25.   <input type="button" value="hi.hide()"
onclick="document.getElementById('hi').style.visibility='hidden'");
// 一個按鈕、一開始顯示 hi.hide() ，當按下後會讓上述 <div id ="hi"><b>你好!</b></div> 這個段落隱藏起來。
</body></html>
```

考生姓名：　　學號：

程式解釋 6：

```
<html><head>
<style>
26.  .menu   { background-color:black; color:white; padding:10px;
       vertical-align:top; width:100px; list-style-type:none; }
// CSS 語法，用來指定功能表 <li id="menu1" class="menu">menu1</li> 的背景為黑色、文字為白色、內框大小為 10 點，並且對齊方式為靠頂，寬度為 100 點，項目前無標記。
27.  .menu a { color:white; text-decoration:none; }
// CSS 語法，用來指定功能表內超連結的背景為白色、且超連結不要有底線。
</style>
<script type="text/javascript">
28. function show(id) {  document.getElementById(id).style.visibility='visible'; }
// 用來讓某個 id 對應元素顯示出來的函數
29. function hide(id) {  document.getElementById(id).style.visibility='hidden'; }
// 用來讓某個 id 對應元素隱藏起來的函數
</script>
</head>
30.  <body onload="JavaScript:hide('popup1');hide('popup2');">
// 一開始就將兩個浮現視窗 popup1, popup2 隱藏起來。
31.     <ul onmouseover="show('popup1');"  onmouseout="hide('popup1')"
       style="position:absolute; left:100px; top:20px">
// 當滑鼠移到本元件上時，就顯示浮現功能表 popup1 ，當滑鼠移出時，就隱藏該浮現功能表。
      <li id="menu1" class="menu">menu1</li>
      <ul id="popup1" class="menu">
32.        <li><a href="JavaScript:alert('1.1');">menu 1.1</a></li>
// 當此項目連結被按下時，顯示對話訊息框，內容為 1.1。
33.        <li><a href="JavaScript:alert('1.2');">menu 1.2</a></li>
// 當此項目連結被按下時，顯示對話訊息框，內容為 1.2。
      </ul>
    </ul>
```

```
</body></html>
```

程式解釋 7 – 網路記事本

檔案：note.htm

```
<!DOCTYPE html><html><head>
34. <meta charset="utf-8" />
// 用 <meta charset="utf-8" /> 告訴流覽器這個文件的文字編碼方式為 UTF-8
</head><body>
35.  <form method="POST" action="/save/[[?path?]]">
// HTML 表單開始，採用 POST 方式傳遞訊息，當按下後會傳送到 /save/[[?path?]] 這個路徑，假如 path 為 hello.txt ，則會傳送到 /save/hello.txt 這個網址中。
36.     <label>檔案路徑：[[?path?]]</label><br>
// 用 label 的方式顯示檔案路徑，例如 檔案路徑:hello.txt。
37.      <textarea name="note" rows=30 cols=80>[[?fileText?]]</textarea><br/>
// 用 textarea 顯示編輯視窗 ，裡面內容原為 [[?fileText?]]  (但在後面的程式中會被取代程檔案內容）
38.     <input type="submit" value="送出"/>
// 送出紐，按下後會啟動表單的 action 動作。
 </form>
</body></html>
```

檔案：save.htm

```
<!DOCTYPE html><html><head>
<meta charset="utf-8" />
</head><body>
39. save:path=<a href="/note/[[?path?]]">[[?path?]]</a>
// 顯示檔案路徑，例如 save:path=hello.txt。
40. <pre>[[?fileText?]]</pre>
// 顯示檔案內容，在後面的程式中 [[?fileText?]]  會被取代為檔案內容。
</body>
</html>
```

考生姓名：　　學號：

檔案：NoteServer.js

```
var path = require('path');
var fs = require("fs");
41. var qs = require('querystring'); // 以 qs 為名稱使用 querystring 模組
42. var express = require("express"); // 以 express 為名稱使用 express 模組
43. var app = express();     // 呼叫 express 模組的建構函數，建立物件夠指定給 app 變數
44. app.listen(80);  // 開啟並佔用 80 port ，開始傾聽對此連接埠的請求。
45. var noteTemplate = ""; // 設定樣版變數 noteTemplate 為空字串。
46. fs.readFile("note.htm", "utf8", function(err, file) {   noteTemplate = file; });
// 讀入樣版檔案 note.htm 到樣版變數 noteTemplate 當中。
47. fs.readFile("save.htm", "utf8", function(err, file) {   saveTemplate = file;});
// 讀入儲存樣版檔案 save.htm 到樣版變數 saveTemplate 當中。
48. var error = function(err, res) { // 定易錯誤處理函數 error。
```

```
49.     if (err) { // 當有錯誤發生時
50.       res.writeHead(404, {'Content-Type': 'text/plain'}); // 輸出 404 找不到網頁的錯誤訊息。
51.       res.end(); // 輸出回應完畢
   }
}
52. var response = function(res, type, text) { //定義回應函數 response
53.    res.writeHead(200, {'Content-Type': type}); // 輸出 200 OK 的回應表頭。
54.    res.write(text); // 輸出 text 到回應訊息中。
55.    console.log(text); // 印出 text 到命令列視窗中
56.    res.end(); // 輸出回應完畢
}
57. app.post('/save/:path', function(req, res){ // 如果網址符合 /save/:path 這個格式，就呼叫此處的匿名函數
58.    console.log('save ' + req.params.path); // 輸出 save + 網址中的 path 欄位
59.    var path = "./"+req.params.path; // 設定 path 變數為 "./"+網址中的 path 欄位
60.    formData = ''; // 設定 formData 為空字串
61.    req.on("data", function (chunk) {  formData += chunk;  }); // 當有 POST 資料時，不斷接收資料加入
formData 變數中
62.    req.on("end", function () { // 當 POST 資料結束時，呼叫此處的匿名函數
63.       form = qs.parse(formData); // 剖析 POST 所傳來的 formData 字串，建立成 form 物件
64.       fs.writeFile(path, form.note, function (err) { // 將 form 的 note 欄位寫入到 path 所指定的檔案中。
65.         if (!err) // 假如檔案寫入沒有錯誤的話
66.           response(res, "text/html", saveTemplate.replace("[[?path?]]", req.params.path) // 回應訊息
67.           .replace("[[?path?]]", req.params.path) // 將 [[?path?]] 取代 (連續三次)
68.           .replace("[[?path?]]", req.params.path).replace("[[?fileText?]]", form.note)); // 然後再將 [[?
fileText?]] 取代為 form.note 的編輯內容。
      });
   });
});
69. app.get('/note/:path', function(req, res){  // 如果網址符合 /note/:path 這個格式，就呼叫此處的匿名函數
70.     console.log('note ' + req.params.path);  // 輸出 note + 網址中的 path 欄位
71.     var path = "./"+req.params.path; // 設定 path 變數為 "./"+網址中的 path 欄位
72.     fs.readFile(path, "utf8", function(err, file) { // 以 UTF8 的方式讀取 path 路徑中的檔案內容
       if (err)
73.      response(res, "text/html", noteTemplate.replace("[[?path?]]", req.params.path)
          .replace("[[?path?]]", req.params.path).replace("[[?fileText?]]", "檔案不存在，可修改後存檔！"));
// 如果有錯 (讀取失敗)，則顯示內容為「檔案不存在，可修改後存檔！」的文字訊息
      else
74.      response(res, "text/html", noteTemplate.replace("[[?path?]]", req.params.path)
           .replace("[[?path?]]", req.params.path).replace("[[?fileText?]]", file));
// 如果沒錯 (讀取成功)，則顯示內容為讀取的檔案內容。
   });
});
console.log('start NoteServer\n');
```