網頁程式設計
JavaScript程式設計

Instructor: 馬豪尚

DOM Tree節點操作

DOM 查找元素

- 通常你要存取 HTML 都是從 document 物件開始。
- document.getElementById(id)
  - 用來根據 id 取得一個 HTML 元素。
- document.getElementsByTagName(name)
  - 用來根據 HTML 標籤 (tag) 名稱取得所有這個標籤的元素集合
  - 返回的結果是一個像陣列 (array) 的物件
  - 每個 HTML DOM 元素也有 getElementsByTagName 方法，用來找該元素下面的子元素

DOM 查找元素

- document.getElementsByName(name)
  - 用來取得特定名稱 (name) 的 HTML 元素集合
  - 返回的結果是一個像陣列 (array) 的物件
- document.getElementsByClassName(names)
  - 用來取得特定類別名稱 (class name) 的 HTML 元素集合
  - 返回的結果是一個像陣列 (array) 的物件

DOM 查找元素

- document.querySelector(selectors)
  - 用 CSS 選擇器 (CSS selectors) 來尋找符合條件且第一個找到的 HTML 元素
  - 每個 HTML DOM 元素也有 querySelector 方法，用來找該元素下面的子元素
- document.querySelectorAll(selectors)
  - 可以用 CSS 選擇器 (CSS selectors) 來尋找所有符合條件的 HTML 元素集合
  - 返回的結果是一個像陣列 (array) 的物件
  - 每個 HTML DOM 元素也有 querySelectorAll 方法，用來找該元素下面的子元素

DOM 新增節點操作

- document.createElement(tagName)
  - 建立一個新的 HTML 元素節點
  - tagName為HTML 元素標籤名稱
- document.createTextNode(str)
  - 建立一個新的文字節點
  - str為文字節點的內容值

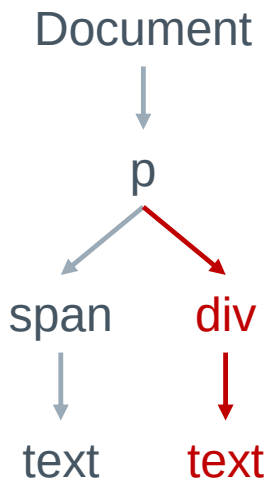
DOM 新增節點操作

- Node.appendChild(aChild)
  - 插入一個新的子元素到現有子元素的最後面
  - aChild為一個節點物件

Example

```
<p id="foo"><span id="s1">first span</span></p>

<script>
    var newDiv = document.createElement('div');
    var newContent = document.createTextNode('aa');
    newDiv.appendChild(newContent);
    var currentDiv = document.getElementById('foo');
    currentDiv.appendChild(newDiv);
</script>
```

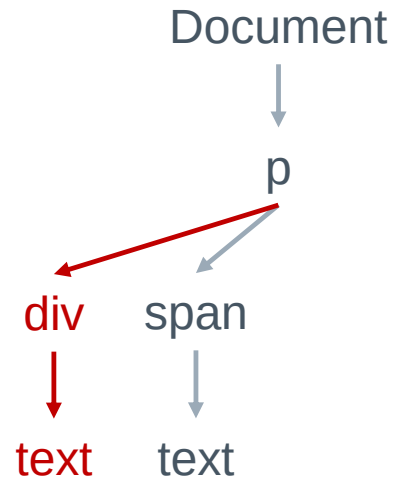


DOM新增節點操作

- Node.insertBefore(newNode, referenceNode)
  - 將一個新的元素加到某個元素的前面
  - newNode為要加入的節點物件
  - referenceNode為參考位置的節點物件

Example

```
var currentDiv = document.getElementById('foo');
var span = document.getElementById('s1');
currentDiv.insertBefore(newDiv, span);
```



DOM 刪除節點操作

- Node.removeChild(child).
  - 用來移除 DOM 節點，會返回移除的節點物件
  - child為節點物件，child要是Node的子節點

Example

```
<div id="container"><div id="nested">12</div>34</div>

var container = document.getElementById('container');
var nested = document.getElementById('nested');
var garbage = container.removeChild(nested);
```

DOM 修改節點操作

- Node.replaceChild(newChild, oldChild)
  - 用新節點來取代某個子節點
  - 新節點可以是某個已存在的節點或是新建立的節點
  - 會返回被取代的節點物件

Example

```
<div id="container"><div id="nested">12</div>34</div>

var nested = document.getElementById('nested');
var parentDiv = nested.parentNode;
var replacedNode = parentDiv.replaceChild(NewNode, nested);
```

DOM 修改節點操作

- Node.cloneNode(false/true)
  - 可以用來複製一個節點
  - 預設不會複製節點的內容，可以傳入參數 true 來複製節點的內容

Example

```
<div id="foo"><span>bar</span></div>

<script>
    var foo = document.getElementById('foo');
    foo.cloneNode(false)
    foo.cloneNode(true)
</script>
```


國立臺中科技大學
NATIONAL TAICHUNG UNIVERSITY OF SCIENCE AND TECHNOLOGY

DOM HTML屬性

DOM HTML 屬性(Attributes)

- DOM 中的屬性: Properties 和 Attributes
  - Properties是 JavaScript DOM 物件上的屬性
    - array.length
    - window.innerWidth
    - Node.firstChild
  - Attributes 是HTML 元素上的屬性
    - HTML 標籤上的 id 、class、value、name等

DOM HTML 屬性(Attributes)

- Element.hasAttribute(attrName)
  - 用來檢查 HTML 元素是否有某個屬性
  - 回傳True/False

Example

```
<a id=" goo" href="http://www.google.com/">google</a>
<script>
   var goo = document.getElementById('goo');
   goo.hasAttribute('xyz');
   goo.hasAttribute('href');
</script>
```

DOM HTML 屬性(Attributes)

- Element.getAttribute(attrName)
  - 用來取得 HTML 元素的屬性值
  - attrName為想取得的屬性(attribute)名稱
- Element物件有提供一些屬性存取的方式
  - Element.id
  - Element.className
  - Element.tagNmae
  - Element.type
  - Element.value
  - Element.href
  - Element.name

DOM HTML 屬性(Attributes)

- Element.attributes
  - 可以取得 HTML 元素上所有的屬性 (attributes)
  - attributes 會返回一個 key/value pair 的 NamedNodeMap 型態物件
  - NamedNodeMap 物件的 key 是一個字串表示屬性名稱，value 則是一個 Attr 物件
    - Element.attributes['id']
  - Attr 物件上的 name 屬性可以取得屬性名稱，Attr 物件上的 value 屬性可以取得屬性值
    - Element.attributes['id'].name → id
    - Element.attributes['id'].value → 該元素節點的id屬性的值

DOM HTML 屬性(Attributes)

- Element.setAttribute(attrName, value)
  - 用來新增 HTML 元素的屬性，如果屬性已經存在則更新其值
  - attrName為要增加屬性(attribute)的名稱
  - value為屬性的值
- Element.removeAttribute(attrName)
  - 用來移除 HTML 元素的某個屬性
  - attrName為要移除的屬性(attribute)的名稱

JavaScript DOM Event

JavaScript Event

- 使用者在瀏覽網頁時會觸發很多事件 (events) 的發生
  - 按下滑鼠
  - 按下鍵盤按鍵
  - 圖片完成下載
  - 表單欄位值被改變
- DOM Event 定義很多種事件型態，讓你可以用 JavaScript 來監聽 (listen) 和處理 (event handling) 這些事件

使用者介面事件

› 使用者介面事件代表與操作瀏覽器相關的事件

事件	說明
load	當瀏覽器將網頁或圖片完成載入時會觸發
unload	當使用者關閉 (卸載) 網頁之後會觸發
error	當圖片或文件(網頁)內容下載發生錯誤時會觸發
resize	當視窗或框架大小被改變時會觸發
scroll	當瀏覽器視窗捲軸被拉動時會觸發
DOMContentLoaded	當HTML文件載入完畢(不用等到樣式表、圖片或影片等資源載完)時會觸發
hashchange	當URL中#符號後面的資料變更時會觸發
beforeunload	當視窗、文件和相關的資源即將離開(卸載)時會觸發此事件，此時文件仍舊是看得到的

鍵盤事件

- 鍵盤事件表示與使用者操作鍵盤相關的事件

事件	說明
keydown	當使用者按下鍵盤按鍵時會觸發
keyup	當使用者按下並放開鍵盤按鍵時會觸發
keypress	當使用者按下並放開鍵盤按鍵後會觸發

滑鼠事件

› 滑鼠事件代表使用者操作滑鼠相關的事件

事件	說明
click	當使用者滑鼠點擊物件時會觸發
dblclick	當使用者滑鼠連點二下物件時會觸發
mousedown	當使用者按下滑鼠按鍵時會觸發
mouseup	當使用者放開滑鼠按鍵時會觸發
mouseout	當使用者滑鼠離開某物件四周時會觸發
mouseover	當使用者滑鼠進入一個元素 (包含進入該元素中的子元素) 四周時會觸發
mousemove	當使用者介於over跟out間的滑鼠移動行為
mousewheel	當使用者在元素上滾動滑鼠滾輪時會觸發

表單事件

› 表單事件表示使用者操作表單的相關事件

事件	說明
input	當<input>、<select>或<textarea>等元素物件的值被輸入時會觸發
change	當<input>、<select>或<textarea>等元素物件的值被變更時會觸發
submit	當使用者提交表單時會觸發
reset	當使用者重設表單時會觸發
select	當使用者在表單欄位選取內容時會觸發
cut	當使用者在表單欄位剪下內容時會觸發
copy	當使用者在表單欄位複製內容時會觸發
paste	當使用者在表單欄位貼上內容時會觸發

焦點事件

- 焦點事件表示當元素物件取得或失去焦點時的時候

事件	說明
focus	當物件被點擊或取得焦點時會觸發
blur	當物件失去焦點時會觸發

事件處理之前

- 要進行事件處理時，先想以下三點
  - 要由哪個元素觸發事件
  - 要觸發哪種事件
  - 被觸發的事件要繫結哪個事件處理程式/事件監聽程式
- 繫結的方法有以下三種
  - 用HTML元素的事件屬性設定事件處理程式
  - 傳統的DOM事件處理程式
  - DOM Level 2 事件監聽程式

用HTML元素的事件屬性設定事件處理程式

- 事件屬性的名稱
  - 事件的名稱前面加上on，全部小寫
    - onclick()
    - onmouseover()
    - onkeydown()

Example

```
<button type="button" onclick="window.alert('Hello, world!');">顯示訊息</button>

<button type="button" onclick="showMsg()">顯示訊息</button>
```

傳統DOM事件處理程式

- 不太建議用HTML元素上的屬性直接設定事件處理程式
  - Html和javascript也應該要分開寫(和CSS一樣)

```
<button type="button" onclick="window.alert('Hello, world!');">
```

```
<button type=“button” id=“btn”>顯示訊息</button>

Js檔案
var btn = document.getElementById('btn');
btn.onclick = showMsg;

function showMsg() {
  window.alert('Hello, world!');
}
```

DOM Level 2 事件監聽程式

- DOM Level 2事件監聽程式是近年來比較常見的做法
- 使用DOM Level 2 事件監聽程式最大優點
  - 可以針對同一個物件的同一種事件設定多個處理程序

DOM Level 2 事件監聽程式

- 使用在addEventListener(event, function[, useCapture])
  - 監聽事件並設定事件處理程式
  - event為監聽的事件名稱
  - function為事件處理程式
  - useCapture為選擇性參數，輸入為布林值
    - 表示當內層和外層元素都有發生參數event指定的事件時，用來指定事件處理函數是要在 Capturing 階段或 Bubbling 階段被執行
    - 預設是false，代表會從內層元素開始執行處理程式
    - 輸入true值，代表會從外部元素開始執行處理程式
- 移除綁定的事件處理函數
  - removeEventListener(event, function)

DOM Level 2 事件監聽程式

- 傳參數進要執行的函式
  - addEventListener(event, function_name.bind(this, some_parameter))
  - 用function_name.bind()方法
    - 會建立一個新函式
    - 該函式被呼叫時，會將 this 關鍵字設為給定的參數
    - 呼叫時，傳入給定順序的參數

Event Bubbling (事件氣泡) vs Event Capturing (事件捕捉)

- DOM 中的事件有傳播 (event flow) 的概念
  - 當 DOM 事件發生時，事件會先由外到內 (capturing phase)、再由內到外 (bubbling phase) 的順序來傳播
  - Event Bubbling
    - 當某個事件發生在某個DOM元素物件上（如：點擊），這個事件會先觸發該物件的事件處理程式，再觸發它的父元素物件的事件處理程式，一直觸發到最上層元素物件，像氣泡從下面浮出來一樣
  - Event capturing
    - 和Event bubbling相反從最上層的元素物件的事件處理程式開始觸發，一直到事件發生的某個DOM元素物件上

Event Bubbling (事件氣泡) vs Event Capturing (事件捕捉)

- 當使用者點擊 li 元素時，事件觸發的順序是
  - Capturing 捕捉階段：document -> <html> -> <body> -> -> ->
  - Bubbling 氣泡階段： -> -> -> <body> -> <html> -> document

```
<html>
<head>
   <title>example</title>
</head>
<body>
   <div>
      <ul>
         <li></li>
      </ul>
   </div>
</body>
</html>
```

DOM 停止事件傳遞

- 當我們想中斷Capture Phase或Bubbling Phase的傳遞
  - Event.stopPropagation()
  - Event.stopImmediatePropagation()
  - 兩者的差別在於，如果今天EventTarget綁定了好幾個listener，想要停止全部與這個EventTarget有關的listener，就必須用Event.stopImmediatePropagation()

練習

- 完成一個猜拳遊戲
- 這個遊戲由mora.html, game.css, game.js三個檔案完成
- 遊戲功能為
  - 當我用滑鼠點擊某一個選項之後，判斷和電腦出的拳輸贏結果如何
- Html網頁內的區塊說明
  - 在html內的 為顯示三個圖片物件
  - 的這個區域要顯示遊戲結果
  - 0的區域顯示獲勝了幾次