DOM

目錄

- DOM
- HTML DOM Methods
- JS插入HTML標籤的兩種方法
- Animation

DOM

https://www.w3schools.com/js/js_htmldom.asp

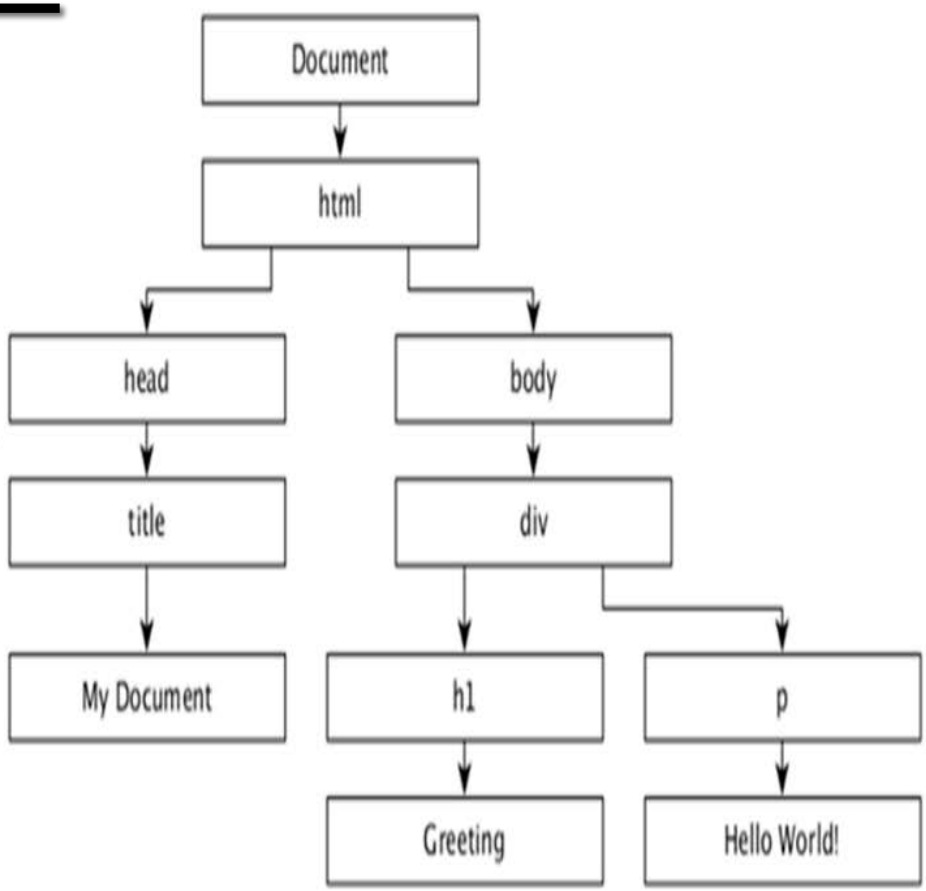
文件物件模型 (DOM)

- **文件物件模型（*Document Object Model, DOM*）**是 HTML、XML 和 SVG 文件的程式介面。它提供了一個文件（樹）的結構化表示法，並定義讓程式可以存取並改變文件架構、風格和內容的方法。DOM 提供了文件以擁有屬性與函式的節點與物件組成的結構化表示。節點也可以附加事件處理程序，一旦觸發事件就會執行處理程序。 本質上，它將網頁或程式語言連結在一起。
- 雖然常常使用 JavaScript 來存取 DOM，但它本身並不是 JavaScript 語言的一部分，而且它也可以被其他語言存取,如 Java, C, 等等

The HTML DOM (Document Object Model)

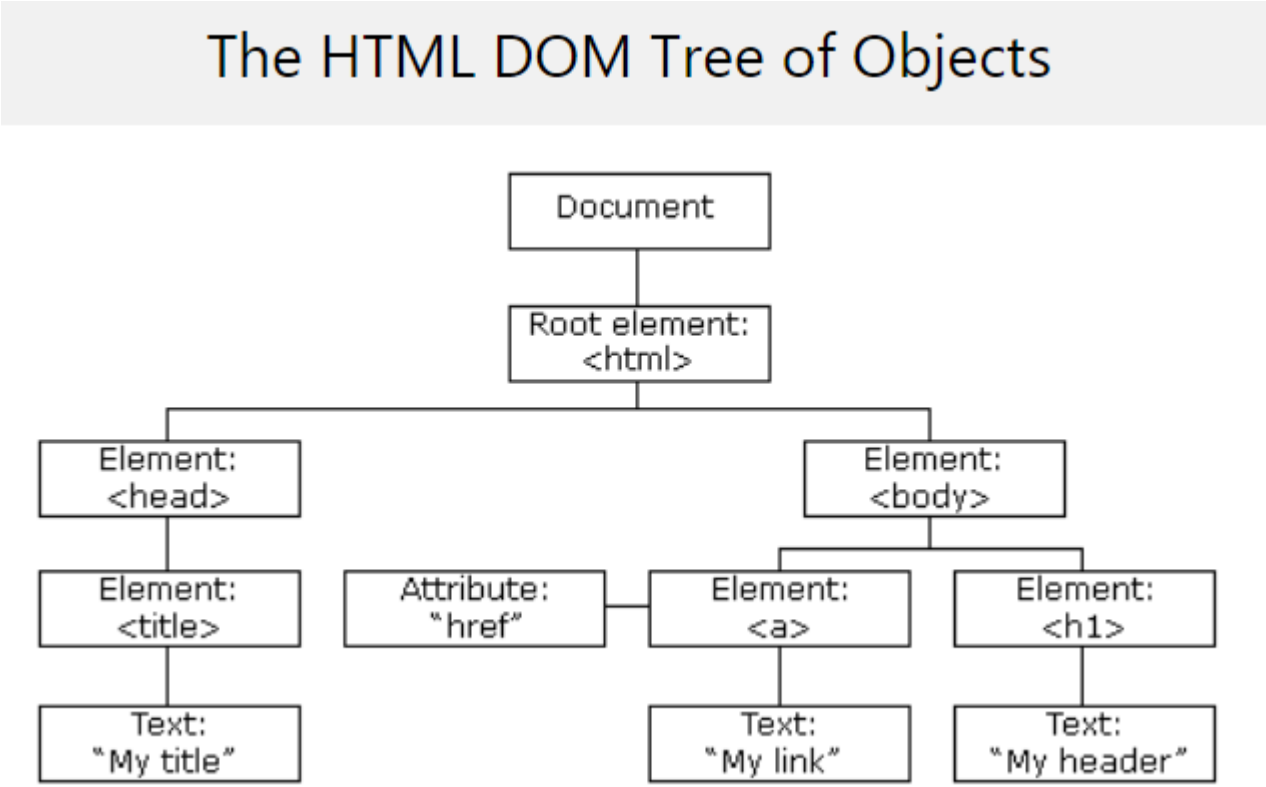
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:

```html
<!DOCTYPE html>
<html lang = "en">
   <head>
      <title>My Document</title>
   </head>

   <body>
      <div>
         <h1>Greeting</h1>
         <p>Hello World!</p>
      </div>
   </body>
</html>
```



The HTML DOM (Document Object Model)

- 元素+元素的屬性



DOM上的操作

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

p.s. JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

What You Will Learn

- How to change the content of HTML elements
- How to change the style (CSS) of HTML elements
- How to react to HTML DOM events
- How to add and delete HTML elements

What is the DOM standard?

- The DOM is a W3C (World Wide Web Consortium) standard.
- The W3C DOM standard is separated into 3 different parts:
  - Core DOM - standard model for all document types
  - XML DOM - standard model for XML documents
  - HTML DOM - standard model for HTML documents

HTML DOM Methods

HTML DOM Methods

- HTML DOM methods are **actions** you can perform (on HTML Elements).
- HTML DOM properties are **values** (of HTML Elements) that you can set or change.

The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object.
- A **property** is a value that you can get or set (like changing the content of an HTML element).
- A **method** is an action you can do (like add or deleting an HTML element).

Example

- The following example changes the content (the innerHTML) of the <p> element with id="demo":

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Page</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

My First Page

Hello World!

- In the example above, getElementById is a **method**, while innerHTML is a **property**.

About the Example

- The getElementById Method:

  The most common way to access an HTML element is to use the id of the element.

  In the example above the getElementById method used id="demo" to find the element.

- The innerHTML Property:

  The easiest way to get the content of an element is by using the innerHTML property.

  The innerHTML property is useful for getting or replacing the content of HTML elements.

  The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Page</h2>

<p id="demo"></p>

<script>
document.getElementById
("demo").innerHTML =
"Hello World!";
</script>

</body>
</html>
```

The HTML DOM Document Object

- The document object represents your web page.
- Finding HTML Elements:

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

- Changing HTML Elements

Property	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.*attribute* = *new value*	Change the attribute value of an HTML element
element.style.*property* = *new style*	Change the style of an HTML element
Method	**Description**
element.setAttribute(*attribute, value*)	Change the attribute value of an HTML element

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new*, *old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Adding Events Handlers

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler code to an onclick event

Finding HTML Objects

- The first HTML DOM Level 1 (1998), defined 11 HTML objects, object collections, and properties. These are still valid in HTML5.
- Later, in HTML DOM Level 3, more objects, collections, and properties were added.

Property	Description	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.applets	Returns all <applet> elements (Deprecated in HTML5)	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.documentElement	Returns the <html> element	3

HTML DOM Elements

Finding HTML Elements

- This page teaches you how to find and access HTML elements in an HTML page. Often, with JavaScript, you want to manipulate HTML elements.
- To do so, you have to find the elements first. There are several ways to do this:
  A. Finding HTML elements by id
  B. Finding HTML elements by tag name
  C. Finding HTML elements by class name
  D. Finding HTML elements by CSS selectors
  E. Finding HTML elements by HTML object collections

A. Finding HTML Element by Id

- The easiest way to find an HTML element in the DOM, is by using the element id.

```html
<!DOCTYPE html>
<html>
  <body>
   <h2>Finding HTML Elements by Id</h2>
   <p id="intro">Hello World!</p>
   <p>This example demonstrates the <b>getElementsById</b> method.</p>
   <p id="demo"></p>

   <script>
   var myElement = document.getElementById("intro");
   document.getElementById("demo").innerHTML =
      "The text from the intro paragraph is " + myElement.innerHTML;
   </script>
 </body>
</html>
```

Finding HTML Elements by Id

Hello World!

This example demonstrates the **getElementsById** method.

The text from the intro paragraph is Hello World!

ex03-01.html

- If the element is found, the method will return the element as an object (in myElement).
- If the element is not found, myElement will contain null.

B. Finding HTML Elements by Tag Name

- This example finds all <p> elements:

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Finding HTML Elements by Tag Name</h2>
    <p>Hello World!</p>
    <p>This example demonstrates the
       <b>getElementsByTagName</b> method.</p>
    <p id="demo">  </p>
    <script>
       var x = document.getElementsByTagName("p");
       document.getElementById("demo").innerHTML =
         'The text in first paragraph (index 0) is: '
         + x[0].innerHTML;
    </script>
  </body>
</html>
```

Finding HTML Elements by Tag Name

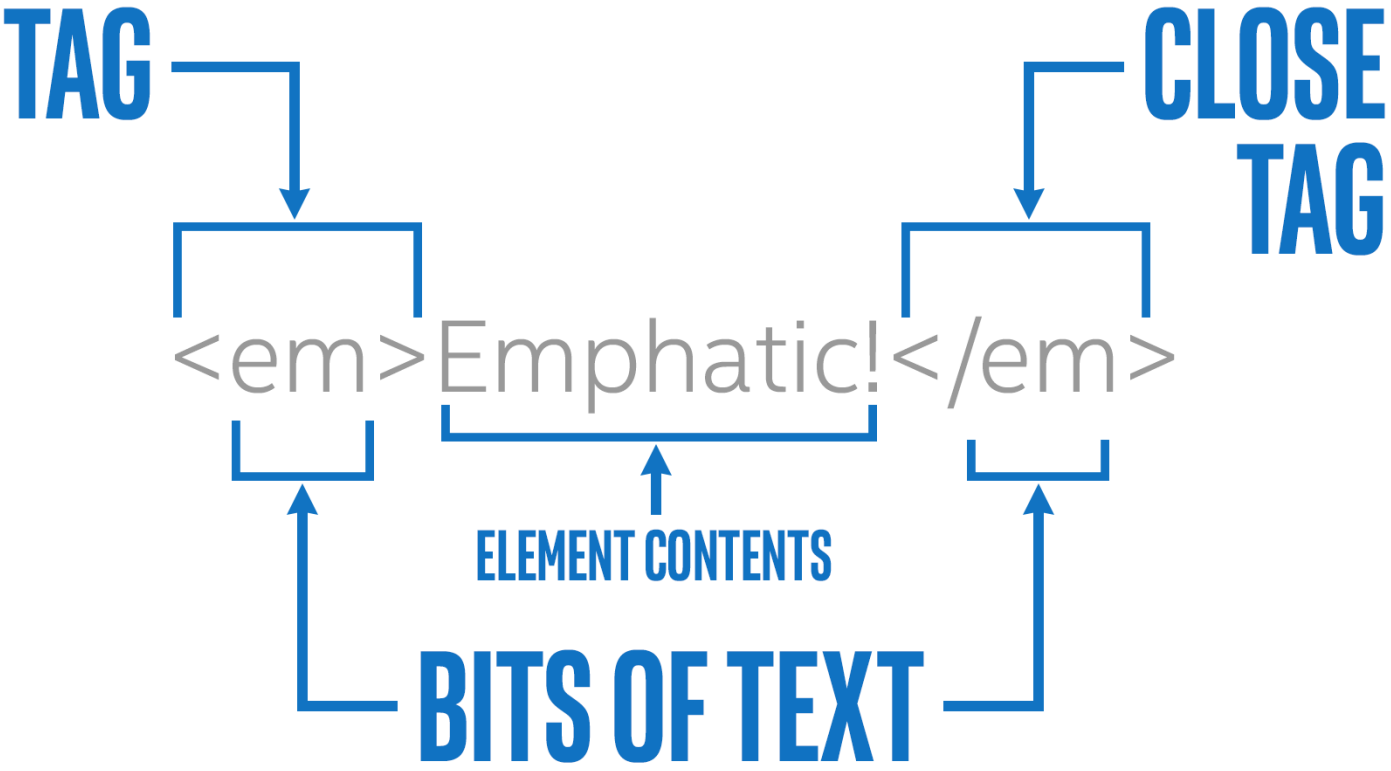
Hello World!

This example demonstrates the **getElementsByTagName** method.

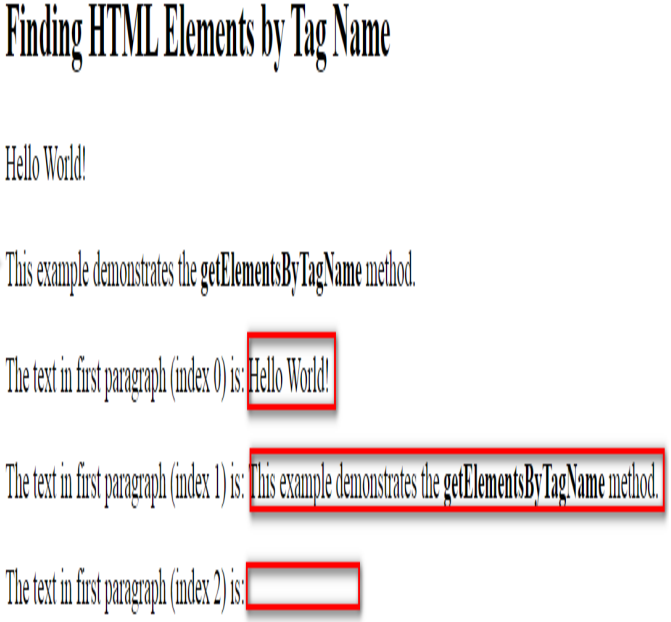
The text in first paragraph (index 0) is: Hello World!

ex03-02.html

Difference between element and tag



```html
<!DOCTYPE html>
<html>
  <body>
    <h2>Finding HTML Elements by Tag Name</h2>
    <p>Hello World!</p>
    <p>This example demonstrates the
       <b>getElementsByTagName</b> method.</p>
    <p id="demo">  </p>

    <p id="demo1">  </p>
    <p id="demo2">  </p>
    <p id="demo3">  </p>

    <script>
       var x = document.getElementsByTagName("p");
       document.getElementById("demo1").innerHTML =
         'The text in first paragraph (index 0) is: '
         + x[0].innerHTML;
       document.getElementById("demo2").innerHTML =
         'The text in first paragraph (index 1) is: '
         + x[1].innerHTML;
       document.getElementById("demo3").innerHTML =
         'The text in first paragraph (index 2) is: '
         + x[2].innerHTML;
    </script>
  </body>
</html>
```



ex03-03.html

C. Finding HTML Elements by Class Name

- If you want to find all HTML elements with the same class name, use getElementsByClassName(). This example returns a list of all elements with class="intro".

```
<!DOCTYPE html>
<html>
<body>
 <h2>Finding HTML Elements by Class Name</h2>
 <p>Hello World!</p>
 <p class="intro">The DOM is very useful.</p>
 <p class="intro">This example demonstrates the
 <b>getElementsByClassName</b> method.</p>
 <p id="demo"></p>
 <script>
   var x = document.getElementsByClassName("intro");
   document.getElementById("demo").innerHTML =
    'The first paragraph (index 0) with class="intro":
'    + x[0].innerHTML;
 </script>
</body>
</html>
```

Finding HTML Elements by Class Name

Hello World!

The DOM is very useful.

This example demonstrates the **getElementsByClassName** method.

The first paragraph (index 0) with class="intro": The DOM is very useful.

D. Finding HTML Elements by CSS Selectors

- If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method. This example returns a list of all <p> elements with class="intro".

```
<!DOCTYPE html>
<html>
<body>
  <h2>Finding HTML Elements by Query Selector</h2>
  <p>Hello World!</p>
  <p class="intro">The DOM is very useful.</p>
  <p class="intro">This example demonstrates the
<b>querySelectorAll</b> method.</p>
  <p id="demo"></p>
  <script>
    var x = document.querySelectorAll("p.intro");
    document.getElementById("demo").innerHTML =
    'The first paragraph (index 0) with class="intro": '
    + x[0].innerHTML;
  </script>
</body>
</html>
```

Finding HTML Elements by Query Selector

Hello World!

The DOM is very useful.

This example demonstrates the **querySelectorAll** method.

The first paragraph (index 0) with class="intro": The DOM is very useful.

E. Finding HTML Elements by HTML Object Collections

- This example finds the form element with id="frm1", in the forms collection, and displays all element values:

```
<!DOCTYPE html>
<html>
<body>
 <h2>Finding HTML Elements Using document.forms</h2>

 <form id="frm1" action="/action_page.php">
  First name: <input type="text" name="fname" value="Donald"><br>
  Last name: <input type="text" name="lname" value="Duck"><br><br>
  <input type="submit" value="Submit">
 </form>
 <p>Click "Try it" to display the value of each element in the form.</p>
 <button onclick="myFunction()">Try it</button>
 <p id="demo"></p>
 <script>
  function myFunction() {
   var x = document.forms["frm1"];
   var text = "";
   var i;
   for (i = 0; i < x.length ;i++) {
    text += x.elements[i].value + "<br>";
   }
   document.getElementById("demo").innerHTML = text;
  }
 </script>
</body>
</html>
```

Finding HTML Elements Using document.forms

First name: Donald
Last name: Duck

Submit

Click "Try it" to display the value of each element in the form.

Try it

Finding HTML Elements Using document.forms

First name: Donald
Last name: Duck

Submit

Click "Try it" to display the value of each element in the form.

Try it

Donald
Duck
Submit

JS插入HTML標籤的兩種方法

https://www.w3schools.com/js/js_htmldom.asp

JS插入HTML標籤的兩種方法

- 在<JavaScript>區域內操控HTML 可使用以下兩種方法：
  1. innerHTML:組完字串後，傳進語法進行渲染。效能快，但有資訊安全的風險，要確保來源沒有問題，要小心使用，因安全性問題有表單輸入的部分儘量不要使用。
  2. createElement：以DOM節點來處理。安全性較高，但效能差。

1. innerHTML:

- **innerHTML vs textContent:** innerHTML可以插入新增一個標籤及其內容，但是textContent的話只能單純增加為文字內容。例如現在希望在h1標籤中增加一個<a href=”#”>我是連結<a>，藉由以下範例來看看結果有什麼不同：

```
<h1 class=innertest></h1>
<h1 class=innertest></h1>
</h1>
<script>
    var inner = document.querySelectorAll('.innertest');
    inner[0].innerHTML = '<a href="#">我是連結</a>';
//要注意的是a標籤內的href="#"，如打成單引號'#'，則會出現錯誤，因a標籤外已使用單引號，所以必須使用雙引號做區隔。
    inner[1].textContent = '<a href="#">我是連結</a>';
</script>
```

我是連結

我是連結

p. s.

第一個h1使用innerHTML，會插入一個a標籤及其內容。

第二個h1使用textContent則是將一整段a標籤及其文字內容當成一串字串輸出。

更多innerHTML想法

- 若是要插入的innerHTML中的某些內容，希望能夠是用變數方式操作，可使用 + 來組字串：

```
<ul class='list'></ul>
<script>
    var ul = document.querySelector('.list');
    var link = "http://www.google.com";
    var name = "Lexi.L";

    ul.innerHTML = "<li><a href='" + link + "'>"
     + name
     + "</a></li>";
</script>
```

- Lexi.L

p. s.
點選Lexi.L可連結至http://www.google.com

2. createElement

- **createElemant() 與appendchild():**如果要增加標籤元素，除了innerHTML還可以使用createElemant，兩者的差別在於innerHTML是組好字串之後整串插入到HTML的方式新增一個標籤內容; 而createElemant是先動態新增一個標籤元素的節點，再對此標籤節點加入內容，最後再用appendchild()，將這個標籤元素新增為一個子節點。

在h1下插入一個標籤

- 在h1下插入一個em標籤，並增加內容及屬性。

```
<h1 class='test'></h1>

<script>
    var em = document.createElement('em'); //動態新增一個em標籤
    em.textContent = "我用createElement新增出來的"; //新增內容至em標籤中
    em.setAttribute('class', 'red'); //對em標籤增加一個屬性class ='red'
    //將整個em的標籤掛到h1底下(要先找到h1，再掛上去)
    document.querySelector('.test').appendChild(em);

    //想查詢看看新增加上去的em標籤屬性class是否為red
    attrget = document.querySelector('.test em').getAttribute('class');
    console.log(attrget);
</script>
```

我用createElement新增出來的

p.s. console.log輸出結果為：red

Exercise

- createElemant()用於for

```
<h1 id='nameid'>產品名稱</h1>
    <ul class='list'></ul>
    <script>
        var products = [{
                name: "經典款T",
                price: 390,
                storage: 10
            },
            {
                name: "刷色牛仔長褲",
                price: 590,
                storage: 2
            },
        ];
        var str = '';
        var i
        for (i = 0; i < products.length; i++) {
            var li = document.createElement('li');
            li.textContent = products[i].name;
            document.querySelector('.list').appendChild(li);
        }
    </script>
```

產品名稱

- 經典款T
- 刷色牛仔長褲

Animation

https://www.w3schools.com/js/js_htmldom.asp

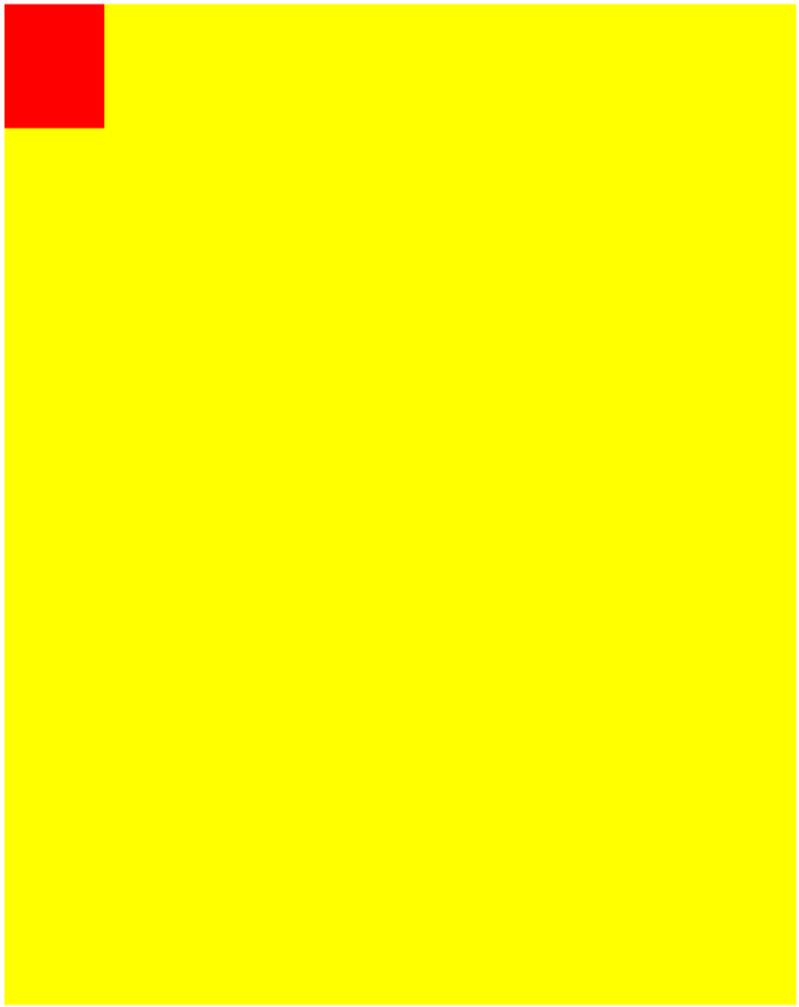
Create the Full Animation

```
var id = null;
function myMove() {
  var elem = document.getElementById("animate");
  var pos = 0;
  clearInterval(id);
  id = setInterval(frame, 5);
  function frame() {
    if (pos == 350) {
      clearInterval(id);
    } else {
      pos++;
      elem.style.top = pos + 'px';
      elem.style.left = pos + 'px';
    }
  }
}
```

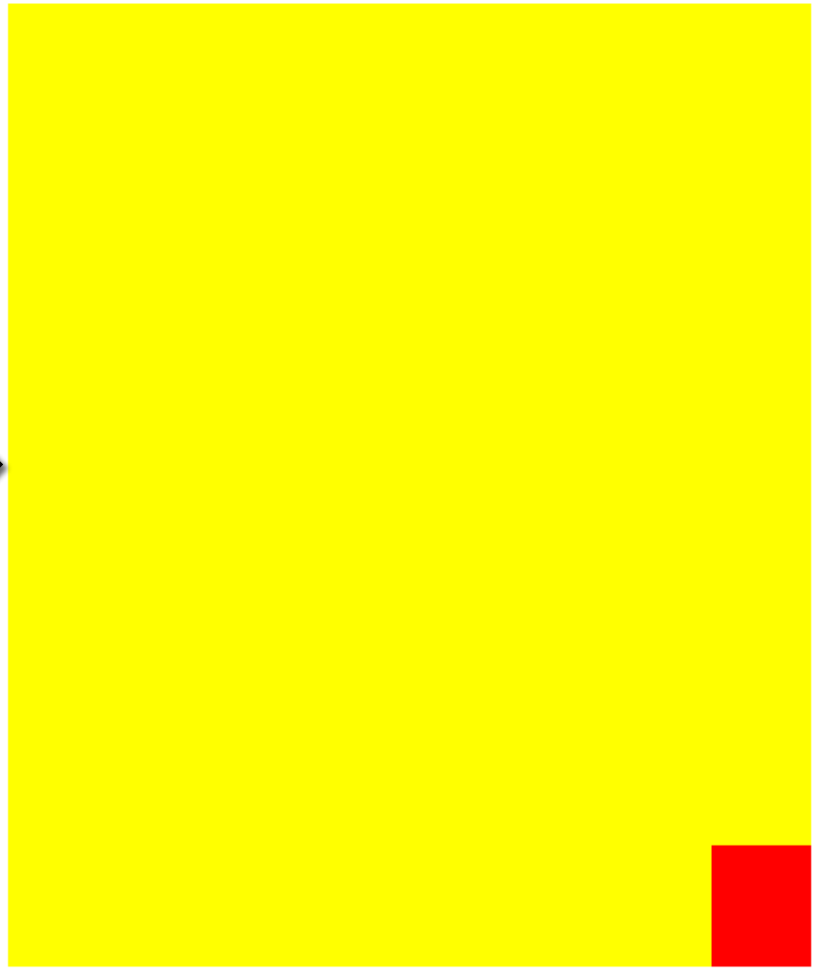
```html
<body>

<p><button onclick="myMove()">Click Me</button></p>

<div id ="container">
  <div id ="animate"></div>
</div>

<script>
var id = null;
function myMove() {
  var elem = document.getElementById("animate");
  var pos = 0;
  clearInterval(id);
  id = setInterval(frame, 5);
  function frame() {
    if (pos == 350) {
      clearInterval(id);
    } else {
      pos++;
      elem.style.top = pos + "px";
      elem.style.left = pos + "px";
    }
  }
}
</script>

</body>
</html>
```

ex03-04.html

Click Me



Click Me



The End