



# jQuery

Write less, do more.

# 市佔率

## Top JavaScript Technologies

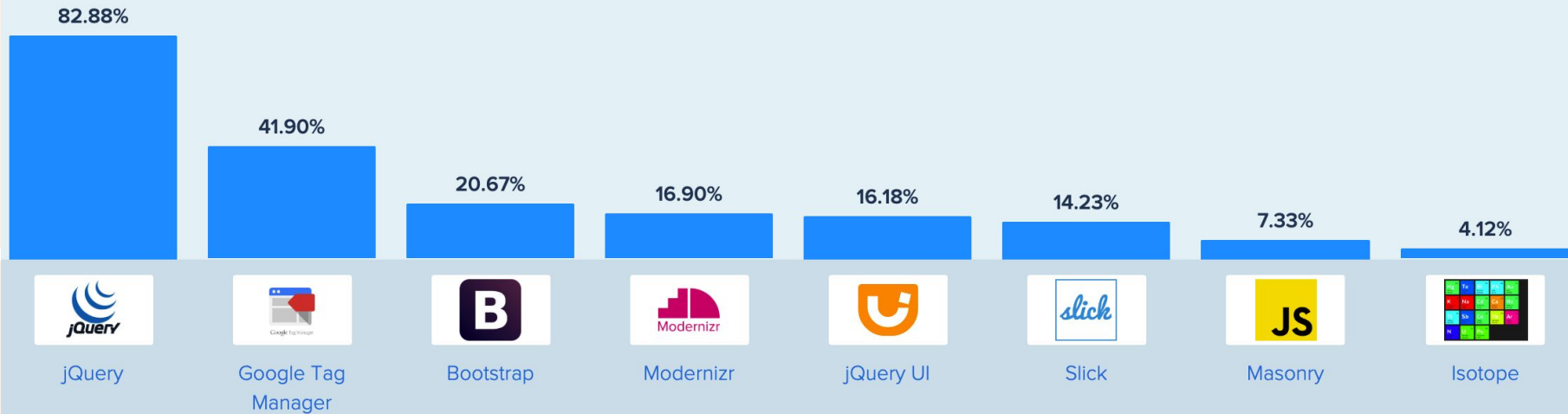
Leading JavaScript technologies share on the web

Top 10K Sites

Top 100K Sites

Top 1M Sites

The Entire Internet



# Contents of jQuery

jQuery有下列特色：

- ❑ 使用多瀏覽器開源選擇器引擎Sizzle(jQuery專案的衍生產品)進行DOM元素選擇<sup>[10]</sup>
- ❑ 基於CSS選擇器的DOM操作, 使用元素的名稱和屬性(如id和class)作為選擇DOM中節點的條件
- ❑ 事件 event
- ❑ 特效和動畫
- ❑ Ajax
- ❑ Deferred 和 Promise 物件來控制非同步處理
- ❑ JSON 解析
- ❑ 通過外掛程式擴充
- ❑ 多瀏覽器支援(不要與跨瀏覽器混淆)



# 事件驅動 VS 資料驅動




## 事件驅動

透過針對某個 DOM 節點做一個 `addEventListener` 監視事件，當事件被觸發的時候執行某個 `function` 產生資料變化或畫面改變。

## 資料驅動

直接將畫面與資料綁定關連性，未來只要資料變動，自動重新渲染畫面更新，不再需要選取 DOM 節點。





# BOM VS DOM



BOM (Browser Object Model, 瀏覽器物件模型)

DOM (Document Object Model, 文件物件模型)

前端開發者就是透過 Javascript 去呼叫 BOM 和 DOM 提供的 API 來控制瀏覽器的行為跟網頁的內容。



## DOM

Document

images

links

form

layers

others

## BOM

Window

Location

History

Navigator

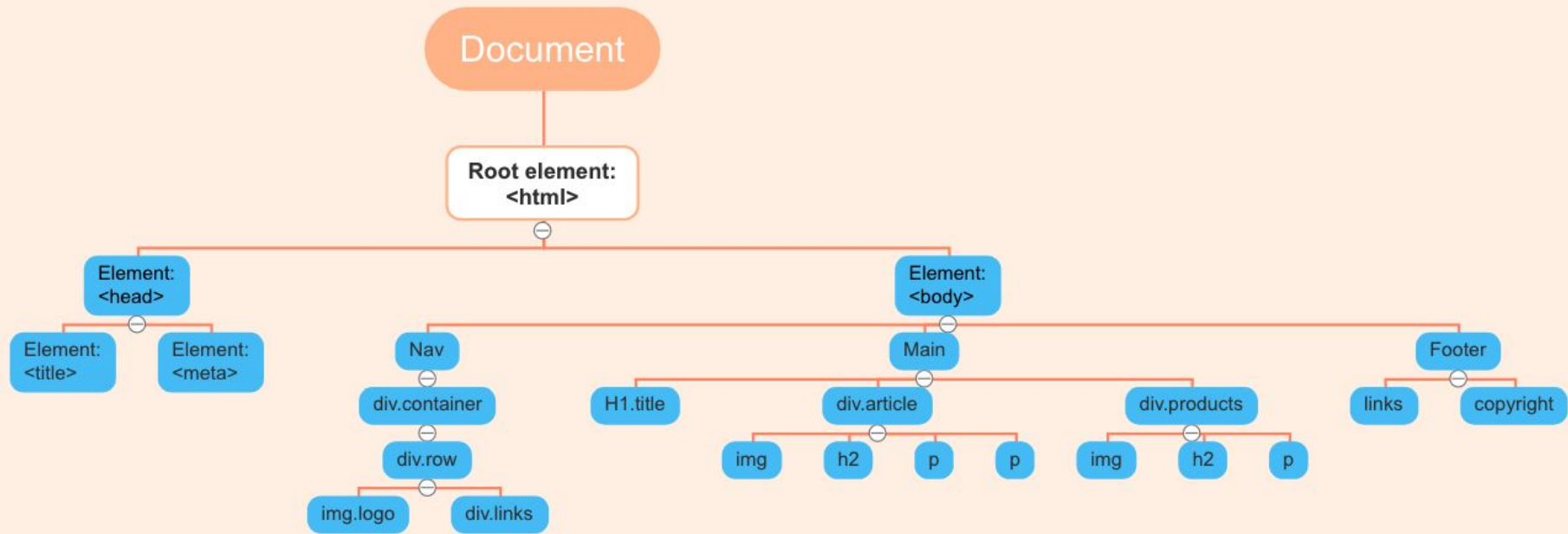
Screen

Popup

Timer

cookie





網頁架構類似一個樹狀圖，每一個 element 都是一個節點(Node)



# \$ (document).ready() VS \$(window).load()



## \$(document).ready(function):

網頁本身的 HTML 載入後就觸發 function, 適合用在想取得網頁 DOM 的時候, 確保一定有該物件, 不用擔心會找不到該物件, 而發生錯誤的情況。可以出現多個。

## \$(window).load(function):

等到 HTML 中引用的css、圖片檔案、內嵌物件(iFrame)等都載入候才會觸發, 執行時機較晚一些, 適合用在等待圖檔要確定圖片長寬數字之類的細微操作。只能出現一次, 如果有多個.load(), 則後蓋前。





## \$ (document).ready() 的實際寫法:



```
$(document).ready(function() {  
    // Do something when .ready() called.  
});
```

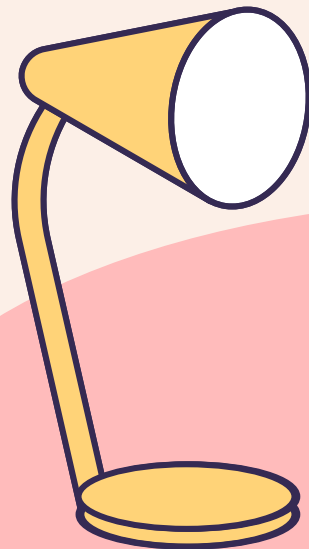
```
$(function() {  
    // short for $(document).ready().
```



```
});
```

# 01

## 載入 jQuery





## 1. 下載至本地端後讀入

<https://jquery.com/download/>

```
<script src="./js/jquery.js"></script>
```

## 2. 直接使用 CDN 方式讀入

<https://code.jquery.com/>

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"  
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohe  
tphbbj0=" crossorigin="anonymous"></script>
```

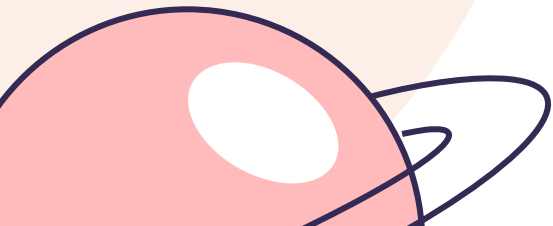






## jQuery slim

從第 3 版開始新增一個 slim 的版本，  
拿掉 ajax 跟 effect 的部份。

避免一直被人詬病檔案過於肥大問題，  
讓不需要使用這兩塊的使用者可以更快  
速的讀取 jQuery。





名稱	大小	種類
 jquery-3.6.0.js	289 KB	JavaScript
 jquery-3.6.0.slim.js	235 KB	JavaScript
 jquery-3.6.0.min.js	90 KB	JavaScript
 jquery-3.6.0.slim.min.js	72 KB	JavaScript

1.全功能無壓縮版本

2.無 ajax & effect 版本

3.全功能的壓縮版本

4.無 ajax & effect + 壓縮的版本





Your donations help fund the continued development and growth of jQuery.

SUPPORT THE PROJECT

Download API Documentation Blog Plugins Browser Support

Search



### Lightweight Footprint

Only 30kB minified and gzipped. Can also be included as an AMD module



### CSS3 Compliant

Supports CSS3 selectors to find elements as well as in style property manipulation



### Cross-Browser

Chrome, Edge, Firefox, IE, Safari, Android, iOS, and more



## Download jQuery

v3.6.0

The 1.x and 2.x branches no longer receive patches.

[View Source on GitHub](#) →

[How jQuery Works](#) →

## What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

## Resources

- [jQuery Core API Documentation](#)
- [jQuery Learning Center](#)
- [jQuery Blog](#)
- [Contribute to jQuery](#)

# 02

## 選擇器 selector





# jQuery 基本用法

`$(selector)` + 動作語法







## 原生 JavaScript selector 寫法:

- `document.getElementById("myID")`
- `document.getElementsByTagName("p")`
- `document.getElementsByClassName("class")`
  
- `document.querySelectorAll("span.a, span.c")`
- `document.querySelector("#id, .class")`



# jQuery selector 寫法縮短很多

前面需要加上 \$(), 裡面類似 css 選擇器用法

- html 標籤選擇器 → `$("div")`
- class 選擇器 → `$(".class")`
- id 選擇器 → `$("#id")`
- 屬性選擇器 → `$("a[href='#']")`
- 一次選擇多個 → `$(".btn, .btns")`

## Form selector

- :button(input, button)
- :checkbox
- :checked
- :disabled
- :enabled
- :file
- :focus
- :image

- :input
- :password
- :radio
- :reset
- :selected
- :submit
- :text

## Basic filter

- :eq() - 選取第 n 個(從0開始計算)
- :even - 選取第偶數個(從0開始計算)
- :odd - 選取第奇數個(從0開始計算)
- :first - 選取第一個
- :last - 選取最後一個
- :gt() - 選取大於第 n 個的(也可以給負數)
- :lt() - 選取小於第 n 個的(也可以給負數)
- :header - 選取 h1~h6

## Basic filter

- :not()- 不選取某個狀態, 範例為只有沒選取狀態下的 checkbox 旁的字會變色

// HTML

```
<div>
  <input type="checkbox" name="alex">
  <span>Alex</span>
</div>
<div>
  <input type="checkbox" name="bill" checked>
  <span>Bill</span>
</div>
```

// jQuery

```
$( "input:not(:checked) + span" ).css( "background", "yellow" );
```



Alex




Bill



# Child Filter



- :first-child - 選取第一個
  - :last-child - 選取最後一個
  - :first-of-type - 選取同類第一個
  - :last-of-type - 選取同類最後一個
  - :nth-child() - 第 n 個
  - :nth-last-child() - 倒數第 n 個
  - :nth-of-type()
  - :nth-last-of-type()
  - :only-child - 選取只有一個的
  - :only-of-type
- 

# Content filter

- :contains()- 選取內容包含xx

// HTML

```
<div class="names">Jason Kidd</div>
```

```
<div class="names">LeBron James</div>
```

```
<div class="names">Jason Williams</div>
```

```
<div class="names">James Harden</div>
```

// jQuery

```
$("div.names:contains('Jason')").css("background", "yellow");
```



## Content filter



- :empty- 選取內容為空的元素

// HTML

```
<table class="table empty">
```

```
<tbody>
```

```
<tr> <td>1</td><td></td> <td>3</td><td></td> </tr>
```

```
</tbody>
```

```
</table>
```

// jQuery

```
$(".empty td:empty").css("background", "yellow");
```





# Content filter

- :has()- 包含 xx 元素

// HTML

```
<div>abc</div>
```

```
<div><p>def</p></div>
```

// jQuery

```
$("div:has(p)").css("background", "yellow");
```

# Visibility Filter

- :hidden- 選取隱藏的
- :visible- 選取看得見的

// HTML

```
<div hidden>abc</div>
```

```
<div><p>def</p></div>
```

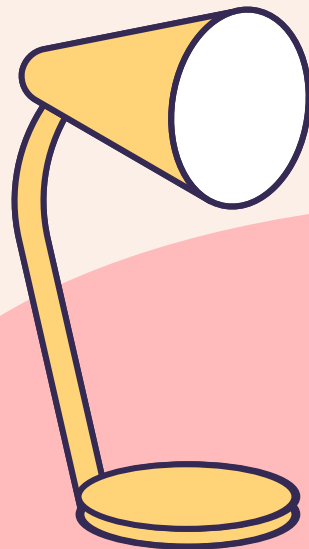
// jQuery

```
$("div:hidden").show();
```

# 02

## 遍歷


## Traversing





# Filtering



- .eq()- 選取第 n 個
  - .even()- 選取第偶數個 (從零開始計算)
  - .odd()- 選取第奇數個 (從零開始計算)
  - .first()- 選取第一個
  - .last()- 選取最後一個
- 

# Filtering

- .has()- 選取含有 xx

// HTML

```
<div>
```

```
  <div class="a1">A1</div>
```

```
</div>
```

```
<div>
```

```
  <div class="a2">A2</div>
```

```
</div>
```

// jQuery

```
$("div").has(".a1").css("background", "red");
```

# Filtering

- .not()- 選取不是 xx 的

// HTML

```
<div class="a1">A1</div>
```

```
<div class="a2">A2</div>
```

```
<div class="a3">A3</div>
```

// jQuery

```
$("div").not(".a1").css("background", "red");
```



# Tree Traversal




- `.children()`- 所有條件相符子元素(只找一層)
- `.find()`- 找條件相符子孫元素





# Tree Traversal




- .parent()- 往上選取一層
  - .closest()- 最接近祖先元素
  - .parents()- 往上選取全部
  - .parentsUntil()- 往上選取全部並設定停止的元素
- 





# Tree Traversal




- .next()- 往後找一個元素
  - .nextAll()- 往後及其之後的元素
  - .nextUntil()- 往後選但到特定元素後停止
- 



# Tree Traversal



- .prev()-往前找一個元素
  - .prevAll()-往前及其之前的元素
  - .prevUntil()-往前選但到特定元素停止
- 

# Tree Traversal

- .siblings()- 自己以外的元素

// HTML

```
<div class="a1">A1</div>
```

```
<div class="a2">A2</div>
```

```
<div class="a3">A3</div>
```

```
<div class="a4">A4</div>
```

```
<div class="a5">A5</div>
```

// jQuery

```
$(".a2").siblings().css("background", "red");
```

# Tree Traversal

- .end()- 回到上一個選取狀態

// HTML

```
<ul class="first">  
  <li class="a1">list item 1</li>  
  <li>list item 2</li>  
  <li class="a2">list item 3</li>  
</ul>  
<ul class="second">  
  <li class="a1">list item 1</li>  
  <li>list item 2</li>  
  <li class="2">list item 3</li>  
</ul>
```

// jQuery

```
$( "ul.first" )  
  .find( ".a1" )  
  .css( "background-color", "red" )  
  .end()  
  .find( ".a2" )  
  .css( "background-color", "green" );
```

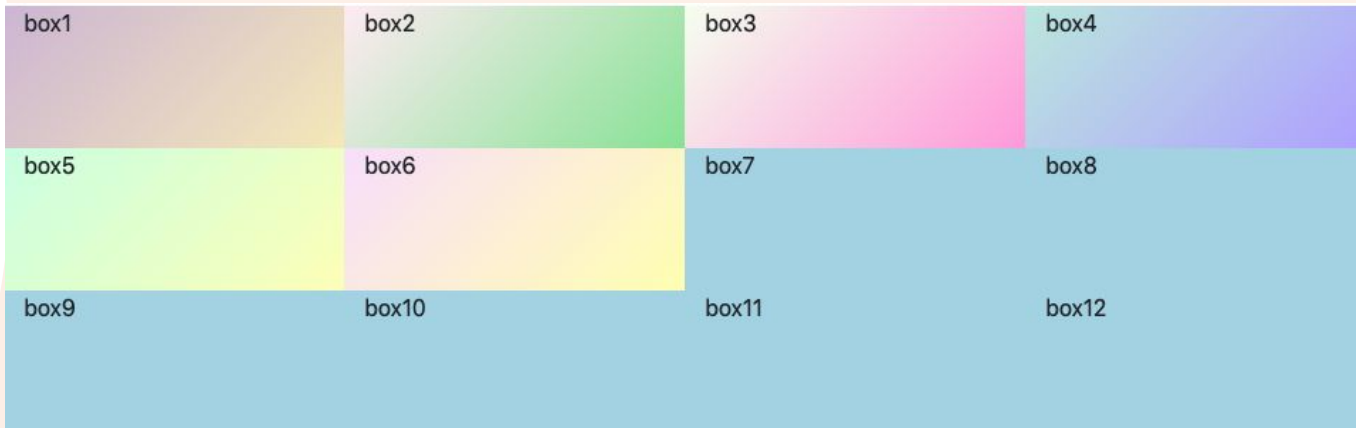
# 練習1: 列表選取

- 點擊 checkbox 後讓整行 highlight
- 用 :checkbox 選擇上層元素

<input checked="" type="checkbox"/>	Elon Musk	0912345678	Elon Musk@gmail.com
<input type="checkbox"/>	Mark Zuckerberg	0912345678	Mark Zuckerberg@gmail.com
<input checked="" type="checkbox"/>	Bill Gates	0912345678	Bill Gates@gmail.com
<input type="checkbox"/>	Larry Page	0912345678	Larry Page@gmail.com

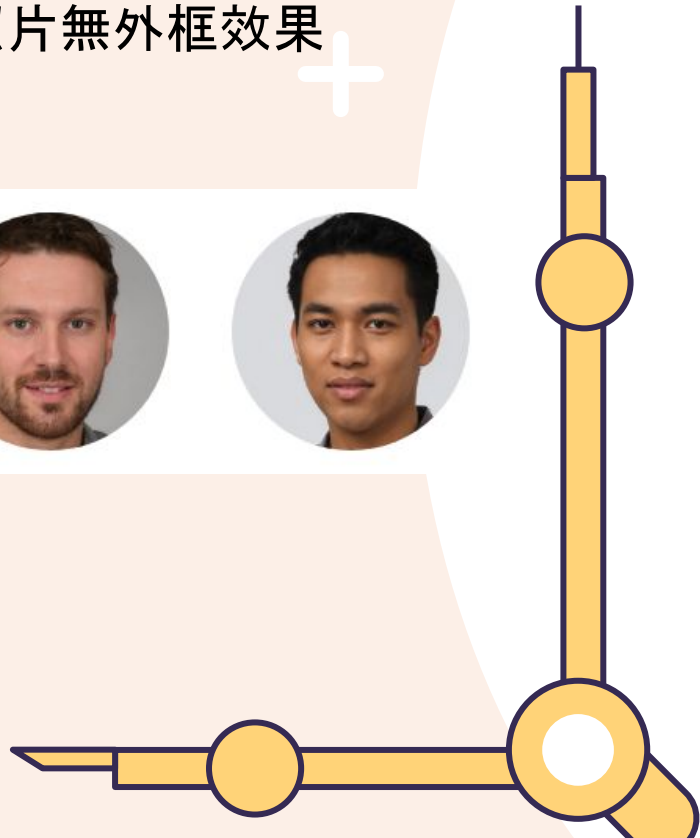
## 練習2: 往後選取

- 點擊 div 後讓後面背景顏色都變成 lightblue



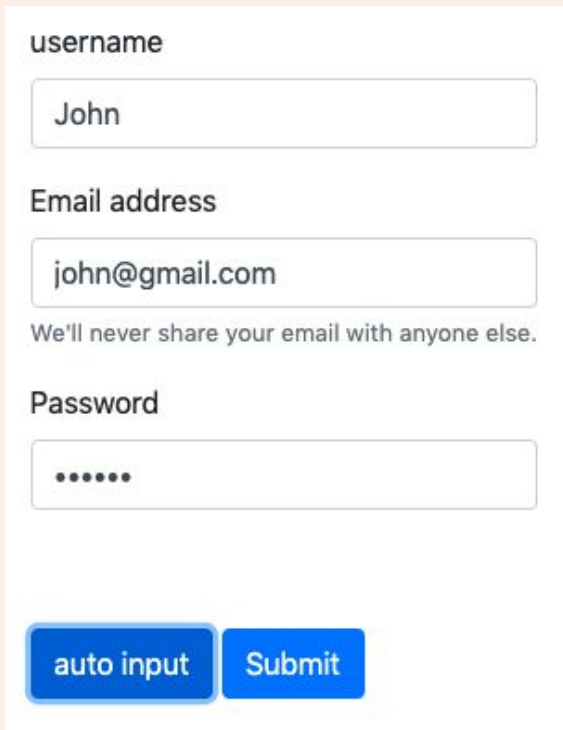
## 練習3: 照片選取

- 點擊照片只有自己會有外框，其他照片無外框效果



# 練習4：一鍵輸入

- 點擊按鈕之後自動輸入預先設定好的使用者資料



username

Email address

We'll never share your email with anyone else.

Password

auto input Submit



03


# 事件處理 event





# jQuery Event



- Mouse event
  - Keyboard event
  - Form event
  - Document/Window event
- 



## 滑鼠事件(Mouse Event)



- .click()- 點擊
  - .dblclick()- 連點
  - .contextmenu()- 右鍵
  - .mousedown()- 滑鼠按下
  - .mouseup()- 離開滑鼠
  - .mouseenter()- 滑鼠移進選擇器
  - .mouseleave()- 滑鼠離開
  - .hover()- 移進與離開
- 

## .hover()

只有 hover 的話滑進滑出都會回傳事件，要分開的話可以用這個方式寫：

```
.hover(function(){
```

```
    //偵測滑入
```

```
}, function(){
```

```
    //偵測滑出
```

```
});
```

## 練習5：輪播牆

- 透過判斷現在滑過第幾個點，將投影片移至相對應的圖片



## .index()

取得目前符合條件的選擇器是第幾個

例:

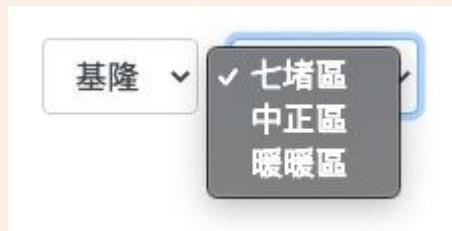
```
$(“li”).click(function(){  
    console.log($(this).index());  
})
```

## 表單事件(Form Event)

- .submit()- 點了可以送出表單
- .focus() 偵測某個 input 變成輸入狀態
- .blur() 偵測離開某個 input 輸入狀態
- .change()- 偵測 input 或 select 變化

## 練習6：下拉式選單連動

點擊第一個下拉式選單，後面資料需要跟著變動



```
let districtObj = [  
  ['七堵', '中正', '暖暖'],  
  ['中正', '大安', '信義'],  
  ['蘆洲', '三重', '新店'],  
];
```





## Document/Window Event



- .scroll()- 偵測滑鼠滾動
- .resize()- 偵測瀏覽器大小變化



## .scrollTop(), .scrollLeft()

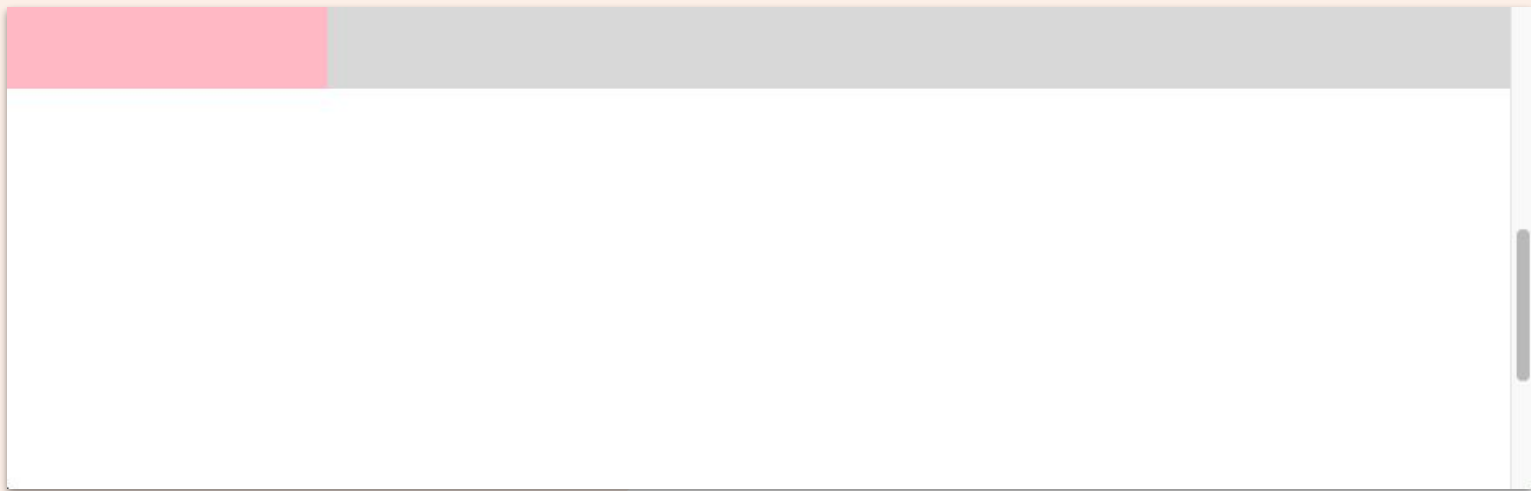
- 偵測捲軸位置或是指定捲軸位置
- ()內無值為偵測位置，有值則為指定位置

## `.width()`, `.height()`

- 偵測 selector 長寬或是指定 selector 長寬
- ()內無值為偵測長寬，有值則為指定長寬

## 練習7：偵測滾動百分比UI

1. 利用 `scrollTop()` 算出目前捲軸位置
2. 與頁面高度對比算出比例
3. 再將比例加到進度條上



## .On()

1. Selector 是後來才產生的時候綁定事件
2. 一個 selector 需要綁定多個事件時
3. 不同事件觸發一樣的動作時

## 1. selector 是動態產生時綁定事件

- 因為 jQuery 會在 HTML 畫完後根據既有的結構將事件綁到 HTML 上，若是靠程式事後產生的話，原本寫的就無法綁到 HTML 上。
- \$(選擇器(原本就存在HTML上之元素)).on( 觸發方式(原本的"click","mouseenter"...等) [, 程式動態產生的元素 ] , 事件處理(function(){}))

# +

## 1. selector 是動態產生時綁定事件

# +

.on( events [, selector ] [, data ], handler )

```
$( "#clickMe" ).on( "click", function() {  
    console.log( $( this ).text() );  
});
```

+

```
$( "ul" ).on( "click", "li", function() {  
    console.log( $( this ).text() );  
});
```

## 2. 一個 selector 需要綁定多個事件時

```
$("#on").on({  
  mouseenter: function(){  
    $("#showAction").text("Mouse Enter");  
  },  
  click: function(){  
    $("#showAction").text("Click");  
  },  
  dblclick: function(){  
    $("#showAction").text("Double Click");  
  }  
});
```



### 3. 不同事件觸發同一個動作時

```
$("#button").on("click mouseenter", function(){  
    //do something  
});
```



**off**



- 專門用在取消由 on 建立的事件

```
$('#btn').off('click', functionName);
```



## one

- 專門用在只想要執行一次就好的時候使用

```
$('#btn').one('click', function(){  
  
});
```

03

# 樣式控制 style



## .css()

- .css('css 敘述')- 取得值
- .css('css 敘述', '值')- 設定值
- 設定多個樣式

```
var style = {  
    color: "#fff",  
    background: "#000",  
};  
$(this).css(style);
```



**.css()**



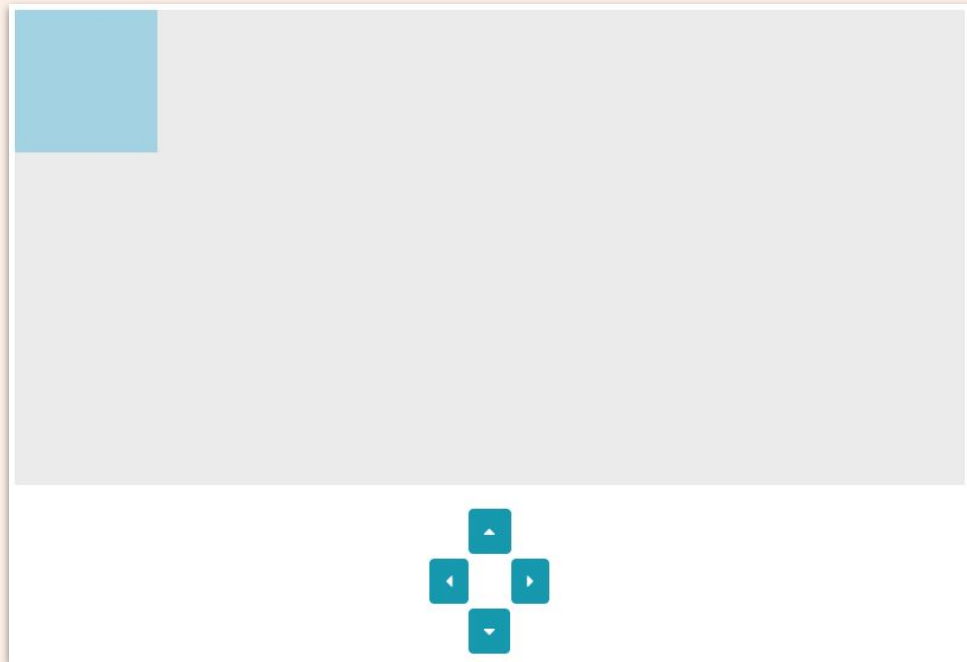
累進値:

ex: `css("left", "+=2");`



## 練習 8：方塊移動


製作一個可以依照使用者操作移動的方塊





# CSS



- .css()- 設定 css
  - .addClass()- 增加 class
  - .removeClass()- 移除 class
  - .toggleClass()- 動態切換 class
  - .hasClass()- 判斷是否有某個 class
- 



## 練習 9：利用按鈕切換文字大小

### jQuery

Lorem ipsum dolor sit amet consectetur adipisicing elit. Aperiam consectetur, iusto ratione, quas nihil exercitationem blanditiis, dolorum numquam adipisci sequi harum facilis deserunt reiciendis ducimus dicta natus maxime eligendi fugiat!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dignissimos, reprehenderit pariatur illo quaerat consectetur eius blanditiis placeat sequi natus, amet ipsam deleniti ratione unde optio, eveniet quas. Officia, minus hic?

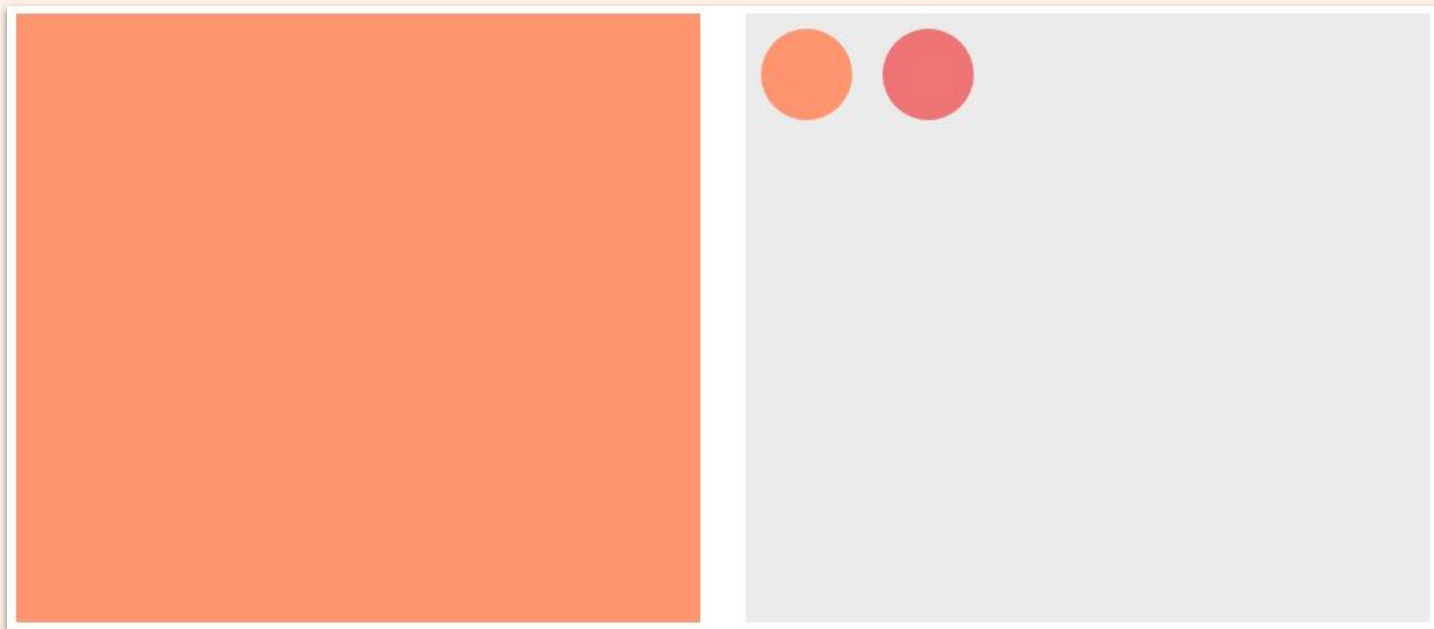
S

M

L

## 練習 10：顏色選取器

- 效果：點擊顏色圓型，左邊出現相同顏色。



# CSS 元素的寬與高

width() - 設置或取得元素的寬度

height() - 設置或取得元素的高度 括號內放值可自訂寬高

innerWidth() - 取得元素的寬度(含 padding)

innerHeight() - 取得元素的高度(含 padding)

注意: 元素內, 不含border等

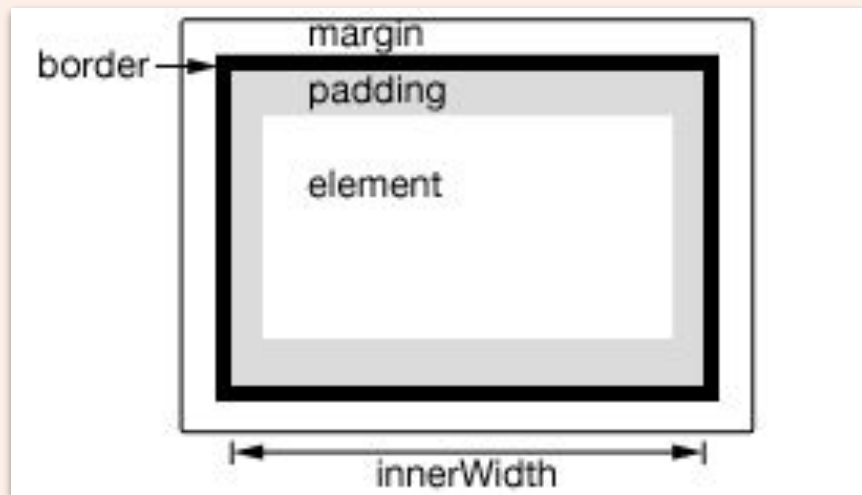
outerWidth() - 取得元素的寬度(包含 padding 和 border)

outerHeight() - 取得元素的高度(包含 padding 和 border)

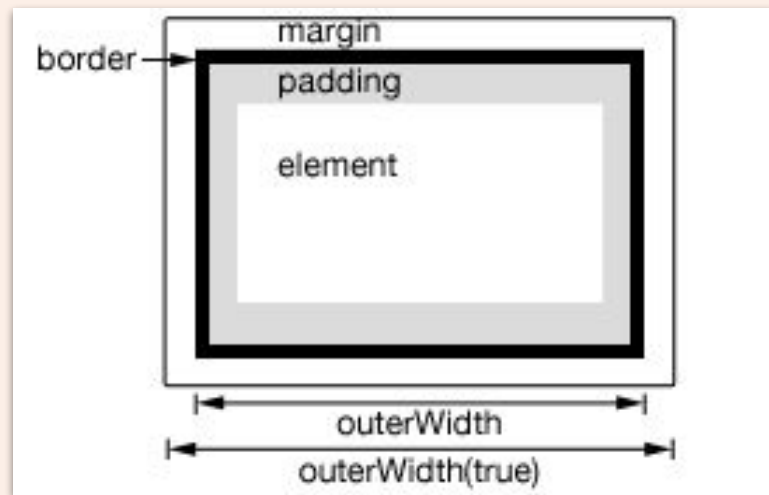
注意: 不含margin, 括號內加入true就包含margin

# CSS 元素的寬與高

innerHTML



outerWidth



## CSS 元素在視窗的絕對位置

offset() - 取得一個物件，包含top和left

offset().left - 取得X座標

offset().top - 取得Y座標

也可以傳入一個物件設定他的位置(數值為字串)

```
$('.block').offset({  
  top: '50', //數值為字串  
  left: '100',  
})
```

## CSS 元素的相對位置

`position()` - 取得一個物件，包含`top`和`left`

`position().left` - 取得X座標

`position().top` - 取得Y座標

使用方法和絕對位置基本上一樣，只是座標參考起始點位置不同。

# CSS 視窗的寬高

取得螢幕寬 `$(window).width()`

取得螢幕高 `$(window).height()`

# CSS 卷軸位置

`$(window).scrollTop()` 卷軸現在Y值

`$(window).scrollLeft()` 卷軸現在的X值



## 練習 11 : Scroll to fixed

- 效果: 製作捲動到一定超過選單高度時, 選單會貼到畫面頂端
- 提示: 偵測 scroll event, 判斷是否大於選單的位置後切換 css
- 利用 `$("#element").offset().top`; 可抓到位置再跟 `scrollTop` 比較判斷

## 練習 12：動態收起選單效果

- 效果: 使用者向下滑動時收起選單，向上滑則重新出現選單。
- 提示: `let scrollNow = $(this).scrollTop();`
- 記錄每次最後 `lastScroll`，並再次跟 `scrollNow` 比對;  
`lastScroll` 大於 `scrollNow` 則隱藏選單，  
`lastScroll` 小於 `scrollNow` 代表使用者向上滑，所以顯示選單。

## 搭配 animate.css

- <https://animate.style/>
- 將 animate.css 讀入後, 就可以使用 class 產生動畫效果
- 可再加上 `animate__infinite` 這個 class 讓動畫無限播放

## 執行一次後移除 class

```
let animate = 'animate__' + $(this).text();  
$(".box").addClass(animate).one('animationend', function() {  
  $(this).removeClass(animate);  
});
```

04


# 取得網頁內容 content





## HTML 操作



- `.text();`
  - `.html();`
  - `.attr();`
  - `.prop();`
  - `.val();`
- 

## HTML 操作 .text()

- 取得或設定 HTML tag 內的文字
- 取值: .text()
- 設值: .text(text)

## HTML 操作 .html()

- 跟.text()類似，但會取得或設定 HTML tag 的HTML內容
- 取值: .html()
- 設值: .html(html)



## HTML 操作 .attr()

- 取得或設定 HTML tag **attribute**屬性 的值
- `<img src="" alt="">`
- 取值 `.attr( attributeName )`
- 設值 `.attr( attributeName, value )`

## HTML 操作 .attr()

使用物件方式設定 attr 的值：

```
.attr({
```

```
  屬性1: 值1,
```

```
  屬性2: 值2
```

```
})
```

## 練習 13 點選後替換圖片



max SW-CK19