使用 jQuery 操作 DOM

從 jQuery 取得或改變網頁元素

什麼是 jQuery？

1. 為建立在 JavaScript 基礎之下的跨瀏覽器函式庫

2. 為解決網路應用中不同平台、技術和開發者帶來的不相容問題

3. 以較少的程式碼達到與 JavaScript 一樣的結果

4. 許多插件使用 jQuery 為基礎開發，可以廣泛應用在網頁設計

取得 DOM 元素

$(CSS選擇器)

說明

‘$’ 錢字號為 jQuery 代表性符號，可取得所有指定元素

實際用例

```
// 取得選擇器 '#navbar a' 匹配的元素
$('#navbar a')
// 取得選擇器 '.block > .content' 匹配的元素
$('.block > content')
```

.each(函式)

說明

取得所有指定元素並執行函式

實際用例

```
$(element).each(function() {
  if ($(this).hasClass('active')) {
    $(this).css({ color: 'red' })
  }
})
```

改變元素 CSS 樣式（不推薦）

.css( )

說明

直接更改元素 css 樣式

實際用例

```
1.  $(element).css({
      color: '#fff'
      backgroundColor: '#666'
    })
```

改變元素 Class（較推薦）

.addClass( )

說明

新增指定的 class

實際用例

```
1.  // 於指定元素新增 class 'active'
    $(element).addClass('active')
2.  // 於指定元素新增多個 class
    $(element).addClass('active move')
```

.removeClass( )

說明

刪除指定的 class

實際用例

```
1.  // 於指定元素刪除 class 'active'
    $(element).removeClass('active')
```

改變元素 Class

.toggleClass( )

說明

判斷有無指定的 class，新增或刪除該 class

實際用例

```
1.  // 判斷指定元素有無 class 'active'，新增或刪除 'active'
    $(element).toggleClass('active')
```

.hasClass( )

說明

判斷是否包含指定的 class

實際用例

```
1.  // 判斷指定元素有無 class 'active'，回傳 true 或 false
    $(element).hasClass('active')
```

改變元素內容

.text( )

說明

取得或以純文字重新設置指定元素內文字

實際用例

```
1.  // 取得為傳值為 'Hello World!!!'
    HTMLElement = '<h1>Hello World<span
    style="display: none;">!!!</span></h1>'
    console.log($(element).text())
2.  // 將指定元素內 HTML 取代為 'Hello'
    $(element).text('Hello')
```

.html( )

說明

取得或重新設置指定元素內的 HTML 標籤

實際用例

```
1.  // 將指定元素內 HTML 取代為 '<h2>Hello</h2>'
    $(element).html('<h2>Hello</h2>')
```

改變元素內容

.prop('outerHTML')

說明

取得包含指定元素的 HTML 標籤

實際用例

```
// 取得 $(element) 的 HTML 內容
$(element).prop('outerHTML')
```

.replaceWith( )

說明

重新設置包含指定元素的 HTML 標籤

實際用例

```
// 將指定元素取代為 '<h2>Hello</h2>'
$(element).replaceWith('<h2>Hello</h2>')
```

插入元素

.append( )

說明

於指定元素內的最後方插入元素

實際用例

```
1.  // 在指定元素內的最後方插入 '<span>Hello</span>'
    $(element).append('<span>Hello</span>')
```

.prepend( )

說明

於指定元素內的最前方插入元素

實際用例

```
1.  // 在指定元素內的最前方插入 '<span>Hello</span>'
    $(element).prepend('<span>Hello</span>')
```

刪除元素

.empty( )

說明

刪除指定元素內的所有元素及文字

實際用例

```
$(element).empty()
```

.remove( )

說明

刪除指定元素

實際用例

```
$(element).remove()
```

其他選取元素的方法

.children( )

說明

取得指定元素的所有子元素

.first( )

說明

取得指定元素的第一個子元素

受空格符影響

.last( )

說明

取得指定元素的最後一個子元素

受空格符影響

.siblings( )

說明

取得指定元素的兄弟元素

.parent( )

說明

取得指定元素的父元素

.closest( )

說明

取得最近的指定父元素

動畫原理及應用

使用 jQuery 製作元素動畫

基礎動畫

.hide( )

說明

控制元素隱藏

.show( )

說明

控制元素顯示

.fadeIn( )

說明

控制元素淡入

.fadeOut( )

說明

控制元素淡出

.slideDown( )

說明

控制元素展開

.slideUp( )

說明

控制元素收合

自定義動畫

.animate(屬性, 時間, 速度曲線, 完成函式)

基本用法

```
$('element').animate({
  width: '20px',
  left: '100px',
}, 1000, 'swing', function() {
  console.log('動畫完成')
})
```

Easing 參考網址

https://easings.net/

細微調整

```
$('element').animate({
  width: '20px',
  left: '+=100px', // 以原有數值計算
}, {
  duration: 1000,
  specialEasing: { // 個別設定easing
    width: 'easeInOutCubic',
    left: 'easeOutBounce'
  }
  complete: function() {
    console.log('動畫完成')
  }
})
```

動畫串連

```
$('element').animate({
  width: '20px',
  left: '100px',
}, 1000, 'swing')
.animate({
  width: '50px',
  left: '+=50px',
}, 500, 'linear')
.delay(500) // 中間延遲
.animate({
  left: 0,
}, 500, 'easeOutBounce')
```

自定義動畫

GSAP - 解決 jQuery animate 無法達到的重播、transform 等功能

默認設置

```
const tl = gsap.timeline({
  paused: true,
  onStart: function() {
    console.log('動畫開始')
  },
  onComplete: function() {
    console.log('動畫完成')
    // seek 設定時間軸
    tl.pause().seek(0)
  }
})
```

to – 元素動作至設定值（使用keyframes）

```
tl.to('element01', { keyframes: [{
    width: '20px',
    x: '100px', // 設定 translateX
    duration: 0.5,
    ease: 'power4.in'
  }, {
    width: '50px',
    x: '+=200px', // 相對位移
    rotation: 45,
    duration: 1,
    delay: 1,
    repeat: 2,
    yoyo: true, // 如同 reverse
    ease: 'expo.inOut'
  }, { ... }]
})
```

to（使用串連並設定執行時間差）

```
tl.to('element02', {
  width: '30px',
  height: '30px',
  x: '100px',
  borderRadius: '50%',
  duration: 1
}, 0) // 執行時間軸
tl.add('timePin', 1) // 1 秒位置設置圖釘
tl.to('element02', {
  x: '+=100px',
  duration: 1
}, 'timePin+=2') // 時間軸為 3 秒時執行

$('button.gsap').click(function() {
  tl.play()
})
```

AJAX 非同步請求

跨伺服器請求資料

什麼是AJAX？

1. AJAX 為非同步的 JavaScript 與 XML 技術

2. 無須重新整理，就能即時地透過瀏覽器與伺服器溝通，取得資料

3. 資料請求回應通常是以三種資料格式之一（HTML、XML、JSON）

4. 因 AJAX 主要於前端執行，並且是按需取資料，減輕了伺服器的負擔

基本用法

jQuery AJAX

```
$.ajax({
    url: 'API網址',
    method: 使用方法（預設 get）,
    data: { 需傳送的資料 },
    success: function(res) { 請求成功時回傳 res },
    error: function( ) { 請求失敗時執行 },
})
```

method 常見方法

```
get / post / put / delete
[GET] /get              取得所有物件
[GET] /get/:id          取得指定物件
[POST] /post            新增物件
[PUT] /put/:id          更新指定物件
[DELETE] /delete/:id    刪除指定物件
```

AJAX 回調函式

非同步性

```
const data = null

$.ajax({
  url: '/',
  success: function(res) {
    data = res
  }
})
console.log(data)  // null
```

callback 回調函式

```
const data = null

function getData(callback) {
  $.ajax({
    url: '/',
    success: function(res) {
      callback(res)
    }
  })
}
getData(function(res) {
  data = res
  getAnotherData(function(res) {
    data = res
  })
})
```

Promise 防止 callback 地獄

```
const data = null

$.ajax({
  url: '/',
}).then(function(res) {
  // 請求成功時回傳
  data = res
  return $.ajax('/another_url')
}, function() {
  // 請求失敗時執行
}).then(function(res) {
  data = res
  return $.ajax('/last_url')
}, function() {
  //
})
```