

物件以及事件觸發

範例 1-1(頁首頁尾-個別選取物件以及觸發事件)

index.html

```
<div id="page">
  <div id="header"></div>
  <div id="mainarea">
    <div id="sidebar">
      <h2>最新回應</h2>
      <p class="topic" author="方勉之">身為一個小前端,就等廖大開課啦!</p><br />
      5月26日 - 方勉之
      <br />
      <br />
      <p class="topic" author="Nicole Liu">你好 請問接下來的開課時間有確定嗎??是幾月呢??</p><br />
      5月12日 - Nicole Liu
      <br />
      <br />
      <p class="topic" author="洧杰">抱歉,目前人數已滿, 且之後有行程規劃,所以暫時不開課了^_^ </p><br />
      5月6日 - 洧杰
      <br />
      <br />
      <p class="topic" author="Kate">現在這課程還有嗎? </p><br />
      4月28日 - Kate
      <br />
      <br />
      <p class="topic" author="盧均林">找sass馬上就找到今天的講者</p><br />
      4月20日 - 盧均林
    </div>
  </div>
  <div id="footer"></div>
</div>
```

common.js

```
// 選取物件
var clickHeader = function () {
    alert("我是header的部分");
}

var clickFooter = function () {
    alert("我是footer的部分");
}
document.getElementById("header").onclick = clickHeader;
document.getElementById("footer").onclick = clickFooter;
```

- document.getElementById
 - 透過Id選擇 html上的 DOM 元件
- onclick
 - 滑鼠點擊事件
- alert
 - 瀏覽器彈出訊息框

範例 1-2(透過class 取得多個DOM 元件)

index.html(同1-1)

common.js

```
// 選取物件
var sidebar = document.getElementsByClassName("topic");

alert("總共選到幾個元件" + sidebar.length);
for (var i = 0; i < sidebar.length; i++) {
    sidebar[i].onclick = function () {
        // alert("作者是" + this.getAttribute("author"));
        console.log("作者是" + this.getAttribute("author"));
    }
}
```

- document.getElementsByClassName
 - 透過class 選取 html上的 DOM 元件
 - 一次會選取多個 DOM 元件並存在陣列中
- 用 alert 的問題
 - 在debug的時候會一直跳出訊息
 - 無法得知 事件位置
- console.log("訊息")
 - 在開發者工具顯示訊息

- 可以查看目前 `log` 的位置
- 循序顯示且不會跳出訊息，方便debug
- 查看陣列裡有多少個物件
 - 陣列.length
- this
 - 代表當下的物件

範例 1-3 (透過css 選擇器語法選取DOM物件)

index.html(同1-1)

common.js

```
// 選取物件
var header = document.querySelector("#header");
var footer = document.querySelector("#footer");
var sidebar = document.querySelectorAll("#sidebar .topic");

header.onclick = function () {
    console.log("我是header的部分");
}

footer.onclick = function () {
    console.log("我是footer的部分");
};

for (var i = 0; i < sidebar.length; i++) {
    sidebar[i].onclick = function () {
        alert("作者是" + item.getAttribute("author"));
    }
}
```

- 使用 `document.getElementById`與 `document.getElementsByClassName` 的問題
 - 只能針對 `Id` 或是 `Class` 做選取，無法像CSS選擇器一樣混合使用
- `document.querySelector`(使用css選擇器)
 - 抓取html上的第一個DOM物件
 - 跟jQuery一樣透過CSS選擇器選擇物件
- `document.querySelectorAll`(使用css選擇器)
 - 抓取html上的多個DOM物件
 - 跟jQuery一樣透過CSS選擇器選擇物件
 - 需透過 `for`迴圈 等 處理array方法做處理

範例 1-4(透過 `addEventListener` 綁定 事件)

index.html(同1-1)

common.js

```
var header = document.querySelector("#header");
var footer = document.querySelector("#footer");
var sidebar = document.querySelectorAll("#sidebar .topic");

header.addEventListener("click",function(){
    console.log("我是header的部分");
})

footer.addEventListener("click",function(){
    console.log("我是footer的部分");
})

console.log("總共選到幾個DOM元件:" + sidebar.length);

for (var i = 0; i < sidebar.length; i++) {
    sidebar[i].addEventListener("click",function(){
        console.log("作者是"+this.getAttribute("author"));
    })
}
```

- 只使用 `onclick` 的問題
 - 這類 方法 只支援 Html DOM 物件綁定
 - 無法靈活 綁定 事件給 DOM 物件
- `addEventListener`
 - 支援 所有 javascript DOM 物件
 - 支援事件
 - `click`
 - `mouseover`
 - `mouseleave`
 - `hashchange`
 - 可以靈活綁定事件

Ajax

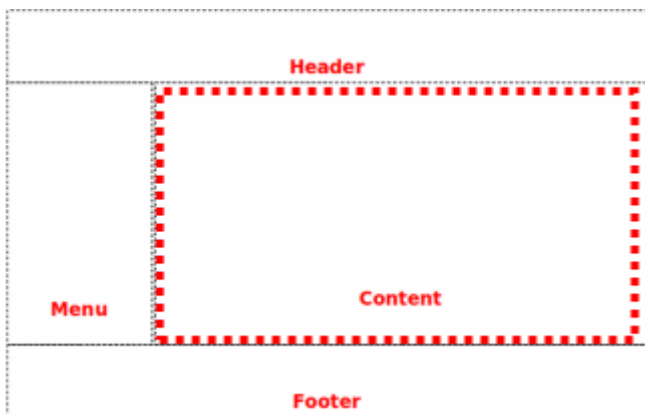
什麼是Ajax

AJAX(Asynchronous JavaScript and XML)主要目的在於提高網頁互動性、速度、可用性。傳統網頁中當使用者按下了送出按鈕時，則必須一定要等待伺服器回應後才可以繼續操作網頁。使用 **AJAX** 方式送出網頁後仍然可以繼續操作畫面，具有更佳良好使用者體驗，不需要更新整個網頁，只更新需要部分即可。

即在當下的頁面在不重新整理頁面(**F5**)的情況下，載入外部的資料。

例子:在註冊畫面的時候，輸入使用者名稱，輸入完時會顯示使用者已存在

Ajax動態載入



以往網頁設計中如果有相同頁首頁尾，一般網頁設計師多半都會使用複製貼上。製作過程中有異動則將全部網頁重新複製貼上一次。

現在可以透過 **AJAX** 動態載入，把頁首頁尾內容動態載入進來，只需要維護一份檔案即可，再也不需要複製貼上了，既省時又省力。

重複的頁面可以透過**Ajax**動態載入，只須要維護一份就好

範例 1-5(頁首頁尾-個別獨立檔案)

index.html、page1.html、page2.html、page3.html

```
<div id="page">
  <div id="header"></div>
  <div id="mainarea">
  <div id="footer"></div>
</div>
```

header.html

```
<div id="headerleft">
  <h1>
    <a href="#">前端工程師之路</a></h1>
    <div class="description">
      記載著在網路世界的學習心得與技術分享
    </div>
</div>
<div id="menulinks">
  <a href="index.html"><span>回首頁</span></a>
  <a href="page1.html"><span>何謂前端</span></a>
  <a href="page2.html"><span>必備技能</span></a>
  <a href="page3.html"><span>未來趨勢</span></a>
</div>
```

footer.html

前端工程師之路 版權所有 Copyright ©

common.js

```
// 選取物件
var header = document.querySelector("#header");
var footer = document.querySelector("#footer");
```

```
fetch('header.html').then(response => {
  return response.text();
}).then(data => {
  header.innerHTML = data;
});
```

```
fetch('footer.html').then(response => {
  return response.text();
}).then(data => {
  footer.innerHTML = data;
});
```

- `fetch` 方法可以動態載入遠端url資料。

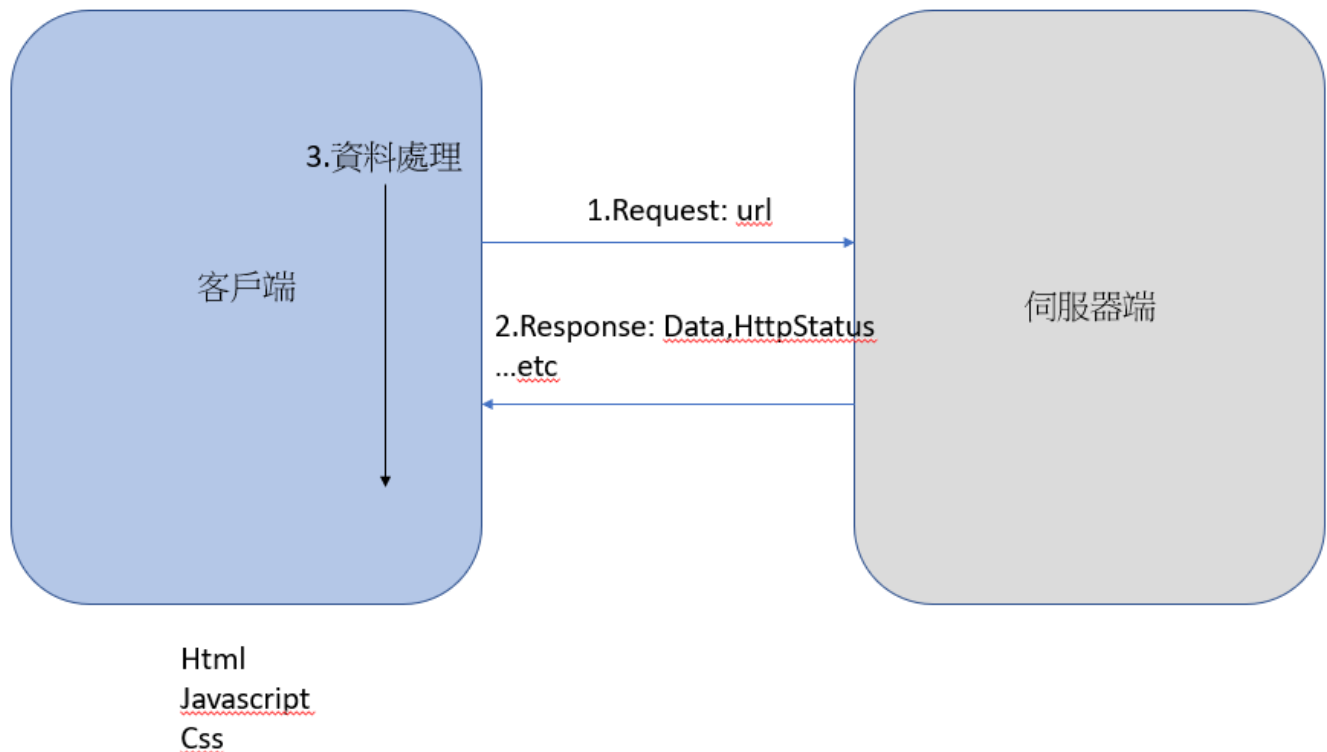
- Fetch 動態載入頁面的基本使用方法

```
fetch(url).then(response => {  
  // 2. 針對 Response 回應作處理  
  return response.text();  
}).then(data => {  
  // 3. 針對取得的資料做處理  
});
```

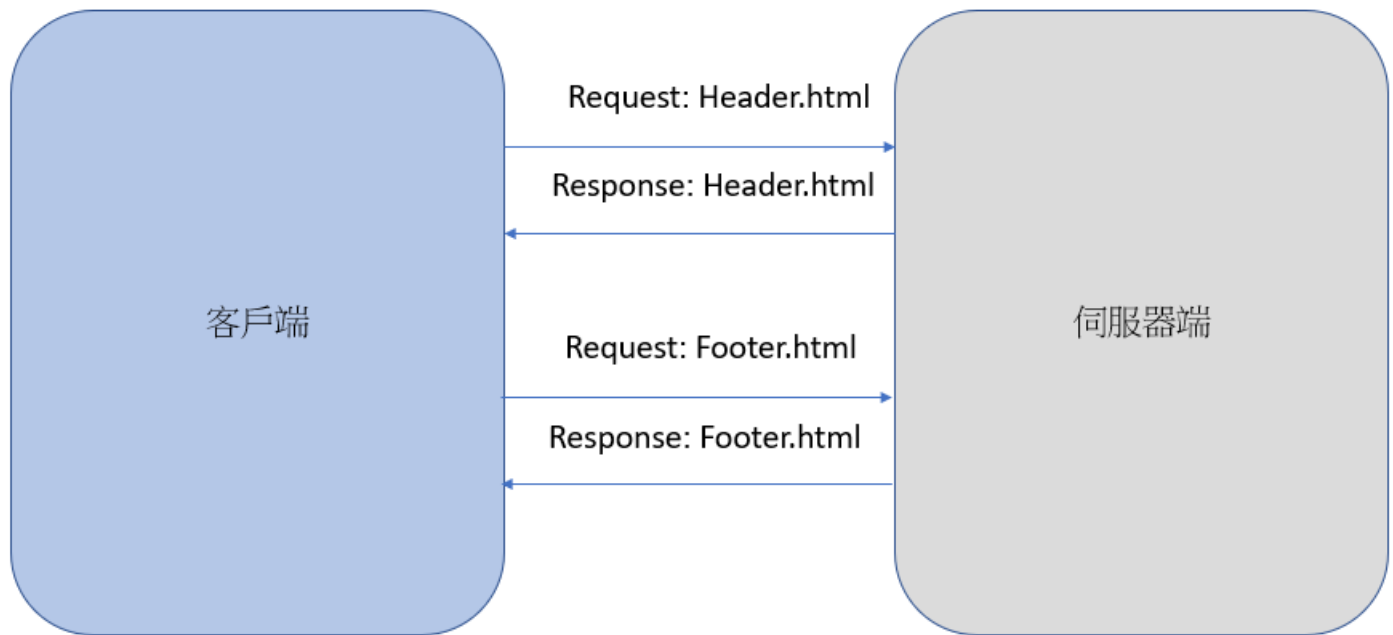
- then 之後的意思

- 代表之後對之前的事件做什麼處理

1. Request指定的Url
2. 針對 Response的資料做處理(資料格式、HttpStatus)
3. 針對 處理完的資料做操作，像是顯示在Html上



- 將response 中的拿到的 text 傳到下一層
- 使用複製貼上維護網頁非常費時費力，當有修改則須重頭複製貼上一次。
- 可以使用 **AJAX** 動態載入的方式來達到相同效果。



範例 1-6(選單-共用並且載入選單)

index.html、Page1.html、Page2.html、Page3.html、Page4.html、Page5.html

```
</head>
<body>
  <div class="menu"></div>
  <div class="content">
```

Menu.html

```
<a href="index.html">首頁</a>
<a href="Page1.html">開課時間</a>
<a href="Page2.html">教學分享</a>
<a href="Page3.html">交流討論</a>
<a href="Page4.html">課程詢問</a>
<a href="Page5.html">聯絡我們</a>
```

common.js


```

fetch('Menu.html').then(response => {
    return response.text();
}).then(data => {
    let url = location.href;
    let href =url.substr(url.lastIndexOf("/") + 1);
    document.querySelector(".menu").innerHTML = data;
    document.querySelector("a[href='" + (href || "index.html") + "']").className = "selected";
});

```

- `location.href` 可以取得目前瀏覽器的網址
- `substr`方法可指定開始位置取出字串到取出長度，第二個參數可省略。(參考下圖)

```

"abcdefg".substr(2,4) //"cdef"
"abcdefg".substr(4,2) //"ef"
"abcdefg".substring(2,4) //"cd"
"abcdefg".substring(4,2) //"cd"

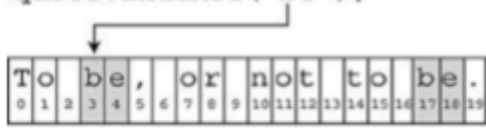
```

- `substring` 方法取出兩個索引數之間字串，第二個參數可省略。(參考上圖)
- `lastIndexOf` 方法可以計算出現文字索引位置。(沒有該字串則回傳-1)

```

quote.indexOf('be');


```



```

quote.lastIndexOf('be');

```



- 短路邏輯(short-circuit evaluation): || 先判斷左是否為真，再判斷右是否為真。

範例 1-7(選單-載入內容)

index.html

```

</div>
<div class="content"></div>
</body>

```

Page0.html~Page5.html

```


<h1>首頁</h1>
<h3>Responsive Web Design 網頁版型設計</h3>

```

common.js

```

// 第一次進網頁的時候，取 menu 第一個 連結載入至頁面
var firstButton = document.querySelector(".menu>a");
firstButton.className = "selected";
loadPageToContent(firstButton.getAttribute("href"));

// 對menu 的每一個選項做事件綁定
document.querySelectorAll(".menu>a").forEach(item => {

    item.onclick = function (event) {
        event.preventDefault();
        // 清空被選擇的物件
        document.querySelector(".selected").className = "";
        this.className = "selected";
        loadPageToContent(this.href);

    }

});

// 將指定的 href頁面 load頁面
function loadPageToContent(href) {
    fetch(href).then(response => {
        return response.text();
    }).then(data => {
        document.querySelector(".content").innerHTML = data;
    });
}

```

- `getAttribute("屬性")` 取得該 DOM 元件的該屬性的值
- `event.preventDefault` 方法可以停止超連結預設動作(包含表單提交)。
- `this` 表示目前被 click 的那個 DOM 物件。
- 問題: 當重新整理頁面的時候，因為url為第一頁，所以頁面會重新到第一頁

範例 1-8(選單-具有獨立網址)

index.html、Page0.html~Page5.html 同範例 1-7
commom.js

```

var menuBar = document.querySelectorAll(".menu>a");

loadPageToContent(document.querySelector("a[href='" + (location.hash.substr(1) || "Page0.html") + "']"));

menuBar.forEach(item => {
    item.onclick=function(){
        document.querySelector(".selected").className = "";
        loadPageToContent(this);
        event.preventDefault();
    }
});

function loadPageToContent(domObject) {
    var href = domObject.getAttribute("href");
    fetch(href).then(response => {
        return response.text();
    }).then(data => {
        domObject.classList.add("selected");
        document.querySelector(".content").innerHTML = data;
        location.hash = href;
    });
}

```

- `location.hash` 可以產生一個帶有#網址。
- 設定 `hash` 不必帶入#字號，取得 `hash` 則會包含#字號。
- 問題: 當回到上下頁時，雖然url改變，但是頁面內容卻沒有改變
 - 原因: 觸發改變頁面的事件 是 `menuBar` 上的連結，不是url

範例 1-9(選單-可使用回上下頁按鈕)

index.html、Page0.html~Page5.html 同範例 1-8

common.js

```

var menuBar = document.querySelectorAll(".menu>a");
// 第一次執行
loadPageToContent(document.querySelector("a[href='" + (location.hash.substr(1) || "Page0.html") + "']"));

menuBar.forEach(item => {
    item.onclick = function (event) {
        location.hash = this.getAttribute("href");
        event.preventDefault();
    }
});

function loadPageToContent(domObject) {
    var href = domObject.getAttribute("href");
    fetch(href).then(response => {
        return response.text();
    }).then(data => {

        domObject.classList.add("selected");
        document.querySelector(".content").innerHTML = data;
    });
}

window.addEventListener("hashchange", function () {
    document.querySelector(".selected").className = "";
    loadPageToContent(document.querySelector("[href='" + (location.hash.substr(1) + "']"));
});

```

- 使用 `window.addEventListener("hashchange",function)`，可以綁定 `hashchange` 事件，當 `hash` 有變動就會觸發function。

範例 1-10(處理找不到該頁面的事件)

index.html、Page0.html~Page5.html 同範例 1-8

common.js

```

var menuBar = document.querySelectorAll(".menu>a");
// 第一次執行
loadPageToContent(document.querySelector("a[href='" + (location.hash.substr(1) || "Page0.html") + "']"));

menuBar.forEach(item => {

    item.onclick = function (event) {

        location.hash = this.getAttribute("href");
        event.preventDefault();
    }
});

function loadPageToContent(domObject) {
    var href = domObject.getAttribute("href");
    fetch(href).then(response => {
        // 當回應正確的時候
        if (response.ok) {
            return response.text();
        }
        // 當回應錯誤的時候
        else {
            // 拋出錯誤訊息
            throw new Error('找不到該網頁');
        }
    }).then(data => {
        menuBar.forEach(item => {
            item.removeAttribute("class");
        });
        domObject.classList.add("selected");
        document.querySelector(".content").innerHTML = data;
    }).catch(
        // 處理錯誤訊息
        function (error) {
            document.querySelector(".content").innerHTML = error.message;
        }
    );
}

window.addEventListener("hashchange", function () {
    loadPageToContent(document.querySelector("[href='" + (location.hash.substr(1) + "']"));
});

```

Http 請求狀態碼

比較常用的狀態碼

- [MDN](#)
- [保哥](#)

重要的分類

- 2xx - 成功 (OK)
- 4xx - 用戶端錯誤 (Client Error)
- 5xx - 伺服器錯誤 (Server Error)

範例 1-11(選單事件-多重事件綁定)

index.html

```
<body>
  <div id="menu"></div>
  <div class="content">
    <h1>課程詢問</h1>
```

menu.html

```
<body>
  <div id="menu"></div>
  <div class="content">
    <h1>課程詢問</h1>
```

index.js

```
fetch('menu.html').then(response => {
  return response.text();
}).then(data => {
  document.querySelector("#menu").innerHTML = data;
  document.querySelectorAll("a").forEach(item => {
    item.onmouseleave = function () {
      var temp = this.text;
      this.innerText = this.dataset.english;
      this.dataset.english = temp;
    }

    item.onmouseenter = function () {
      var temp = this.text;
      this.innerText = this.dataset.english;
      this.dataset.english = temp;
    }
  })
});
```

- 先將menu.html的資料塞到 #menu中

- 再針對每一個連結綁定兩個事件，一個是`mouseleave`、一個是`mouseenter`，將暫存在`english attribute` 的文字塞到`text`

範例 1-12(選單事件-多重事件判斷)

`index.html`、`menu.html` 同範例1-11

`index.js`

```
fetch('menu.html').then(response => {
  return response.text();
}).then(data => {
  document.querySelector("#menu").innerHTML = data;
  document.querySelectorAll("a").forEach(item => {
    item.onmouseleave = function () {
      var temp = this.text;
      this.innerText = this.dataset.english;
      this.dataset.english = temp;
    }

    item.onmouseenter = function () {
      var temp = this.text;
      this.innerText = this.dataset.english;
      this.dataset.english = temp;
    }

    item.onclick = function (event) {
      event.preventDefault();
      fetch(this.href).then(response => {
        return response.text();
      }).then(data=>{
        document.querySelector("#content").innerHTML = data;
      });
    }
  })
});
```

- 新增了一個`click`事件
- `this` 是指當下的這個`dom`物件
- 透過`fetch`將`response`的值塞進`content`裡

範例 1-13(頁首頁尾-共用相同檔案)

`index.html`、`page1.html`、`page2.html`、`page3.html` 同範例 1-1
`common.html`

```

<div>
  <div id="header">
    <div id="headerleft">
      <h1>
        <a href="#">前端工程師之路</a></h1>
      <div class="description">
        記載著在網路世界的學習心得與技術分享
      </div>
    </div>
    <div id="menulinks">
      <a href="index.html"><span>回首頁</span></a>
      <a href="page1.html"><span>何謂前端</span></a>
      <a href="page2.html"><span>必備技能</span></a>
      <a href="page3.html"><span>未來趨勢</span></a>
    </div>
  </div>
  <div id="footer">
    <div>前端工程師之路 版權所有 Copyright</div>
  </div>
</div>

```

common.js

```

fetch('common.html').then(response => {
  // 將拿到的資料作處理，只回傳text的部分
  return response.text();
}).then(data => {
  // console.log(data);
  // 將 string 轉成DOM物件
  const doc = new DOMParser().parseFromString(data, "text/xml");

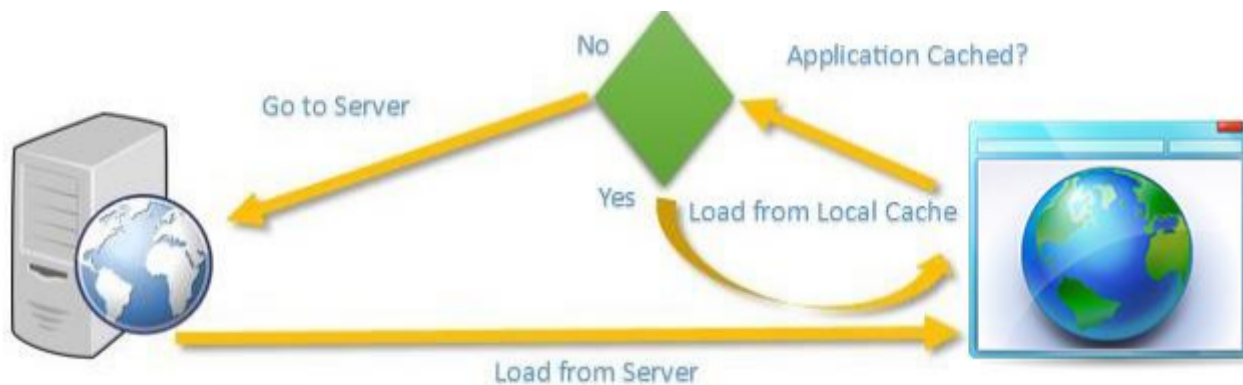
  // 在該DOM的物件中只取 header的部分
  let headrContent = doc.querySelector("#header").innerHTML;

  // 在該DOM的物件中只取 footer的部分
  let footContent = doc.querySelector("#footer").innerHTML

  document.querySelector("#header").innerHTML = headrContent;
  document.querySelector("#footer").innerHTML = footContent;
});

```

- fetch 取得的 data 初始為string格式
- 需透過 DOMParser.parseFromString() 將string 轉成 DOM element，才能使用選擇器
 - DOMParser.parseFromString(string,"轉換類型")
- 減少跟伺服器請求檔案的數量可以加速網頁載入速度。



- 由於瀏覽器為了加快網頁存取效率，會將檔案進行快取動作，但也有可能造成伺服器上檔案更新後，重新整理瀏覽器還是舊內容，所以必須要清除瀏覽器快取。
- 當開啟開發人員工具時可右鍵重新整理按鈕圖示，顯示重新整理選項的清單。

範例 1-14(選單事件- 透過更新 DOM 移除事件)

index.js

```

document.querySelector("#checkbox").addEventListener("change", function () {
    var register = document.querySelector("#register");
    if (this.checked) {
        register.addEventListener("click", registerEvent);
    } else {
        var new_element = register.cloneNode(true);
        register.parentNode.replaceChild(new_element, register);
    }
});

function registerEvent() {
    location.href = "../4-2/index.html";
};

```

index.html

```

<div class="checkbox">
    <input type="checkbox" id="checkbox">
    <label for="checkbox">我已閱讀並接受使用條款和隱私條例</label>
    <div id="register"></div>
</div>

```

- checkbox是否有被勾選
 - DOM 元件.checked
- 絕對路徑與相對路徑
 - 絕對路徑

- 絕對路徑指的是這個檔案在本機端或是網路上的絕對位置
- EX. "D:\近期使用檔案", "http: [//www.pchome.com.tw](http://www.pchome.com.tw)"
- 相對路徑
 - 相對路徑指的是相對於當前檔案的位置
- opacity
 - 不透明度
- 透過更新 DOM 可以移除綁定事件。
- 複製一個新的 DOM物件後，該複製的物件不會帶有事件，並且取代掉舊的物件