HTML5 API

林新德

shinder.lin@gmail.com

https://github.com/shinder/html5-api-examples

0. Web APIs

- 主題列表： https://developer.mozilla.org/en-US/docs/Web/API
- 參考網站： http://html5index.org/ 。

0.1 將介紹的主題

- 1. 新增的標籤與屬性
- 2. Media: Audio and Video
- 3. Canvas
- 4. SVG
- 5. Drag and Drop
- 6. File and FileReader
- 7. Web Storage
- 9. Server Sent Event
- 10. Web Sockets
- 11. AJAX File Upload
- 12. Web Workers
- 13. Chrome 啟動圖示
- 14. 離線使用
- 15. Web Notifications
- 16. Geolocation

0.2 工具網站

- 檢測瀏覽器支援 HTML5 的程度

  https://html5test.com/

- 查看某個功能在各瀏覽器的支援情況

  https://caniuse.com/

  https://zh.wikipedia.org/wiki/W3C推薦標準

  - 工作草案（WD, Working draft）
  - 候選推薦標準（CR, Candidate recommendation）
  - 提案推薦標準（PR, Proposed recommendation）
  - W3C推薦標準（REC, W3C recommendation）

0.3 環境準備

- 部份內容會使用到後端環境，故會使用 node/express 環境

```
// 1. 安裝 express-generator
$ sudo npm i -g express-generator
// 2. 使用 express-generator 建立專案，-e 使用 ejs
$ express -e html5-site
// 3. 安裝 packages
$ cd html5-site
$ npm i
// 4. 將 package.json 的啟動修改，使之使用 nodemon 啟動（之前必須全域安裝 nodemon）
// 5. 測試啟動伺服器 (http://localhost:3000)
$ npm start
// 6. Ctrl-C 停止伺服器
// 7. 安裝 serve-index
$ npm i serve-index
// 8. 修改 app.js
        const serveIndex = require('serve-index');
        // app.use('/', indexRouter); // 加入註解
        app.use('/', serveIndex('public', {'icons': true})); // 加入此行

// 9.再到 http://localhost:3000 可以看到 public 內的檔案列表
```

1. 新增的標籤與屬性

- Semantic Elements
- Form Elements
- Contenteditable

1.1 Semantic Elements

https://www.w3schools.com/html/html5_semantic_elements.asp

<article>
<aside>
<details>
<figcaption>
<figure>
<footer>
<header>
<main>
<mark>
<nav>
<section>
<summary>
<time>

1.2 表單的 input

https://www.w3schools.com/html/html_form_input_types.asp

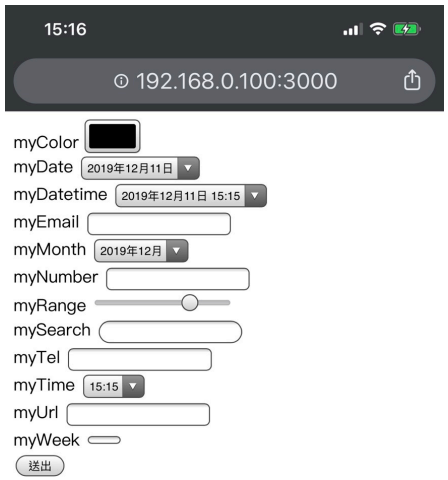
```
<input type="button">
<input type="checkbox">
<input type="file">
<input type="hidden">
<input type="image">
<input type="password">
<input type="radio">
<input type="reset">
<input type="submit">
<input type="text">
```

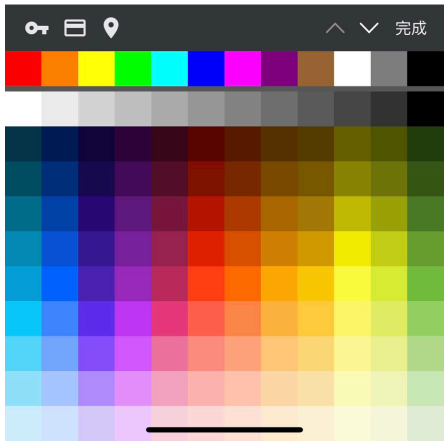
- input 新的 type 屬性值

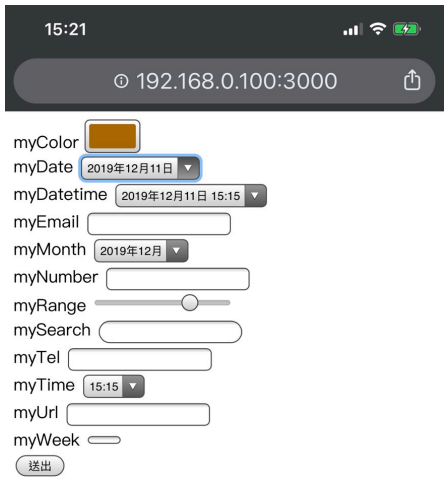
```
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="email">
<input type="month">
<input type="number">
<input type="range">
<input type="search">
<input type="tel">
<input type="time">
<input type="url">
<input type="week">
```

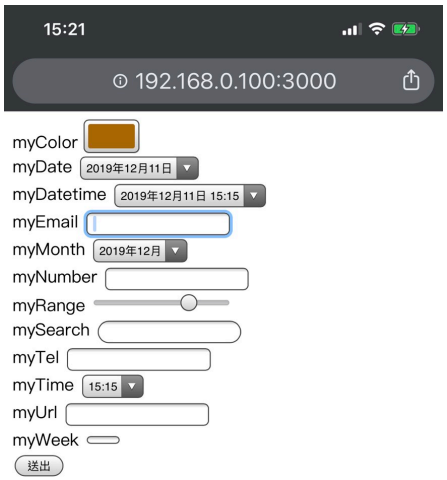
jQuery UI: datepicker

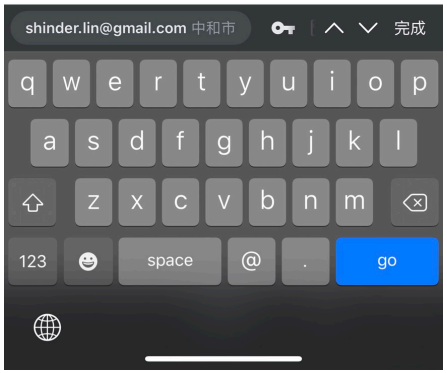
https://github.com/shinder/html5-api-examples/blob/master/public/form-input-type.html











- https://www.w3schools.com/html/html_form_attributes.asp

- input 新的屬性

```
autocomplete
autofocus
height and width
list (搭配 datalist)
min and max
multiple
pattern (regexp)
placeholder
required
step
```

- form 新的屬性

```
autocomplete
novalidate
```

https://github.com/shinder/html5-api-examples/blob/master/public/form-input-pattern.html

```
<form action="" onsubmit="return false;" novalidate autocomplete="on">
    <label for="myName">姓名：</label>
    <input type="text" name="myName" id="myName" required autofocus><br>

    <label for="myEmail">電郵：</label>
    <input type="email" name="myEmail" id="myEmail"><br>

    <label for="myGroup">組別：</label>
    <input type="text" name="myGroup" id="myGroup" list="mylist">
    <datalist id="mylist">
        <option value="甲組"></option>
        <option value="乙組"></option>
        <option value="丙組"></option>
    </datalist><br>

    <label for="myTel">手機號碼：</label>
    <input type="tel" name="myTel" id="myTel"
        pattern="^09\d{2}-?\d{3}-?\d{3}$"><br>
    <input type="submit">
</form>
```

- 自訂驗證訊息

```
const myPass = document.querySelector('#myPass');
const myPass2 = document.querySelector('#myPass2');

myPass2.addEventListener('blur', function(event){
    console.log(event);
    if(myPass.value !== myPass2.value) {
        // 只要有設定，送出前就會顯示
        myPass2.setCustomValidity('密碼確認欄必須和密碼欄相同內容');
    } else {
        myPass2.setCustomValidity(''); // 取消
    }
});
```

form-custom-validity.html

```
let inputs = document.querySelectorAll('input');
const myValidate = function(event){
    const t = event.target;
    const v = t.validity;
    let msg = '';
    switch(true){
        case v.valueMissing:
            msg = '必填欄位';
            break;
        case v.typeMismatch:
            msg = '請填寫正確的類型';
            break;
        case v.patternMismatch:
            msg = '請填寫規定的格式';
            break;
    }
    t.setCustomValidity(msg);
};
inputs.forEach(function(inp){
    inp.addEventListener('blur', myValidate);
});
```

form-custom-validity-2.html

1.3 Contenteditable

div-contenteditable.html

```
<div id="info" contenteditable="true">
        123
</div>
<script>
    info.addEventListener('input', function(event){
        console.log(info.innerText);
    });
</script>
```

2. Embed Media: Audio and Video

- 在頁面內嵌入多媒體內容。
- <audio> 標籤為 HTMLAudioElement 類型。
- <video> 標籤為 HTMLVideoElement 類型。
- HTMLAudioElement 和 HTMLVideoElement 皆繼承自 HTMLMediaElement 。
- 特有的事件定義於 HTMLMediaElement。
- 由於聲音可以是沒有外觀的，可以用 Audio 類別建立。

2.1 Audio

data開頭的是說明而已
相機拍照也會把照片資訊壓縮存在meta裡面

https://github.com/shinder/html5-api-examples/blob/master/public/audio-tag.html

```
<audio src="media/196381__minitauross__panflute-loop.mp3"
                    data-src="mp3, wav, ogg"
    preload="auto"  data-preload="auto, metadata, none"
    loop            data-loop="循環播放"
    autoplay        data-autoplay="自動播放，目前政策是無效果"
    controls        data-controls="控制器"
    >
    <a href="media/196381__minitauross__panflute-loop.mp3">下載音樂檔</a>
</audio>
```

```
<audio controls>
    <source src="horse.ogg" type="audio/ogg">
    <source src="horse.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
</audio>
```

- 使用 **Audio** 物件動態載入聲音檔

https://github.com/shinder/html5-api-examples/blob/master/public/audio-object.html

```
<button id="doLoad">載入</button>
<button id="doPlay" style="display: none;">播放或停止</button>
<script>
    const doLoad = document.querySelector('#doLoad');
    const doPlay = document.querySelector('#doPlay');
    const au = new Audio();
    const handler = (event)=>{
        console.log(` ${event.type}
            duration: ${au.duration}, currentTime: ${au.currentTime},
            volume: ${au.volume}, muted: ${au.muted}, paused: ${au.paused} `);
    };
    au.addEventListener('loadedmetadata', handler);
    au.addEventListener('timeupdate', handler);
    au.addEventListener('ended', handler);

    doLoad.onclick = (event)=>{
        au.src = 'media/196381__minitauross__panflute-loop.mp3';
        doLoad.style.display = 'none';
        doPlay.style.display = 'block';
    };
    doPlay.onclick = ()=> au.paused ? au.play() : au.pause();
</script>
```

2.2 Video

- Demo 的影片檔：https://www.webmfiles.org/demo-files/

https://www.w3schools.com/howto/howto_css_fullscreen_video.asp

https://github.com/shinder/html5-api-examples/blob/master/public/video-tag.html

```
<video src="media/big-buck-bunny_trailer.webm"
                          data-src="webm, mp4, avi, ogv"
    preload="metadata"    data-preload="auto, metadata, none"
    loop                  data-loop="循環播放"
    autoplay              data-autoplay="自動播放，目前政策是無聲才能播"
    controls              data-controls="控制器"
    poster="images/bunny-poster.png"
                          data-poster="未播放時替代圖片"
    width="320"
    height="240"
    muted
    >
    <p>沒有支援 video 標籤時顯示</p>
</video>
```

- 動態載入影片檔（一）

```
<video id="player" controls></video>
<div id="btnGroup">
    <button id="video1">video 1</button>
    <button id="video2">video 2</button>
    <button id="doPlay">play or pause</button>
</div>
<script>
    const videos = [
        'media/big-buck-bunny_trailer.webm',
        'media/elephants-dream.webm',
    ];
    const player = document.querySelector('#player');
    const btns = {
        video1: document.querySelector('#video1'),
        video2: document.querySelector('#video2'),
        doPlay: document.querySelector('#doPlay'),
    };
    // 下頁繼續...
```

- 動態載入影片檔（承上頁）

```
    const playerListener = event=>{
        console.log(` ${event.type}
            duration: ${player.duration}, currentTime: ${player.currentTime},
            volume: ${player.volume}, muted: ${player.muted}, paused: ${player.paused} `);
    };

    player.addEventListener('loadedmetadata', playerListener);
    player.addEventListener('timeupdate', playerListener);
    player.addEventListener('ended', playerListener);

    const btnsListener = (event)=>{
        switch(event.target) {
            case btns.video1:
                player.src = videos[0];
                break;
            case btns.video2:
                player.src = videos[1];
                break;
            case btns.doPlay:
                player.paused ? player.play() : player.pause();
                break;
        }
    };
    document.querySelector('#btnGroup').addEventListener('click', btnsListener);
</script>
```

- 字幕 WebVTT
- https://developer.mozilla.org/zh-TW/docs/Web/API/Web_Video_Text_Tracks_Format

```
<video src="media/big-buck-bunny_trailer.webm"
     preload="auto" controls
     poster="images/bunny-poster.png">
    <track kind="subtitles" srclang="正體中文" src="media/bunny-zh.vtt" default />
    <track kind="subtitles" srclang="English" src="media/bunny-en.vtt" />
</video>
```

https://github.com/shinder/html5-api-examples/blob/master/public/video-subtitles.html

```
WEBVTT

NOTE 這是VTT註解

1
00:00:05.000 --> 00:00:08.000
This is the first subtitle.

2
00:00:12.000 --> 00:00:20.000
This is the second.

3
00:00:20.000 --> 00:00:25.000
Third
```

3. Canvas

- 線與多邊形
- 矩形
- 圓、弧、扇形
- 圖片
- 文字
- 影片截取
- Canvas lib: pixi.js, three.js, frabic.js

3.1 畫線與多邊形

https://github.com/shinder/html5-api-examples/blob/master/public/canvas-line-1.html

```
<!-- 預設 300px * 150px -->
<canvas id="myCanvas" width="600" height="400"></canvas>
<script>
    const myCanvas = document.querySelector('#myCanvas');
    const cnt = myCanvas.getContext('2d');

    cnt.beginPath(); // 重置 path
    cnt.moveTo(50, 50);
    cnt.lineTo(200, 300);
    cnt.lineTo(400, 50);
    cnt.strokeStyle = 'orange'; // 設定畫筆顏色
    cnt.lineWidth = 20;// 設定畫筆粗細
    cnt.stroke(); // 在路徑上畫線
</script>
```

https://github.com/shinder/html5-api-examples/blob/master/public/canvas-line-2.html

```
<form name="form1">
    <input type="checkbox" name="isClosedPath"
id="isClosedPath">
    <label for="isClosedPath">封閉</label> ,
    <label for="lineCap">線段端點樣式</label>
    <select name="lineCap" id="lineCap">
        <option value="butt">butt</option>
        <option value="round">round</option>
        <option value="square">square</option>
    </select> ,
    <label for="lineJoin">端點銜接樣式</label>
    <select name="lineJoin" id="lineJoin">
        <option value="miter">miter</option>
        <option value="round">round</option>
        <option value="bevel">bevel</option>
    </select>
</form>
```

```
let isClosedPath = document.form1.isClosedPath.checked;
let lineCap = document.form1.lineCap.value;
let lineJoin = document.form1.lineJoin.value;

cnt.clearRect(0, 0, myCanvas.width, myCanvas.height); // 清除畫布
cnt.beginPath(); // 重置 path
cnt.moveTo(50, 50);
cnt.lineTo(200, 300);
cnt.lineTo(400, 50);
isClosedPath ? cnt.closePath() : false;
cnt.strokeStyle = 'orange';
cnt.lineWidth = 40;
cnt.lineCap = lineCap;
cnt.lineJoin = lineJoin;
cnt.stroke();
```

3.2 畫矩形

https://github.com/shinder/html5-api-examples/blob/master/public/canvas-rect.html

```
const myCanvas = document.querySelector('#myCanvas');
const cnt = myCanvas.getContext('2d');

cnt.strokeStyle = 'orange'; // 設定畫筆顏色
cnt.lineWidth = 20;// 設定畫筆粗細
cnt.lineJoin = 'round';
cnt.strokeRect(50, 50, 100, 100); // (x, y, width, height)

cnt.fillStyle = 'lightblue'; // 填滿的顏色
cnt.fillRect(50, 200, 100, 100);

// 陰影效果
cnt.shadowOffsetX = 10; // 位移
cnt.shadowOffsetY = 10;
cnt.shadowColor = 'rgba(0,0,0,.5)';
cnt.shadowBlur = 4; // 模糊效果
cnt.strokeRect(350, 50, 100, 100);
cnt.fillRect(350, 200, 100, 100);
```

3.3 畫圓、弧、扇形

https://github.com/shinder/html5-api-examples/blob/master/public/canvas-arc.html

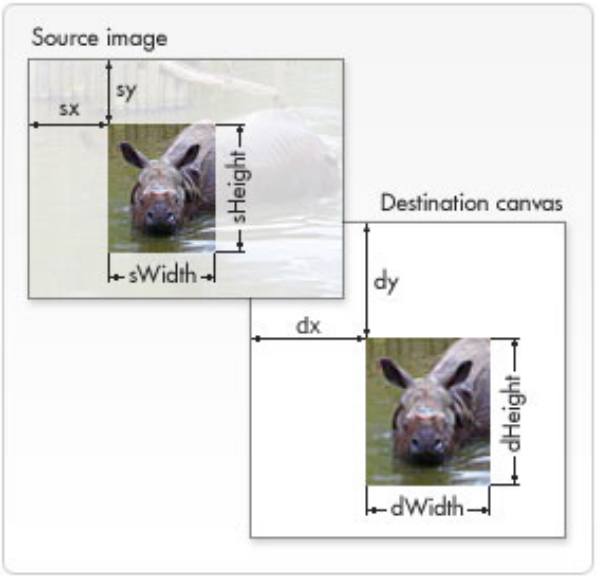
```
const myCanvas = document.querySelector('#myCanvas');
const ctx = myCanvas.getContext('2d');

ctx.strokeStyle = 'red'; // 設定畫筆顏色
ctx.lineWidth = 6; // 設定畫筆粗細
ctx.fillStyle = 'lightblue'; // 填滿的顏色

// arc(x, y, radius, startAngle, endAngle, anticlockwise);
// arc(圓心x, 圓心y, 半徑, 起始角度, 結束角度, 是否為逆時針方向);
ctx.arc(420, 60, 50, 0, Math.PI*2); // 圓形
ctx.stroke();
```

3.4 畫圖片

https://developer.mozilla.org/zh-TW/docs/Web/API/Canvas_API/Tutorial/Using_images




```
// 定點繪製
drawImage(image, x, y)

// 縮放
drawImage(image, x, y, width, height)

// 切割影像
drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)
```

- 等比例縮放：canvas-image-1.html

```
<script>
    const myCanvas = document.querySelector('#myCanvas');
    const ctx = myCanvas.getContext('2d');
    let w, h;
    const showImg = ()=>{
        ctx.clearRect(0, 0, myCanvas.width, myCanvas.height);
        switch(myWay.value){
            case '0':
                ctx.drawImage(img, 0, 0);
                break;
            case '1':
                h = img.height * myCanvas.width/img.width;
                ctx.drawImage(img, 0, 0, myCanvas.width, h);
                break;
            case '2':
                w = img.width * myCanvas.height/img.height;
                ctx.drawImage(img, 0, 0, w, myCanvas.height);
                break;
        }
    };
    const img = new Image();
    img.addEventListener('load', showImg);
    img.src = 'images/bunny-poster.png';
    myWay.addEventListener('change', showImg)
</script>
```

- 從客戶端圖檔繪入：canvas-image-2.html

```
<canvas id="myCanvas" width="600" height="400"></canvas>
<br>
<input type="file" id="myFile" style="display: none" accept="image/*">
<button onclick="myFile.click()">繪製圖檔</button>
```

```
const myCanvas =
document.querySelector('#myCanvas');
const ctx = myCanvas.getContext('2d');

const img = new Image();
const reader = new FileReader();
img.addEventListener('load', event=>{
    ctx.drawImage(img, 0, 0);
});
reader.addEventListener('load', ()=>{
    img.src = reader.result;
});

myFile.addEventListener('change', ()=>{
    reader.readAsDataURL(myFile.files[0]);
});
```

3.5 畫文字

canvas-text.html

```
const myCanvas = document.querySelector('#myCanvas');
const ctx = myCanvas.getContext('2d');
ctx.fillStyle = 'red';
ctx.strokeStyle = 'blue';
ctx.lineWidth = 2;

ctx.textAlign = 'center';
ctx.textBaseline = 'middle';

ctx.font = 'bold 72px serif';
ctx.fillText('Hello Canvas', 300, 100);
ctx.font = 'bold 72px arial';
ctx.strokeText('Hello Canvas', 300, 200);
ctx.fillText('哈囉', 300, 300);
```

3.6 截取影片

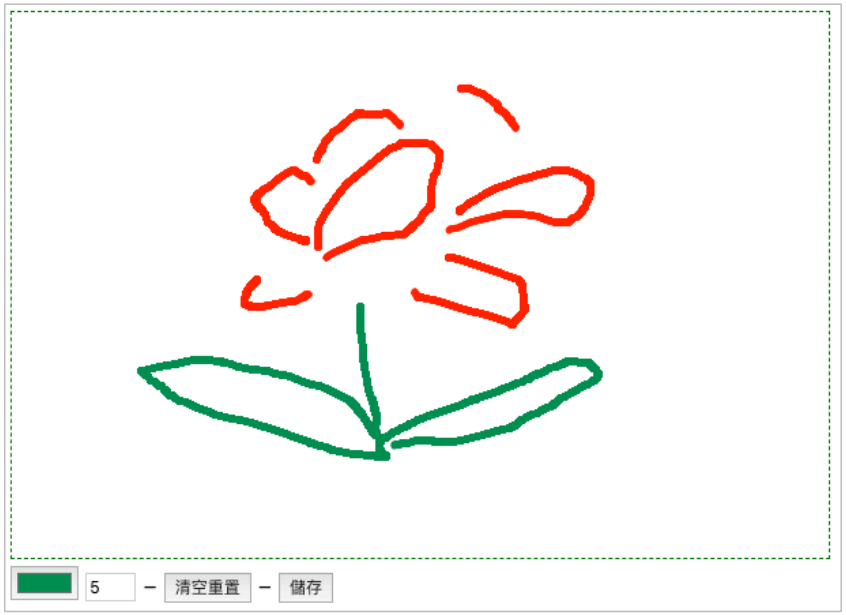
canvas-video.html

```
<video src="media/big-buck-bunny_trailer.webm" preload="auto" controls></video>
<br>
<button onclick="takePic()">截取</button>
<br>
<canvas id="myCanvas"></canvas>
<script>
    const myCanvas = document.querySelector('#myCanvas');
    const ctx = myCanvas.getContext('2d');
    const video = document.querySelector('video');

    const takePic = ()=>{
        myCanvas.setAttribute('width', video.videoWidth);
        myCanvas.setAttribute('height', video.videoHeight);
        ctx.drawImage(video, 0, 0);
    };
</script>
```

3.7 塗鴉練習

https://github.com/shinder/html5-api-examples/blob/master/public/canvas-draw-app.html



4. SVG

- 網頁向量圖標準
- XML
- 有自己的標籤定義
- 可使用 style
- 可放入 HTML 檔內，可使用 DOM API 操作
- 方便使用繪畫軟體製作，例如：Adobe illustrator, Inkscape
- https://developer.mozilla.org/en-US/docs/Web/SVG/Element
- 有名的 lib：D3.js, snap.svg

4.1 常見的SVG標籤

- <svg>：頂層標籤，設定顯示區大小。
- <line> ：線
- <rect> ：矩形
- <circle> ：圓形
- <ellipse> ：橢圓
- <polygon> ：多邊形
- <path> ：路徑
- <g> ：群組

- svg-demo.html

```
<style>
    svg {
        width: 200px;
        height: 100px;
    }
    .c1 { fill: red; }
    .c1:hover { fill: blue; }
    .c2 { fill: orange; }
    .c2:hover { fill: purple; }
</style>
```

```
<svg viewBox="0 0 200 100" xmlns="http://www.w3.org/2000/svg">
    <polygon class="c1" points="0,100 50,25 50,75 100,0" />
    <polygon points="100,100 150,25 150,75 200,0"
             fill="yellow" stroke="black" />
</svg>
<script>
    const poly = document.querySelector('.c1');
    poly.onclick = ()=>{
        poly.getAttribute('class')==='c1' ?
            poly.setAttribute('class','c2') : poly.setAttribute('class','c1');
    }
</script>
```

5. Drag and Drop （拖拉功能）

- dragstart: 開始拖拉
- drag: 正在拖拉（類似 mousemove）
- dragend: 結束拖拉

- dragenter: 有拖拉物進入時
- dragover: 有拖拉物在範圍內移動時
- dragleave: 有拖拉物離開時
- drop: 拖拉物在範圍內放開時

- dragover 的處理器內要呼叫 preventDefault()，才能觸發 drop 事件。
- <a> 和 <img> 原本就有預設拖拉的功能。

拖拉測試：drag-drop-1.html

```
const ball = $('.ball');
const rect2 = $('.rect2');
const listener = function(event){
    console.log(event.type, event.target);
};
// ball.on('drag', listener);  // 正在拖拉（類似 mousemove）
ball.on('dragstart', listener);  // 開始拖拉
ball.on('dragend', listener);  // 結束拖拉

rect2.on('dragenter', listener);  // 拖入
rect2.on('dragleave', listener);  // 拖出

// 正在上面拖拉
rect2.on('dragover', function(event){
    // 一定要在 dragover 呼叫 preventDefault(), drop 事件才會發生
    event.preventDefault();
});
rect2.on('drop', listener);
```

```
<div id="rect">
    <div class="ball" draggable="true"></div>
    <div class="rect2"></div>
</div>
```

拖拉測試二：drag-drop-2.html

```html
<div class="container">
    <div class="rect" style="left: 100px">
        <div class="ball" id="ball1" draggable="true">1</div>
        <div class="ball" id="ball2" draggable="true"
             style="background-color: #ff3228;">2</div>
        <div class="ball" id="ball3" draggable="true"
             style="background-color: #26c213;">3</div>
    </div>
    <div class="rect"></div>
</div>
```

```javascript
const ball = $('.ball');
const rect = $('.rect');

ball.on('dragstart', function(event){
    // 設定轉移的資料
    event.originalEvent.dataTransfer.setData('text', event.target.id);
});
rect.on('dragover', function(event){
    event.preventDefault();
});
rect.on('drop', function(event){
    // 取得轉移的資料
    let id = event.originalEvent.dataTransfer.getData('text');
    const b = $('#' + id);
    $(this).append(b);
});
```

拖拉測試三：drag-drop-3.html

```
const ball = $('.ball');
const rect2 = $('.rect:eq(1)');

ball.on('dragstart', function(event){
    // 設定轉移的資料
    event.originalEvent.dataTransfer.setData('text', event.target.id);
});
rect2.on('dragover', function(event){
    event.preventDefault();
});
rect2.on('drop', function(event){
    // 取得轉移的資料
    let id = event.originalEvent.dataTransfer.getData('text');
    if(! id) return;
    const b = $('#' + id).clone(); // 複製 jQuery 元素物件
    b.removeAttr('id');
    rect2.append(b);
});
```

6. File and FileReader

- 由於安全性考量，不能直接存取檔案。
- 透過使用者的操作（允許）可有條件的讀取檔案。
- `<input type="file">` 可讓使用者決定要讀取或上傳的檔案。
- File 類型的物件用來表示檔案的實體，但無法使用 JS 取得確切的磁碟路徑。
- FileReader 類型的物件用以讀取檔案的內容。

6.1 FileList 類型

- event. dataTransfer.files: 從檔案管理員拖拉進來
- https://github.com/shinder/html5-api-examples/blob/master/public/file-drag-drop.html
- fileField.files: 表單內檔案欄位
- https://github.com/shinder/html5-api-examples/blob/master/public/file-field.html

```
<input type="file" id="fileFiled" style="display: none;"
     multiple accept="image/*">
<div class="container">點擊選取檔案</div>
<script src="javascripts/jquery-3.4.1.js"></script>
<script>
    const container = $('.container');
    const fileFiled = $('#fileFiled');
    container.on('click', function(event){
        fileFiled.click();
    });
    fileFiled.on('change', function(event){
        // fileFiled[0] 才是 element, fileFiled 是 jQuery 物件
        for(let i=0; i<fileFiled[0].files.length; i++){
            container.append('<br>' + fileFiled[0].files[i].name);
        }
    });
</script>
```

6.2 FileReader 類型

- 顯示文字檔內容
- `reader.readAsText(file)`
- https://github.com/shinder/html5-api-examples/blob/master/public/fileReader-text.html

- 讀取圖檔並顯示
- `reader.readAsDataURL(file)`
- https://github.com/shinder/html5-api-examples/blob/master/public/fileReader-images.html

```html
<input type="file" id="fileFiled" style="display: none;" multiple accept="image/*">
<button>add image</button>
<div class="container"></div>
```

```javascript
const container = $('.container');
const fileFiled = $('#fileFiled');
$('button').click(function(event){
    fileFiled.click();
});
const readerLoaded = event=>{
    let image = new Image();
    image.height = 150;
    image.src = event.target.result; // 將結果設定給 img
    container.append(image)
};
fileFiled.on('change', function(event){
    let file, reader;
    for(let i=0; i<fileFiled[0].files.length; i++){
        file = fileFiled[0].files[i];
        reader = new FileReader();
        reader.addEventListener('load', readerLoaded);
        reader.readAsDataURL(file);
    }
});
```

7. Web Storage

- 前端常用的儲存資料技術：Cookie, Web Storage, Indexed DB。
- 採「同源政策」（same-origin policy）。
- Web Storage 包含兩個物件：localStorage 和 sessionStorage。
- 儲存格式：key-value pair 文字格式。
- 兩個物件的方法皆相同。
- 容量限制約 5M ~ 10M，依瀏覽器實作而定。
- localStorage 適用於整個網站。
- sessionStorage 只適用於單一個瀏覽器視窗或分頁。

7.1 Web Storage 方法

- **length** 取得資料筆數
- **key(n)** 取得索引值為 n 的 key
- **setItem(key, value)** 設定項目
- **getItem(key)** 讀取項目
- **removeItem(key)** 移除項目
- **clear()** 刪除所有項目

- 儲存資料
- https://github.com/shinder/html5-api-examples/blob/master/public/localStorage-setItem.html

```html
<form name="form1">
    <label for="name">姓名</label>
    <input type="text" name="name" id="name" required><br>
    <label for="age">年齡</label>
    <input type="number" name="age" id="age" required><br>
    <label for="mobile">手機</label>
    <input type="text" name="mobile" id="mobile" required
        pattern="^09\d{2}-?\d{3}-?\d{3}$" placeholder="09XX-XXX-XXX"><br>
    <input type="submit" value="儲存">
</form>
```

```javascript
const prefix = 'friend-book;;';
document.form1.onsubmit = ()=>{
    const t = new Date().getTime();
    let data = {
        name: document.form1.name.value,
        age: document.form1.age.value,
        mobile: document.form1.mobile.value,
        time: t,
    };
    localStorage.setItem(prefix+t, JSON.stringify(data));
    document.form1.reset();
    alert('儲存完成！');
    return false;
};
```

- 讀取資料
- https://github.com/shinder/html5-api-examples/blob/master/public/localStorage-getItem.html

```
<table>
    <thead>
    <tr>
        <td>姓名</td>
        <td>年齡</td>
        <td>手機</td>
    </tr>
    </thead>
    <tbody id="the-list"></tbody>
</table>
```

```
const prefix = 'friend-book;;';
const tbody = $('#the-list');
const showList = ()=>{
    tbody.html('');
    for(let i=0; i<localStorage.length; i++) {
        let key = localStorage.key(i);
        if(key.indexOf(prefix)===0) {
            let d = localStorage.getItem(key);
            d = JSON.parse(d);
            tbody.append(`<tr><td>${d.name}</td><td>${d.age}</td><td>${d.mobile}</td></tr>`);
        }
    }
};
showList();
```

- 讀取列表加刪除功能
- https://github.com/shinder/html5-api-examples/blob/master/public/localStorage-list.html

```
const showList = ()=>{
    tbody.html('');
    for(let i=0; i<localStorage.length; i++) {
        let key = localStorage.key(i);
        if(key.indexOf(prefix)===0) {
            let d = localStorage.getItem(key);
            d = JSON.parse(d);
            d.key = key;
            tbody.append(`<tr><td><a href="javascript:removeItem('${key}')">X</a></td>
                <td>${d.name}</td>
                <td>${d.age}</td><td>${d.mobile}</td></tr>`);
        }
    }
};
showList();

const removeItem = (key)=>{
    if(!confirm('確定要移除項目？')) return;
    localStorage.removeItem(key);
    showList();
};
```

問題：列表有按照新增順序嗎？ 如何解決？

- localStorage 儲存圖片
- https://github.com/shinder/html5-api-examples/blob/master/public/localStorage-draw-app.html

```
// 儲存項目
const saveCanvas = ()=>{
    const imgTxt = myCanvas.toDataURL("image/png").replace("image/png", "image/octet-stream");
    let key = 'shinder-draw;;' + new Date().getTime();
    localStorage.setItem(key, imgTxt);
    alert('儲存完成');
    renewList();
};

// 載入項目
const loadCanvas = (key)=>{
    if(!key) return;
    const img = new Image();
    img.onload = ()=>{
        ctx.drawImage(img, 0, 0);
    };
    img.src = localStorage.getItem(key);
}
```

9. Server Sent Event

- 用戶端瀏覽器停留在該頁面（保持連線）的情況下，伺服端可以間隔一段時間，將 UTF-8 資料推用到用戶端。
- 資料只能單向由伺服端到用戶端。

在連線情況下可以持續的傳送資料，但只能單向傳送。

- 伺服端送出的檔頭：

```
Content-Type: text/event-stream
Cache-Control: no-cache
Connection: keep-alive
```

- 資料格式：

```
: 可以放註解\n
event: myevent\n
id: 20\n
retry: 5000\n
data: 傳送的資料\n\n
```

```
: 註解
event: 自訂的事件名稱，預設為 message
id: 訊息id（可以是字串）
retry: 重新連線毫秒數，預設為 5 秒
data: 傳送的資料
```

- SSE 測試 - 客戶端
- https://github.com/shinder/html5-api-examples/blob/master/public/server-sent-event.html

```
<div id="info">123</div>
<script>
    const info = document.querySelector('#info');
    const es = new EventSource('/try-sse');

    es.addEventListener('open', function(event) {
        info.style.backgroundColor = 'orange';
        console.log(new Date().toLocaleString(), event);
    });
    es.addEventListener('message', function(event) {
        // console.log('message: ', event);
        info.innerHTML = event.lastEventId + ': ' + event.data;
    });
    es.addEventListener('error', function(event) {
        console.log(new Date().toLocaleString(), event);
    });
</script>
```

- SSE 測試 – 伺服端
- https://github.com/shinder/html5-api-examples/blob/master/app.js

```
app.get('/try-sse', (req, res) => {
    let id = 30;
    res.writeHead(200, {
        'Content-Type': 'text/event-stream',
        'Cache-Control': 'no-cache',
        'Connection': 'keep-alive',
    });
    setInterval(function () {
        let now = new Date();
        res.write('id: ' + id++ + '\n');
        res.write('data: ' + now.toLocaleString() + '\n\n');
    }, 2000);
});
```

10. WebSocket

- 作 client-server 雙向的資料交換（溝通）。
- 有「狀態」的連線方式（HTTP為無狀態的連線方式）。
- 傳輸的資料以文字為主。
- 網路七層架構（OSI 模型https://zh.wikipedia.org/wiki/OSI模型）中，WebSocket 屬於第7層（HTTP也是），Socket 則屬於第4層。
- 可以用 Socket 去實作上層的應用，但要處理的事務相當煩瑣。為了讓通訊更為簡便，所以有 WebSocket 的誕生。
- 常用來實作「聊天室」、「即時客服」等應用。
- 需要用 WebSocket 伺服器。
- ws 使用 port 80，wss 使用 port 443。

http port: 80, https port: 443

理論有7層，實際只有5層(理論的5~7是同一層)

http的協定,需要client發出req,server回應res,但以前常常發一個req,res就斷線
session就是來解決這種無狀態問題(不知道現在是誰)

WebSocket不需要遵守同源政策。

10.1 WebSocket 客戶端 API

- 使用 WebSocket 類別建立連線物件。
- open 事件：建立連線時觸發。
- message 事件：資料送達時觸發，event.data 可取得資料。
- close 事件：連線關閉時觸發。
- error 事件：發生錯誤時觸發。
- bufferedAmount 屬性：資料使用緩衝區的大小。使用 send() 方法時，資料會先放在緩衝區排隊送出。
- readyState 屬性：連線狀態（唯讀）。

常數	值	描述
CONNECTING	0	連線尚未打開。
OPEN	1	連線已打開，可以進行通訊。
CLOSING	2	連線正在進行關閉程序。
CLOSED	3	連線已關閉／連線不能打開。

- 使用現成的後端功能測試 WebSocket。
- http://websocket.org/echo.html
- https://github.com/shinder/html5-api-examples/blob/master/public/web-socket-echo.html

```
let socket;
const myOpen = event=>{
    info.innerHTML += `--- 已連線 ---<br>`;
    console.log(socket.binaryType);
};
const myMessage = event=>{
    info.innerHTML += `接收到的訊息: ${event.data}<br>`;
};
const myClose = event=>{
    info.innerHTML += `=== 已斷線 ===<br>`;
};
document.addEventListener('click', event=>{
    switch(event.target.id){
        case 'connectBtn':
            if(socket && socket.readyState===WebSocket.OPEN)
                return;
            socket = new WebSocket('ws://echo.websocket.org');
            socket.addEventListener('open', myOpen);
            socket.addEventListener('message', myMessage);
            socket.addEventListener('close', myClose);
            break;
        case 'disconnectBtn':
            if(socket && socket.close) socket.close();
            break;
        case 'sendBtn':
            if(socket && socket.send) socket.send(myInput.value);
            break;
    }
});
```

```
<button id="connectBtn">連線</button>
<button id="disconnectBtn">斷線</button>
<br>
<input type="text" id="myInput">
<button id="sendBtn">送出</button>
<div id="info"></div>
```

10.2 ws: a Node.js WebSocket library

- 官網：https://www.npmjs.com/package/ws。
- 安裝：**`npm i ws`**
- 可以是獨立的 server 也可以和 Node 的 http server 使用相同的 domain 和 port。
- 說明文件：https://github.com/websockets/ws/blob/master/doc/ws.md

- 以 ws 建立 echo server
- 在 /bin/www 內，在 server 物件建立之後，加入：

```
require('../routes/ws-echo')(server);
```

- /routes/ws-echo.js 的內容：

```
const WebSocket = require('ws');
const createEchoServer = server=>{
    const wsServer = new WebSocket.Server({server});

    wsServer.on('connection', (ws, req)=>{
        console.log('連線數:', wsServer.clients.size);
        console.log('ip: ' + req.connection.remoteAddress);
        console.log('port: ' + req.connection.remotePort);
        ws.on('message', message=>{
            ws.send(message);
        });
        ws.send('連線了!');
    });
};
module.exports = createEchoServer;
```

ws: websocket的物件和client端對應

- 以 ws 建立簡易聊天室「伺服端」
- 在 /bin/www 內，在 server 物件建立之後，加入：

```
// require('../routes/ws-echo')(server);
require('../routes/ws-chat')(server);
```

- /routes/ws-chat.js 的內容：

```
const WebSocket = require('ws');
const createChatServer = server=>{
    const wsServer = new WebSocket.Server({server});
    const map = new Map(); // 存放對應的名稱
    wsServer.on('connection', (ws, req)=>{
        map.set(ws, {name: ''}); // 設定對應的物件
        ws.on('message', message=>{
            const mObj = map.get(ws); // 取得對應的物件
            let msg;
            if(! mObj.name){
                mObj.name = message;
                msg = `${mObj.name} 進入，人數：${wsServer.clients.size}`;
            } else {
                msg = `${mObj.name}: ${message}`;
            }
            wsServer.clients.forEach(c=>{
                if(c.readyState===WebSocket.OPEN){
                    c.send(msg);
                }
            });
        });
    });
};
module.exports = createChatServer;
```

這裡Map是類型
不是方法

- 簡易聊天室「客戶端」

```
<input type="text" id="user" placeholder="請輸入名字">
<button id="connectBtn">連線</button><br>
<textarea id="msgArea" cols="50" rows="30" readonly></textarea><br>
<input type="text" id="msg"><button id="sendBtn">送出</button>
```

```
let socket;
const myOpen = event=>{
    socket.send(user.value);
};
const myMessage = event=>{
    msgArea.value += `${event.data}\n`;
};
document.addEventListener('click', event=>{
    switch(event.target.id){
        case 'connectBtn':
            if(! user.value){
                alert('請輸入名字');
                return;
            }
            if(socket && socket.readyState===WebSocket.OPEN)
                return;
            socket = new WebSocket(`ws://${location.host}`);
            socket.addEventListener('open', myOpen);
            socket.addEventListener('message', myMessage);
            break;
        case 'sendBtn':
            if(socket && socket.send) socket.send(msg.value);
            break;
    }
});
```

11. Ajax files uploading

- 建立 /tmp-uploads 資料夾，暫存上傳的檔案。
- 建立 /public/img-uploads 資料夾，存放上傳的圖檔。
- 安裝 multer 套件
- 安裝 uuid 套件
- 單檔前端：https://github.com/shinder/html5-api-examples/blob/master/public/ajax-upload-single.html
- 多檔前端：https://github.com/shinder/html5-api-examples/blob/master/public/ajax-upload-multi.html
- 後端： https://github.com/shinder/html5-api-examples/blob/master/routes/ajax-upload.js

上傳頭貼會用到

- 上傳單一檔案的表單（前端 1）

```
<form action="" name="form1" onsubmit="return doAjax();" enctype="multipart/form-data">
    <label for="user">姓名</label>
    <input type="text" name="user" id="user"><br>
    <label for="avatar">大頭貼</label>
    <input type="file" name="avatar" id="avatar" accept="image/*"><br>
    <img src="" alt="" id="avatar-img" width="300px"><br>
    <label for="description">簡介</label><br>
    <textarea name="description" id="description" cols="50"></textarea><br>
    <button type="submit">上傳</button>
</form>
```

```
// 取得原有的內容
fetch('/json/profile.json')
    .then(response=>response.json())
    .then(obj=>{
        user.value = obj.user || '';
        document.querySelector('#avatar-img').src = obj.avatar || '';
        description.value = obj.description || '';
    })
    .catch(ex=>console.log('ex:', ex));
```

- 上傳單一檔案的表單（前端 2）

```
// 選擇檔案時預覽
avatar.addEventListener('change', event=>{
    const reader = new FileReader();
    reader.addEventListener('load', event=>{
        document.querySelector('#avatar-img').src = reader.result;
    });
    if(avatar.files[0]){
        reader.readAsDataURL(avatar.files[0]); // 讀取並轉換為 base64 格式
    }
});
```

```
const doAjax = ()=>{
    const fd = new FormData(document.form1);
    fetch('/uploads/profile', {
        method: 'POST',
        body: fd,
    })
        .then(response=>response.json())
        .then(obj=>console.log(obj));
    return false;
};
```

- 上傳單一檔案的表單（後端 1）

```
const express = require('express');
const multer = require('multer');
const uuidv4 = require('uuid/v4');
const fs = require('fs');
const extMap = { // 檔案類型的副檔名的對應
    'image/jpeg': '.jpg',
    'image/png': '.png',
    'image/gif': '.gif',
};
```

```
// multer 儲存的設定
const storage = multer.diskStorage({
    destination: (req, file, cb)=>{
        cb(null, __dirname + '/../public/img-uploads');
    },
    filename: (req, file, cb)=>{
        let ext = extMap[file.mimetype]; // 副檔名
        if(ext)
            cb(null, uuidv4() + ext);
        else
            cb(new Error('上傳的檔案格式錯誤'));
    }
});
const fileFilter = (req, file, cb)=>{
    if(extMap[file.mimetype])
        cb(null, true); // 接受檔案
    else
        cb(null, false); // 不接受
};
const upload = multer({
    storage: storage,
    fileFilter: fileFilter,
});
```

- 上傳單一檔案的表單（後端 2）

```
// 承上頁 ...
const router = express.Router();
router.post('/profile', upload.single('avatar'), function(req, res) {
    let data; // 要存檔的資料
    const output = { body: req.body }; // 輸出給前端，先預放傳入的 POST 資料
    try {
        data = require(__dirname + '/../public/json/profile'); // 取資料
    } catch (ex) {
        data = {}; // 如果找不到檔案，使用預設值
    }
    data.user = req.body.user; // 變更資料
    data.description = req.body.description;
    if(req.file && req.file.originalname){ // 若有上傳檔案
        data.avatar = '/img-uploads/' + req.file.filename; // 儲存包含路徑
        output.upload = data.avatar;
    }
    fs.writeFile(__dirname+'/../public/json/profile.json', JSON.stringify(data), error=>{
        if(error) return console.log(error);
    });
    res.json(output);
});
module.exports = router;
```

- 上傳多個檔案

```
<form action="" name="form1" onsubmit="return false;">
    <label for="myFiles">選了就上傳</label>
    <input type="file" name="myFiles" id="myFiles" accept="image/*" multiple><br>
    <div id="img-container"></div>
    <textarea name="description" id="description" cols="50" readonly></textarea>
</form>
```

```
myFiles.addEventListener('change', event=>{
    const fd = new FormData(document.form1);
    fetch('/uploads/multiple', {
        method: 'POST',
        body: fd,
    })
        .then(response=>response.json())
        .then(ar=>{
            description.value = JSON.stringify(ar);
            imgCont.innerHTML = '';
            ar.forEach(el=>{
                imgCont.innerHTML += `<img src="${el}" alt="" width="300px">`;
            });
        });
});
```

```
router.post('/multiple', upload.array('myFiles'), function(req, res) {
    const output = [];
    req.files.forEach(file=>{
        output.push('/img-uploads/' + file.filename);
    });
    res.json(output);
});
```

12. Web Worker

- 兩種型態：**Dedicated** Worker（針對某個頁面）和 **Shared** Worker（可在同源不同頁面分享運算的結果）。
- Worker 是以 JS 檔的方式存在，類型為 **`Worker`** 或 **`SharedWorker`** 。
- Workers 雙方透過 message 事件接收資料，透過 postMessage() 方法傳送資料。
- 在 Dedicated Worker 的 JS 檔內，可定義onmessage() 方法接收資料；直接呼叫 postMessage() 傳送資料。
- 在主程式，可定義 dedicatedWorker.onmessage() 方法接收資料；呼叫 dedicatedWorker.postMessage() 傳送資料。
- Shared Worker 的物件必須以其 port 子物件去啟動（start），及做資料的傳送及接收。
- 限制：不能操作 DOM。
- 參考資料：https://developer.mozilla.org/zh-TW/docs/Web/API/Web_Workers_API/Using_web_workers

12.1 Dedicated Worker

- 沒有使用 Worker：dedicated-worker-main-1-no.html
- 使用 Worker：dedicated-worker-main-1.html

```
<input type="text" placeholder="測試輸入欄"><br>
<button onclick="start(event)">開始算質數</button>
<div id="info"></div>
<script>
    const worker = new Worker('javascripts/dedicated-worker-prime-numbers.js');
    worker.onmessage = event=>{
        let {primes, length, howLong} = event.data;
        info.innerHTML = `${length} 個, ${howLong} 毫秒`;
    };
    const start = event=>{
        event.target.style.display = 'none';
        worker.postMessage('start');
    };
</script>
```

- javascripts/dedicated-worker-prime-numbers.js

```
const getPrimeNumbers = (num=2e7)=>{
    const startTime = new Date().getTime();
    const pn = [2]; // 存放質數的陣列

     // 略...

    return {
        primes: pn,
        length: pn.length,
        howLong: new Date().getTime() - startTime
    }
};

// this, self: DedicatedWorkerGlobalScope
self.onmessage = event=>{
    if(event.data==='start'){
        postMessage(getPrimeNumbers());
    }
};
```

12.2 Shared Worker

- 只要有一個頁面使用Shared Worker，那個 Worker 將會一直存在。
- Shared Worker 頁面： shared-worker-main-1.html 、 shared-worker-main-2.html

```
<input type="text" placeholder="測試輸入欄"><br>
<div id="info"></div>
<script>
    const worker = new SharedWorker('javascripts/shared-worker-prime-numbers.js');

    worker.port.onmessage = event=>{
        const data = event.data;
        const pn = data.primeNumbers[data.primeNumbers.length-1];
        info.innerHTML = `${data.action} - 質數：${pn}, ${data.runningTime} msec`;
    };

    worker.port.postMessage('Hello');
</script>
```

- 使用 Chrome 測試 Shared Worker 時，請使用：chrome://inspect/#workers
- javascripts/shared-worker-prime-numbers.js

```
const pn = [2]; // 存放質數的陣列
let runningTime;
const getPrimeNumbers = (num=2e7)=>{
    const startTime = new Date().getTime();
    // 略...
    runningTime = new Date().getTime() - startTime;
};
getPrimeNumbers();

self.onconnect = event=>{
    const port = event.ports[0]; // MessagePort
    port.onmessage = event=>{
        port.postMessage({
            primeNumbers: pn,
            runningTime: runningTime,
            action: event.data,
        });
    };
};
```

13. 啟動圖示

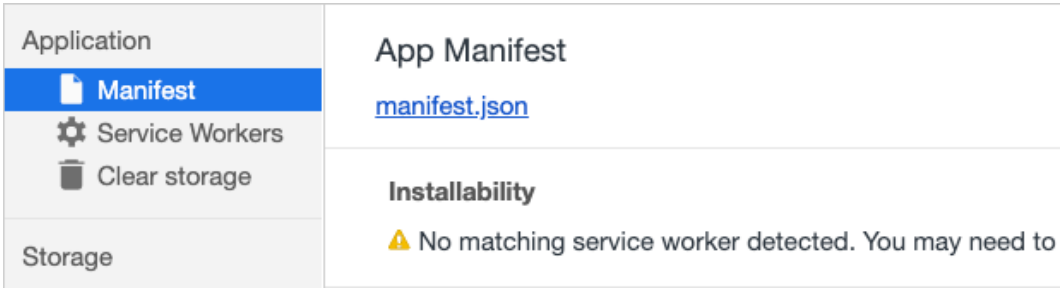
- Native app vs Web app
- Native app ：效能較好、啟動圖示、存取裝置、離線使用。
- Web app 如何追趕上 Native app？
- PWA: Progressive Web App https://ithelp.ithome.com.tw/articles/10186584
- Web app manifest 為 JSON 文件，功能是將 Web app 安裝到設備的主畫面，其中包含名稱、圖示和描述等相關資訊。 https://developer.mozilla.org/zh-TW/docs/Web/Manifest
- 由於 Chrome 和 Chrome for Android 的支援程度較好，故將以 Chrome 為主要環境。
- Manifest 產生器： https://app-manifest.firebaseapp.com
- Manifest 驗證器： https://manifest-validator.appspot.com
-

13.1 Manifest 內容 MDN Manifest

```
{
    "name": "Web app 名稱",
    "short_name": "短名，啟動圖示上顯示的名稱",
    "start_url": "點擊圖示時開啟的網頁",
    "display": "常用 standalone",
    "background_color": "#fff",
    "description": "Web app 描述",
    "icons": [
        {
            "src": "images/touch/homescreen192.png",
            "sizes": "192x192",
            "type": "image/png"
        }
    ],
    "related_applications": [
        {
            "platform": "play",
            "url": "https://play.google.com/store/apps/details?id=000"
        }
    ]
}
```

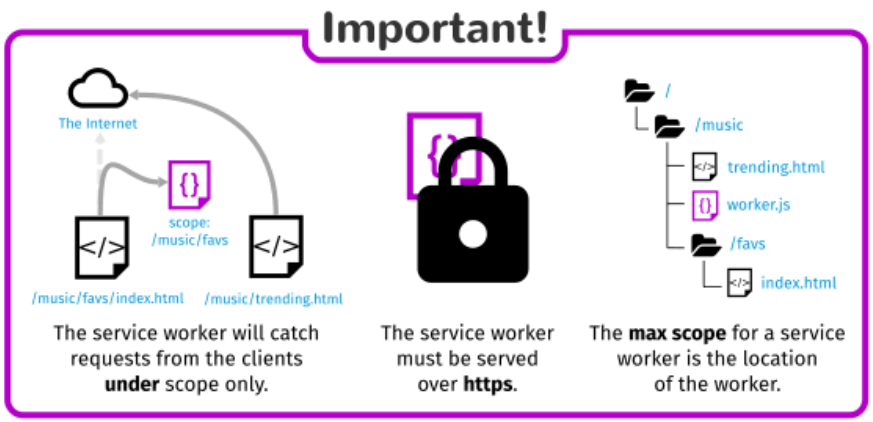
13.2 建立 App: Manifest

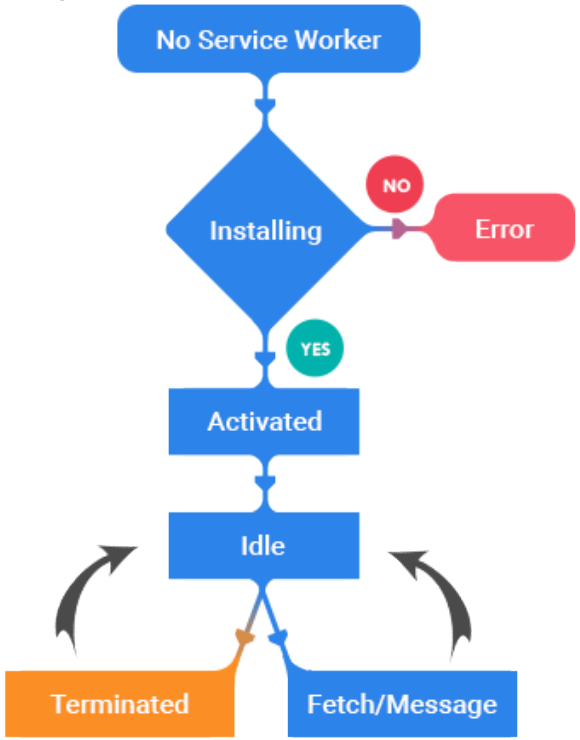
- 在 /public 資料夾內新增 myApp 資料夾。
- 將 canvas-draw-app.html 複製到資料夾內。
- 使用 https://app-manifest.firebaseapp.com/ 建立所需檔案。
- 修改 .html 加入：
  - `<link rel="manifest" href="./manifest.json">`
- 測試頁面，並使用開發人員工作，查看 Application 分頁



13.3 建立 App: Service Worker

- 使用 Service Worker: Using_Service_Workers





- https://www.cronj.com/blog/browser-push-notifications-using-javascript/

- 註冊 Service Worker

```
navigator.serviceWorker
        .register('service-worker.js')
        .then(reg=>{
                console.log('註冊成功：' + reg);
        })
        .catch(err=>{
                console.log('註冊失敗：' + err);
        })
```

- Service Worker 基本事件處理器

```
self.addEventListener('install', function(e) {});
self.addEventListener('activate', function(e) {});
self.addEventListener('fetch', function(e) {});
```

- 可以安裝但還不能離線使用

14. 離線使用

https://github.com/shinder/html5-api-examples/blob/master/public/myApp/service-worker.js

```
let cacheName = 'v1';
self.addEventListener('install', function(event) {
    event.waitUntil(
        caches.open(cacheName).then((cache) => {
            return cache.addAll([
                '/myApp/canvas-draw-app.html',
                '/myApp/manifest.json',
                '/myApp/images/icons/icon-144x144.png',
            ]);
        })
    );
});
```

安裝時將檔案放入快取區，才能離線使用

```
self.addEventListener('fetch', function(event) {
    console.log(event); // 查看 chrome://inspect/#service-workers
    event.respondWith(
        caches.match(event.request).then((response) => {
            return response || fetch(event.request);
        })
    );
});
```

取得檔案資料時，先由快取區取得

```
self.addEventListener('activate', event=>{
    event.waitUntil(
        caches.keys().then( keyList=>{
            return Promise.all(keyList.map( key=>{
                if (key !== cacheName) {
                    return caches.delete(key);
                }
            }));
        })
    );
});
```

啟動時可以移除舊版本的快取

15. Notifications

- Chrome 通知設定 chrome://settings/content/notifications
- 連結桌面的通知功能
- 測試網址 https://tw.yahoo.com/
- 使用Notification 類型
- 檢視目前允許狀況 ***Notification***.permission
  - granted : 允許
  - denied : 封鎖
  - default : 預設
- 要求權限 ***Notification***.requestPermission()

https://github.com/shinder/html5-api-examples/blob/master/public/myApp/notification-my.html

```
const btn = document.querySelector('button');
const checkNotification = function(){
    console.log(Notification.permission);
    switch (Notification.permission) {
        case 'granted':
            console.log('允許');
            btn.style.display = 'block';
            break;
        case 'denied':
            console.log('封鎖');
            break;
        default:
            console.log('未決定');
            // 要求權限
            Notification.requestPermission(function(){
                if(Notification.permission==='granted'){
                    btn.style.display = 'block';
                }
            });
    }
};
checkNotification();
btn.onclick = function(){
    let ntf = new Notification('MyTitle', {
        body: '說明文字',
        icon: './images/icons/icon-144x144.png'
    });
};
```

16. Geolocation

- 使用 navigator.geolocation.getCurrentPosition()
- 由於屬於隱私權資料，所以會有詢問要求權限的對話框。
- https://github.com/shinder/html5-api-examples/blob/master/public/geolocation-map.html

```
<div id="myMap"></div>
<div id="info"></div>
<script src="https://maps.googleapis.com/maps/api/js?key="></script>
<script>
const positionHandler = (position)=>{
  const lat = position.coords.latitude; // 緯度
  const lng = position.coords.longitude; // 經度
  console.log(lat, lng);
  //取得地圖座標
  const latlng = new google.maps.LatLng(lat, lng);
  const myOptions = {zoom: 13, center: latlng, mapTypId: google.maps.MapTypeId.ROADMAP};
  //建立地圖
  const map = new google.maps.Map(document.querySelector('#myMap'), myOptions);
  //建立標記
  const marker = new google.maps.Marker({ position: latlng, title: "標記的名稱" });
  marker.setMap(map); // 把標記放入地圖
  // *** 利用座標取得地址 ***
  const geocoder = new google.maps.Geocoder();
  geocoder.geocode({ 'latLng': latlng }, function (result, status) {
    console.log(result);
    if (status === google.maps.GeocoderStatus.OK) {
      const address = result[0].formatted_address; //取得地址
      document.querySelector('#info').innerHTML = address;
    }
  })
};
// 取得目前位置經緯度
navigator.geolocation.getCurrentPosition(positionHandler, error=>{
  alert(JSON.stringify(error));
});
</script>
```

王孝弘老師的參考資料

http://html5api.azurewebsites.net/