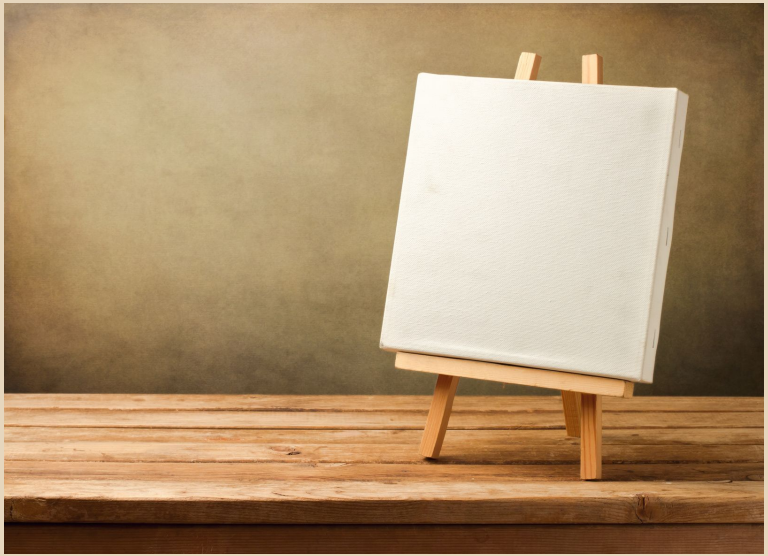
HTML5 Canvas

Lecturer: Resnick Chang

HTML5 APIs

- Drag and Drop
- Geolocation
- Web Storage
- Web Database (Firefox 無)
- Video and Audio Support
- SVG and Canvas
- Application Cache
- Web Workers



Canvas

<canvas> Tag

- <canvas> 標籤用來定義html中的一個畫布，並使用Javascript呼叫 html canvas API來繪製圖形。
- canvas上可以
  - 繪製各種幾何圖形，如線段，圓
  - 可以加上文字，陰影與效果
  - 也可以用來繪製影像，並儲存影像

定義Canvas

- 在canvas用來繪圖之前，必須先定義
  - **id**, **width**, **height**
  - 上述屬性可以透過html定義，也可以透過JS定義

```
<canvas id="myCanvas" width="600" height="400"> </canvas>
```

canvas的預設內容

- 萬一遇到瀏覽器不支援，或者canvas程式載入失敗
  可以給定一個預設文字，提供canvas無法顯示時，瀏覽器應顯示的內容

```
<canvas id="myCanvas" width="600" height="400">
    <p>萬一Canvas掛掉，就顯示這段文字</p>
</canvas>
```

繪製的第一步

- 使用DOM選擇器取得Canvas

```
var canvas = document.getElementById('myCanvas');
if (canvas.getContext) {
    var ctx = canvas.getContext('2d');


    //繪製圖形的程式碼寫在這邊



} // close if
```

繪製矩形

- fillRect(x, y, width, height)
  - 畫出一個填滿的矩形。
- strokeRect(x, y, width, height)
  - 畫出一個矩形的邊框
- clearRect(x, y, width, height)
  - 清除指定矩形區域內的內容，使其變為全透明。

Example

```
function draw() {
  var canvas = document.getElementById('canvas');
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d');

    ctx.fillRect(25,25,100,100);
    ctx.clearRect(45,45,60,60);
    ctx.strokeRect(50,50,50,50);
  }
}
```

動動手:

- 繪製如下的兩個長方形，讓它們重疊在一起

路徑繪製

- beginPath()
  - 產生一個新路徑，產生後再使用繪圖指令來設定路徑。
- closePath()
  - 閉合路徑好讓新的繪圖指令來設定路徑。
- 路徑 API
  - moveTo, lineTo等指令
- stroke()
  - 畫出圖形的邊框。
- fill()
  - 填滿路徑內容區域來產生圖形。

Example: 繪製三角形

```
function draw() {
  var canvas = document.getElementById('canvas');
  if (canvas.getContext){
    var ctx = canvas.getContext('2d');

    ctx.beginPath();
    ctx.moveTo(75,50);
    ctx.lineTo(100,75);
    ctx.lineTo(100,25);
    ctx.fill();
  }
}
```

路徑API: beginPath, moveTo

- beginPath()
  - 產生一個路徑，表面下，路徑會被存在一個次路徑 (sub-path) 清單中，例如直線、曲線等，這些次路徑集合起來就形成一塊圖形。
  - 每一次呼叫這個方法，次路徑清單就會被重設，然後我們便能夠畫另一個新圖形。
- moveTo(x, y)
  - 移動畫筆到指定的(x, y)座標點
  - 當初始化畫布或是呼叫beginPath()，通常會想要使用moveTo()來指定起始點，我們可以用moveTo()畫不連結的路徑

路徑API: lineTo

- lineTo(x, y)
  - 從目前繪圖點畫一條直線到指定的(x, y)座標點。
  - 本方法接受x, y參數作為線條結束點的座標位置，至於起始點則視前一個繪圖路徑，由前一個繪圖路徑的結束點作為起始點，當然，起始點也可以用moveTo()方法來變更。

```
function draw() {
  var canvas = document.getElementById('canvas');
  if (canvas.getContext){
    var ctx = canvas.getContext('2d');

    // Filled triangle
    ctx.beginPath();
    ctx.moveTo(25,25);
    ctx.lineTo(105,25);
    ctx.lineTo(25,105);
    ctx.fill();

    // Stroked triangle
    ctx.beginPath();
    ctx.moveTo(125,125);
    ctx.lineTo(125,45);
    ctx.lineTo(45,125);
    ctx.closePath();
    ctx.stroke();
  }
}
```

路徑API: arc

- arc(x, y, radius, startAngle, endAngle, anticlockwise)
  - 畫一個弧形
  - 本方法接受五個參數:
    - x, y代表圓心座標點,
    - radius代表半徑,
    - startAngle, endAngle分別代表沿著弧形曲線上的起始點與結束點的弧度, 弧度測量是相對於x軸,
    - anticlockwise為true代表逆時針作圖、false代表順時針作圖。

```
function draw() {
  var canvas = document.getElementById('canvas');
  if (canvas.getContext){
    var ctx = canvas.getContext('2d');

    for(var i=0;i<4;i++){
      for(var j=0;j<3;j++){
        ctx.beginPath();
        var x              = 25+j*50;               // x 座標
        var y              = 25+i*50;               // y 座標
        var radius         = 20;                    // 弧形半徑
        var startAngle     = 0;                     // 起始弧度
        var endAngle       = Math.PI+(Math.PI*j)/2; // 結束弧度
        var anticlockwise  = i%2==0 ? false : true; // 順時針逆時針

        ctx.arc(x, y, radius, startAngle, endAngle, anticlockwise);

        if (i>1){
          ctx.fill();
        } else {
          ctx.stroke();
        }
      }
    }
  }
}
```

路徑API: rect

- rect(x, y, width, height)
  - 畫一個左上角位於(x, y)、寬width、高height的矩形。
  - 呼叫這個方法，moveTo()方法會以(0, 0)參數被自動呼叫所以目前的下筆點跟者自動被設為預設座標。

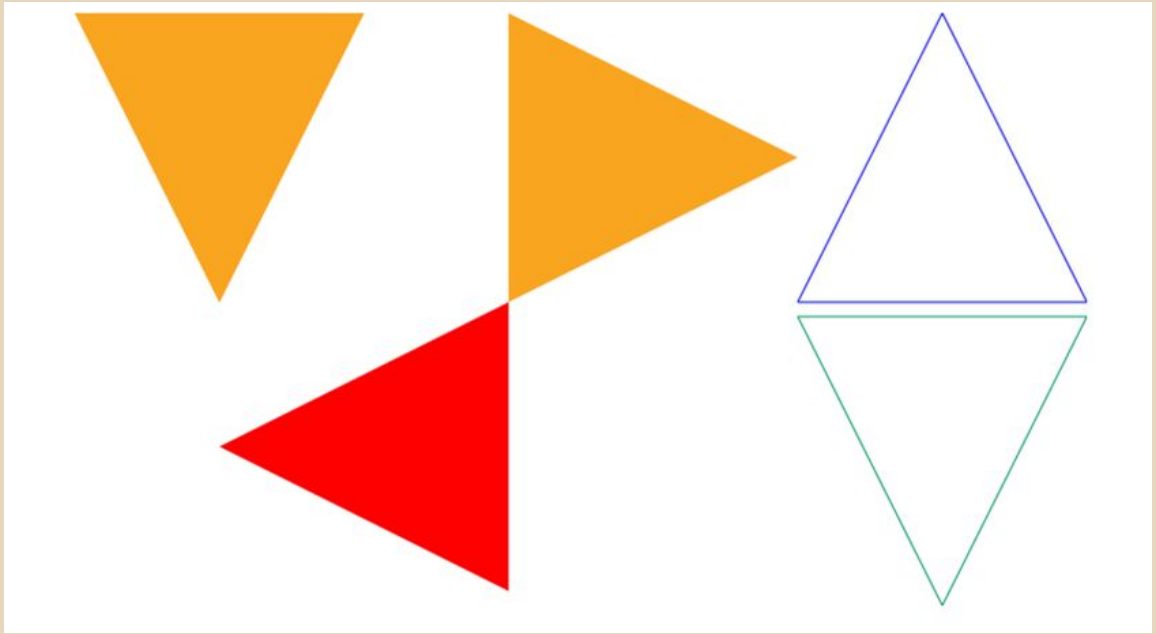
漸層

- createLinearGradient方法用來設置漸層。
- createLinearGradient方法的參數是(x1, y1, x2, y2)，其中x1和y1是起點坐標，x2和y2是終點坐標。通過不同的坐標值，可以生成從上至下、從左到右的漸變等等。

```
var myGradient = ctx.createLinearGradient(0, 0, 0, 160);
myGradient.addColorStop(0, "#BABABA");
myGradient.addColorStop(1, "#636363");
ctx.fillStyle = myGradient;
```

動動手:

- 繪製出下面五個三角形



繪製文字

- canvas上的文字也是畫上去的，繪製文字時相關的API如下
  - font - 定義字型
  - fillText(text,x,y) - 在 canvas 上繪製實心的字串
  - strokeText(text,x,y) - 在 canvas 上繪製空心的字串

Example: Hello World

```
var canvas=document.getElementById("myCanvas");
var ctx=canvas.getContext("2d");
ctx.font="30px Arial";
ctx.fillText("Hello World",10,50);
```

```
var canvas=document.getElementById("myCanvas");
var ctx=canvas.getContext("2d");
ctx.font="30px Arial";
ctx.strokeText("Hello World",10,50);
```

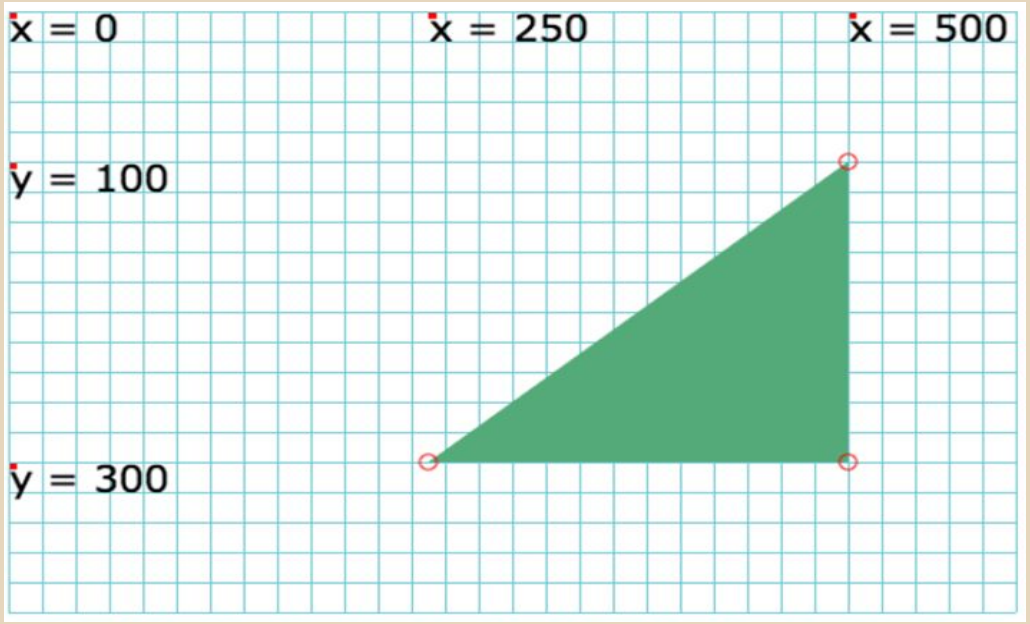
陰影

```
ctx.shadowOffsetX = 10; // 設置水平位移
ctx.shadowOffsetY = 10; // 設置垂直位移
ctx.shadowBlur = 5; // 設置模糊度
ctx.shadowColor = "rgba(0,0,0,0.5)"; // 設置陰影顏色

ctx.fillStyle = "#CC0000";
ctx.fillRect(10,10,200,100);
```

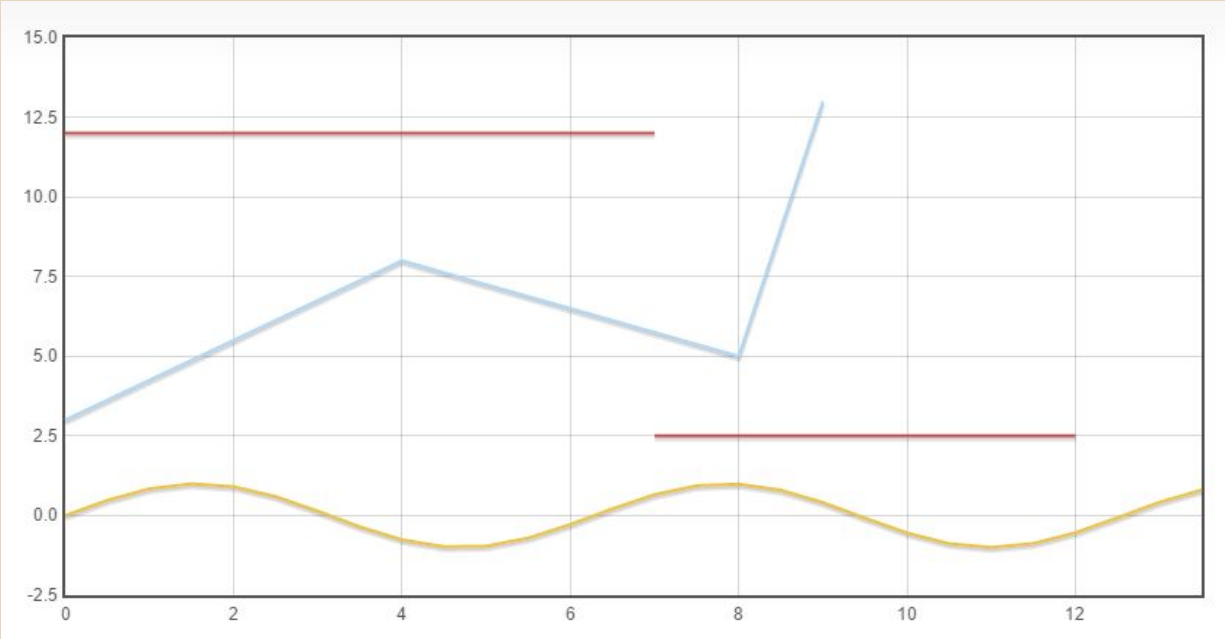
動動手:

- 透過上面所有API繪製下面網格與文字



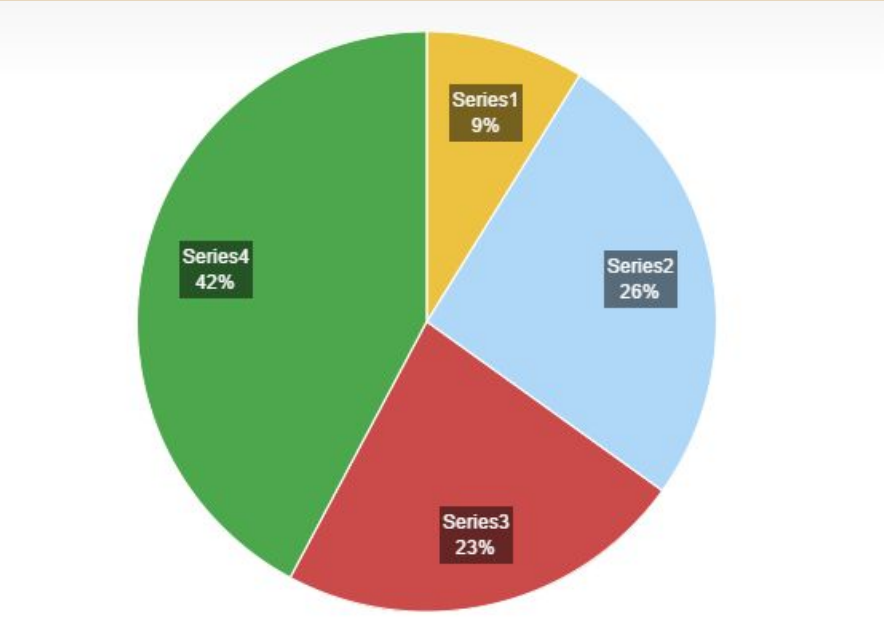
動動手: 繪製基本座標圖

- 繪製如下座標圖



動動手:

- 繪製如下圓餅圖



影像處理

- 使用drawImage可以將圖片繪製在canvas上

```
var image = new Image();
image.onload = function() {
  var canvas = document.createElement("canvas");
    canvas.width = image.width;
    canvas.height = image.height;
    canvas.getContext("2d").drawImage(image, 0, 0);

    return canvas;
}
image.src = "image.png";
```

draeImage

- drawImage()方法接受三個參數，
  - 第一個參數是圖像文件的DOM元素（即img標籤），
  - 第二個和第三個參數是圖像左上角在Canvas元素中的坐標，上例中的(0, 0)就表示將圖像左上角放置在Canvas元素的左上角。

```
var img = new Image();
img.src = "image.png";
ctx.drawImage(img, 0, 0); // 設置對應的圖像對象，以及它在畫布上的位置
```

getImageData, putImageData

- imageData對象有一個data屬性，它的值是一個一維陣列。該陣列的值，依次是每個像素的紅、綠、藍、alpha通道值，因此該陣列的長度等於 圖像的像素寬度 x 圖像的像素高度 x 4，每個值的範圍是0-255。
- 這個陣列可讀寫，因此通過操作這個陣列的值，就可以達到操作圖像的目的。修改這個陣列以後，使用putImageData方法將陣列內容重新繪製在Canvas上。

語法

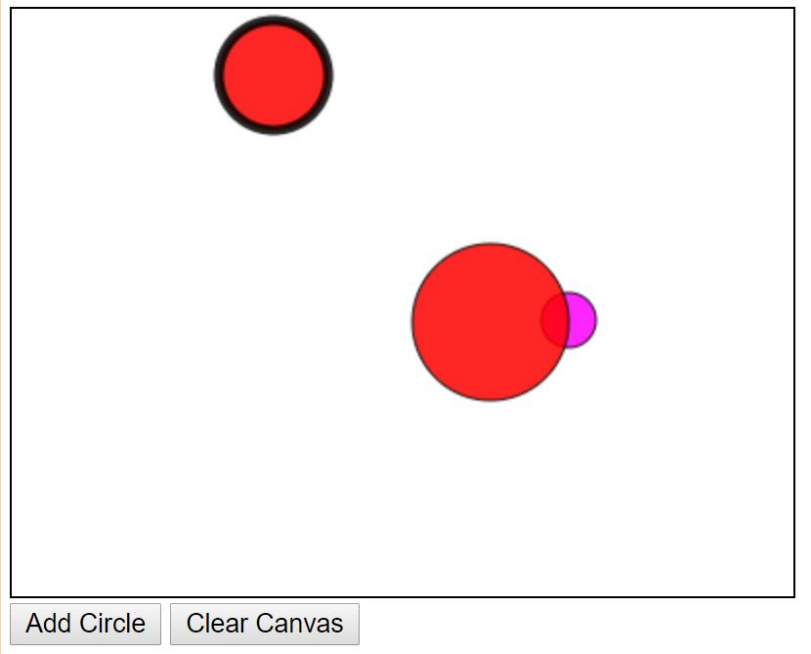
```
var imageData = context.getImageData(0, 0,
canvas.width, canvas.height);
context.putImageData(imageData, 0, 0);
```

Example

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
var img=document.getElementById("scream");
ctx.drawImage(img,10,10);
```

綜合實例

- 建構一個可以互動的canvas，可點選畫面上的元素



Video

HTML5 Video

- html 5 支援video標籤，來直接播放影片，無須使用flash
- 目前支援的格式

Format	IE	Firefox	Opera	Chrome	Safari
Ogg	No	3.5+	10.5+	5.0+	No
MPEG 4	No	No	No	5.0+	3.0+
WebM	No	No	10.6+	6.0+	No

如何使用

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

<video src="movie.ogg"  width="320" height="240"
controls="controls">
Your browser does not support the video tag.
</video>

</body>
</html>
```

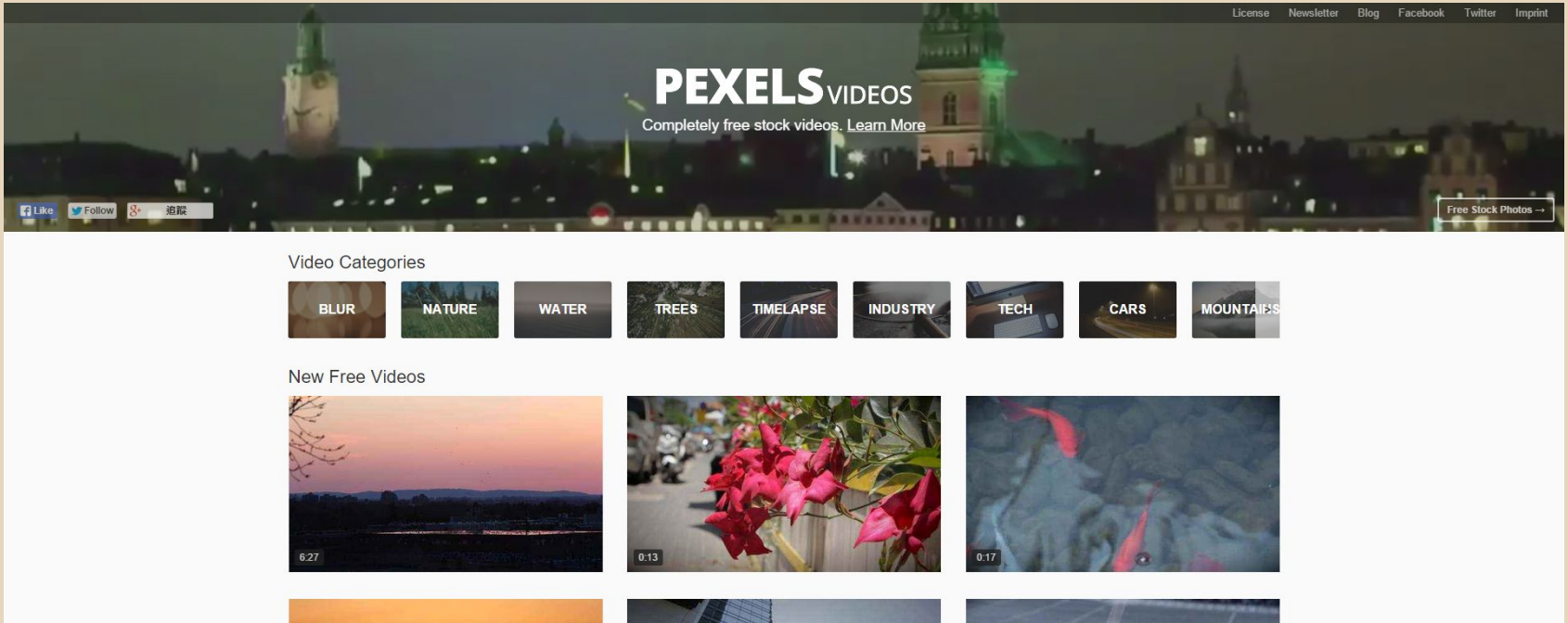
為影片提供不同的來源

- 同一個影片可以有不同的影片格式作為source

```
<video width="320" height="240" controls="
controls">
  <source src="movie.ogg" type="video/ogg" />
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.webm" type="video/webm" />
Your browser does not support the video tag.
</video>
```

影片素材

- Pexels Videos: https://videos.pexels.com/



Example

```
<!DOCTYPE html>
<html>

<body>
    <div style="text-align:center">
        <button onclick="playPause()">播放/暫停</button>
        <button onclick="makeBig()">放大</button>
        <button onclick="makeSmall()">縮小</button>
        <button onclick="makeNormal()">普通</button>
        <br>
        <video id="video1" width="420">
            <source src="mov_bbb.mp4" type="video/mp4">
            <source src="mov_bbb.ogg" type="video/ogg"> 您的瀏覽器不支持 HTML5 video 標籤。
        </video>
    </div>
```

```
    <script>
    var myVideo = document.getElementById("video1");
    function playPause() {
        if (myVideo.paused)
            myVideo.play();
        else
            myVideo.pause();
    }
    function makeBig() {
        myVideo.width = 560;
    }
    function makeSmall() {
        myVideo.width = 320;
    }
    function makeNormal() {
        myVideo.width = 420;
    }
    </script>
</body>

</html>
```

Video的API

方法	描述
addTextTrack()	向音頻/視頻添加新的文本軌道。
canPlayType()	檢測瀏覽器是否能播放指定的音頻 /視頻類型。
load()	重新加載音頻/視頻元素。
play()	開始播放音頻/視頻。
pause()	暫停當前播放的音頻/視頻。

Vidoe Event

事件	描述
abort	當音頻/視頻的加載已放棄時觸發。
canplay	當瀏覽器可以開始播放音頻 /視頻時觸發。
canplaythrough	當瀏覽器可在不因緩衝而停頓的情況下進行播放時觸發。
durationchange	當音頻/視頻的時長已更改時觸發。
emptied	當目前的播放列表為空時觸發。
ended	當目前的播放列表已結束時觸發。
error	當在音頻/視頻加載期間發生錯誤時觸發。
loadeddata	當瀏覽器已加載音頻 /視頻的當前幀時觸發。
loadedmetadata	當瀏覽器已加載音頻 /視頻的元數據時觸發。

Vidoe Event

loadstart	當瀏覽器開始查找音頻/視頻時觸發。
pause	當音頻/視頻已暫停時觸發。
play	當音頻/視頻已開始或不再暫停時觸發。
playing	當音頻/視頻在因緩衝而暫停或停止後已就緒時觸發。
progress	當瀏覽器正在下載音頻/視頻時觸發。
ratechange	當音頻/視頻的播放速度已更改時觸發。
seeked	當用戶已移動/跳躍到音頻/視頻中的新位置時觸發。
seeking	當用戶開始移動/跳躍到音頻/視頻中的新位置時觸發。
stalled	當瀏覽器嘗試獲取媒體數據，但數據不可用時觸發。

Vidoe Event

suspend	當瀏覽器刻意不獲取媒體數據時觸發。
timeupdate	當目前的播放位置已更改時觸發。
volumechange	當音量已更改時觸發。
waiting	當視頻由於需要緩衝下一幀而停止時觸發。