1. 初步

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Training 99</title>

<script>

let ga = {

res:{ // 因為一定要保證影像身音載入完成!!!

imgs:{explosion: "imgs/particle-explosion.png", plane: "imgs/plane.png"}, //影像列表

sounds:{explosion: "sounds/sound-explosion.mp3", bullet: "sounds/sound-bullet.mp3"}, // 聲音列表

loaded: 0,

total:4 //這裡先寫死，一班可以後端去撈

}

}; // 如果預期程式大，讓全域變數只有一個!

window.addEventListener("load", function(){

ga.loadResources(); // 載入所有

});

ga.loadResources = function(){ // 把全部載入行為包在ga內 (當然也是可以直接寫在ga內! 但會怕一大包!!!!!)

ga.loadImages();

// ga.loadSounds();

}

ga.loadImages = function(){ //彭: 他習慣因為只在loadResources用，所以縮排

// 找出物件中的每個成員

for(let name in ga.res.imgs){

// alert(name + ":" + ga.res.imgs[name])

ga.loadImage(name, ga.res.imgs[name])

}

}

ga.loadImage = function(name, src){ // 單張圖片的載入

let img = new Image();

img.src = src;

img.onload = function(){

// 圖片載入完成

ga.res.imgs[name] = this; // 載入完從原本字串變物件

}

}

</script>

</head>

<body>

<div id="loader"></div>

<div id="main" style="display:none">Main</div>

<div id="game" style="display:none">Game</div>

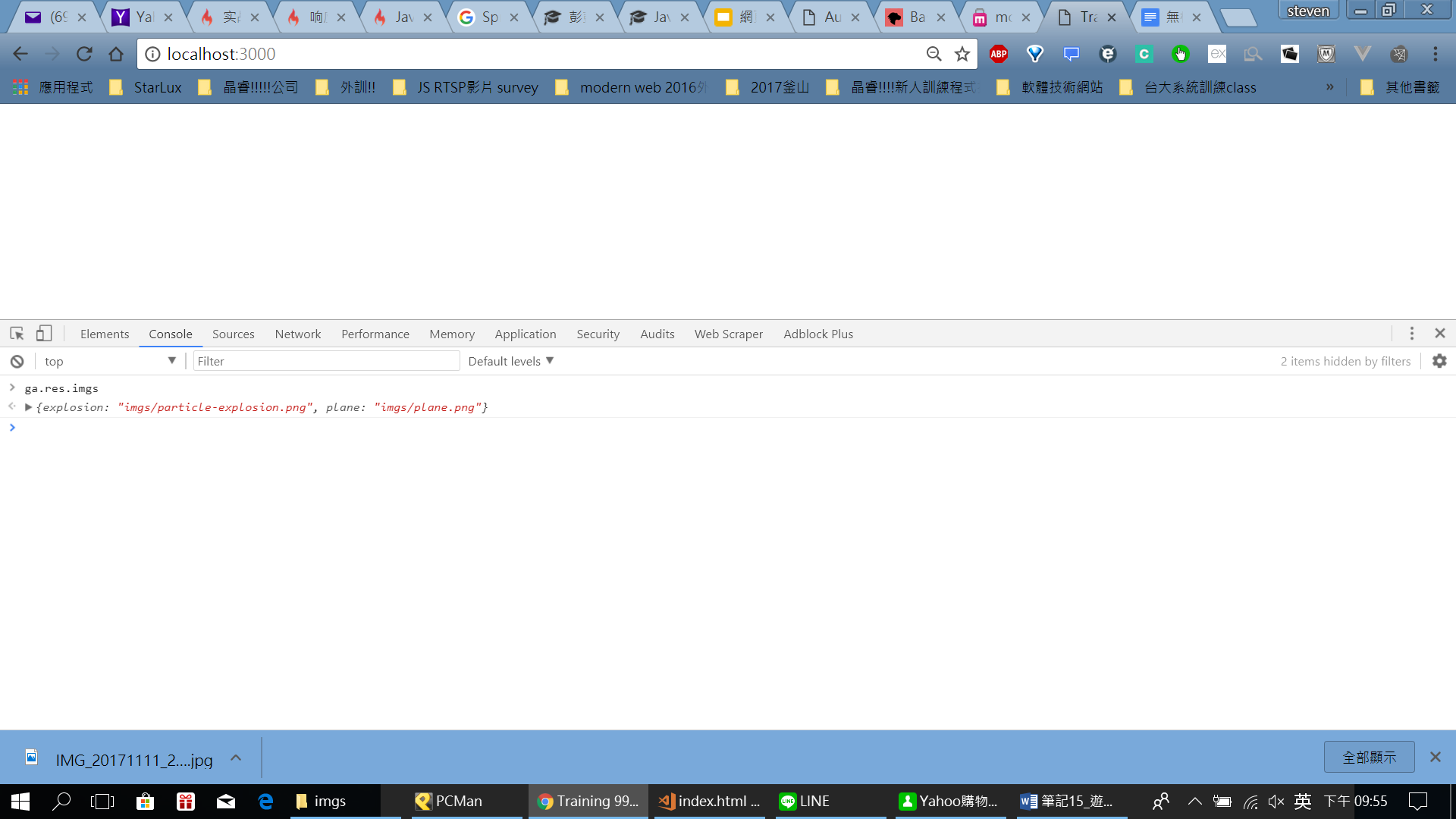
</body>

</html>

上面這邊loadresources 就是要把我們的全域指到的string 變成圖片物件:

例如假如我先把onload那邊跑的ga.loadResource拿掉

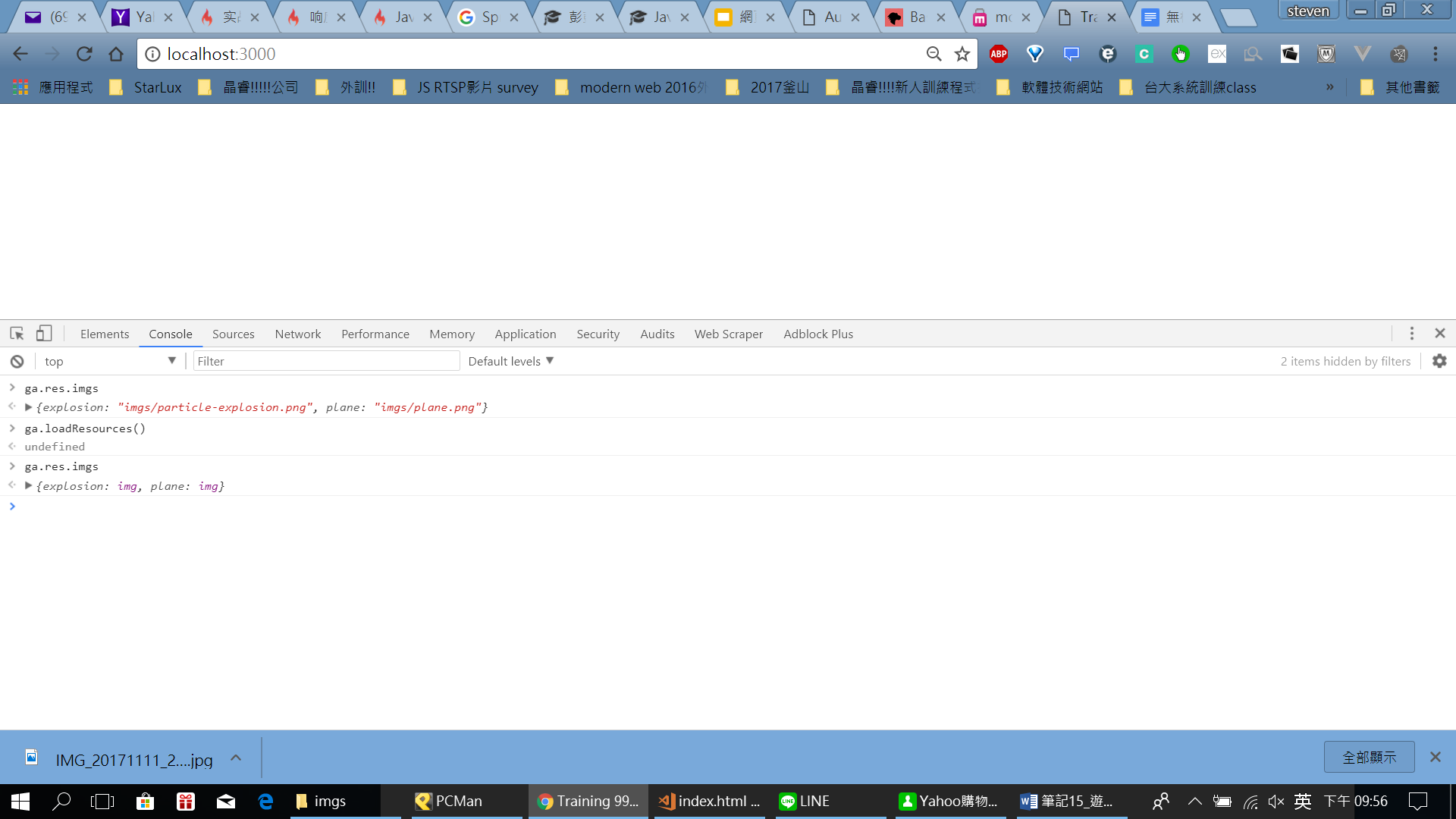
到browser那邊輸入ga.res.imgs會看到顯示:

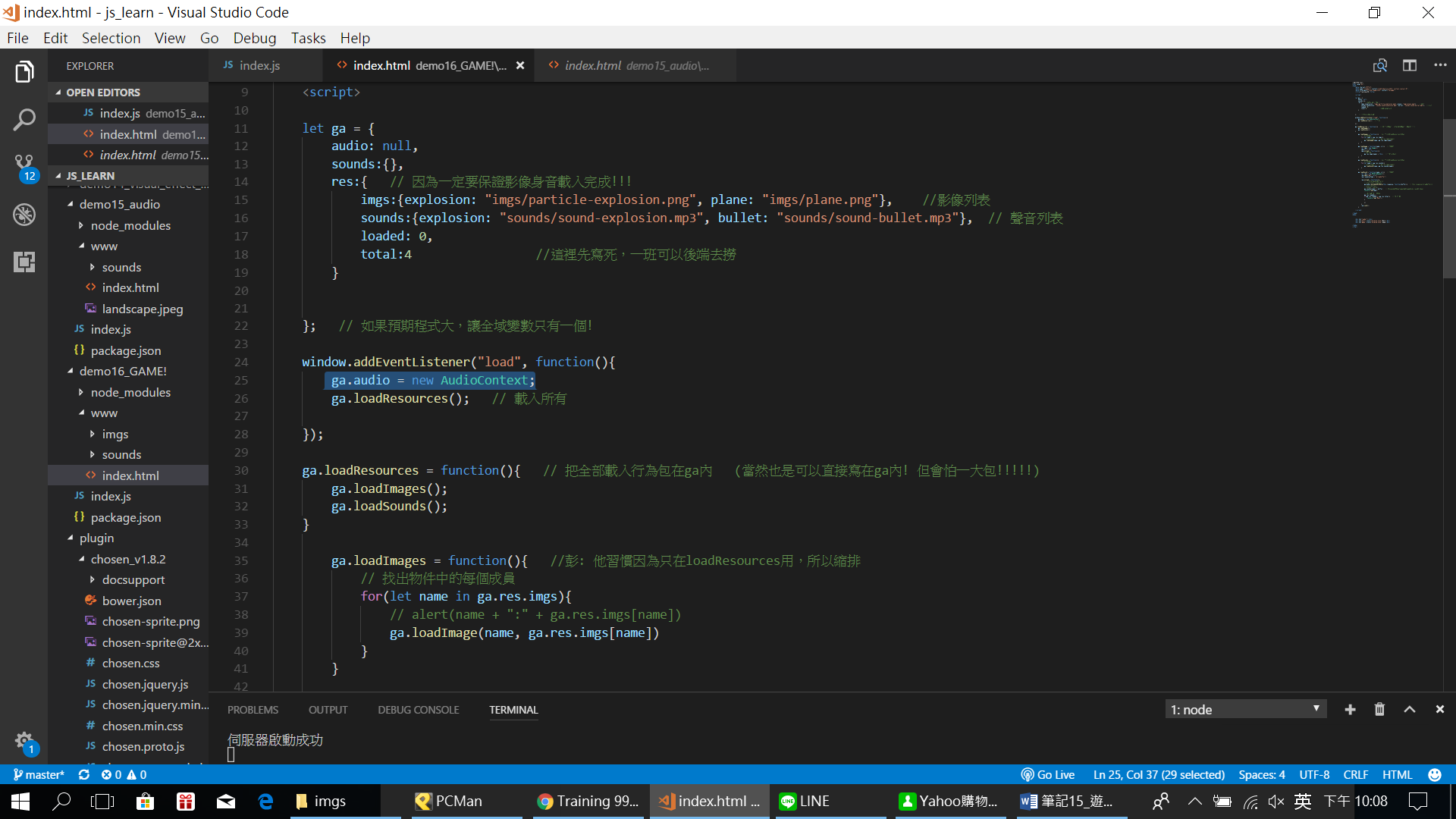


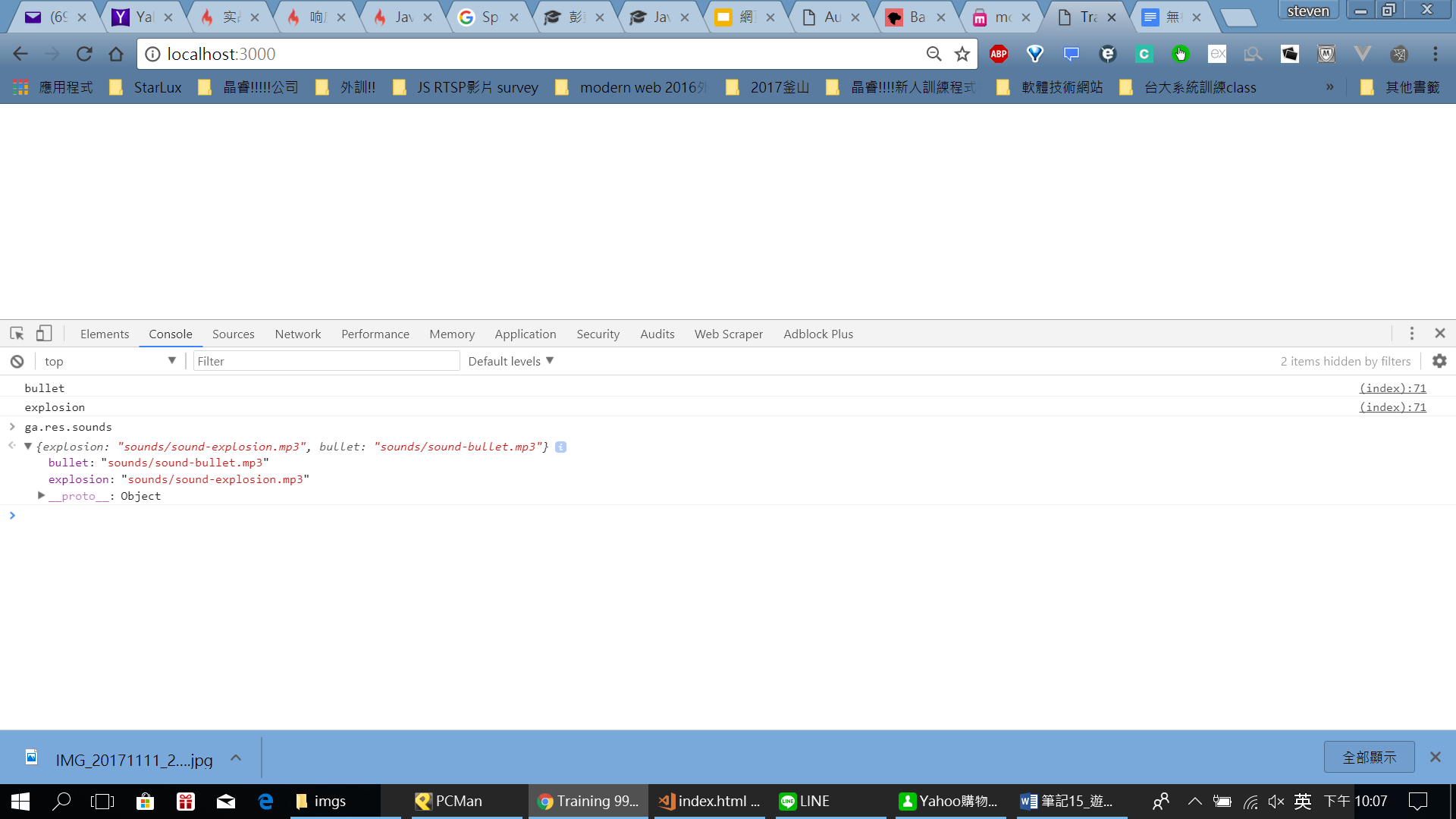
這時候去跑:

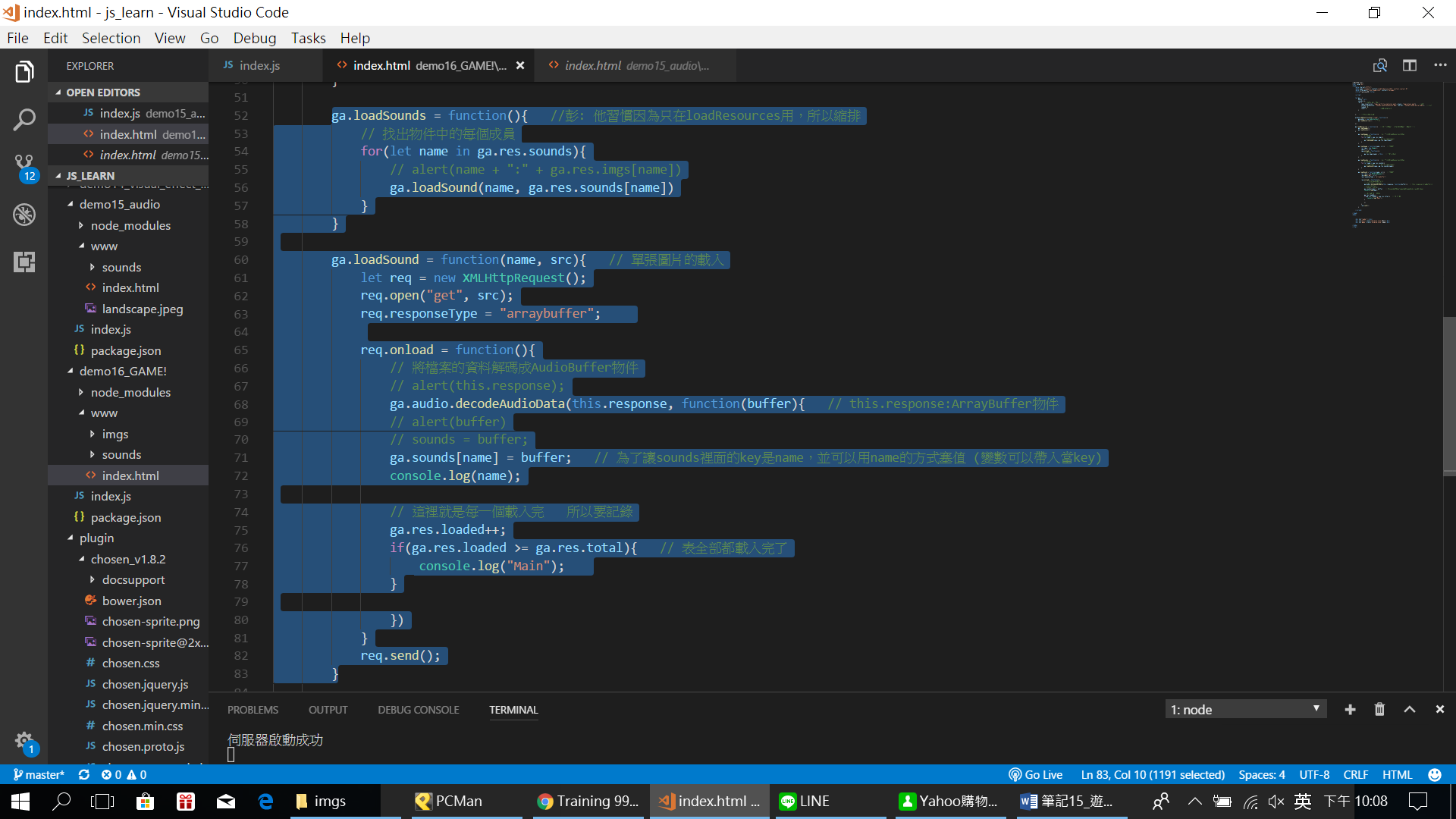
ga.loadResources:

再重新秀就會發現轉成圖片物件了!!!:

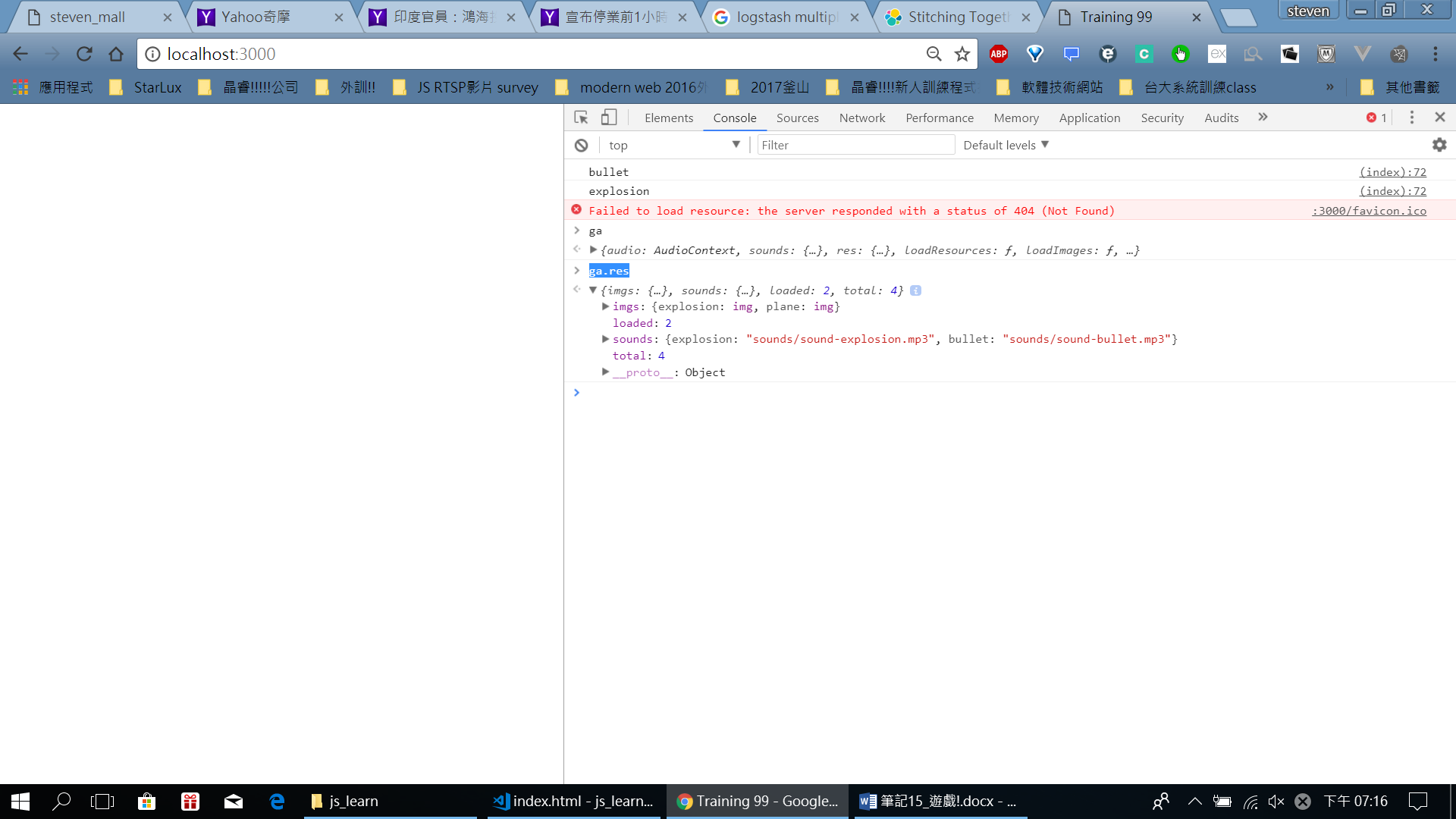


1. 補上sound:
2. 





1. Console輸入可看到仔入完成:



1. 先來寫個載入程式:
2. // 每個載入來改數量
3. ga.resourceLoaded = function(){
4. // 這裡就是每一個載入完 所以要記錄
5. ga.res.loaded++;
6. // 弄個載入百分比XD
7. let process = 100 \* ga.res.loaded / ga.res.total;
8. document.getElementById("loader").innerHTML = process + "%"; // 顯示出來
9. if(ga.res.loaded >= ga.res.total){ // 表全部都載入完了
10. console.log("Main");
11. }
12. }

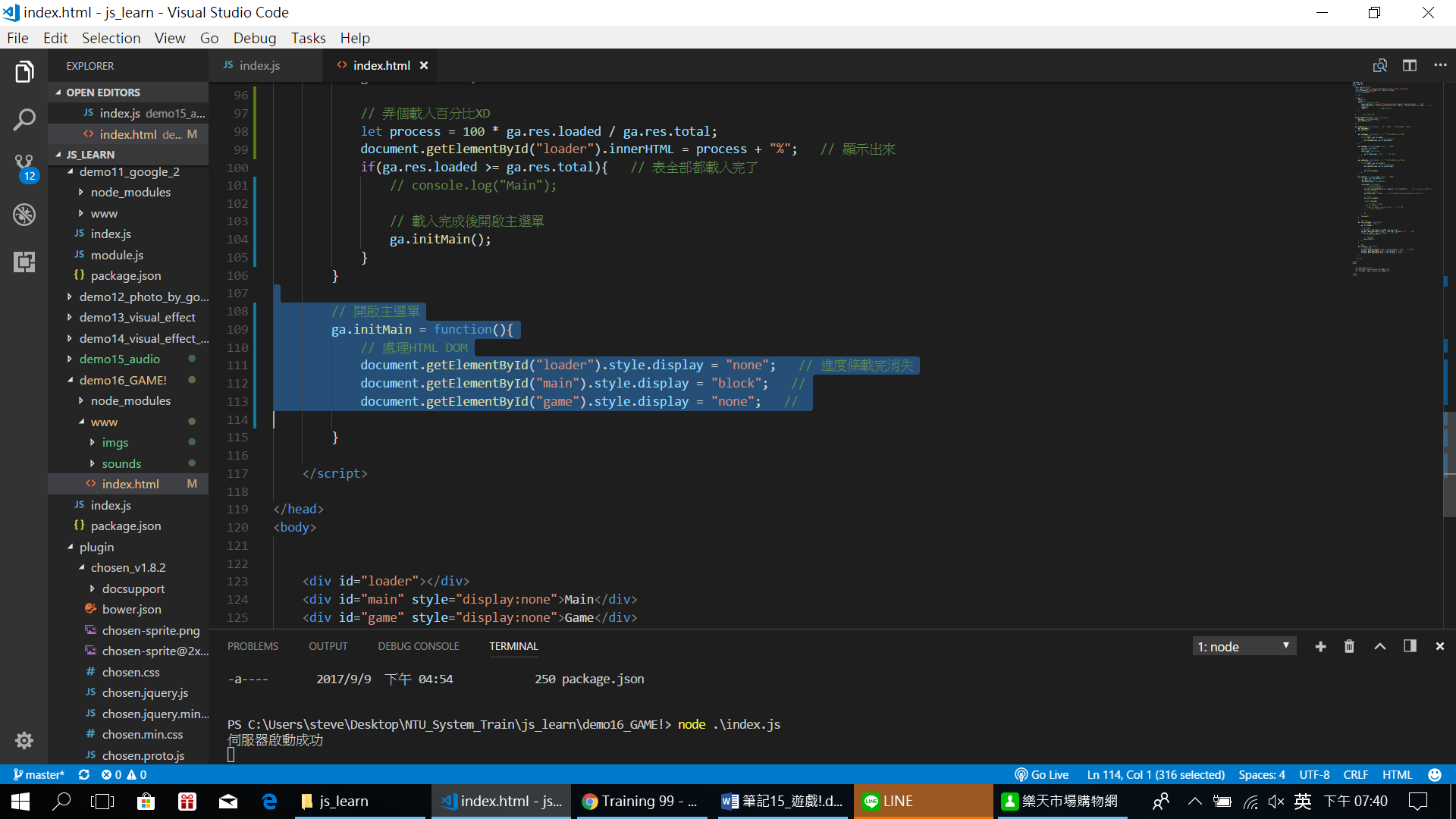
其中

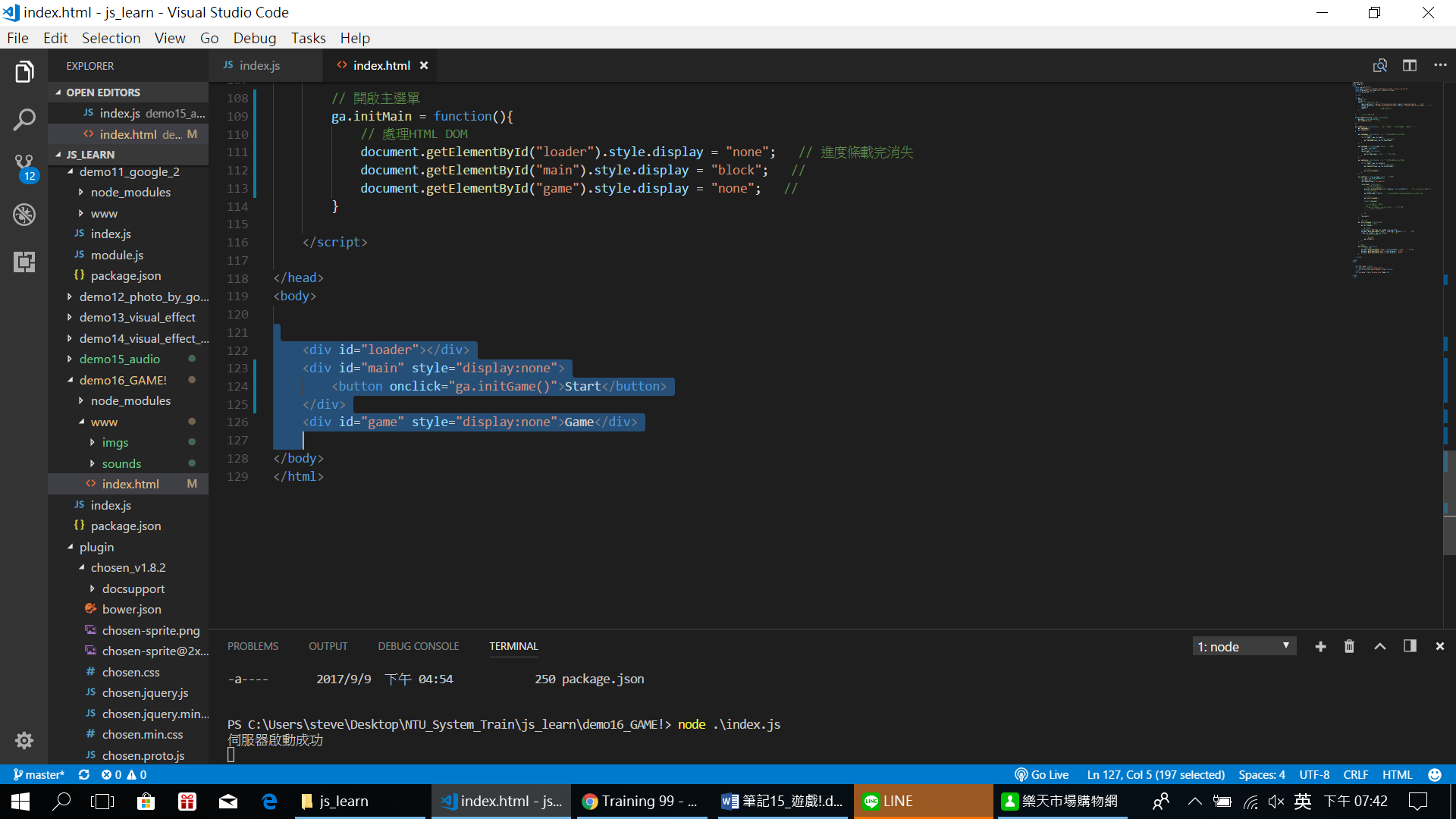
let process = 100 \* ga.res.loaded / ga.res.total;

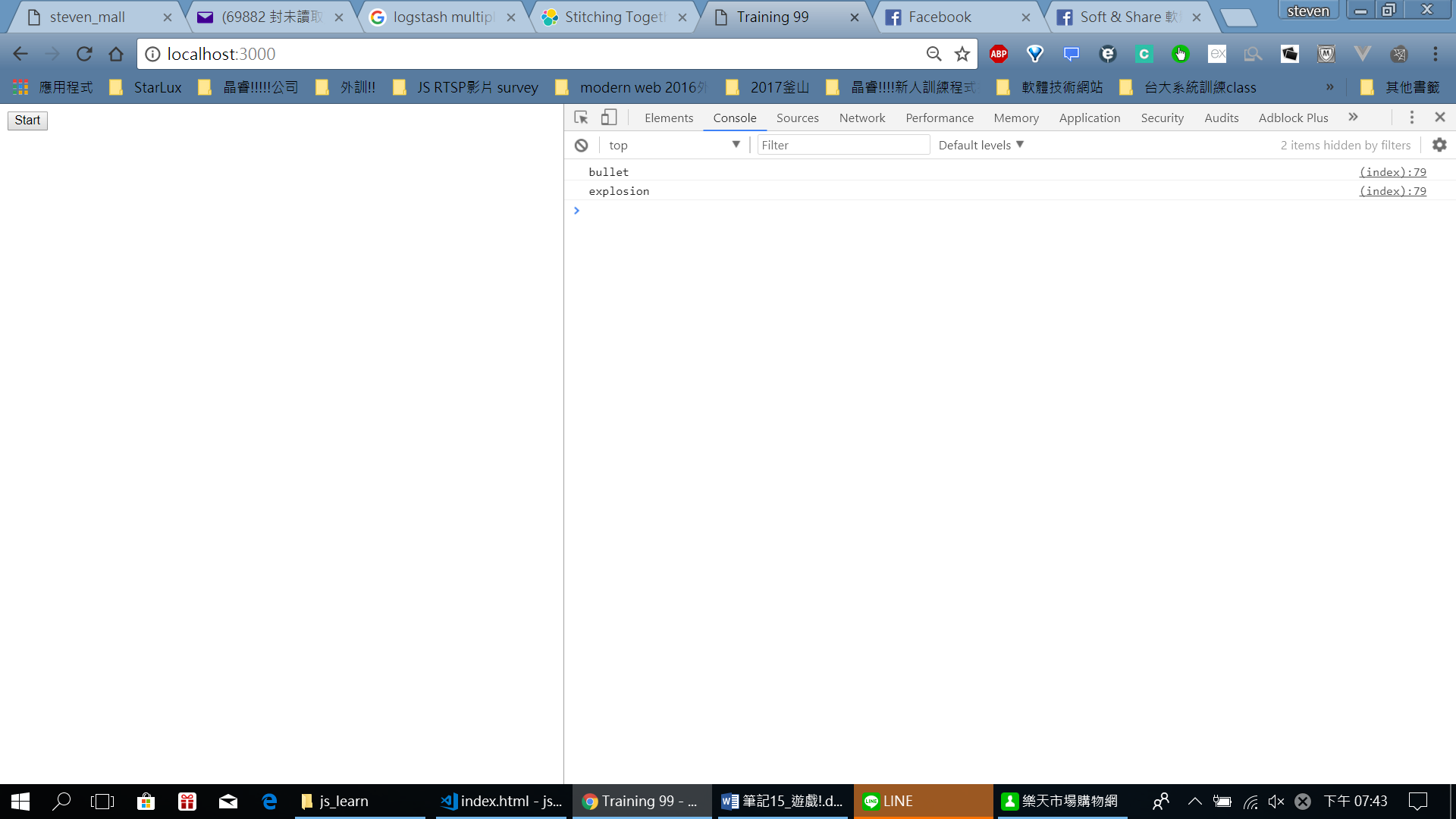
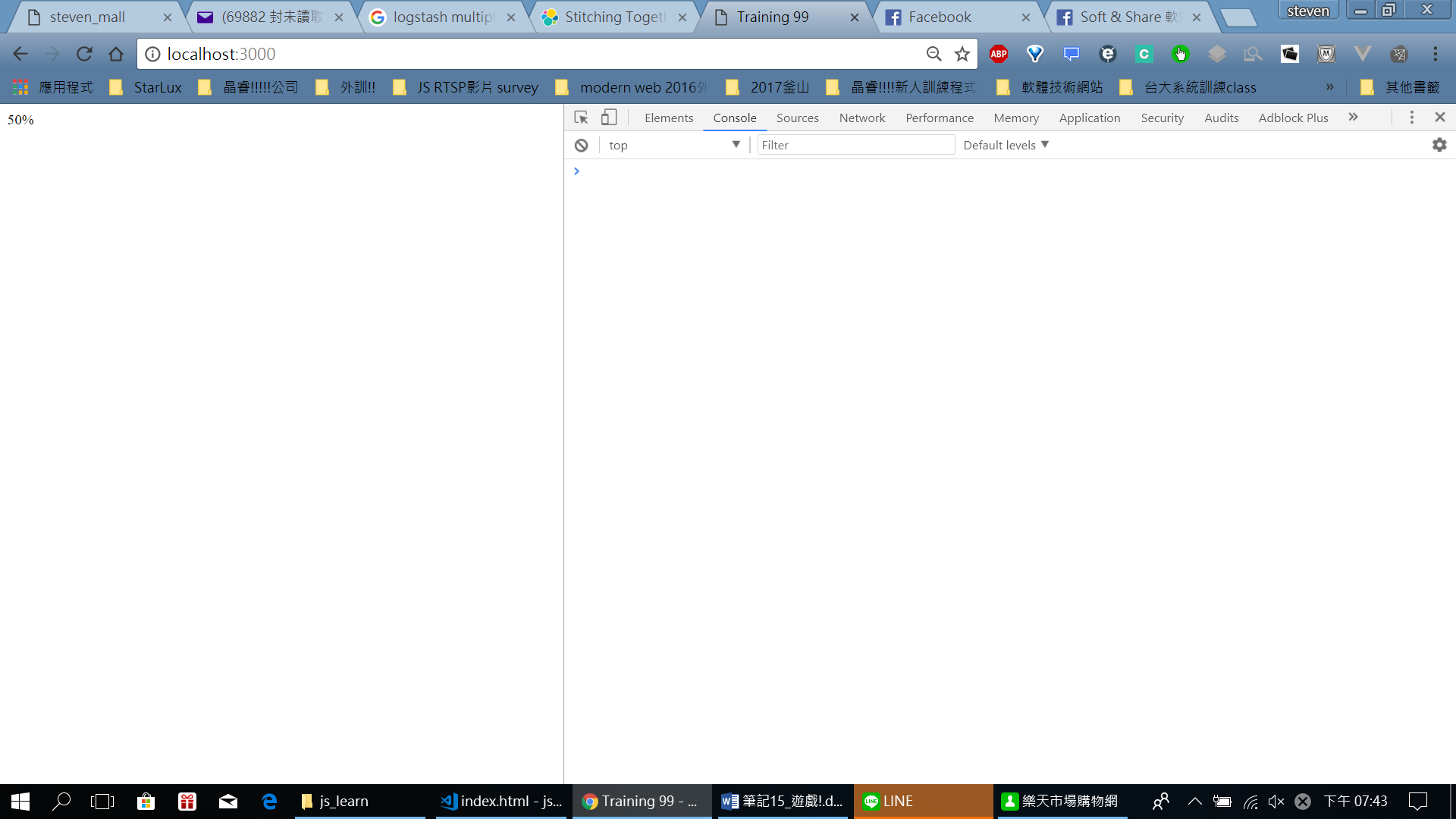
document.getElementById("loader").innerHTML = process + "%"; // 顯示出來

就是放進度條XD 網頁上放這個進度條很簡單!

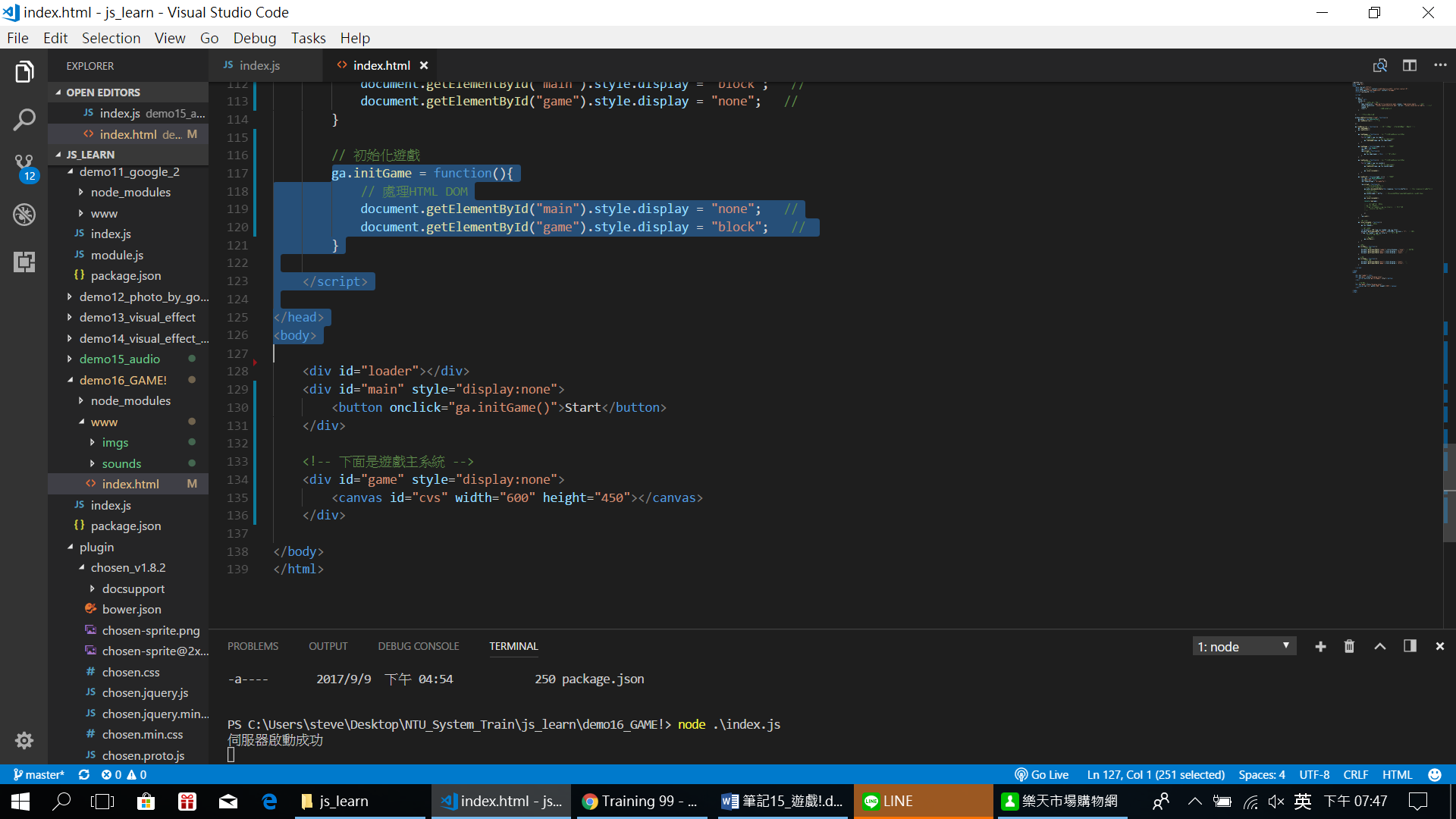
6. 當載入完成後 隱藏進度條，顯示主選單:



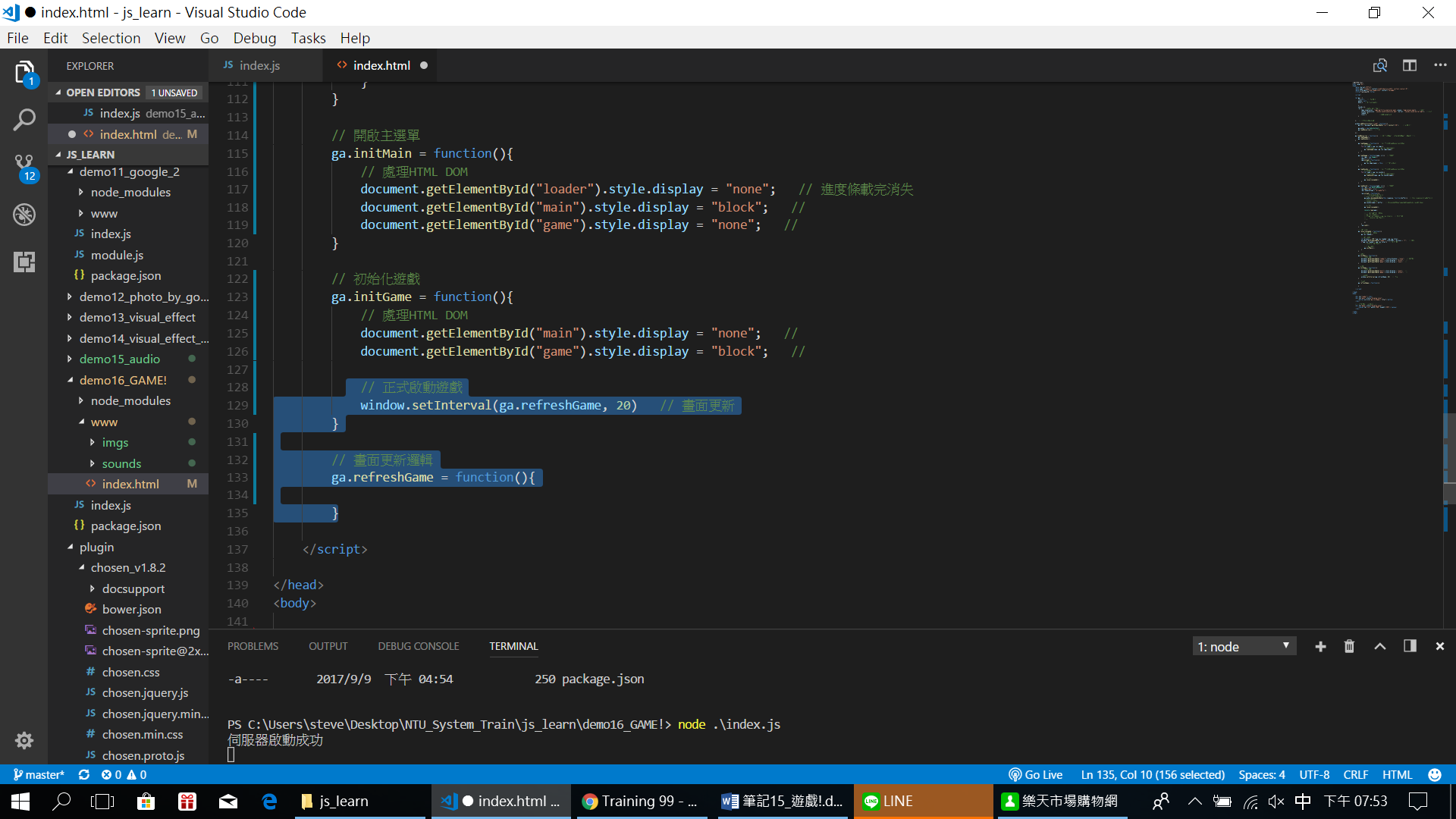




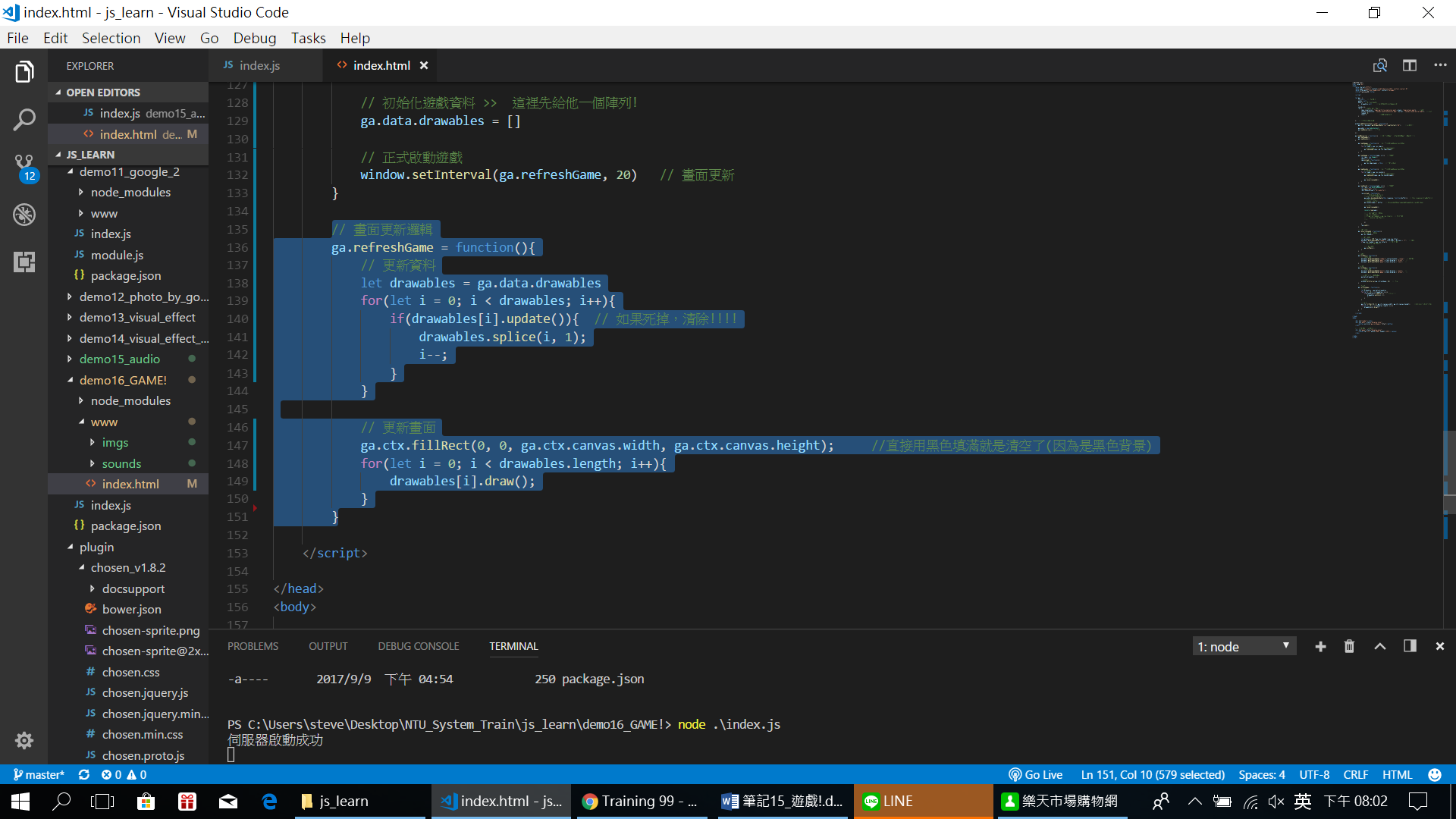
˙7.初始化遊戲

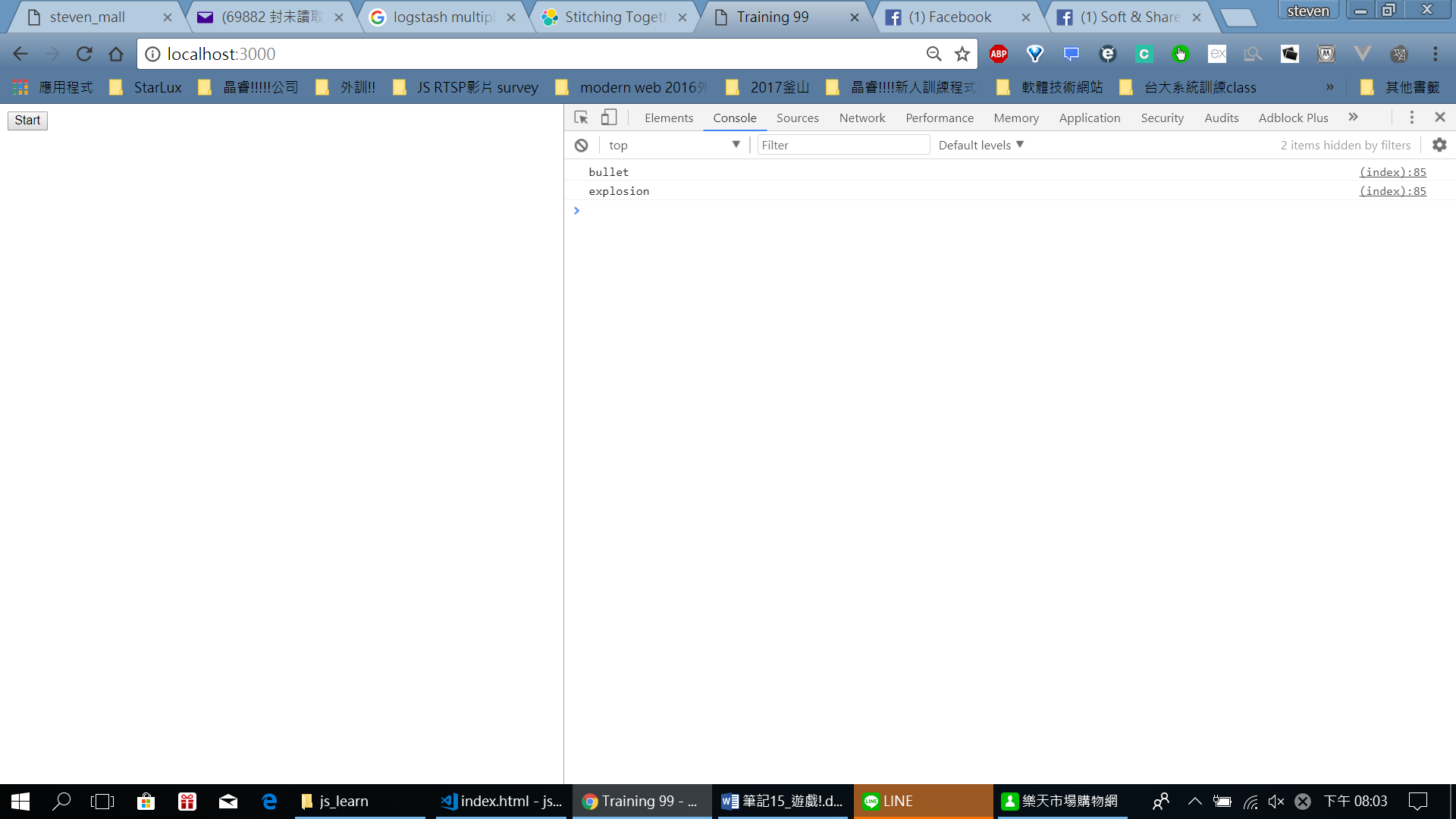


8. 更新畫面:

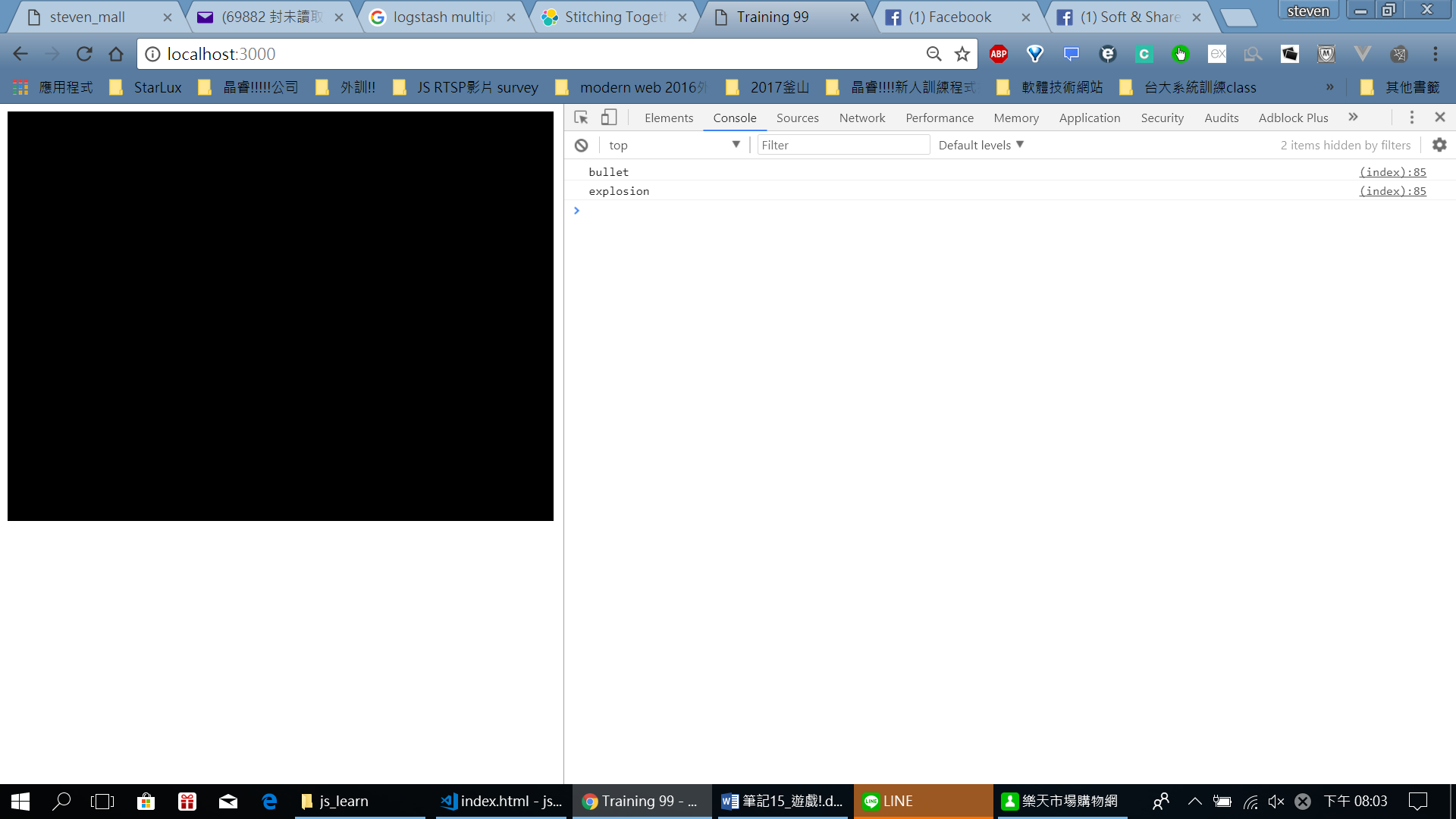


9. 撰寫更新畫面:

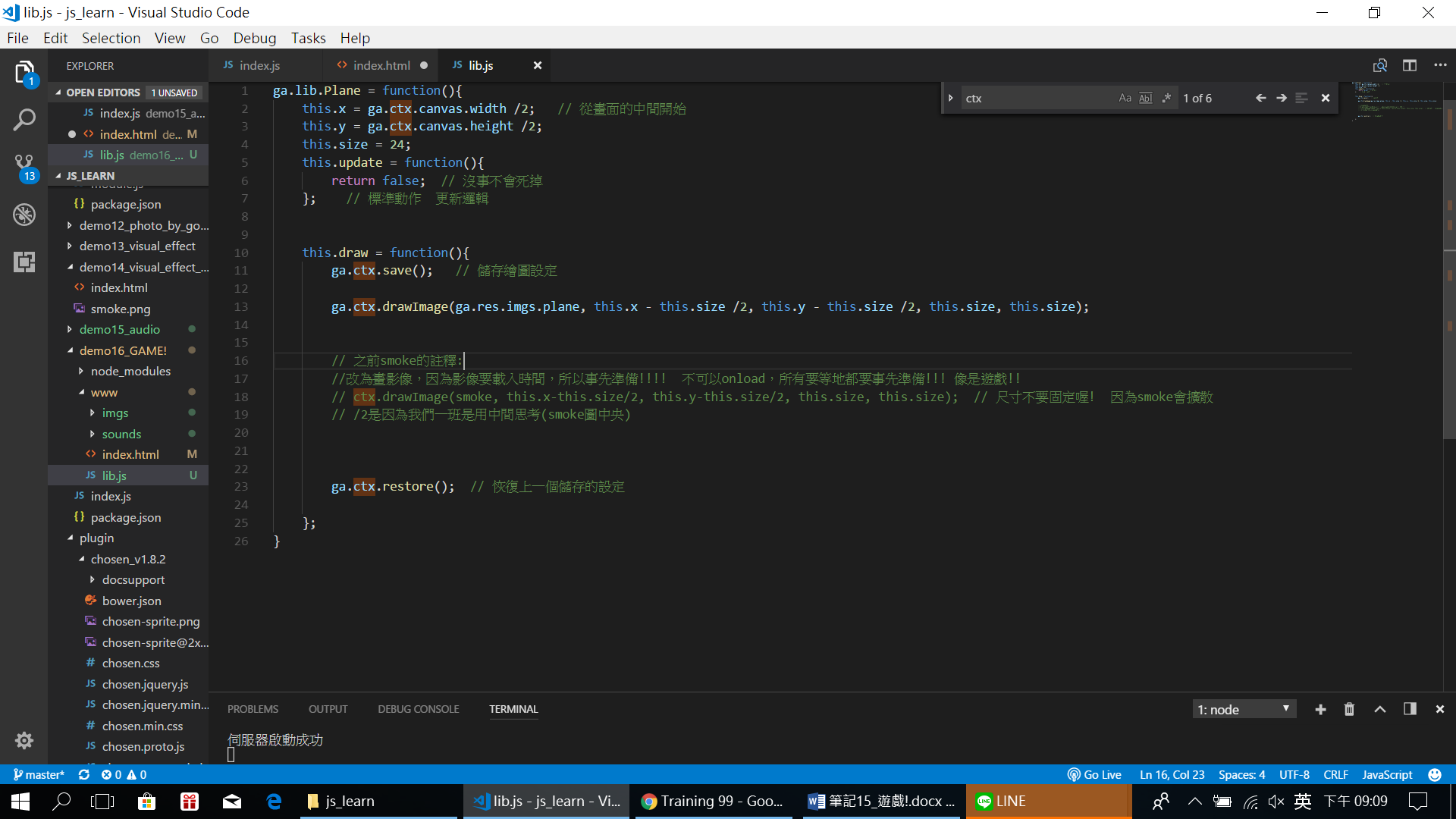




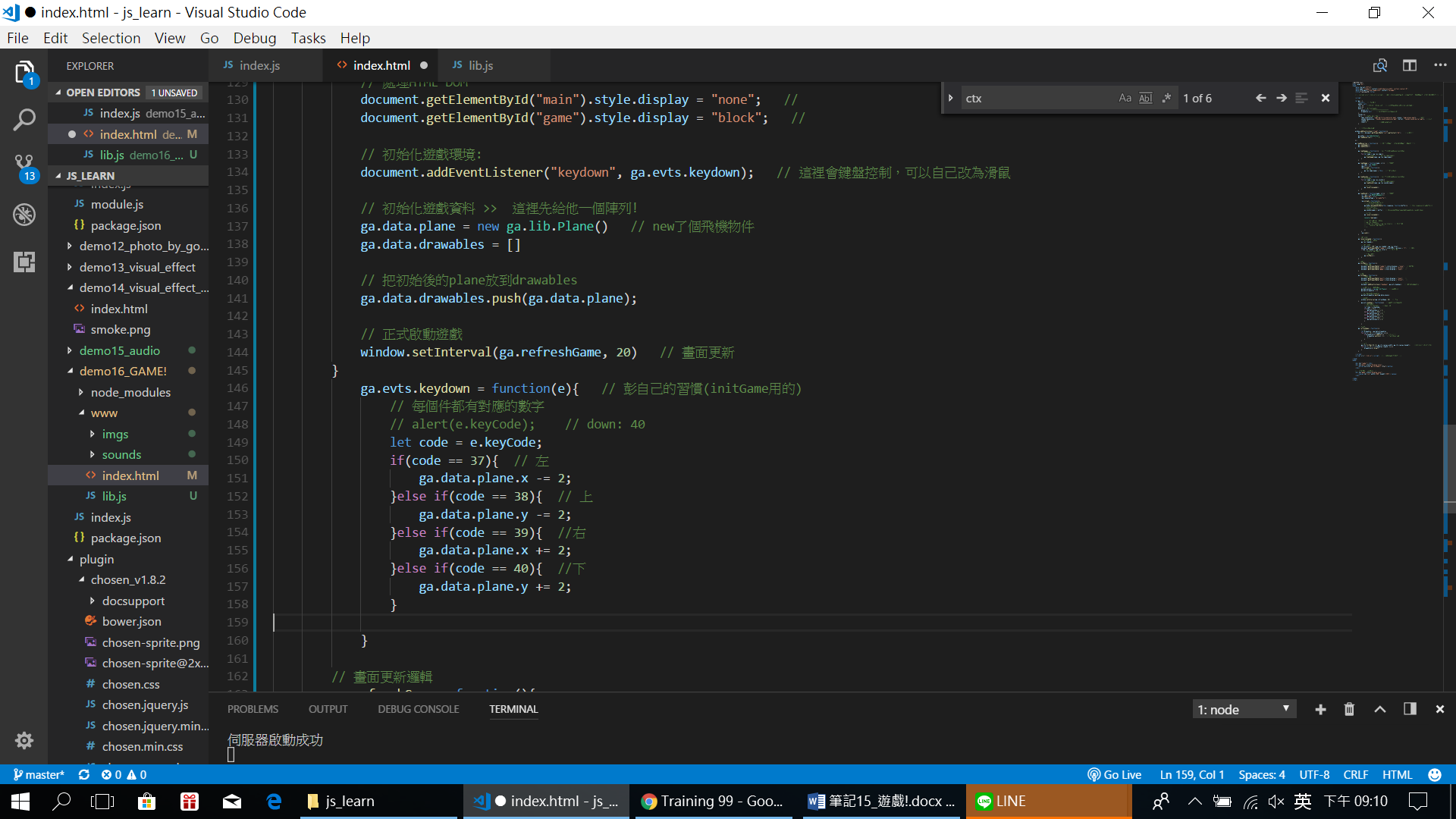
點擊後全黑:

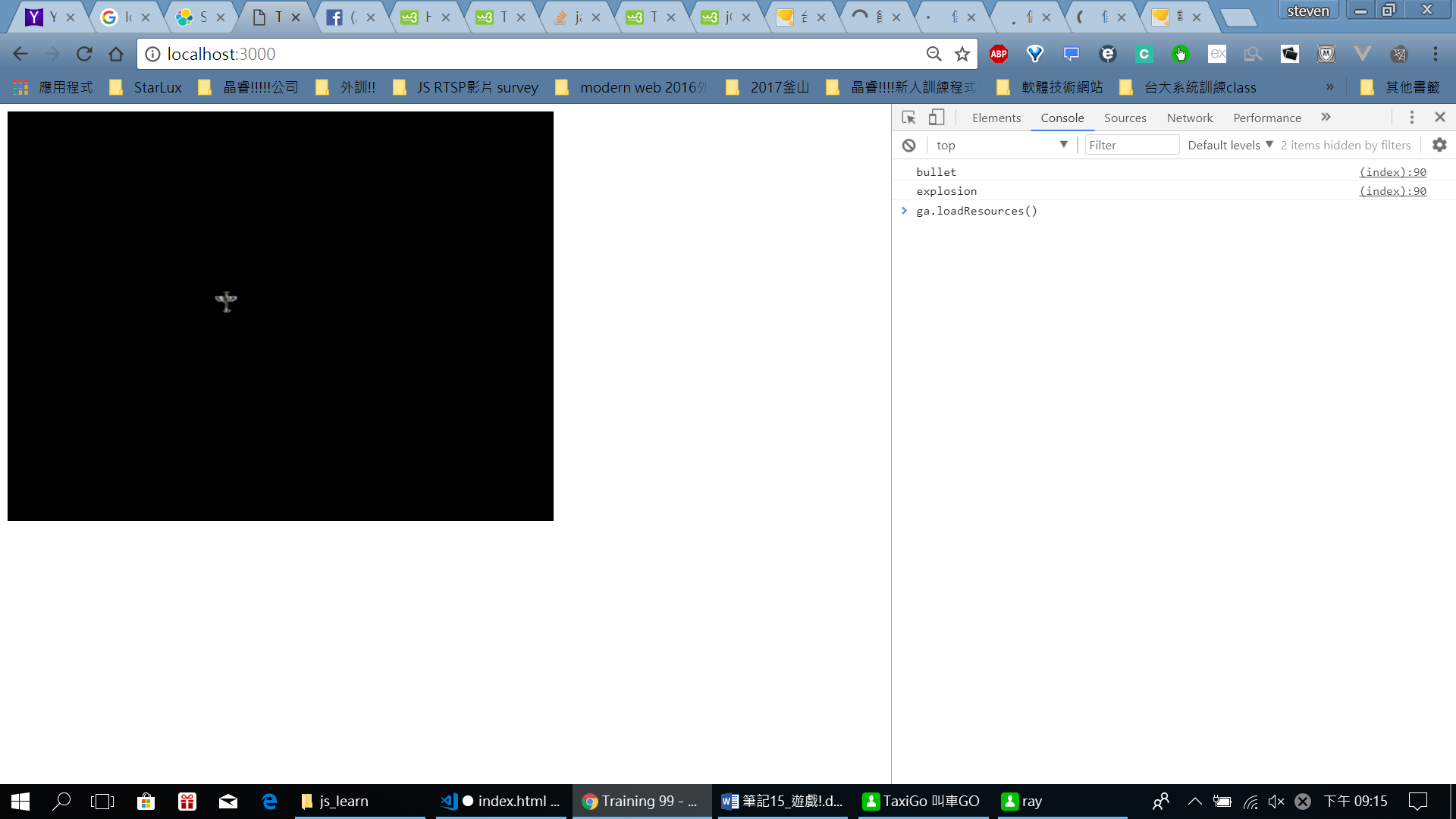


10. 改類別:



新增鍵盤event:





發現鍵盤會頓…

>> 改成記錄狀態:

目前code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Training 99</title>

<!-- <script src="./lib.js"></script> 這裡先引入就錯了，因為code在上面ga就會找不到，因為ga定義在下面 所以可以放在ga下面 或是把要引入的東西都放在最下面!!!> -->

<script>

let ga = {

ctx:null, // 後面初始化遊戲用

audio: null,

lib: {}, // 類別那邊要用! 在libs.js定義 // 這邊因為順序問題，一定要先讓瀏覽器看到，lib.js那邊用才可以喔!

evts:{},

data:{ // 遊戲中runtime的物件

plane:null,

drawables:null // 此初始化是在啟動遊戲時(initGame)初始化!

},

keys:{left:false, top: false, right: false, bottom: false,

space: false // space等等拿來加速用的

}, // 記錄你有興趣的鍵盤 (因為原本的event作法會頓，改為紀錄鍵盤，再根據鍵盤紀錄做事情!!!!!!)

sounds:{},

res:{ // 因為一定要保證影像身音載入完成!!!

imgs:{explosion: "imgs/particle-explosion.png", plane: "imgs/plane.png"}, //影像列表

sounds:{explosion: "sounds/sound-explosion.mp3", bullet: "sounds/sound-bullet.mp3"}, // 聲音列表

loaded: 0,

total:4 //這裡先寫死，一班可以後端去撈

}

}; // 如果預期程式大，讓全域變數只有一個!

window.addEventListener("load", function(){

ga.ctx = document.getElementById("cvs").getContext("2d"); // 初始化遊戲用!!

ga.audio = new AudioContext;

ga.loadResources(); // 載入所有

});

ga.loadResources = function(){ // 把全部載入行為包在ga內 (當然也是可以直接寫在ga內! 但會怕一大包!!!!!)

ga.loadImages();

ga.loadSounds();

}

ga.loadImages = function(){ //彭: 他習慣因為只在loadResources用，所以縮排

// 找出物件中的每個成員

for(let name in ga.res.imgs){

// alert(name + ":" + ga.res.imgs[name])

ga.loadImage(name, ga.res.imgs[name])

}

}

ga.loadImage = function(name, src){ // 單張圖片的載入

let img = new Image();

img.src = src;

img.onload = function(){

// 圖片載入完成

ga.res.imgs[name] = this; // 載入完從原本字串變物件

}

}

ga.loadSounds = function(){ //彭: 他習慣因為只在loadResources用，所以縮排

// 找出物件中的每個成員

for(let name in ga.res.sounds){

// alert(name + ":" + ga.res.imgs[name])

ga.loadSound(name, ga.res.sounds[name])

// 聲音載入成功後:

ga.resourceLoaded();

}

}

ga.loadSound = function(name, src){ // 單張圖片的載入

let req = new XMLHttpRequest();

req.open("get", src);

req.responseType = "arraybuffer";

req.onload = function(){

// 將檔案的資料解碼成AudioBuffer物件

// alert(this.response);

ga.audio.decodeAudioData(this.response, function(buffer){ // this.response:ArrayBuffer物件

// alert(buffer)

// sounds = buffer;

ga.sounds[name] = buffer; // 為了讓sounds裡面的key是name，並可以用name的方式塞值 (變數可以帶入當key)

// 聲音載入完:

ga.resourceLoaded();

console.log(name);

// // 這裡就是每一個載入完 所以要記錄

// ga.res.loaded++;

// if(ga.res.loaded >= ga.res.total){ // 表全部都載入完了

// console.log("Main");

// }

})

}

req.send();

}

// 每個載入來改數量

ga.resourceLoaded = function(){

// 這裡就是每一個載入完 所以要記錄

ga.res.loaded++;

// 弄個載入百分比XD

let process = 100 \* ga.res.loaded / ga.res.total;

document.getElementById("loader").innerHTML = process + "%"; // 顯示出來

if(ga.res.loaded >= ga.res.total){ // 表全部都載入完了

// console.log("Main");

// 載入完成後開啟主選單

ga.initMain();

}

}

// 開啟主選單

ga.initMain = function(){

// 處理HTML DOM

document.getElementById("loader").style.display = "none"; // 進度條載完消失

document.getElementById("main").style.display = "block"; //

document.getElementById("game").style.display = "none"; //

}

// 初始化遊戲

ga.initGame = function(){

// 處理HTML DOM

document.getElementById("main").style.display = "none"; //

document.getElementById("game").style.display = "block"; //

// 初始化遊戲環境:

document.addEventListener("keydown", ga.evts.keydown); // 這裡會鍵盤控制，可以自己改為滑鼠

document.addEventListener("keyup", ga.evts.keyup); // 改為記錄鍵盤狀態後: 放開鍵盤也要記錄狀態

// 初始化遊戲資料 >> 這裡先給他一個陣列!

ga.data.plane = new ga.lib.Plane() // new了個飛機物件

ga.data.drawables = []

// 把初始後的plane放到drawables

ga.data.drawables.push(ga.data.plane);

// 正式啟動遊戲

window.setInterval(ga.refreshGame, 20) // 畫面更新

}

ga.evts.keydown = function(e){ // 彭自己的習慣(initGame用的)

// 每個件都有對應的數字 // 但這裡做完發現畫面會頓(雖然windows可以設定)

// alert(e.keyCode); // down: 40

// let code = e.keyCode;

// if(code == 37){ // 左

// ga.data.plane.x -= 2;

// }else if(code == 38){ // 上

// ga.data.plane.y -= 2;

// }else if(code == 39){ //右

// ga.data.plane.x += 2;

// }else if(code == 40){ //下

// ga.data.plane.y += 2;

// }

// 上面的寫法會頓，所以改為紀錄狀態>>>>>>>> 為了繞過系統設計

let code = e.keyCode;

if(code == 37){ // 左

ga.keys.left = true;

}else if(code == 38){ // 上

ga.keys.top = true;

}else if(code == 39){ //右

ga.keys.right = true;

}else if(code == 40){ //下

ga.keys.bottom = true;

}else if(code == 32){

ga.keys.space = true;

} //空白建

}

ga.evts.keyup = function(e){ // 彭自己的習慣(initGame用的)

// 上面的寫法會頓，所以改為紀錄狀態>>>>>>>> 為了繞過系統設計

let code = e.keyCode;

if(code == 37){ // 左

ga.keys.left = false;

}else if(code == 38){ // 上

ga.keys.top = false;

}else if(code == 39){ //右

ga.keys.right = false;

}else if(code == 40){ //下

ga.keys.bottom = false;

}else if(code == 32){

ga.keys.space = false;

} //空白建

}

// 畫面更新邏輯

ga.refreshGame = function(){

// 更新資料

let drawables = ga.data.drawables

for(let i = 0; i < drawables.length; i++){

if(drawables[i].update()){ // 如果死掉，清除!!!!

drawables.splice(i, 1); // 從陣列中i的元素，刪除一個

i--;

}

}

// 更新畫面

ga.ctx.fillRect(0, 0, ga.ctx.canvas.width, ga.ctx.canvas.height); //直接用黑色填滿就是清空了(因為是黑色背景)

for(let i = 0; i < drawables.length; i++){

drawables[i].draw();

}

}

</script>

<script src="./lib.js"></script> <!-- 乾脆放這裡，ga的後面使用的前面! -->

</head>

<body>

<div id="loader"></div>

<div id="main" style="display:none">

<button onclick="ga.initGame()">Start</button>

</div>

<!-- 下面是遊戲主系統 但初始化的ctx -->

<div id="game" style="display:none">

<canvas id="cvs" width="600" height="450"></canvas>

</div>

</body>

</html>

ga.lib.Plane = function(){

this.x = ga.ctx.canvas.width /2; // 從畫面的中間開始

this.y = ga.ctx.canvas.height /2;

this.size = 24;

this.speed = 2; //建立飛機速度

this.update = function(){ // 後來改為紀錄鍵盤狀態後:

let speed = this.speed;

if(ga.keys.space){

speed \*= 2;

}

// 不要用else if (行為不一樣喔! 這樣會更順 因為會重疊)

if(ga.keys.left){

this.x -= speed;

}

if(ga.keys.top){

this.y -= speed;

}

if(ga.keys.right){

this.x += speed;

}

if(ga.keys.bottom){

this.y += speed;

}

return false; // 沒事不會死掉

}; // 標準動作 更新邏輯

this.draw = function(){

ga.ctx.save(); // 儲存繪圖設定

let img;

if(ga.keys.space){

img = ga.res.imgs.explosion; // 當按下space時變成爆炸圖!!!

}else{

img = ga.res.imgs.plane;

}

ga.ctx.drawImage(img, this.x - this.size /2, this.y - this.size /2, this.size, this.size);

// 之前smoke的註釋:

//改為畫影像，因為影像要載入時間，所以事先準備!!!! 不可以onload，所有要等地都要事先準備!!! 像是遊戲!!

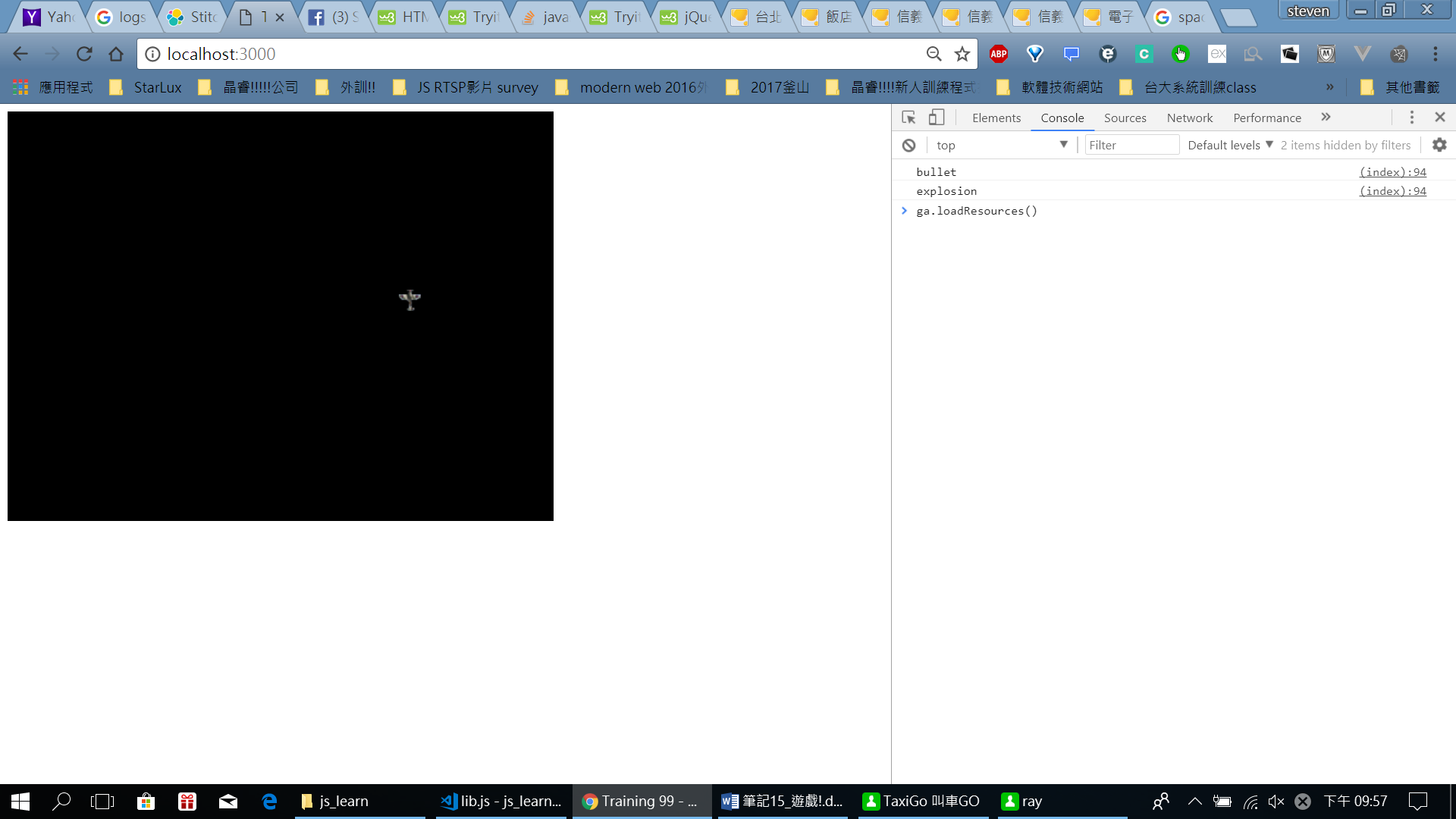
// ctx.drawImage(smoke, this.x-this.size/2, this.y-this.size/2, this.size, this.size); // 尺寸不要固定喔! 因為smoke會擴散

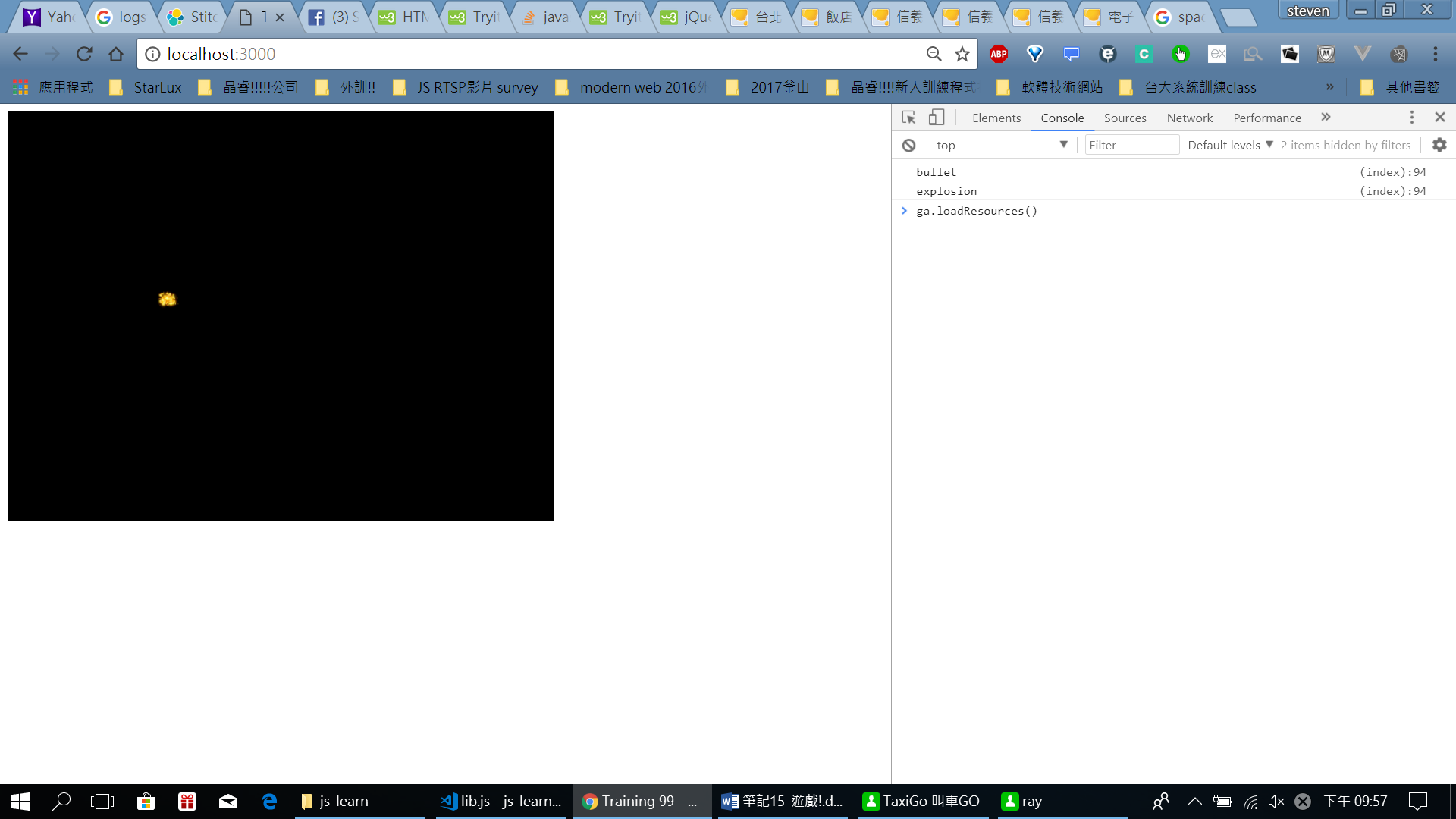
// /2是因為我們一班是用中間思考(smoke圖中央)

ga.ctx.restore(); // 恢復上一個儲存的設定

};

}





再來

1. 設計彈幕系統
   1. 加入回合數的概念 >> 為了不要讓頻率那麼高
   2. 主要邏輯重製前幾周的煙幕系統
2. 設計子彈的位置和運動方式
   1. 起始位置、運動方向
   2. 切分成4種模式(上下，下上、左右、右左)
   3. 畫圖形
3. 等等

如何做碰撞偵測: