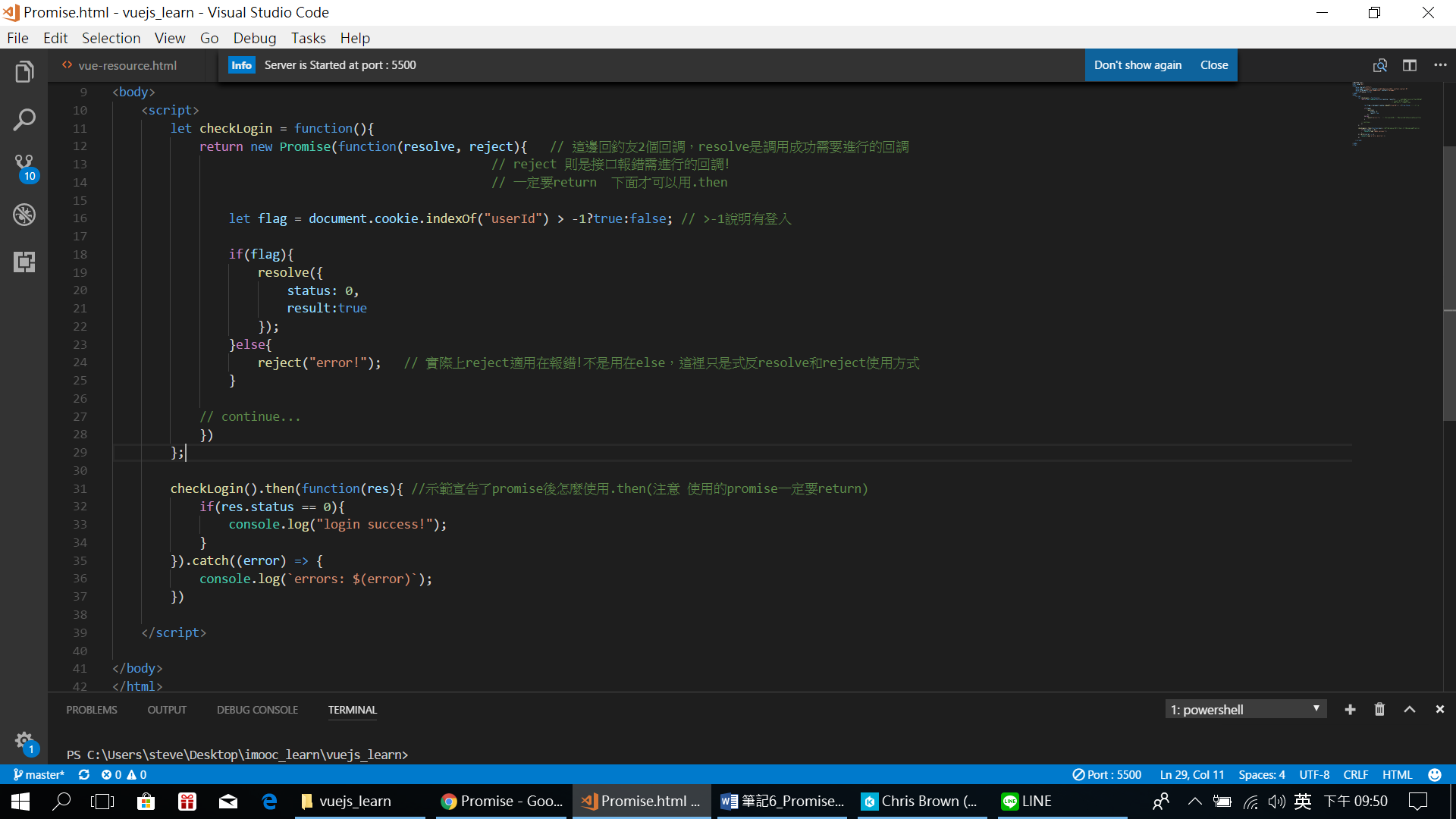
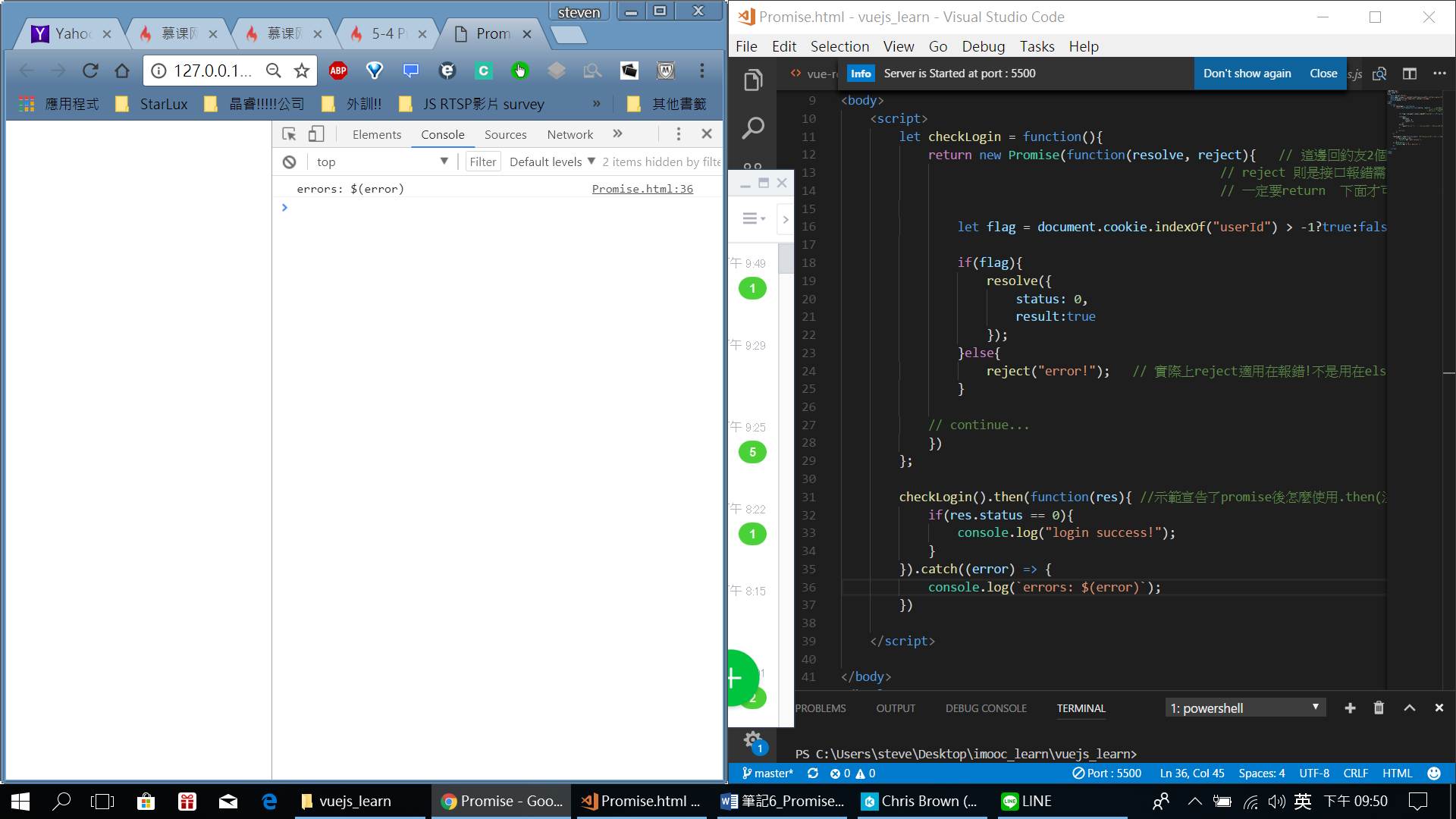
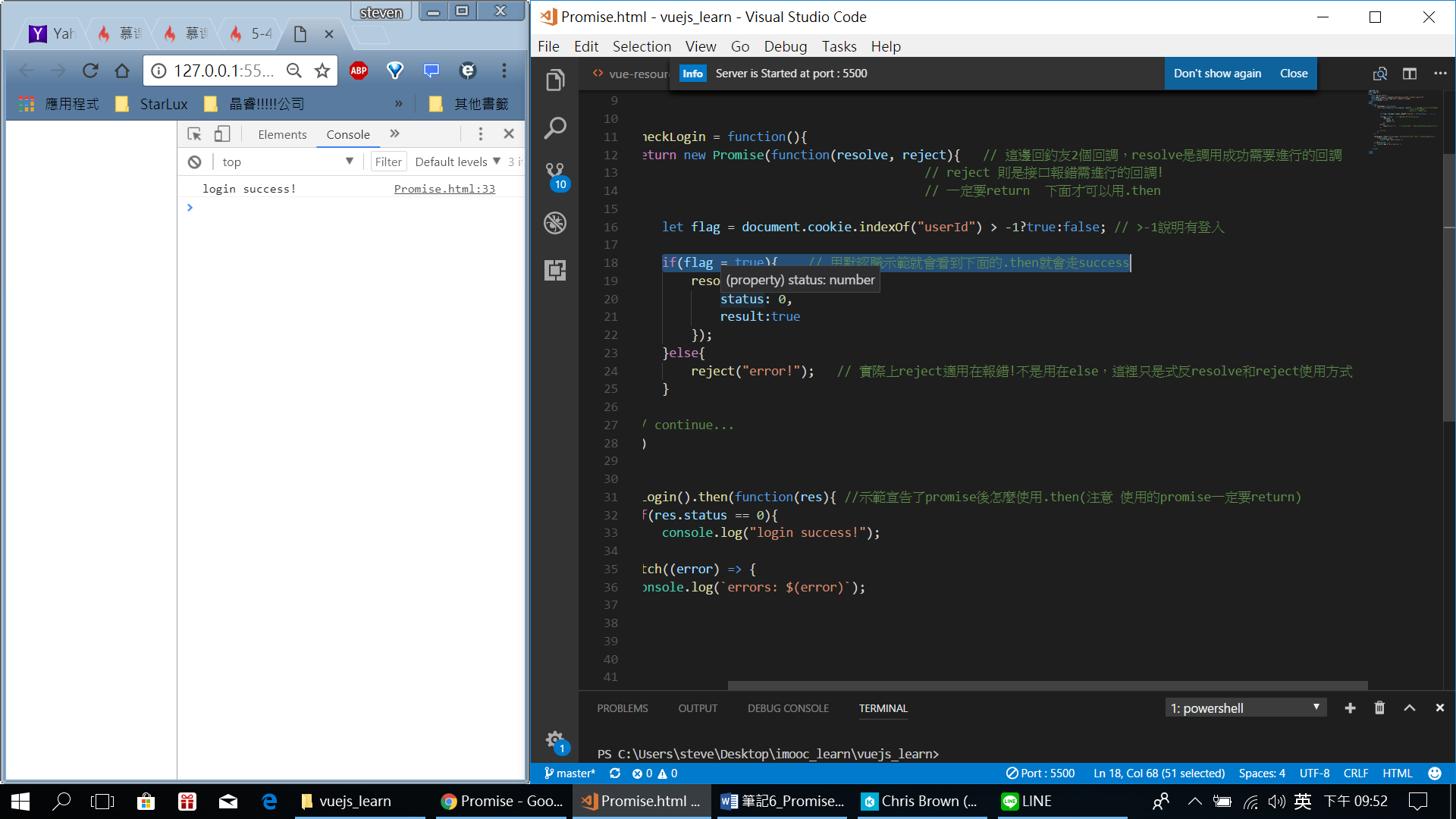
重要 多聽在做筆記!!!!

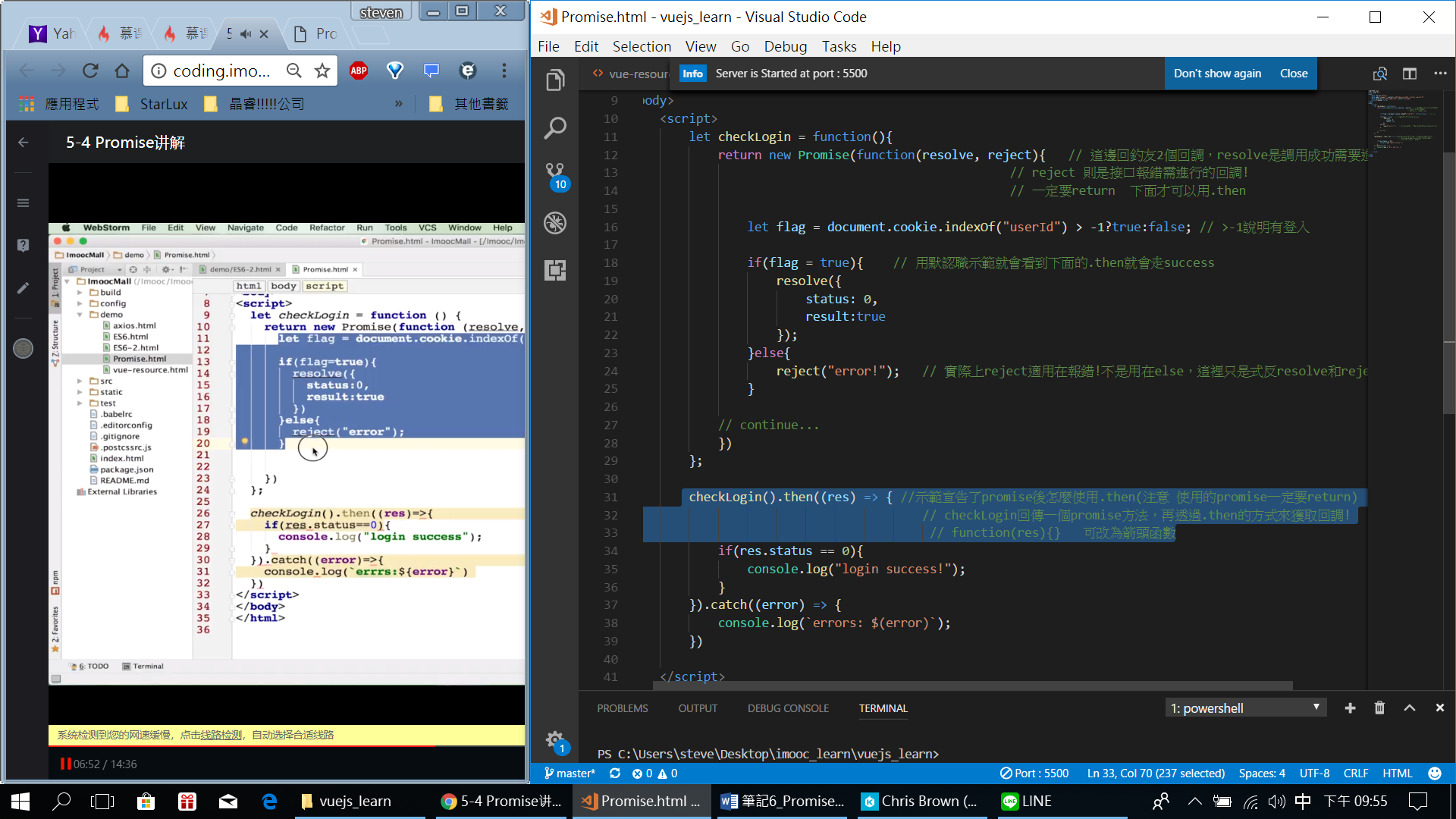




2. 可以看到給默認值示範:



3. 可改為箭頭韓式:



<以後: 登入的邏輯都可以放到.Promise裡面去>想是把登入成功的結果放到resolve 再透過.then的方式來進行獲取

>>>>>>>>>>>

例如我們可以這樣寫:

// 假如我們想演示: 當login成功後，想用.then的方式獲取用戶信息>>

// 所以也加個getUserInfo的promise物件吧!!!

let getUserInfo = () => {

return new Promise((resolve, reject) => {

let userInfo = {

userId: 101

};

resolve(userInfo) // 因為這裡不是真的去串接後台，我們先是者把userInfo寫死在上面XD

});

}

所以:

checkLogin().then((res) => { //示範宣告了promise後怎麼使用.then(注意 使用的promise一定要return)

// checkLogin回傳一個promise方法，再透過.then的方式來獲取回調!

// function(res){} 可改為箭頭函數

if(res.status == 0){

console.log("login success!");

// login 成功就去getUserInfo

return getUserInfo(); // 但因為getUserInfo也是Promise物件，所以下方.catch後面那邊還可以再進行.then!

}

}).catch((error) => {

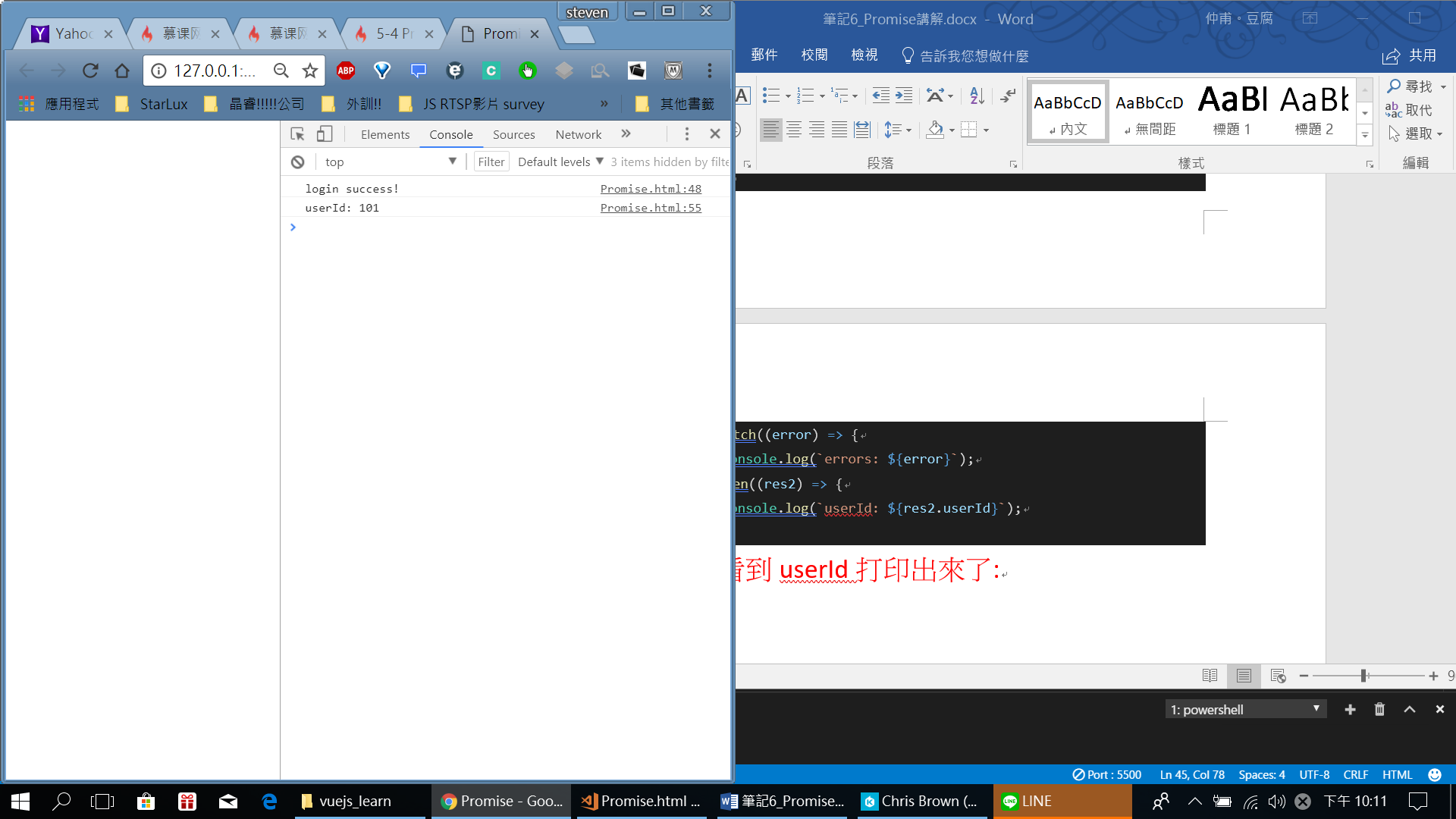
console.log(`errors: ${error}`);

}).then((res2) => {

console.log(`userId: ${res2.userId}`);

})

就可以看到userId打印出來了:



>>>>>>>>>>>>

用這種Promise的方式，的鏈式調用大量減少了以前寫一堆瘋狂的callback代碼!!!!

厲害的是>>>>>> Promise其實還提供了多種API>> 像是 Promise.all 可以同事放入多個promise物件:

// 厲害的是>>>>>> Promise其實還提供了多種API>> 像是 Promise.all 可以同事放入多個promise物件:

Promise.all([checkLogin(), getUserInfo()]).then(([res1, res2]) => { // checkLogin()和 getUserInfo因為都是Promise物件所以才可以放進來喔!!!!

// .then後面因為是接前面兩個的回調，所以也2個都放在[]中

console.log(`result1: ${res1.result}, result2:${res2.userId}`) // res1和res2分別就對應了上面2個promise物件的resolve那邊的值!!!

});

舊可以看到他印出來囉!

