**深入JavaScript程式設計**

<https://paper.dropbox.com/doc/JavaScript-HEMc19Y9Qg7LaTtz4G38J>

**作業**

請使用 github 展示你的 code。

本作業希望不要耗費大家太多的時間，希望可以把小專或大專要做的東西使用 git 和 npm 管理

1. 首先，將原本的專案用 git 管理並上傳到 github
2. 再來，用 npm 管理你的專案並將一些專案開始用 node\_module 引入
3. 使用 GSAP 做個小動畫，同時，可以偷老師的 code 來改～ 別緊張！
4. 希望大家會用 parcel 跟 express ，對不同 url 呈現不同的內容，更為加分！
5. 希望大家可以互相合作，互相討論交流，當然也讓大家互相評分！你可以看到好的作品在 paper 上 comment +1 然後留下你覺得好在什麼地方，還有哪裡要改進，有好留言的分數也會依數量 + 0.5
6. 歡迎大家錄 loom 解說你的作品的原理與用意

|  |  |
| --- | --- |
| 姓名 | 作品連結 |
| 張婕妤 |  |
| 黃姿華 | <https://github.com/Chloe63360/JS-Homework>  <https://codepen.io/chloe63360/pen/VwvjRON> |
| 潘冠竹 |  |
| 林宜錞 |  |
| 姚佳君 | <https://github.com/amy123881/homework20200328> |
| 許文馨 |  |
| 莊雯茜 | [https://github.com/yuma-akane/sukudai](https://github.com/yuma-akane/sukudai202003) |
| 李瑋筑 |  |
| 吳軒毅 |  |
| 林威君 | <https://github.com/WEI003737/project-mid>   <https://codepen.io/Wei003737/pen/NWGqxxJ> |
| 吳佳陵 |  |
| 李元劭 |  |
| 林寄韻 |  |
| 余信和 |  |
| 李胤銘 |  |
| 尤紹安 |  |
| 陳品蓉 |  |
| 毛乙丞 |  |
| 吳佩芫 |  |
| 吳玉虹 |  |
| 謝佳芳 |  |
| 顏秀如 |  |
| 蔡宛錚 |  |
| 郭子瑄 |  |
| 蔡凱婷 |  |

**大綱**

~~Module 1. JavaScript基本語法~~

~~1-1: 基本類型~~

~~1-2: 運算子和運算元~~

~~1-3: 流程控制~~

~~Module 2. 基本集合類型~~

~~2-1: Object~~

~~2-2: Array~~

~~2-3: for/in迴圈~~

~~Module 3. 操作DOM~~

~~3-1: 取得DOM元素~~

~~3-2: 建立DOM元素~~

~~3-3: 刪除DOM元素~~

**Module 1. 深入DOM元素**

**1-1: DOM 屬性操作**

var oHtml = document.documentElement;

var oHead = oHtml.firstChild;

var oBody = oHtml.lastChild;

//JS 用法

var oHead = oHtml.childNodes[0];

var oBody = oHtml.childNodes[1];

//正式從 ChildNode 的提取方法

var oHead = oHtml.childNodes.item(0);

var oBody = oHtml.childNodes.item(1);

.parentNode             //父母元素

.previousSibling      //前兄弟姊妹元素

.nextSibling             //後兄弟姊妹元素

.ownerDocument   //回傳一個此節點所屬的的頂層 document 物件

<https://www.itread01.com/content/1547905895.html>

**1-2: 自訂屬性 data-\***

* data-\* attribute: 能夠添加一些自己需要用到的屬性名稱
* 例子: <https://codepen.io/txs/pen/gOpdNGQ?editors=1111>
* jQuery 的用法:

console.log($('#slider').data('type')); //  "slideShow"

**1-3: 元素與樣式**

* + style 屬性的優先級和通過style設置內聯樣式是一樣的，並且在css層級樣式中擁有最高優先級
  + 一些不同的寫法

// 在单个语句中设置多个样式

elt.style.cssText = "color: blue; border: 1px solid black";

// 或者

elt.setAttribute("style", "color:red; border: 1px solid blue;");

// 设置特定样式，同时保持其他内联样式值不变

elt.style.color = "blue";

* + Inline 樣式

<!DOCTYPE HTML>

<html>

<body style="font-weight:bold;">

    <div style="color:red" id="myElement">..</div>

 </body>

</html>

<https://developer.mozilla.org/zh-CN/docs/Web/API/HTMLElement/style>

**Module 2. DOM元素集合**

**2-1: querySelectorAll與NodeList**

* .querySelectorAll 回傳一個 NodeList
* NodeList 為*動態集合（live collection）但有時為 static 看下一節*

var parent = document.getElementById('parent');

var child\_nodes = parent.childNodes;

console.log(child\_nodes.length); // let's assume "2"

parent.appendChild(document.createElement('div'));

console.log(child\_nodes.length); // should output "3"

**2-2: HTMLCollection**

[HTMLCollection](https://www.w3schools.com/js/js_htmldom_collections.asp)

* *An HTMLCollection object is an array-like list (collection) of HTML elements.*

[NodeList](https://www.w3schools.com/js/js_htmldom_collections.asp)

* *物件是節點的集合*
  + **NodeList.length**
  + **NodeList.forEach()**
  + **NodeList.item(index)**

document.getElementsByTagName('div');

*// HTMLCollection() [...] { some-id: div#some-id.xxx}*

document.querySelectorAll('div')

*// NodeList() [...] <- 只有 List 而已*

// Testing HTMLCollection

var blocks = document.getElementsByTagName('div');

console.log(blocks.length); *// assume output 1*

document.children[0].appendChild(document.createElement('div')) *// 亂加個 div*

console.log(blocks.length); *// will output 2*

// Testing NodeList

var blocks = document.querySelectorAll('div');

console.log(blocks.length); *// assume output 1*

document.children[0].appendChild(document.createElement('div')) *// 亂加個 div*

console.log(blocks.length); *// will output 1*

**2-3: NodeList不是Array**

const nodelist = document.querySelectorAll('div');

const nodelistToArray = Array.apply(null, nodelist);

// 之後 ..

nodelistToArray.forEach(...);

nodelistToArray.map(...);

nodelistToArray.slice(...);

// 等等...

// ES 2015 之後的作法

const nodelist = [...document.querySelectorAll('div')]; // 回傳一個實際的陣列

// 之後 ..

nodelist.forEach(...);

nodelist.map(...);

nodelist.slice(...);

// 等等...

<https://blog.csdn.net/qq_30100043/article/details/53391308>

**Module 3. 函式**

**3-1: 函式定義**

* + 函式的名稱。
  + 包圍在括號()中，並由逗號區隔的一個函式參數列表。
  + 包圍在大括號{}中，用於定義函式功能的一些JavaScript語句。

function square(number) {

  return number \* number;

}

**3-2: 呼叫 call**

let author  = { favoriteNumber: 1}

let andy    = { favoriteNumber: 666}

// console.log(author.favoriteNumber);

function doubleFavoriteNumber() {

    return this.favoriteNumber \* 2;

}

var a = doubleFavoriteNumber.call(author);

let twoNum = doubleFavoriteNumber.call(andy)

console.log(a);

console.log(twoNum);

let author  = { favoriteNumber: 1}

function doubleFavoriteNumber(n) {

    return this.favoriteNumber \* n;

}

var a = doubleFavoriteNumber.call(author, 2);// this: author, n:2

console.log(a);

function add(a, b) {

  return a + b;

}

console.log(add(1, 2));                                                *// 3*

console.log(add.call(null, 1, 2));                        *// 3*

console.log(add.apply(null, [1, 2]));                *// 3*

console.log(add.apply(null, 1, 2));                        *// Uncaught TypeError: CreateListFromArrayLike called on non-object*

<https://ithelp.ithome.com.tw/articles/10195896>

**3-3: 綁定 bind**

* 語法：fn.bind(this, arg1, arg2..., argn)

function add(a, b) {

  return a + b;

}

add.call(null, 1, 2);                        *// 3*

add.call(null, 1, 4);                        *// 5*

add.apply(null, [1, 2]);                *// 3*

add.apply(null, [1, 4]);                        *// 5*

var add1 = add.bind(null, 1);

console.log(add1(2));                        *// 3*

console.log(add1(4));                        *// 5*

**Module 4.  表單**

**4-1: 文字欄**

<https://codepen.io/PJCHENder/pen/zvQXbq>

<https://codepen.io/PJCHENder/pen/zvQXbq>

**4-2&3: 單選項目 & 多選項目**

<https://codepen.io/txs/pen/GRJYKyv>

<https://codepen.io/txs/pen/GRJYKyv>

*  SelectedIndex 不瞭解 要在講一次！ [@txshon t](/ep/profile/16Vtz98QRUunv3Rpqfnl80dI2qYf2678oXaV5sBHjDyvGamk)

**Module 5. 深入表單**

**5-1&2: 檔案上傳**

<https://pjchender.blogspot.com/2019/01/js-javascript-input-file-upload-file.html>

[**https://developer.mozilla.org/zh-TW/docs/Web/JavaScript/Obsolete\_Pages/Obsolete\_Pages/Obsolete\_Pages/例外處理語法/try...catch\_語法**](https://developer.mozilla.org/zh-TW/docs/Web/JavaScript/Obsolete_Pages/Obsolete_Pages/Obsolete_Pages/例外處理語法/try...catch_語法)

base64 image: <https://jsfiddle.net/casiano/Xadvz/>

// STEP 1: select element and register change event

const imagePreview = document.querySelector('[data-target="image-preview"]');

const spinner = document.querySelector('[data-target="spinner"]');

const fileUploader = document.querySelector('[data-target="file-uploader"]');

//加一個 很像 onChange 當他改變 執行 handleFileUpload

fileUploader.addEventListener("change", handleFileUpload);

async function handleFileUpload(e) {

    // try... catch finally 偵錯用

  try {

    const file = e.target.files[0];

    setUploading(true);

    if (!file) return;

    const beforeUploadCheck = await beforeUpload(file);

    if (!beforeUploadCheck.isValid) throw beforeUploadCheck.errorMessages;

    const arrayBuffer = await getArrayBuffer(file);

    const response = await uploadFileAJAX(arrayBuffer);

    alert("File Uploaded Success");

    showPreviewImage(file);

  } catch (error) {

    alert(error);

    console.log("Catch Error: ", error);

  } finally {

    e.target.value = '';  // reset input file

    setUploading(false);

  }

}

// STEP 2: showPreviewImage with createObjectURL

// If you prefer Base64 image, use "FileReader.readAsDataURL"

function showPreviewImage(fileObj) {

  const image = URL.createObjectURL(fileObj);

  imagePreview.src = image;

}

// STEP 3: change file object into ArrayBuffer

function getArrayBuffer(fileObj) {

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

    const reader = new FileReader();

    // Get ArrayBuffer when FileReader on load

    reader.addEventListener("load", () => {

      resolve(reader.result);

    });

    // Get Error when FileReader on error

    reader.addEventListener("error", () => {

      reject("error occurred in getArrayBuffer");

    });

    // read the blob object as ArrayBuffer

    // if you nedd Base64, use reader.readAsDataURL

    reader.readAsArrayBuffer(fileObj);

  });

}

// STEP 4: upload file throguth AJAX

// - use "new Uint8Array()"" to change ArrayBuffer into TypedArray

// - TypedArray is not a truely Array,

//   use "Array.from()" to change it into Array

function uploadFileAJAX(arrayBuffer) {

  // correct it to your own API endpoint

  return fetch("https://jsonplaceholder.typicode.com/posts/", {

    headers: {

      version: 1,

      "content-type": "application/json"

    },

    method: "POST",

    body: JSON.stringify({

      imageId: 1,

      icon: Array.from(new Uint8Array(arrayBuffer))

    })

  })

    .then(res => { //回應

      if (!res.ok) {

        throw res.statusText; //說錯誤！

      }

      return res.json();// 成功回傳 json

    })

    .then(data => data) // 回傳 data

    .catch(err => console.log("err", err)); // 印出錯誤

}

// STEP 5: Create before upload checker if needed

function beforeUpload(fileObject) {

  return new Promise(resolve => {

    const validFileTypes = ["image/jpeg", "image/png"]; //我只吃這兩種格式

    const isValidFileType = validFileTypes.includes(fileObject.type);

    let errorMessages = []; // 空的Array 為了 error 訊息 建的

    if (!isValidFileType) {

      errorMessages.push("You can only upload JPG or PNG file!");// 塞一個錯誤訊息

    }

    const isValidFileSize = fileObject.size / 1024 / 1024 < 2; //小於 2mb

    if (!isValidFileSize) {

      errorMessages.push("Image must smaller than 2MB!"); //塞一個錯誤訊息

    }

    resolve({

      isValid: isValidFileType && isValidFileSize,

      errorMessages: errorMessages.join("\n")

    });

  });

}

function setUploading(isUploading) {

  if (isUploading === true) {

    spinner.classList.add("opacity-1");

  } else {

    spinner.classList.remove("opacity-1");

  }

}

**5-3: JSON 來上傳檔案**

1. 取得使用者上傳檔案：在 HTML 中建立 <input type="file" onChange={handleUpload} /> 來取得使用者上傳的檔案。
2. 得到該檔案的 Blob：在 handleUpload 的 e.target.files 中可以取得該檔案的 **Blob**。
3. 轉成 ArrayBuffer：透過 FilerReader() 來轉成 **ArrayBuffer** 的格式。在 reader.onLoad 的時候，可以透過 reader.result 來取得 ArrayBuffer。
4. 轉成 Uint8Array：接著透過 new Uint8Array() 把這個 ArrayBuffer 轉成陣列，但要特別注意，**轉出來的是「類陣列（TypedArray）」而不是真正的陣列**，因此在送出 AJAX 之前需要先轉成真正的陣列。
5. 轉成真正的陣列：透過 Array.from() 把剛剛的 **Uint8Array** 轉成真正的陣列。
6. 轉成 JSON 格式：如果直接對 Uint8Array 執行 JSON.stringify() 會得到錯誤的結果，記得要先使用 Array.from() 才可以使用 JSON.stringify()

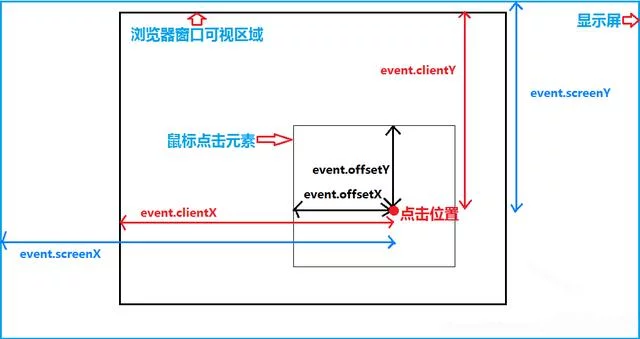
exclamation mark️透過 new Uint8Array() 轉換出來的陣列會是一個「**類陣列（Typed Array）**」，可以透過 Array.from() 等方式轉換成真正的陣列。

**Module 6. 事件處理器 Event Handler**

**~~6-1: 不同形式的事件處理器 onclick onmouseover 的之前講過了~~**

**6-2: 滑鼠事件**

* + **MouseEvent** 介面表示了由使用者經指標裝置（如滑鼠）進行互動所發生的事件。常見的 MouseEvent 實作事件包括了 [click](https://developer.mozilla.org/zh-TW/docs/Web/Reference/Events/click)、dblclick、mouseup 與 mousedown
* **event.pageX、event.pageY**
* 類似於event.clientX、event.clientY，但它們使用的是文檔坐標而非窗口坐標。這2個屬性不是標準屬性，但得到了廣泛支持。IE事件中沒有這2個屬性。
* **event.offsetX、event.offsetY**
* 滑鼠相對於事件源元素（srcElement）的X,Y坐標，只有IE事件有這2個屬性，標準事件沒有對應的屬性。
* **event.screenX、event.screenY**
* **滑鼠相對於用戶顯示器螢幕左上角的X,Y坐標。標準事件和IE事件都定義了這2個屬性**



* 原文網址：<https://kknews.cc/news/r3pzzr.html>
* 額外 React 範例: <https://codtrepen.io/mokargas/pen/LWRMdb>

**6-3: 事件模型**

* + Traditional model - Example 1  :  <https://codepen.io/txs/pen/wvaELLv>
    - **this就會設定為當時觸發事件的元素**
  + Inline model          - Example 2  : <https://codepen.io/txs/pen/KKpxjOg>
    - 不建議這種寫法了！
    - 跟這個同等意思

**document.getElementById('btn1').onclick = function() {**

**handle(this);**

**};**

* + 有些事件觸發時會有預設的動作
    - 表單的submit事件預設會發送表單
    - 超鏈結<a>的click事件預設會連結至指定網頁
  + 取消預設事件動作，可以讓事件處理器傳回false
  + 或在 function 內寫 preventDefault() → Google 一下！
  + Example 3: <https://codepen.io/txs/pen/GRJXVJb>
  + Example 4: <https://codepen.io/txs/pen/oNXPKjQ>

**Module 7. Git 版本控制**

**7-1: 什麼是 Git?**

* + Git 是一個分散式版本控制軟體，由 Linux 的發明人於 2005 創作，最初的目的是為了更好地管理 Linux核心 (Eat your own dog food)
  + 分散式的好處:
    - 大家都有完整資料庫
    - 大家都能獨立工作
    - 大多的運作不必網路，Server 壞了可再找一台機器復原
  + 最大的好處: Github 上有很多好用的 project，可以少寫一些 code，多做一些事！

1. 安裝 Git

* cd : 打開目錄

    cd 目錄名稱      // 打開目錄

    cd ..          // 回到上一層

* ls: 察看目錄內有什麼東西

    ls             // 後面部用加東西 可以直接顯示

* mkdir: 新建資料夾

    mkdir 新增資料夾名稱 //

* rm 刪除檔案

rm 目標檔案        // 刪除檔案

rm -r 目標資料夾    // 刪除目標資料夾

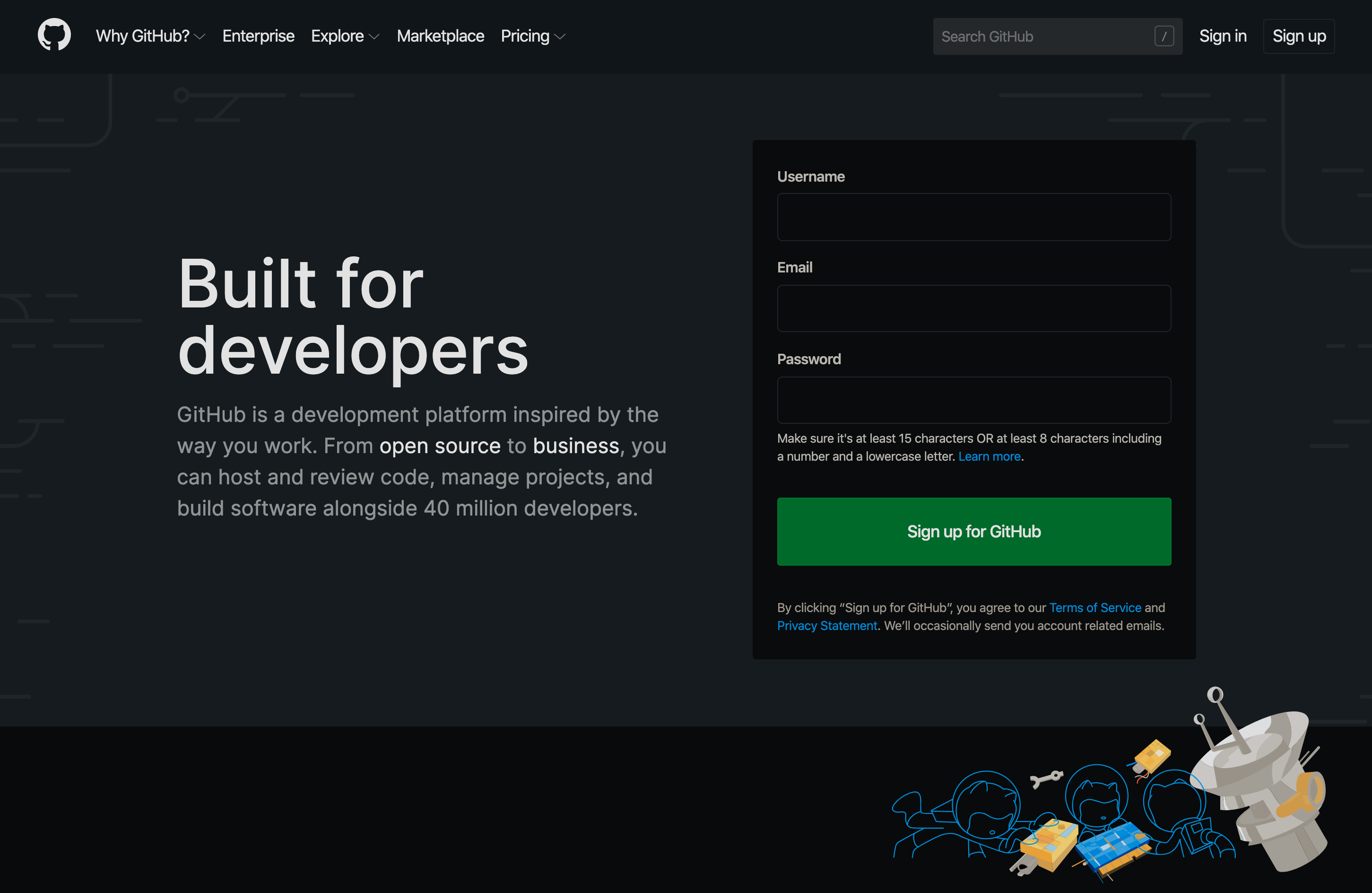
* 打開電腦裡的 git bash

git clone https://github.com/txs/mmmh57.js.git

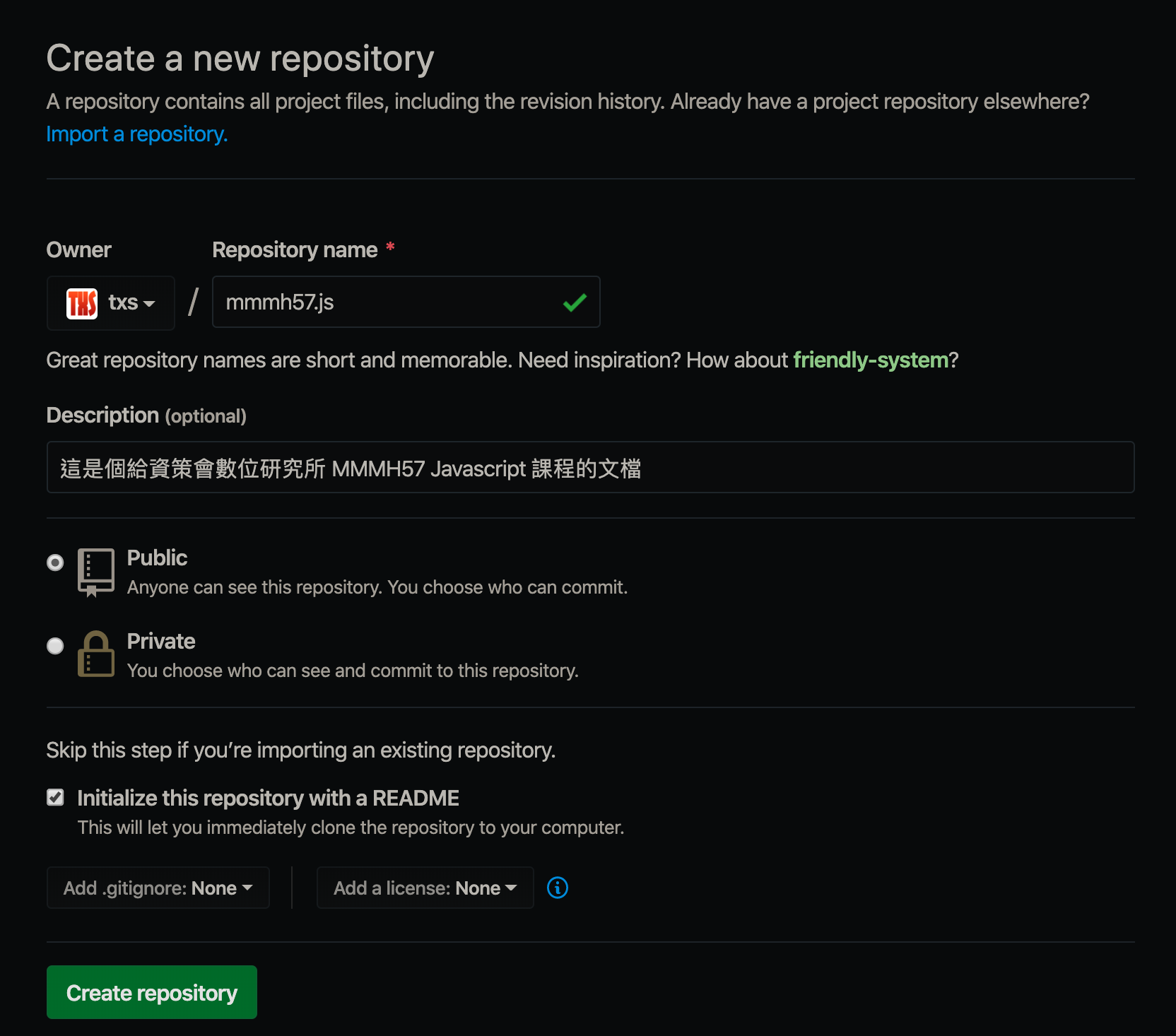
**7-2: 什麼是 Github?**

**GitHub**是透過[Git](https://zh.wikipedia.org/wiki/Git)進行[版本控制](https://zh.wikipedia.org/wiki/%E7%89%88%E6%9C%AC%E6%8E%A7%E5%88%B6)的軟體原始碼代管服務平台

* 建立第一個 Github 帳號 <https://github.com>



* 創建第一個 Github repo



* 檢查電腦是否已經安裝 git，
* 若無安裝在此連結安裝，<https://git-scm.com/downloads>。
* 建立本地的git

1. 到你要建立的目錄，右鍵選擇git bash here
2. 設定你的 email 與 名字駐記

git config --global user.email "you@example.com"

git config --global user.name "Your name"

1. 初始化你的專案

**git init**

1. 將所有的檔案加入 git 的範圍

**git add .**

1. 將加入的檔案變化作標記

**git commit -m "註釋語句"**

1. 將github上的位址加入到 remote 表單中

 git remote add origin \*\*\*git位置

1. 上傳github之前，要先pull一下

**git pull origin master**

1. 上傳程式碼到github，會要你輸入 github 的帳密

**git push -u origin master**

git 推步上去怎麼辦

如果有設定，當下次執行 git push 指令的時候，它就會用來當預設值。舉例來說：

$ git push -u origin master

下一次就可以

$ git push

<https://gitbook.tw/chapters/github/push-to-github.html>

7-3: git 常用語法

1. git init：初始化 git。
2. git status：git 目前版本狀態查詢。
3. git add：git add +檔案，將檔案加入版本控制。例：git add 123，將檔案名稱123加入版本控制。
4. git add .：一次將目前所有檔案都加入版本控制。
5. git commit –m “版本敘述”：不進入vim，直接在 terminal 新增 commit message。例：git commit –m “ first virtion”。
6. git log：查看版本歷史紀錄。每個版本上方會寫commit 後面接一串英數交雜，這個就是版本(編)號。
7. git log --oneline：查看簡單的版本歷史紀錄，此時的版本號會是簡易的前七碼。
8. git checkout：git checkout + 完整的版本號。將版本更改到版本號的那個版本。
9. git checkout master：將版本更改到最新的版本。
10. .gitignore：不想要被版本控制的檔案。

<https://hackmd.io/@Yu040419/SyHrpos6V>

**Module 8. Markdown 文本**

**8-1: 什麼是 Markdown?**

Markdown的特性

* 輕量化
* 易讀易寫
* 圖片，圖表、數學式都有支援
* 目前許多網站都廣泛使用
* 有 HTML 擴增功能

哪些服務用 Markdown

* Jupyter notebook
* GitHub
* Reddit (美國PTT)
* Stack Exchange
* OpenStreetMap
* SourceForge
* 撰寫[電子書](https://zh.wikipedia.org/wiki/%E9%9B%BB%E5%AD%90%E6%9B%B8)

一些老師遇到的缺點

* 沒有左右的排版辦法，想要左邊放圖右邊放字卻需要嵌入 HTML

**8-2: Markdown 語法**

標題 : 一到六個 # 來區表示 H1→H6

# This is an H1

## This is an H2

###### This is an H6

**分隔線:** 一行中用三個或以上的星號、底線、減號來建立一個分隔線

\*\*\*

\_\_\_\_\_\_\_

-------------------

**插入連結**

[Markdown 語法說明](http://markdown.tw/ "Title")

**自動連結**

<http://markdown.tw/>

**無序清單**

\* 星號

+ 加號

- 減號

有序清單

1. 有序

2. 清單

引用

> This is an example by Andy

**待辦事項**

- [ ] 待辦事項

- [x] 已完成事項

**程式碼 用反斜包起來**

s = "Js syntax highlighting"

console.log('Hello Js')

**圖片**

![Alt text](/path/to/img.jpg "Optional title")

**8-3: Markdown 工具**

VS Code 內建即可

Github 可直接改 README.md

Apple 裝置: Bear - notes <https://bear.app>

Chrome: MADO 不過時間有點久了<https://chrome.google.com/webstore/detail/mado/gmmlaihnodfojphcmjeemhaeajaldcdj?hl=zh-TW>

StackEdit: 看起來不錯 <https://stackedit.io>

Markdown 線上工具: <https://dillinger.io>

**Module 9. 網頁測試伺服器**

**9-1: 安裝node和使用http**

安裝 Node.js: <https://nodejs.org/zh-tw/download/>

建立一個 server.js

var http = require('http'); *// 1 - 載入 Node.js 原生模組 http*

var server = http.createServer(function (req, res) {   *// 2 - 建立server*

       res.write('Hello World!');

*// 在此處理 客戶端向 http server 發送過來的 req。*

});

server.listen(3000); *//3 - 進入此網站的監聽 port, 就是 localhost:xxxx 的 xxxx*

console.log('Node.js web server at port 3000 is running..')

**9-2: 掛載不同的 url 頁面**

處理 http request

* + request 物件可以用來得到一些從http server 發過來的請求，例如url、http header 及data。
  + response 物件則送出回應給 Http 送來的請求。
*   [@txshon t](/ep/profile/16Vtz98QRUunv3Rpqfnl80dI2qYf2678oXaV5sBHjDyvGamk) 補上 http req 的補充資料

adServer.js : 不同 url 寫入不同的 html 文本

var http=require('http');

var server=http.createServer(function(req,res){

        if(req.url=='/'){

            res.writeHead(200,{'Content-Type':'text/html'});

            res.write('<html><body>This is Home Page.</body></html>');

            res.end();

        }else if(req.url=='/student'){

            res.writeHead(200,{'Content-Type':'text/html'});

            res.write('<html><body>This is student Page.</body></html>');

            res.end();

        }else if(req.url=='/admin'){

            res.writeHead(200,{'Content-Type':'text/html'});

            res.write('<html><body>This is admin Page.</body></html>');

            res.end();

        }else

            res.end('Invalid Request!');

});

server.listen(3000);

**9-3: 使用 http 掛載 HTML**

fsServer.js

var http = require('http');

var fs = require('fs');

var server = http.createServer(function(req, res){

    fs.readFile('test.html',function (err, data){

        res.writeHead(200, {'Content-Type': 'text/html','Content-Length':data.length});

        res.write(data);

        res.end();

    });

})

server.listen(3000);

Module 11. GSAP

11-1: 什麼是 GSAP

<https://greensock.com/gsap/>

* 古人云: “ 吾好 flash, 但吾往矣～ ”
* GSAP: 用 JS 來寫類 FLASH，有FLASH的方便，JS的效能！

<https://www.youtube.com/watch?v=YqOhQWbouCE&&feature=youtu.be>

<https://youtu.be/YqOhQWbouCE>

如何快速使用

1. 將我的 15-3 的 codepen 範例複製到 jsfiddle。( jsfiddle 有播放鍵run的功能 )
2. 引入 GSAP 的 js <https://cdnjs.cloudflare.com/ajax/libs/gsap/2.0.1/TweenMax.min.js>
3. 在HTML script 迴圈後面加入這段 code

TweenLite.to('.ball',2,{

  x:200

})

* TweenLite是GSAP的主體核心，它創建的基本動畫，是其他類別模塊的基礎。一般都會搭配插件/ CSSPlugin以實現DOM元素的動畫（也就是我們最熟悉的動畫了）。
  + TimelineLite是一個叫做時間軸的動畫容器，它用於對多個動畫進行有序的組織與控制。
  + TimeLineMax是TimelineLite的升級版，在TimelineLite的基礎之上，可以有更高級的組織與控制。
  + TweenMax是GSAP集合包，除前面3個之外，還包括plugins裡的常用插件以及easing裡的緩動函數補充。
* to → 到哪裡 所以是目標 後面會解釋語法
* ‘.ball’ → 這是 css selector
* 2 → 秒數
* obj → 內含 x:200 往左移動 200px

<https://jsfiddle.net/txshon/xrqmt1ck/embedded>

<https://jsfiddle.net/txshon/xrqmt1ck/embedded>

**11-2** TweenLite 的基本動畫

* 首先，我們從純 JS 角度來看 他如何來改值

<https://jsfiddle.net/txshon/g0k2mysL/embedded>

* <https://jsfiddle.net/txshon/g0k2mysL/embedded>

再來，我們來做基礎的繪圖

<https://jsfiddle.net/txshon/xtpw0f9e/embedded>

<https://jsfiddle.net/txshon/xtpw0f9e/embedded>

GSAP用x、y表示transform的translateX和translateY。 TweenLite.to(target, duration, vars)的第一個參數target可以是選擇符，因此這裡就是選取id為ball1的元素，執行時長為2s的動畫，從當前位置移動到translateX(200px)的位置。

相同效果

TweenMax.to(obj, 1, {x:100});

var myTween = new TweenMax(obj, 1, {x:100})

var myTween = TweenMax.to(obj, 1, {x:100});

多個物件共同進行動畫

TweenMax.to([obj1, obj2, obj3], 1, {x:100});

多物件程式繪圖

<https://jsfiddle.net/txshon/5qybmegd/embedded>

<https://jsfiddle.net/txshon/5qybmegd/embedded>

ref: <https://www.tweenmax.com.cn/api/tweenmax/TweenMax.to(>)

邊跑邊旋轉

<https://jsfiddle.net/txshon/huomcg4x/embedded>

<https://jsfiddle.net/txshon/huomcg4x/embedded>

重複

* 無限重複， repeat 值為 -1

<https://greensock.com/docs/v2/TweenMax/repeat()>

<https://jsfiddle.net/txshon/f4r3qtav/embedded>

<https://jsfiddle.net/txshon/f4r3qtav/embedded>

**11-3: GSAP delay 等待 & ease 線性**

Delay: 等待

<https://jsfiddle.net/txshon/kmvzec6p/embedded>

<https://jsfiddle.net/txshon/kmvzec6p/embedded>

Ease: 調整線性

<https://jsfiddle.net/txshon/dn9j4qzy/embedded>

<https://jsfiddle.net/txshon/dn9j4qzy/embedded>

Ease 可視化工具: <https://greensock.com/docs/v2/Easing>

**Module 12. GSAP 進階**

**12-1: GSAP to, from, fromTo 語法**

* to 從開始到最後 輸入目標條件
* from 從最後到開始 輸入原始條件
* fromTo: 輸入起始條件與目標條件

TweenLite.fromTo(target, duration, fromVars, toVars)

* delay, ease 只能放在 toVars

from 範例

<https://jsfiddle.net/txshon/gv16p5zc/embedded>

<https://jsfiddle.net/txshon/gv16p5zc/embedded>

fromTo Example

<https://jsfiddle.net/txshon/an437huv/embedded>

<https://jsfiddle.net/txshon/an437huv/embedded>

FromTo lock center

<https://jsfiddle.net/txshon/64b2zpke/embedded>

<https://jsfiddle.net/txshon/64b2zpke/embedded>

**12-2: GSAP TimeLineLite() 時間軸**

var tl = new TimelineLite();

tl.from("#ball1", 1, {

    y: "-=60px",

    autoAlpha: 0

}).from("#ball2", 1, {

    x: "+=60px",

    autoAlpha: 0

}).from("#ball3", 1, {

    y: "+=60px",

    autoAlpha: 0

}).from("#ball4", 1, {

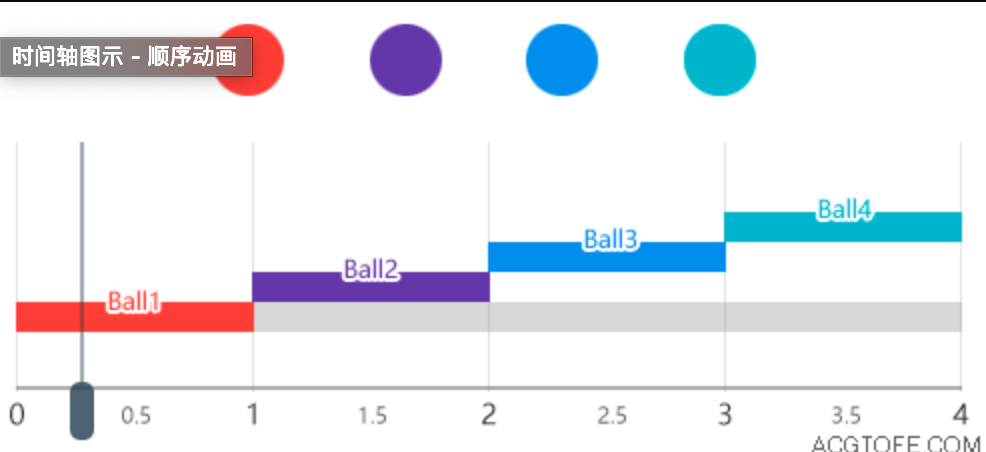
    x: "-=60px",

    autoAlpha: 0

});

<https://jsfiddle.net/txshon/e4zgnh73/embedded>

<https://jsfiddle.net/txshon/e4zgnh73/embedded>



那如果我們不要一個結束後才做第二個

tl.from(target, duration, vars, position)

position 是播放的時間的位置

var tl = new TimelineLite();

tl.from("#ball1", 1, {

    y: "-=60px",

    autoAlpha: 0

}).from("#ball2", 1, {

    x: "+=60px",

    autoAlpha: 0

}, "-=0.7").from("#ball3", 1, {

    y: "+=60px",

    autoAlpha: 0

}, "-=0.7").from("#ball4", 1, {

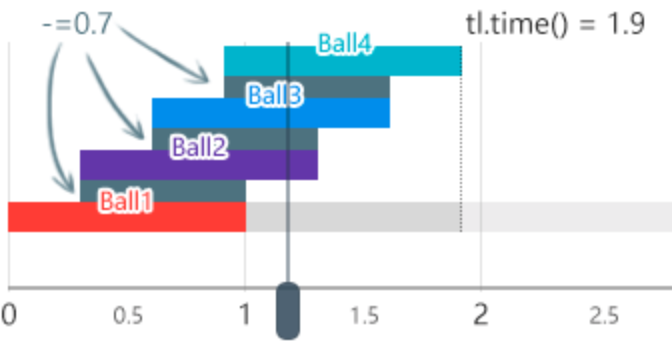
    x: "-=60px",

    autoAlpha: 0

}, "-=0.7");

<https://jsfiddle.net/txshon/78d10L9s/embedded>

<https://jsfiddle.net/txshon/78d10L9s/embedded>



**12-3: GSAP stragger 相同動畫重複**

<https://jsfiddle.net/txshon/h5rsgbek/embedded>

it[https://jsfiddle.net/txshon/h5rsgbek/embedded](https://jsfiddle.net/txshon/h5rsgbek/embedded" \t "_blank)

**Module 10. 使用npm&yarn**

**10-1: 使用npm建立專案**

* 當我們要開始一個新專案時，npm init 來起始你的專案，全部都是使用預設的參數 -y

npm init

npm init -y

設定作者資訊

$ npm config set init.author.name [YOUR NAME]

$ npm config set init.author.email [YOUR EMAIL]

**10-2: 安裝 yarn**

[Yarn](https://yarnpkg.com/zh-Hans/) 是一個新的 package 管理器，它替代了 npm 客戶機或其他包管理器的現有工作流，同時保持與 npm 註冊表的兼容性。它具有與現有工作流程相同的功能集，同時運行速度更快，更安全，更可靠，簡單來說用 Yarn 來安裝套件比起 npm 更來得快速！

Yarn 是 Facebook 自家團隊與 Exponent、 Google、Tilde 所合作開發的套件管理工具

安裝快很多 穩定性好 不易出錯 安全

* macOS用戶

brew install yarn

* Windows用戶
* 直接利用 npm 就能裝起來囉

npm install yarn *--g*

安裝結束後來測試是否安裝成功！

yarn *--version*

|  |  |  |
| --- | --- | --- |
| npm install | yarn install | 安裝 json.package 所有依賴 |
| npm install [package] | (N/A) | Yarn不支援直接安裝套件 |
| npm install --save [package] | yarn add [paakage] | 儲存在 json.package中的dependencies |
| npm install --global [package] | yarn global add [package] | 安裝在電腦全域中 |
| npm uninstall --save [package] | yarn remove [package] | 移除dependencies某套件 |
| rm -rf node\_modules && npm install | yarn upgrade | 更新node\_modules |

**10-3: npm 好用的套件**

<https://www.npmjs.com/package/bootstrap>

<https://www.npmjs.com/package/moment>

<https://www.npmjs.com/package/express>

Express 是個熱門的 node.js 後端，他讓複雜的 node.js API 變的簡單使用

const express **=** require('express')

const app **=** express()

app.get('/', function (req, res) {

  res.send('Hello World')

})

app.listen(3000)

<https://www.npmjs.com/package/request>

Request 是一個非常好用異步單類 API 的整合工具

**import** request **from** "request";

request("http://www.google.com", (error, response, body) => {

  console.log("error:", error);

  console.log("statusCode:", response && response.statusCode);

  console.log("body:", body);

});

<https://www.npmjs.com/package/lodash>

Lodash 一個 js 的工具 能夠有很多特殊的 js 資料處理語法

**import** \_ **from** "lodash";

\_.defaults({ a: 1 }, { a: 3, b: 2 }); *// { 'a': 1, 'b': 2 }*

\_.partition([1, 2, 3, 4], n => n % 2); *// [[1, 3], [2, 4]]*

<https://www.npmjs.com/package/@ionic/core>

<https://www.npmjs.com/package/font-awesome>

<https://www.npmjs.com/package/jquery>

<https://medium.com/html-test/>

**Module 13. 好用的專案工具**

**13-1: Marktext**

* 好用的 Markdown 工具，解決想寫個文檔卻要一直查語法的問題
* 首先，我們先去 Mark Text 官網下載。
* <https://marktext.app>
* 安裝好後，我們用他來編輯我們的 專案的 README.md
* 他可以幾乎不需要知道語法的方式 處理表格 超連結 與標題轉換大小

**13-2: 什麼是 Parcel?**

<https://parceljs.org>

* 超快幾乎不用設定的一個打包工具
* 相對來說另外一個大家常用的 Webpack 相對的複雜切不易設定！
* 超快？有多快？

**Benchmarks**

|  |  |
| --- | --- |
| Bundler | Time |
| browserify | 22.98s |
| webpack | 20.71s |
| **parcel** | **9.98s** |
| **parcel - with cache** | **2.64s** |

* 自動幫你處理 Babel, PostCSS, and PostHTML，如果之後你要自己處理也是可以。
* Hot Module 當東西更新了，他會自動幫你更新頁面
* 自動 Code splitting，讓你只載入你需要的東西

<https://parceljs.org>

Material Design: <https://material.io/design/introduction/>

Bootstrap Parcel: <https://github.com/yasminzy/custom-bootstrap>

**13-3: 安裝 Parcel**

<https://parceljs.org/getting_started.html>

Yarn:

yarn global add parcel-bundler

yarn init -y

查看 index.html

<html>

<body>

  <script src="./index.js"></script>

</body>

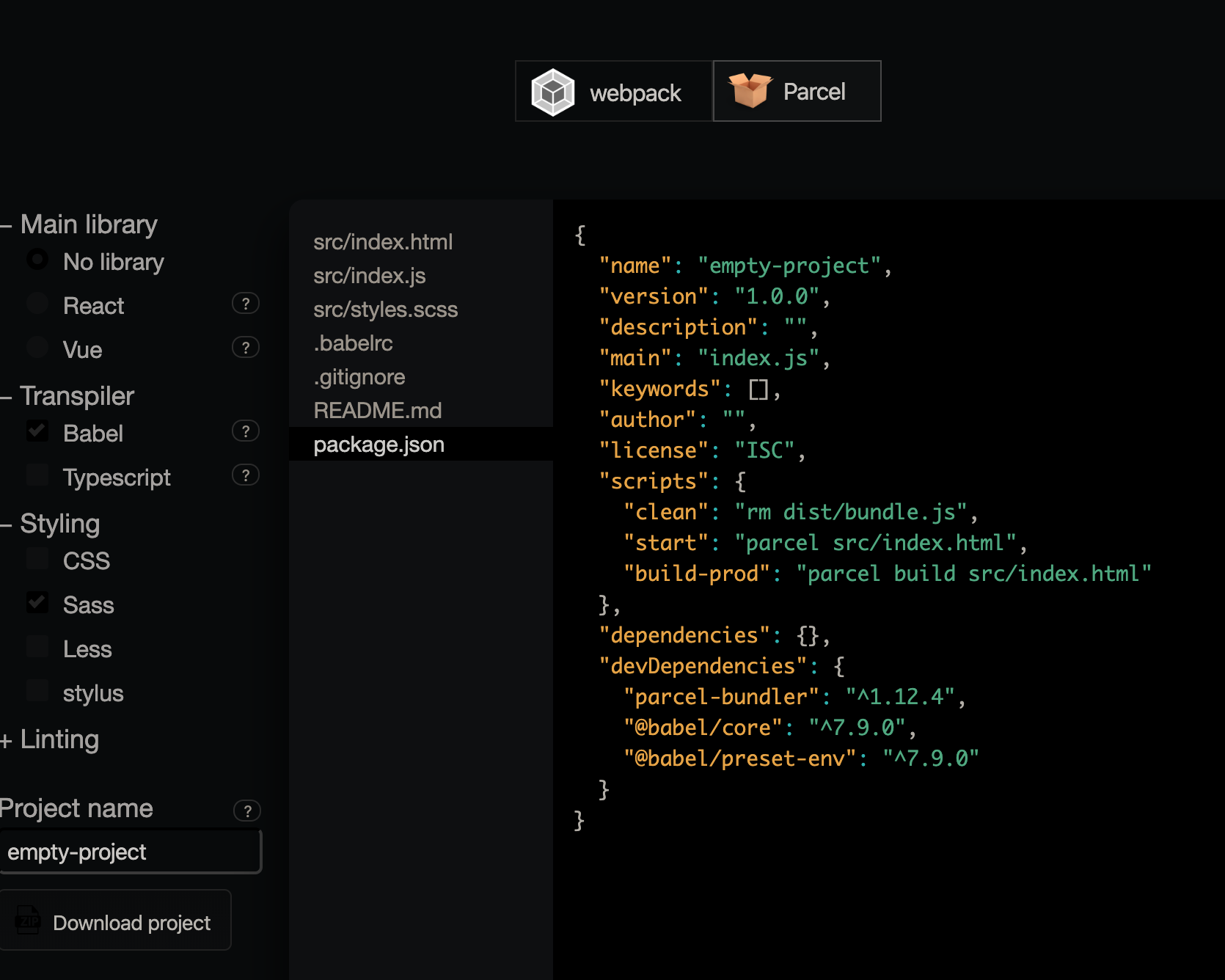
</html>

他的 js 會掛載在 index.js 下

**直接使用** [**https://createapp.dev/parcel**](https://createapp.dev/parcel) **來建立你要的包**

這邊我們選

* Main library
  + No
* Transpiler
  + Babel (ES6)
* Styling
  + Sass



將下載檔案內的zip 解壓即可使用

但我們已經設定好一個專案 我們把 package.json 的 scripts 與 dependecies 複製貼好

再來 把 src 底下的檔案放到我們的專案 同時刪除自創的 index.html index.js

yarn install

再來 src/index.html 內的 bundle.js 因為找不到路徑 需要改成 index.js

package.json 內有 scripts

當我們要跑來測試網站時，我們在 VS code 的 terminal 跑 npm start! → demo

* 我們可已 ctrl 再對連結按一下 來到我們的網頁看一下
* 目前是沒有東西的
* 打開 console 我們可以看到 hello world 代表成功
* npm start 自動會幫你把東西打包到 dist 而且可已 hard reload
* 現在這個網站有 hard reload 可以改動 html 內的東西 存檔 將自動刷新

當我們要把網站部署時，我們先刪掉 dist  跑 build-prod

yarn run build-prod

打包成最小 且做優化的方式 直接丟掉伺服器就可以用了

**有趣的術語**

* <https://zh.wikipedia.org/zh-tw/吃自己的狗粮>



**Module 14. Express**

**14-1: 什麼是 Express?**

* Express 是最小又靈活的 Node.js Web 應用程式架構，具有大量的 HTTP 公用程式方法與中介軟體供您支配，能夠快速又輕鬆的建立完整的 API。

MERN → MongoDB Express React Node.js

* 安裝 Express

yarn add express

yarn global add express-generator

14-2: 寫出第一支 Express 網站訊息

const express = require('express');

const app = express(); //建立一個Express伺服器

app.listen(3000); //告訴server聽取3000這個Port

const express = require('express');

const app = express();

app.get('/', **function** (req, res) {

  res.send('Express is excellent!')

});

app.listen(3000, function(){

  console.log('Example app is running on port 3000!');

});

14-3:  Express 判斷 URL 顯示不同內容

app.get('/', **function**(req, res) {

    res.send('Send root!')

});

app.get('/about', **function**(req, res) {

    res.send('Send about page!')

});

app.get('/about/about-me', **function**(req, res) {

    res.send('Send about-me page!')

});

從 req.params 拿參數

app.get('/post/:id', **function**(req, res) {

  res.send(req.params.id);}

);

app.get('/about/:name/:nickname', **function**(req, res) {

  res.send(req.params.name + ' is so ' + req.params.nickname);

});

**在 Express 中提供靜態檔案**

如果要提供影像、CSS 檔案和 JavaScript 檔案等之類的靜態檔案，請使用 Express 中的 express.static 內建中介軟體函數。

將含有靜態資產的目錄名稱傳遞給 express.static 中介軟體函數，就能直接開始提供檔案。舉例來說，使用下列程式碼在名稱是 public 的目錄中，提供影像、CSS 檔案和 JavaScript 檔案：

app.use(express.static('public'));

現在，您可以載入位於 public 目錄中的檔案：

http://localhost:3000/images/kitten.jpg

http://localhost:3000/css/style.css

http://localhost:3000/js/app.js

http://localhost:3000/images/bg.png

http://localhost:3000/hello.html

而我們可以把 static 的 dist 的靜態檔案來顯示

<https://github.com/witchard/tutorial-react-parcel-express>

**14-3-1: 如何把 html 掛上去無.html 檔名**

首先，安裝 path 套件

yarn add path

const path  = require('path')

app.get('/about', function(req, res) {

    let fileLocation = path.join(\_\_dirname, '../dist', 'about.html')

    res.sendFile(fileLocation);

})

這個需要用到 path package

npm install --save path

app.use(express.static(path.join(\_\_dirname, 'public'),{index:false,extensions:['html']}));

<https://stackoverflow.com/questions/16895047/any-way-to-serve-static-html-files-from-express-without-the-extension>

<link rel="stylesheet" href="node\_modules/bootstrap/dist/css/bootstrap.min.css">

<https://www.npmjs.com/package/@openfonts/noto-sans-tc_chinese-traditional>

Cindy Hsu

Q: 如果想把網頁公開讓其他人看到，要怎麼做？例如診所的網站想公開放網路上

你會需要以下幾個東西

1. 一台 server 例如 google cloud platform, linode or digitalocean

* 建議用 google cloud 原因是因為以上平台不會速度差太多，但以台灣的位置來說，google 的伺服器在彰化彰濱工業區，連線上會比較快。再來，Google 是雲服務，當你的服務越來越大，或甚至只是你的中醫店爆紅，其他的 hosting 要自己做server擴增， Google 有 api 當流量大，可以幫你開很多台機器，讓看網頁不卡 ( 你只要付錢就好 )。

1. Domain 網域 指向正確的 ip 與 port 讓大家可以找到你的網址，並記住你的網址！

**影片**

1-1: DOM 屬性操作    -  <https://youtu.be/qmf92_xwBaY>

1-2: 自訂屬性 data-\*  -  <https://youtu.be/GnKlUBJkuoc>

1-3: 元素與樣式          -  <https://youtu.be/fhE-kXdHsXo>

2-1: querySelectorAll與NodeList  - <https://youtu.be/fCysFLhU4tM>

2-2: HTMLCollection - 錄影中遺失 需要重錄

2-3: NodeList不是Array - <https://youtu.be/P4w4Jl7LXX0>

2-3-1: NodeList to Array 轉換實作 - <https://youtu.be/2UtNXl_f2ek>

3-1: 函式定義   -  <https://youtu.be/6tpKBnCxmxU>

3-2: 呼叫 call  -  <https://youtu.be/mDmXSjnzfPc>

3-3: 綁定 bind - <https://youtu.be/qrU6JbyO124>

4-1: 文字欄 - <https://youtu.be/lBGs9WowoWg>

4-2&3: 單選項目 & 多選項目 - <https://youtu.be/vLvmaCeKkdU>

5-1: 檔案上傳 - <https://youtu.be/w4L4Vx_Rufc>

5-2: 檔案上傳解釋 - <https://youtu.be/ZEt0-4IwxIs>

6-1&2: 滑鼠事件 - <https://youtu.be/VKyexXhwUQc>

6-3: 事件模型 - <https://youtu.be/crJQzPzFeZk>

7-1: 什麼是 Git? - 沒錄到

7-1-1: Git 在 windows 上與 Bash 講解 - <https://www.loom.com/share/9683532d04f44a84895559bb803550c8>

防疫說明-週二再來上課-遠端教學 - <https://www.loom.com/share/28e48f9ed9864f88853610399b9e656b>

7-2: 什麼是 Github? - <https://www.loom.coㄋm/share/3d074d0a08574f0e9c7a2837f0eea498>

8-1 - <https://www.loom.com/share/c592e001f2e74ad9a2189b73dd71c4e9>

8-2 - <https://www.loom.com/share/56792e2be91b483d8c847a1b26621604>

8-3 - <https://www.loom.com/share/0b6bb34f76d2448a9bd83441bca25d59>

9-1 - <https://youtu.be/7CmAdvdyTdo>

9-1-1: Node http server 之如何在 VS code, terminal 與 localhost 間切換 - <https://youtu.be/hDQR5vpOxrk>

9-2: 掛載不同的 url 頁面文本 - <https://youtu.be/1kX9VSMav9Y>

9-3

10-1: 使用npm建立專案 - <https://youtu.be/A3RfdMwpJes>

10-2: 安裝 yarn - <https://youtu.be/K0ZJ4PUDcGA>

10-3: npm 好用的套件 - <https://youtu.be/Czf9Wv9j4J0>

13-1: Marktext 好用的 Markdown 編輯器 - <https://youtu.be/546IneFZGho>

13-2: 什麼是 Parcel 打包器? - <https://youtu.be/xqRxXL8irmg>

13-2-1: 什麼是 Material Design? - <https://youtu.be/utHk0Ip9xAk>

13-3: 安裝 Parcel 打包器 - <https://youtu.be/H9v_855juYw>

9-r: 網頁測試伺服器 - <https://youtu.be/P_O1Us4Uezo>

14-1: 什麼是 Express? - <https://youtu.be/cXEfbg5feQ4>

14-2: 寫出第一支 Express 網站訊息 - <https://youtu.be/XtoSh1OK924>

14-3: Express 判斷 URL 顯示不同內容 - <https://youtu.be/SBFqES0NKYY>

14-3-1: 如何把 html 掛上去無 .html 檔名 - <https://youtu.be/BXGc1Dl5S7g>