目錄

您好，我是這本手冊的作者，廖洧杰，同時也是一名前端工程師，爲了讓想成爲前端的朋友能夠更加瞭解職涯生態，於是我撰寫了這本超過七萬字的前端工程師養成手冊，裡頭包含了我個人在業界上的酸甜苦辣，以及這些年我輔導超過兩百位從零基礎到成功獲得前端工程師工作的經驗談，希望它能伴隨著你一同成長，讓自己如願成爲前端工程師。(閱讀更多)

2018/1/1 廖洧杰

六角學院

時勢造英雄，奠定前端工程師的未來發展

來到這裡，我相信你已經對前端工程師的工作內容有些輪廓，這裡我將會順便講解前端工程師的由來，以及主要負責的工作項目有哪些讓你知曉。

歷史淵源

在以前根本沒有分所謂的前端與後端，所有內容都是一個工程師包辦全部，或是設計師兼網頁排版，也就是網頁設計師。那爲什麼會多了前端工程師這個名詞呢？原因是瀏覽器逐漸進步的緣故，十幾年只要你會寫一個 HTML，就算介面超醜沒 CSS 也可從中賺取非常好的收入，隨著瀏覽器能夠有很多嶄新功能，也讓網頁技術也跟著進步，以致於工程師的技術水平要求也水漲船高，舉例來說：

- CSS：設計的東西也開始講求美感，需運用 CSS 還原設計稿樣式
- AJAX：在 2007~2008 年，AJAX 技術帶來嶄新的發展，讓使用者能夠不換頁便能更新網頁內容
- 使用者體驗：電子商務也在同時間盛行，所做的介面也需要講求使用體驗，好讓使用者能夠經由網站引導達成他們想做的事情
- 瀏覽器：各個瀏覽器如雨後春筍般出現，初期只能用「災難」來形容，毫無共通標準可言，一個 HTML+CSS 排版各自表述
- 網頁效能：不只能讓網頁動，同時也會考量效能進行優化

  從上面的細節你可以依稀看出前端工程師的職缺爲何會脫穎而出，因爲要得已經不是一個會動的網頁，而是具備優良使用者體驗的網頁介面了。

  同時也需兼備左右腦的運作，美感與邏輯也需要一定的程度，才能夠將設計師提供的設計稿忠實地在網頁上呈現，我們甚至會說 1px 也不差地還原到前端介面上便是如此。

前端工程師職缺開始流行的重大關鍵

而眞正開始奠定前端工程師的蓬勃發展，我個人認爲是在 2007~2012 開始有了飛躍性的發展，原因如下：

智慧型手機盛行

第一代 iPhone 在 2007 年問世，讓大家對手機有了足夠的信心，但也造就了許多開發上的難度，例如說每個手機系統上的瀏覽器非常多元，在 2012 年時，Android 系統竟多達 4000 種型號，也伴隨著螢幕解析度的多樣性，小則 240px，大則 1920px，這對沒有前端工程師的軟

體公司來說無疑是個頭痛的問題，因爲光程式商業邏輯就已經無暇應付，更何況還需要處理瀏覽器 client 端上的功能是否能運作正常。

響應式網頁設計

2010年 Ethan Marcotte 發表了 Responsive Web Design (響應式) 設計，以 CSS3 Media Queries 的設計核心以解決網頁解析度問題，也打破了大家都認爲手機版網頁只能是單欄式排版，而是能夠自適應螢幕解析度適時呈現單、雙、三欄式設計方式

綜上所述，軟體公司也開始發現，前端介面無形間增加不少 Loading，原本全包的工程師光是資料庫商業邏輯就忙得要死，接下來又還得確保各載具上瀏覽器跟解析度問題而忙上加忙，而這個工作你也沒辦法交由設計出身的 UI 來負責，於是乎「前端工程師」的職稱也依時勢推出。

前端工程師的主要工作內容

現階段大家都會提到只要是前端就一定要會 XX 語言，要會那個框架、要會什麼工具，但就以我的觀點來說，前端工程師的工作項目應該是最瞭解「網頁瀏覽器」的工程師。

你應當成爲瀏覽器的先鋒，當 w3c 出了任何的草案、ECMAScript 出了哪些規範時，瀏覽器也會適版本號依序地納入新語法，在大家還在用舊語法實作功能，你早能領先一步看出該瀏覽器技術預期將會成熟，並納入未來專案導入項目中，讓自己的公司有足夠的技術能量，成爲業界的領頭羊。當瀏覽器顯示介面有問題時，你能透過內建的開發除錯工具找出緣由並加以解決，不論是預期外的 BUG 或是效能問題都是你該挽起袖子解決的問題。

所以在實作一個前端介面時，至少你該先瞭解網站客群特性，例如完全沒有在用 IE，以 Chrome 爲主，甚至到 can i use 觀察語法支援性。如果你在不瞭解瀏覽器需求的情況下便貿然進行開發，我可以說

你是個不及格的前端工程師

。我時常遇到許多朋友都吃過不少這樣的虧，當客户對瀏覽器一知半解時，也沒有訂下瀏覽器支援性的合約，以致於開發結束後才發現客户族群都是用特定瀏覽器，那剛好你用的語法都是最新最潮的 beta 版語法，只有少部分主流瀏覽器有支援，以致於花了更多時間在重構，那就非常不好了。

已往多方領域發展，不局限於網頁介面

很多人都會認爲前端工程師只是負責「網頁排版」，但已現在的趨勢來說，前端工程師能夠發展的空間已經多到你沒辦法想像，這有助於歸功在 JavaScript 身上，現在的前端可以用 JavaScript 幹很多事情，例如說：

- electronjs：寫一個跨平台的桌面應用程式
- Node.js：跨足後端，或倚賴核心開發各種應用程式
- React Native：設計雙平台 APP
- pixijs：透過 WebGL 開發動畫核心，跨足 3D 世界
- A-Frame：將 3D 介面以 VR 模式進行運作

以上隨便列舉的項目，都足夠你投入個好一陣子，讓自己能夠成爲前端特定領域的翹楚，所以當你在向他人講解前端時，應該知道如何介紹前端的廣度了吧？

最後再補充各類前端介紹文章，讓你對前端工程師有著更深入的瞭解

- 爲什麼我們找不到前端工程師
- 我要成爲前端工程師
- 2017年前端工程師技能樹

設計師投入前端的不二法門

在前端職涯中，我常遇到許多視覺科系背景的設計師來詢問投入前端領域的可能性，我可以打包票和你說是百分之兩百適合，但礙於許多人會對所謂的「前端工程師」、「前端設計師」、「網頁設計師」的工作內容感到黑人問號，因爲有些設計師本身也會一些網頁排版，那麼到底要具備哪些能力，才可以稱得上自己有投入到「前端」呢？

前端工程師與前端設計師的差異

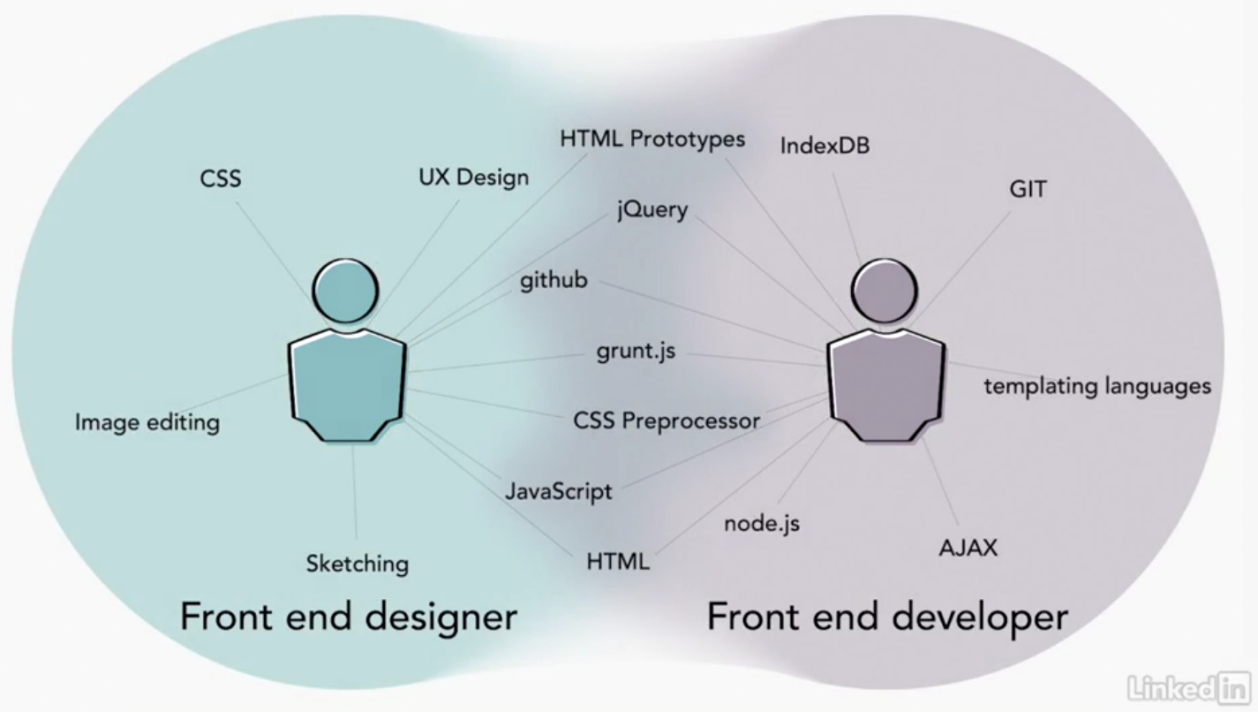
在講解前端之前，先附上一張圖，左邊是前端設計師，右邊是前端工程師。前端設計師會比較著墨在 UI/UX 設計並轉爲網頁格式，他們能夠確保所設計出來的 UI 能夠在網頁上忠實呈現。

其實這段是比前端工程師還要來得有優勢的，就比方說前端工程師是必須拿到設計稿才能開始規劃前端架構，但是前端設計在設計 UI 時，就可以開始思考 CSS 模組化架構、JS 實作可行性等邏輯，而前端工程師則必須被動獲得設計稿後才有辦法開始規劃，流程上就慢了好幾拍。

在我輔導的學生當中，有設計經驗的設計師轉前端設計，拿的第一份薪水的平均值也高於前端工程的第一份工作，理由很簡單，畢竟前端設計除了前端技能外也包含了 UI 設計，無形間將自己打造爲 T 型人才薪水也較高些。

若要分兩者的工作內容，我會說左邊是比較偏向「網頁 UI 視覺呈現」，右邊是偏向「程式邏輯」，雖然學的都還是 HTML、CSS、JS，但前端設計在學這些技術時是偏向「網頁視覺動畫呈現」爲導向，前端工程是偏向「API整合、JavaScript 開發」爲重點。

那麼你說你是設計師想要跨足右邊整個內容可以嗎？當然也可以，畢竟任何東西都可以靠後天培養的。



網頁設計師、前端設計師的差異

有很多人會搞不清楚，網頁設計師與前端設計師的差異，其實工作內容是差不多的，就是需要設計UI，同時也必須將 UI 轉化成前端介面。但若要以程式程度來細分的話，先不講設計功力，這裡用前端火侯來舉例：

不靠譜的前端設計師

- 當設計完前端介面後，就將整包網頁檔丟過去給後端，但不瞭解後端開發邏輯以及是否好套版
- 不懂版本控制語言，所以也不懂該怎麼跟其它工程師協作程式碼
- 當架構變大時，不瞭解如何透過工具或程式語言進行優化 (gulp、Webpack、CSS 預處理器、NPM)
- 不瞭解瀏覽器渲染原理，以致於在 debug 時，不知道該如何透過瀏覽器內建的除錯工具來除錯
- 僅會套 jQuery Plugin 實作動畫效果，更進階要自己客製因爲不懂 JS 底層與瀏覽器應用導致處處綁手綁腳
- 還不太熟機 WEB UI 介面，同時不清楚何謂 Guideline，例如 input 在 desktop、mobile 在各瀏覽器的渲染規則

靠譜的前端設計師

- 比任何人都瞭解一個 HTML TAG 在各載具上的瀏覽器的兼容性。
- 在圖形設計上，瞭解 CSS 的繪圖能力，不會任何東西都匯出圖形，能用 CSS 設計出來

且又不影響網頁效能爲主。

- 知道 SVG、PNG、JPG 的使用時機，並瞭解螢幕 Retina 機制，不會設計會讓圖片糊掉的致命性錯誤。
- 瞭解 JS、CSS 在開發網頁動畫效果的任何細節，擅長使用 SVG、Canvas、WebGL 設計各種 animation
- 知曉後端資料庫合作邏輯，當 CODE 有問題時，會從 Git repo 拉下來切 branch，修改完再 merge 丟給後端。
- 當前端介面需要統一化時，會主動出公司內部的 WEB Guideline，，方便其它同仁遵守規範進行設計
- 會相當專注於網頁元素的設計細節，例如 web font、image、loading、文字設定(行距、字距、粗細、字形大小、標題權重)

  而我認識這些靠譜的前端設計師，薪水級距也在年薪 80~120w左右，也是一個結合本身設計所長，開闢出自己的新道路的一個新的職涯方向，前端設計師。

觀察職稱與工作內容，瞭解廠商究竟是要找什麼樣的人才

最終你還是要觀察公司在職缺網站填寫的工作內容，因爲我還是會看到有些廠商職缺寫「前端工程」，但竟然要包含設計 UI，或是要找前端設計或網頁設計，但工作內容都是跟「平面設計」有關，寫 CODE 權重不到兩成。那以前端設計師來說，設計與寫 CODE 的時間差不多都一半一半，或者是寫 CODE 時間仍然偏多。

建議與求才廠商詢問工作比重，才能確保自己想投入的技術是否能在公司發揮長才，在靠譜的前端設計師，我也列舉了一些投入方向，若你有意轉職前端設計師，也歡迎找我諮詢更多細節 :D

初出社會：沒有人一開始就知道自己要做什麼

寫這篇文時，我其實蠻羨慕現在看這篇文章的你/妳，因爲我花了四、五年時間才確認自己要投入前端工程師的方向。相信大家在決定要投入什麼工作領域時，都會不斷地自我懷疑，自己眞的做得到嗎？如果我眞的頭洗下去我會不會後悔？找朋友/前輩詢問自己是不是眞的做得到？所以每次當我在輔導學生的時候，就很喜歡拿我自己當作案例來分享，以前的我也是和各位一樣經歷過這段時期。

初出社會徬徨無助

我高中是念普通高中社會組，大學是念多媒體設計系，在學業上我那時其實沒有考慮太多，只求能 ALL PASS 就好，雖然高中、大學教授在授課時也時常叨念我們要學個一技之長，否則出社會是找不到工作的。說眞的，在那時我也只當耳邊風聽聽，直到 2008 年大學畢業後進入職場，打開 104 才發現自己在學校所會的東西沒一個是我覺得能夠派上用場的。

在掙扎了一個禮拜後，覺得日子還是得過，所以決定先去麥當勞工作一陣子，期間再好好思考未來的方向。

人生總是有許多曲折變化，但你是否有抓住機會？

就當我跑完麥當勞面試，主管也覺得我 ok，我也覺得可能下個禮拜就可以正式上工時，陰錯陽差地接到了一通來自我媽的電話，說要我來她經營的美容 SPA 店裡，她有位客人對我的經歷有興趣... 咦？

我就一邊騎著車呢一邊心裡各種小劇場，「WTF？？？」、「我有什麼經歷我自己都不知道？？」、「老媽你不要鬧了啊啊」，就在心裡各種 OS 時我就來到我媽店裡，也同時遇到了我生命中第一個貴人，周姐。

以前我不知道該怎麼描述她，現在我可以說在那時對她的第一印象就是氣場很強的人，這感覺就很像是你不認識他，但光從他的外表就可以知道對方是個很強的人，就有點像是孫悟空漫畫一樣，他們都會有一種「氣」的存在，當某個人變超強的時候，他們的神秘感應器就會說：「我感到遠方的XXX的氣變得越來越強了！」，大概就是這樣的感覺。

聊了一陣子後才知道原來周姐有一間 SI 開發公司 (SI：System Integration)，最近有在承接遊戲論壇開發的專案，所以希望一個懂資訊又懂論壇操作的資訊助理幫忙，也提到說有興趣的話，他們目前也缺網頁 web 工程師，之後有興趣也可以轉到工程部門，於是問我有沒有興趣去那裡幫忙？我幾乎考慮沒幾秒就答應了，於是我就婉拒麥當勞，隔天就到資訊公司上班了。

如果沒有方向，不妨從自己的興趣下手

我後來思考了下，爲什麼當時會那麼快答應這份工作。一部分原因是被周姐氣場震懾到，想要在強者身邊工作外。主要原因也有可能本身對寫程式有興趣。

最早的啓蒙是在高一時，我參加了一個班級網頁比賽，那時用的軟體是FrontPage，完全不用懂程式語言就可以拖拉出網頁出來實在很吸引人，我就抱著一本 FrontPage 書一邊搞網頁，看到畫面一一被我刻出來，感到非常有成就感。我也曾經有打開程式碼區塊看了一下，結果看到密密麻麻的程式碼就宣告放棄，後來作品得到了優選，也啓發出我對前端之路的萌芽。

直到大學時我才開始重拾程式，雖然那時多媒體設計系是傾向 3D 設計，但我反而對程式有興趣。像是大一修 HTML、CSS 時，發現原來這些東西是我以前做 FrontPage 的底層，於是就比較認眞去研究，所以程式課程大部分都是我在罩其它學生，最後我的畢業專題是做 用 Flash 寫 as2.0 程式碼，操控我在 3D 軟體匯出來的圖片來設計一個即時戰鬥遊戲。有覺得很厲害嗎？我那時覺得一點也不，各種邏輯都是從網路上找到各種範例所拼湊出來的功能，雖然在畢業展覽上大家都玩得很盡興，在別人誇讚我時只會感到羞愧。

也因爲如此，當我聽到周姐說他們有工程師部門時，可能我內心對程式的嚮往也油然而生，可以的話我想要看一下我大學畢業寫得爛程式碼跟大神工程師寫得有何差別，也讓我在那時毅然決然投入資訊公司。

如果都在猶豫沒有執行，那便會永遠止步不前

我相信看著這篇文章的你，其實已經對前端有著莫名的嚮往，或許你也和我一樣產生相同的漣漪，第一次做出網頁並呈現在瀏覽器上感到喜悅，做出一些小項目覺得自己好像蠻有天份的，或是看完文章後發現其實對某某領域也曾經有相同的共鳴，那麼就相當推薦您去試試看那條道路。

你有可能踏出那一步，發現可行就繼續做下去，也有可能踏出去後發現不適合自己，那也沒有關係，至少你靠自己的力量知道自己適合什麼、不適合什麼，人生的道路上本來就是在逐漸瞭解自己，有些時候總是需要一些事件的衝擊慢慢掌握本身的強項與短處，共勉之。

結尾

我：「老媽妳說她來店裡好幾次，爲啥妳都不知道她自己有開系統公司啊？」
母：「因爲她每次來都很累的感覺，做沒幾分鐘就睡著，所以我也不知道她在做啥。」
我：「……」(內心OS該不會是血汗公司吧，天啊啊啊

資訊爆炸的年代，周遭的人總要你多學一項技能

當我進入公司以後，公司組成可略分爲企劃、工程、美術、行政部門，假使你有工作經驗的話，其實很清楚如果你是以「助理」身份進去的話，基本上每個事情你都會碰到一些，所以公司大大小小的事情都有涉獵，我在那時候的心態其實就是盡量讓自己抱持著開放吸收各種事物，因爲原本我進來的主要原因就是希望從中找到我自己有興趣的方向。

當然，如果你在看這本書的讀書已經有意打算投入前端工程師，而你找的職缺是前端工程師，也被公司錄取時，假使公司要你做本業以外的事情時，就要特別小心。像是最常見的案例是公司本身是應徵前端工程師，但沒過幾個月前端的 loading 變少時，可能就會希望你轉後端工程師。所以在面試時請盡量要詢問清楚前端工程在該公司的工作內容與項目，以避免雙方產生不必要的誤會。

我在那時還是比較喜歡寫程式，所以就一直窩在美術部門，那爲什麼不是工程師部門呢？因爲那時候還沒有專職的前端工程師，甚至連這個名稱都還沒有出現。所以通常「網頁排版」的工作就看設計部門跟工程部門誰對這領域強，誰就領去做。那就剛好公司的設計部門是負責到這段，所以我就跟在美術部門的大前輩學網頁排版，雖然我進來沒幾個月她就辭職了，但硬著頭皮看著前人的網站專案，還是有摸出一點東西的。但我那時還是無法理解發明網頁瀏覽器的開發者是不是白癡，爲什麼相同的程式碼會在 IE6、7、8 會有不同的結果，還必須針對瀏覽器寫特殊語法去除錯，那一段時間眞的是我不堪回首的過去。

我們回歸主題，前幾年在公司期間大家可能看我菜吧，或多或少都會給我一些建議，例如說：

公司希望你什麼都摸一點

工程部門知道我的狀況後，聊天時都會開玩笑吐槽我說「蛤你只會 HTML、CSS 哦，這樣不夠啦，要不然你也寫後端好了」、「你寫網頁要再多學個 xx 語言會比較好」，當他們提到的技術名詞，我都會把他記錄起來，然後自己去嘗試在下班時間去摸索。但就業前半年眞的是一個學習死循環，當我還在吸收他們昨天講的技術，而且也買書來看時。隔天又會再提了好多技術名詞。搞得我非常焦慮，我那時候每天上班都在想我一定是瘋了才會跑來寫網頁。

到最後我才發現這是一個必經的過程，因爲寫網頁只是專案中的一個小項目，當你要將你的網頁部署到伺服器上，中間過程有很多的眉角需要注意。例如 MIS 問你網址、DNS、IIS、遠端主機設定有沒有跟對方要到？我的網頁圖片跟 CSS 路徑寫相對定位，後端抱怨資料都抓不到？周姐抱怨我做的網頁太肥，伺服器 loading 太久。雖然這些東西沒有和網頁有正面的相關，但卻都是爲了讓專案網站能如期上線，各部門都會顧慮到的的細節。

所以假使你也是一樣在這個階段，前半年到一年會有這樣想法其實還蠻正常的，業界會稱做這東西叫做「技術債」，因爲你瞭解開發一個系統或網站的原理太少，導致跨部門溝通容易不順利，所以這些債你遲早還是得還的。我自己也是跟工程部門合作兩三個專案後才覺得自己聽到的火星技術名詞有逐漸減少的趨勢。

你問一百個人意見，就會獲得一百個不同的答案

我相信你在這條道路上一定會有很多人給你意見，但你要記得我說的話，自己的方向要自己決定，他們無法幫您走完人生道路。

不論是已經走在這條路上的前輩，比你早一點就業的同輩，甚至你的家人、情侶，他們都會依照自己在社會上的經歷，提供給妳較好走的方向，但眞的適合你嗎？他們又不是你，怎麼會知道這樣的方向確實正確呢？

就很像是說你問 PHP 開發者，要挑選哪個語言來學後端，他也會依照自己的經驗較爲主觀推薦你 PHP，因爲他在這領域夠久，知道哪裡學習資源多，自己也有足夠的開發經驗，所以才敢給你這建議。

你發現到了嗎？所有人都會因爲自己的價值觀跟經歷提供給你各種建議。就連你在看這篇文章時，也是依照我在前端領域的主觀經驗談。

當然你可以多問各種前端領域朋友問意見，但是要記得，你還是必須依照你的想法，摒除掉主觀意識過強的立場內容，萃取出適合你自己的方向。不要隨波逐流，別人說什麼你就信什麼，否則容易淪落爲多頭馬車，做任何事情都無法有成。

學技能不代表要你就得成爲該領域的專家

我還記得我剛進公司時，一開始和美術設計的摩擦還蠻多的，舉個切圖的例子來說吧，當設計師設計好介面後，都會幫我順便切圖。但有時候設計師因爲對網頁 UI 的掌握度不夠時，就容易切錯。例如背景可以用 background-repeat 延伸，沒有考慮內容資料變多時，介面需要進行調整。

雖然溝通了兩三個專案，但一直沒有好轉。

我：「這樣不是辦法，我明天下班前沒把他搞出來，工程部沒打爆我的頭才怪」
設計：「我也沒辦法啊...我又不懂網頁，要不然你學 photoshop 好了？」
我：「蛤？？我又不會設計，哩賣鬧啊」
設計：「又沒要你學全部...你只要學怎麼切出你要的圖片就好啦」
我：「哦....」

於是乎設計幫我安裝好 photoshop ，花了三小時教我基本操作跟切圖的細節後，自己也能開開心心的切出自己想要的圖片啦，雖然切圖也會增加我的工作時間，但總比設計切出不能用的圖片來回溝通的時間還好多了。

從這個事件也讓我挖掘到自己的一個盲點，當我一聽到要學新東西就會感到恐懼，好像要學得多麼博大精深，非要成爲大師才是眞的學完。但其實並不是那麼回事，人一天也只有 24 小時，你一定會有自己的本業在身上，那爲了要提升你對技術的廣度，你勢必要試著主動學習能夠讓你跨部門溝通流暢的額外基礎技能。

就像我的案例一樣，我學 Photoshop 不代表我要成爲一名設計師，我只想學到的廣度只是方便我可以切圖而已。

我主動學後端語言，不代表我未來就要成爲後端工程師了，我只打算之後我跟任何後端工程師溝通時，可以瞭解他們的後端與資料庫邏輯，減少重複鬼打牆所浪費的時間。

但做這些事時，我還是會以我是一名前端工程師自居，我學習其它技能只是提昇技術的廣度，讓我能夠更加順暢地與其它部門合作，未來能跟團隊共同設計出一個成功的產品，共勉之。

如何有效率地 Google 尋找問題並學習新技術

當我就任三、四個月後，設計部專門負責網頁切版的大前輩離開了，知道這消息實在晴天霹靂，代表著這重責大任要落在我身上了。那時候我切出來的版型常被工程部吐槽，都會被碎念「啊你這版型不行啦，我程式沒辦法接」、「我這裡要跑重複資料，要 repate 的話 HTML 標籤要從哪裡 copy 到哪裡？」、「你這 Layout 是要抓哪啊？你寫得眞得很"巢"耶...我眞的完全看不懂」。所以那時候每次將版型丟給後端那幾個小時都會如座針氈，深怕下一秒又被召喚過去吐槽。

我們公司大部分都承接系統開發，幾乎所有的網頁都會接成動態應用程式，程式語言跟資料庫是用 .NET + MSSQL，後端工程師的嘴雖然很會碎念，但對我這新人還是會耐著性子和我討論，那時常和我配合的工程師叫做阿杜，進來已經三四年了，整個就是一隻老鳥。

我：「所以你會切版嗎？要不然你教我幾招啊，你都會 .NET 了，HTML、CSS 應該還可以吧？」
阿杜：「金拍謝...我會得還沒你多，你可能要自己多研究，以後公司靠你了，你是未來的棟梁。」
我：「眞的假的啊...哩賣騙肖..」
阿杜：「眞的，我在公司只負責動態程式跟資料庫邏輯而已，之前都設計部在負責，排版整個忘光！Google 有很多資料你多參考啦，我不會也是自己 Google 的！」

實在太哭爸了，我眞的超級無言，每次阿杜要我自立自強解決問題時，都會靠北一句「以後公司靠你了，你是未來的棟梁，你可以的!!」來敷衍我，我就會很無言地回到座位上繼續解 IE6、7、8 的 BUG，彷彿大海撈針般從 Gogle 老師探索我想要找尋的答案。

講完我的故事，接下來就來分享一下自己如何用 Google 來解決問題吧。

學任何技能都是從模仿開始

我一開始在學習 HTML、CSS 時，基本的語法概念是有的，當我想要學得更深入時，去網路上看別人部落格，絕大部分都發現到會要你去 W3C 好好看過一遍再來開始學網頁排版。後來我也是照做了，但越做越想睡，雖然基礎知識有了，但我到底該如何運用這些標籤來設計出我要的網頁版型出來，整個就是毫無頭緒。而在當我學 jQuery、JS 時也有這樣的想法。

但一直重複地 Google 撞牆後，還是有發現到一些有用的學習方式。其中一個就是去看別人寫的範例程式碼，網路上不乏會有很多開發者分享自己做網頁版型的範例，裡頭的語法已經讓我有大概的印象，只要瞭解裡頭的語法，我也可以跟他做到一樣的事情，而這種方法也讓我開始改變學習方式，在一開始學新技術前，我會先去學一點基礎的語法，甚至買書來看。等

到稍微有些小成後，就開始去搜尋關鍵字找一些簡單的範例來吸收，有看不懂的語法就回頭翻書或 Google 加深觀念。那時我也會 關鍵字打「website template」、「網頁版型」之類的關鍵字，將別人做得很完整的範例整個看過一遍加深觀念。

沒有人天生就是天才，就像是給你關刀，你知道怎麼亂揮舞，但是刀是拿來斬人的，人會亂移動，你要有效率的斬人一開始是絕對沒譜的，自然就會想去看關公施展招式的過程中學個一招半式，然後自己再去練習怎麼斬，進而內化成自己的武功心法。

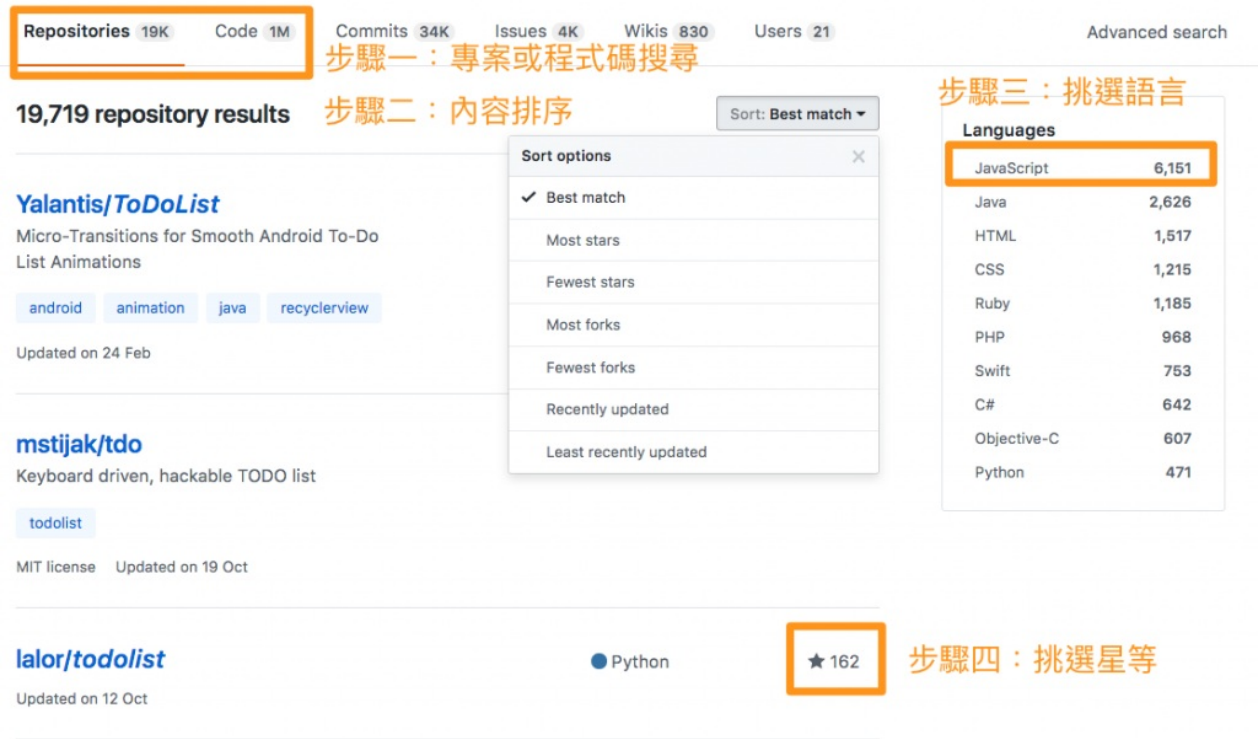
所以才會說一開始學新技術時，先瞭解一些基礎語法後，就從網路上尋找各種範例來練習吧。

先從自己看得懂得程式碼下手

既然要找範例，也記得要找自己稍微看得懂得東西來看。絕對不要找那種跟自己層次差太多的文章，只會讓自己的學習曲線變高，挫折感變重。

這觀念就有點像是高中在上微積分，老師在台上口沫橫飛，你完全聽不懂，舉手問老師，老師重新講解還是丈二金剛摸不著頭緒，於是只好請教隔壁的同學。同學因爲和你的背景相似，比較瞭解你的程度，所以在口語解釋上你比較能聽懂他在講啥。

除了透過網路搜尋別人的開發部落格外，Github搜尋也是個好方法，它目前已經是世界最大的程式碼集聚地，例如說你想用 JavaScript 寫一個todolist，就會看到下圖畫面。



步驟一：像是你打了 todolist 關鍵字， Rspositories 是別人的專案名稱、Code 則是觀看別人的片段程式碼，像我很喜歡看 Code，例如我學會了一個 JavaScript 的 forEach，但想要多找些範例加深自己觀念時，輸入關鍵字就有很多 code 可以參考。

步驟二~四：我自己比較習慣用「Best Match(關鍵字吻合度)」、「Most stars(星等排序)」來下拉尋找，當我在學新技術時，我會反其道而行找星星最少的，因爲現在的我剛學會某個程式，想要更進一步瞭解更多，所以我自然要找我的「同學」，或是比我厲害一點的，通常星星多表示他寫的 code 很漂亮，或整合了許多功能，這樣反而會阻礙我只是想學一個觀念的途徑。除非是說我要找一個熱門的套件，例如網頁動態日曆、燈箱效果時，自然是去找星星最多且有持續在更新的專案會比較 ok。

所以請記得在學程式時，你應該找的學習資源是「比你厲害一點的同學」，而不是大神，當你吸收了同學的養分越變越強，看的 star 越高也都看得懂得時候，恭喜你，你慢慢掌握這技術了。

看不懂程式碼，線上影音教學是你的好朋友

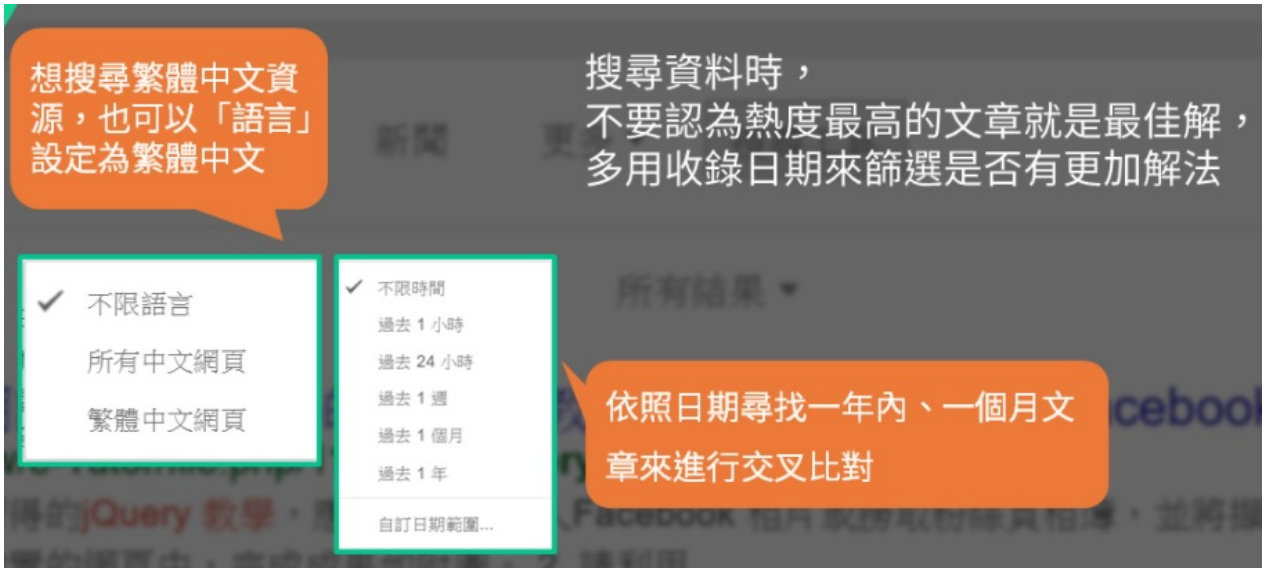
有的時候你還是會遇到看了別人部落格，但自己在練習時還是無法做出一樣的功能，有可能是部落格寫錯、少貼一些 code、環境不一樣等諸多原因，所以我也會習慣用youtube來下關鍵字，看是否有人有教學錄影。另外udemy也是一個不錯的線上課程平台，主要也以程式課程居多，同時也有許多免費程式課程，優惠期間也長有特價 300 優惠，雖然大部分都是英文課程，但幾乎都有支援字幕，甚至透過Google 翻譯軟體即時翻譯。

這樣好處在於至少照著影片上的講師一起做，或下載他的範例程式碼，也有助於你學習程式上會比較順利，想什麼時候上課就自己安排時間學習。

不要尋找過時的文章

技術更新快速，假使你在找解法是三四年前的文章時就要保持懷疑猜測態度，不要盡信，較好的方式可以用 Google 搜尋的下方的功能列。 像是如果你英文不好，也可以先搜尋繁體中文，若還是不行就搜尋「所有中文網頁」，可以進階找到簡體教學，時間區間抓一年以內其實資訊就還蠻新的了。

經由過濾和交叉比對後，找的文章自然也就比較新，或是到目前最大的 程式論壇stackflow下關鍵字，找投票贊成數多的文章來觀看也 ok。



簡報網站都會有意想不到的乾貨

我在學新技術的時候，還蠻常會去能夠上傳簡報的服務網站看看的，通常有人整理技術簡報並上傳，往往都是講師級的 level，簡報品質自然壞不到哪裡去。我自己最常去的則是SlideShare與speakerdeck

圖片搜尋

有時候在找靈感時，也會用圖片搜尋的方式，舉例來說我要找一個jQuery slider的效果，就可以先初步過濾哪些 slider 做得不錯再點擊觀看文章，像是一些動畫效果的關鍵字，之前我也有整理一份起來，也提供連結給各位參考。

結尾

希望以上這些方式能夠打通你對 Google 關鍵字的想像搜尋空間，除了單純文字吸收，其實還是有不少可以幫助學習的管道。

--

阿杜：「阿你是 Google 到了沒辣!!」
我：「還沒啊，你這麼會估你來估啊。」
阿杜：「我等到要度估了啦怎麼估!!」
我：「.....」

番茄鐘工作法 - 設立短、中、長期里程碑

經過一年洗禮後，主管交辦下來的項目大部分都能如期完成，就算有困難，部門間彼此照應，倒也覺得沒太大問題，原本對自己能力開始洋洋得意時，周姐和主管竟然讓我自己去負責兩個專案，一個公司內部系統專案，另一個是公司官網改版，我當下就傻了。

周姐：「洧杰，這兩個東西就交給你了，你規劃一下，然後再把各期間的時程壓給我。」
我：「....(一臉傻逼樣)」

壓時程？但我只會網頁排版啊，代表我也要問下工程部、設計部時程，然後東西也要規劃 UI 介面、動線流程，而且又是公司門面，這麼多事情到底該怎麼想啊，而且又要壓時程，這不是企劃部該做的事嗎？

周姐可能也看出我的心思，一臉不耐煩地說：「現在企劃部門每位都在準備拉明年的案子，都沒時間做這件事。況且，你不要以爲事情都是你一個人全部做完，你只是先將項目拉出來，問各部門時程，你只要幫忙控管專案就好！」

我：「喔...」

最後，我還是搞砸了，我比預期時間超出整整兩個月才交付項目，才意識到以前工作都是被動式接收小項目工作，當需要多人共事與專案控管時，才發現要去注意一大堆細節，除了自己的事情外，重要的是還要管「人」，除了同事還有客户與第三方合作廠商。

所以東西只要一 delay 我就開始找各種藉口給自己。還不是工程部門拖拖拉拉的、還不是周姐不喜歡哪個版面設計一直改，後面也根本沒辦法動。時間那麼少我自己的東西都做不完還管得了別人？我只是進來做網頁排版的，爲啥要我做企劃的事？更不好的是，該驗收的時程點我開始選擇裝死，上層問進度我會用盡各種理由去搪塞，最後就進入到一個很負面的死循環，每天上班壓力超大，又會覺得自己很菜，不敢向前輩催進度，所以整個進度大緩慢，我眞的恨不得自己十項全能，都我一個人做就好，這樣我也知道我何時才做得好。最後開始逃避專案，故意去做自己擅長的工作內容，讓自己很忙，不正視自己眞正該負責的內容。

就在那時我一直過得生不如死，罪惡感越來越重，直到有一次我拖延一個禮拜，上層跟周姐都沒再問我進度，我還天眞以爲他們終於不想管了而沾沾自喜。當然，事情沒有這麼簡單，又過一個禮拜後被叫去會議室被飆了超久，而我也是眞的被罵醒了，才開始正視自己的問題。

所以這篇番茄鐘文章，也算是寫給八年前的自己，如果早點知道這些時間管理的心法，那段時期我也不會過得那麼痛苦，同時分享給各位。

什麼是番茄鐘？

番茄鐘是一個心法觀念，工作 25 分鐘後可以休息 5 分鐘，結束一個循環代表你嗑了一顆番茄。網路上其實也有不少文章可以參考。你可能會想說，那到底該如何運用呢？先不要急，我先分享一些心法給您。

複雜的事情可以試著把它碎片化，先從簡單的事情下手

還記得我 2012 年讀在職碩士準備論文時也很頭痛，光是上班就很累了，還要準備量化研究與準備論文，感覺好龐大，總有一股完全做不完的 fu，那時候教授就指導我說，你應該是要試著將困難的問題拆解成一個個小問題，再逐一攻略，先處理簡單的問題，再把一些困難的問題繼續拆解，每週固定產出進度，積沙成塔自然容易水到渠成。

雖然說起來很簡單，但做起來自然需要花時間習慣這樣的模式，所以當下就和指導教授約每週固定一次的 meeting 來解決問題，而這也讓我培養起將困難的問題簡單化的做事心法。

每天上班時先列最重要的幾件事情出來

許多事情都有輕重緩急，但有些東西一定是你今天該完成的進度，我自己習慣將項目寫在紙或trello上，工作項目大概會有 7~12 件，每當我做完一件事情時，我就會用筆把它劃掉，會有一股莫名的治癒成就感。

當一個大專案需要一個月時程做完時，我會在每週的週一規劃一個禮拜要做完哪些項目，週五當做一個小里程碑來進行小驗收，而且訂在週五的好處是隔天就放假了，你完成一個小里程碑，你和部門同事都會覺得事情向前邁進一大步，不會覺得事情都沒有做完，假放起來更爽。

不要習慣加班

有些人會習慣加班，這時候就要問問自己，是否是你自己效率不好，每次有比較多的工作，就會心裡安慰自己反正加班一下就完成了，而反而讓它變成常態？而且假使是常態性加班，主管其實也會質疑你是不是效率不好？與其如此，爲什麼不找尋有效率的方式來完成工作，或找出自己的工作時間小偷，並正視問題加以解決呢？

時間一直倒數，你會有時間壓迫感

我是用這套軟體，網路與 APP 也很多，找一個自己喜歡的就好，它會在我的電腦桌面上一直進行倒數，所以當我看著它倒數時，心裡也會不自覺地感到時間快到了，得盡快把任務做完。身體會不自覺地想要挑戰在時限內完成。

當一天規劃要做哪幾件事情，就會思考我今天預期要吃五顆番茄，第一顆番茄我想同時完成 A、B項，下午再花四顆番茄去完成 C、D、E 項。

容易進入心流模式

所謂的心流模式就是你會進入一個相當專注的狀態，因爲在番茄鐘督促你時限時，你自然會打起精神將事情做好。當在這狀態時有時候我預估要花九顆番茄才能寫好的程式，只要花三顆就做完了，相信你也曾經有過這樣的體驗，當然，要容易進入心流模式還需要一些工作上的心法，例如自己創造一個不被打擾的空間、把通訊軟體都關掉等等，有機會的話後面再繼續談。

另外工具是死的，人是活的，像是有時進入心流模式後，我就不繼續用番茄鐘了，讓自己持續專注享受這狀態快速完成工作。所以它有時候又會像是一種「儀式」，當我開啓表示要認眞上班，在持續專注的情況下時間自然就過得很快。不過還是建議工作一段時間還是要多起身走走，身體循環才會好。

不要怕得罪人，做事要看結果

就像是我的故事一樣，以前我很菜，不敢向前輩壓時程催進度，時程 delay 被上層罵，也被同事抱怨，變得裡外不是人。公司要的是結果，就算你眞的很努力了，但是用錯方法或是不尋求建議導致時程因此延宕，客户對公司產生不信任感，對公司眞的是好事嗎？

後來這觀念也影響我可以很正面地跟同事們溝通正事，厚著臉皮討論時程催進度，畢竟大家在同一條船幫忙賺錢，沒錢我們哪來的薪水呢？

不只用在工作，也能用在學習上

當你想學習一個新技能時，也可以利用以上的心法，首先先訂立你學這個東西，希望能做出什麼成果？這個目標越具體越好，否則你自己也會不曉得學這技能到哪時才會結束，自然會想要偷懶。

這裡就以初學者學習 HTML、CSS 當做例子吧，學 HTML、CSS 可以做到哪些例子？自然是做出一個網站嘍！所以你的方向就可以寫說我要做出一個個人介紹的網站開發去列出你的學習項目，並將任務碎片化。

這裡我就列出一個學習階段項目出來：

一、大項目 - 尋找資源

1. 中項目

    - 尋找 2~3 個 學習資源，最好是找能夠讓你完成一個網站的範例尤佳
2. 中項目
    - 詢問友人、上網尋找評價關於這幾個學習資源的評比

二、大項目 - 找到線上課程開始上課(大項目)

1. 中項目
    - 完成 1~3章
2. 中項目
    - 完成 4~6章
    - 小項目 - 第四章
    - 小項目 - 第五章
    - 小項目 - 第五章 float 看了課程原理不懂，需要多找些範例
    - 小項目 - 第五章 position 看了課程原理不懂，需要多找些範例
    - 小項目 - 第六章
3. 中項目
    - 完成 7~9章

三、大項目 - 開始實作自己的個人網站

1. 中項目
    - 思考網站版型規劃 (鉛筆稿)
2. 中項目
    - 網站排版
3. 中項目
    - 網站部署

從上面可以看出，做事情時先規劃大項目出來，越籠統也沒關係，然後再從大項目切成中項目，中項目如果太難再繼續切小項目，再來就從最簡單的開始實作，並規劃這一個小項目預期要幾顆番茄才可以完成小項目。每一週你可以訂立至少要完成一個大項目或中項目，然後再從每一天的零碎時間撥 1~2 顆番茄來達成你的目標，這樣才會是比較具體的學習方式。

最後，我相當建議你要有一個自己的線上筆記本，像是quip就還不錯，我找了之前我學 ionic 寫 APP 的線上筆記本給各位參考，你可以看得出來我將一些常用的指令與筆記寫了上去，另外該學習的項目也放上去。如果是別人整理的東西，那並不是你自己的，只有你整理自己的筆記，才會知道你做了哪些練習。時間一久忘記某某指令時還可以翻一下自己的筆記，彷彿時光機般紀錄著當時你的學習狀況，其實是很不錯的。

另外也可以將一些你雖然當下看不懂的文章，但你又覺得他是一個很重要的懶人包或原理解釋，我也習慣先放著，等到我對該技術越來越熟時，再去看艱澀原理文章，往往觀念就整個貫通了。

結語

以上便是我個人一些工作流程的心法，也提供各位參考與學習，最後再附上以前我常用的 chrome 插件工具

- StayFocusd：可以限制一天你只能看特定軟體的分鐘數，例如一天只能看 20 min FB，超過就鎖定畫面
- Life Is Too Short：每打開 chrome，就會倒數你的人生，督促你不要浪費時間在蠢事上
- TodoList：習慣用的 todolist 軟體

盡早培養「自主解決問題」的能力

在講這個主題前，我要來分享個人經歷，在創立高雄前端社群後，常會有人敲我訊息說對前端有興趣，想瞭解要到什麼程度才有辦法滿足職場需求。

當然在業界久了，自然可以推薦一些學習資源與方向，也協助幾位朋友投入前端，但過沒幾個月，他們不約而同出現以下對話。

朋友A：「洧杰問你一下，主管要我研究前端的 xxx，但都沒有頭緒怎麼辦？」
我：「哦，這在網路上有蠻多教學資源的，有先搜尋嗎？」
朋友A：「有啊，但都好難....怎麼辦壓力好大」

朋友B：「老闆要我用一個 HTML5 JS API 搭配 Google Map 開發一個東西，你那裡有 CODE 可以參考嗎？」
我：「那段整合我沒碰過，你有看下 Github 有沒有相似的專案？」
朋友B：「你找比較快啦，之前都你提供資源給我的，給我個範例後面的我會再自己搞定啦 >”<」
我：「...囧」

此時才意識到自己反而害了他們，以前都習慣幫他們找適當的教學資源方便他們消化，但並沒有培養他們自行尋找學習資源，並內化成知識的能力。用常見的俗語就是「只給他們魚，但並沒有給他們釣竿」。

也因如此，在協助學生轉職前端時，我都會請他們做以下事項：

培養關鍵字搜尋的敏銳度

我在六角學院協助學生轉職前端時，當我覺得對方學到一個深度後，就會開始幫助對方去試著自主學習新語法。例如你會 HTML、CSS 基礎排版後，接下來就會請他去研究什麼是 transition，接著試著用 animation 設計一個動畫效果。如果是 JavaScript ，懂陣列、函數、物件操作後，就會再進一步投入中階語法以提升他們對程式的掌握度。

接著他們就會依照我前面章節提到的「如何有效率地 Google 尋找問題與解法」開始自主搜尋答案，畢竟我並不能幫學生培養搜尋關鍵字敏銳度，因爲大腦是他自己的，我沒那麼強可以幫他的大腦升級。

在你吸收一個問題時，你該下什麼關鍵字可以找到最接近的答案，成功與失敗數千次後，大腦就會自動建立專屬於你自己的搜尋流程引擎，遇到問題時，直覺就知道到某個技術論壇會比你到 Google 大海撈針有效，或是印象曾經有在 youtube 找過豐富資源等等，當你這流程優化到個極致後，自然能幫助你在找問題能夠更聚焦地找出答案。

我在幫學生培訓時，也會出一些題目，讓學生們能夠培養「自主解決問題」的能力，並實作出前端功能提昇技術廣度，這裡也釋出部分題目，提供各位朋友參考與學習。

評估技術複雜性

你需要有個認知，當你進入公司開發產品時，實作一個功能絕對會是很多個技術組合起來。舉例來說有個頁面要有好幾個時鐘，並即時顯示各時區的時分秒資訊。你可能知道 JavaScript 已經可以產生時間，但爲了要顯示各時區，就還必須瞭解 UTC、GMT 是什麼，那些國家的時區要如何歸類，以及 JS 與 CSS 該如何控制，才有辦法做出一個動態時鐘出來。這在程式世界是家常便飯，因爲你不是「學程式」，而是「設計程式做出人們可以用的服務」。

當主管在問你是否能實作出一個產品的重要功能時，你是否有辦法評估技術複雜性？

只知道七八成，其它兩成你完全沒聽過，不過曾經從網路或朋友得知早已有實作方案，自然當下回應這部分是可行。

只知道五成，另外五成你當下沒把握，雖然沒辦法即時回應主管是否能做到，所以告知給你兩三天研究，嘗試在最短的時間內做出一個最小可行性產品確保 ok，以回應主管這功能核心的技術都能達成，讓主管回應上層此技術的可行性。

如果是完全不懂的東西，主管只好給你個時限去研究，那你有沒有辦法將複雜的問題細分化，投入過程中瞭解自己距離目標還有多遠。假使如實交付，你能夠肯定自己，如果需要延期，也絕對不要拖到最後一天才告知主管，這樣他也沒辦法向客户交代。

人在遇到未知會感到惶恐是理所當然，在程式世界上，更是家常便飯。所以會建議各位面對它，並善用番茄鐘來瞭解自己的投入狀況，當你將這個循環投入數百數千次後，自然就離資深工程路也不遠矣。

先找出自己可以吸收的內容

以前我在實作一個主管交辦的功能時，會在網路上搜尋各種教學，有時候我會發現某個東西就是我要的，不過還是需要改一些內容才能放在我的專案上，但卻搞了很久都弄不出來。我甚至會亂下一些語法，自己也不知道自己在幹嘛，只求其中一行程式碼可以動，我就可以準時下班。想當然爾最後都沒搞出來，只留一肚子鳥氣回家。

這問題也是出在於技術債欠得太多，明明這功能你會有兩三個核心原理必須先搞懂觀念，但是你視而不見，試圖硬改現成的 CODE 期待奇蹟發生，這本來就是很天真的想法。

欲速則不達，先找自己可以吸收的程式碼，等到觀念融會貫通後，做出自己想要的功能還不難嗎？

複製貼上是家常便飯

在我的前端生涯中， ctrl+c、ctrl+v 算是排名前五名的熱鍵，例如要引用別人的動畫效果、一段複雜的正規表達式不知道怎麼寫，於是上網找解法。你要瞭解任何事情都要自己從零寫到有，自然是不可能的事情。

例如我是後端工程師，因爲專案都必須處理複雜的商業資料庫邏輯，但剛好前端缺人手，所以只好自己用 Bootstrap 框架處理前端問題，自己不用寫太多 CSS，也能夠排出堪用的版型。或是顧客想要一個比較炫的轉場效果，你沒有 idea，只好去codepen尋找靈感，並看是否有適合的效果。

工程師其中一個重要工作是在「整合」，一個中大型專案上絕對會是用了好幾個套件、部分邏輯不懂所以複製stackflow投票數高的程式碼，同時也包含你自己爲產品寫的商業邏輯程式碼。

你也不用覺得一直複製貼上很丟臉，覺得自己沒做啥事，在持續複製貼上過程中也會貼出一種感覺，提升你對 CODE 的敏感度，但要記得自己是「有意識的貼上」，而不是死馬當活馬醫亂貼而完全不懂原理，如果程式碼有太多部分是你不懂的東西，最後專案爆炸的機會也會跟著變高。

判斷問題對錯，不要全盤盡信

在網路上找的東西不要全然相信，例如我常遇到學生請我 code review，發現一段程式碼是錯的，詢問他才得知那段 code 是複製某個部落格工程師上的 CODE。結果才發現因爲程式版本變更，CODE 也跟著不兼容了。

也呼籲在搜尋問題時，都要抱有「持疑」的心態，吸收到的知識就大膽假設小心求證。寫程式沒有正確答案，今天可以跑的程式碼，明天瀏覽器版本可能就不能用了，你無法找出「最完美」的程式碼，但至少你能找出「當下適合情境」的程式碼。

提升解決問題的思維和方法

以我上面的貼的題目連結，你會發現前端不單純是做網頁開發，現在的 JavaScript 可以做的事情相當多，那爲了要培養解決問題的思維，就必須深耕自己的能力，很多人都會喜歡拿「投入一萬個小時才有辦法成爲專家」的話來講，這裡我想說所謂個一萬的小時，不是你已經會的東西重複做一萬個小時，而是知道自己哪裡不足，刻意去補上你不懂的前端領域知識，除了提昇技術深度外，同時也在加強你遇到問題時解決問題的能力，如此一來遇到任何事情時都能依照自身的一套心法去解決問題。

寫爛 CODE 是學程式必經之路

在我進公司寫程式的時候，前幾年都覺得自己在產出一堆噁爛的 CODE，業界還有一個術語叫做「義大利麵的程式碼」，意思就是你寫的 CODE 就像是麵條雜亂交錯，完全無法看出你的程式脈絡。

尤其是網站要準備上線時我就會特別緊張，都會想拉同事幫忙看一下能不能這樣寫。

阿杜：「靠北只要可以動就好了，管那麼多幹嘛？？」
我：「.....」

設計師同事：「嗯..你網頁跟我設計稿很一致啊，應該沒有什麼問題吧？」
我：「喔....」

於是乎網站上線三個月後相安無事還譏笑我顧慮太多！！！

可惡你們都不懂我內心的糾結，我裡面下的 CSS 程式碼都從網路上複製亂貼亂測試，網路上說這樣就能兼容 IE6、7、8 我就傻呼呼的貼上去也沒問題？那時候流行 table 排版，裡面的結構 table 包 table 包 table 巢狀到不行你們也覺得沒關係？我 jQuery 效果亂貼跑馬燈就出來，問我怎麼寫？我只會跟妳說就這六行 jQ CODE 跟一個 CSS 載入就會動，你說原理哦？不好意思他 CSS 寫得很外星語言我完全看不懂欸。

以致於剛進公司前幾年的上線網站都不敢跟人說那些是我做的，深怕別人會打開原始碼，然後恥笑我的 CODE，恥笑公司怎麼會請到這麼菜的人來做網頁排版。

相信各位在開始學寫程式的時候，也一定會有我以上的心路歷程，那麼到底該如何調整心態度過這段時期呢？就讓我娓娓道來吧。

自己要先寫出爛 CODE 才知道要怎麼改進

寫程式同時也是培養邏輯的過程，以網頁排版來說你學了 HTML、CSS ，可能在網頁排版邏輯並沒有融會貫通，自己實作出個網站。JavaScript 學了一些流程控制、變數函數，自己硬幹出一個超陽春版的todolist代辦清單，雖然當下你都會覺得自己做得東西超爛，但你確實靠自己的想法做出一個「可以正常運作的東西」，更重要的是當東西做出來，才有辦法去優化，思考下一步要如何擴充功能，或做完後去搜尋其它人是怎麼寫的並加以練習。

所以我都會跟初學者說，你們就放心寫爛 CODE 吧！一位成功的資深工程師的背後都有著一堆爛 CODE，再從中去思考如何讓自己更好，沒有人天生就是天才，信手拈來就寫出一行行好 CODE，所以拋棄你的羞恥心，初期先以「先求有再求好」的心態來學習程式吧！

不要還不會走就想學飛

比較常見的例子就是 HTML、CSS 還沒實際做出一個版型就想直接用 Bootstrap，或是 JavaScript 才剛會 變數、if、for 等基本語法，就想直接學 SPA 框架 (vue、react、angular)。

以前我也會這樣，當社群上有人分享他開發的習慣與設計模式後，我就依樣畫葫蘆去模仿，然後還洋洋得意地以爲自己功力大幅提昇。但實際用幾天後才發現，裡面有太多抽象的設計模式需要理解，或是你的專案根本不需要用到如此複雜的開發架構，導致自己痛不欲生。

所以會建議不要好高騖遠，好好地打穩基礎功才是王道，所以每當有人問我前端趨勢是哪些時，介紹完後我都還是會耳提面命地說，先把基礎功學好吧。要用 CSS 框架，至少底層的 CSS3 media queries 瞭解響應式原理，你才知道當版型有超出預期的調整時，自己也能寫 CSS 去客製，想學 JS SPA 框架，至少先把 AJAX 原理搞熟、知道網址 router 邏輯、JS執行生命週期再去碰才不會覺得各種觀念卡卡。

透過專案時程壓力來練功

當我培育線上學生投入到一個我認可的階段時，我都會建議他們盡快去就職將能力培養起來。

有時候透過自學你所能涉獵的範圍有限，例如一個專案從開始到結束的過程、跨部門溝通，更重要的是程式碼也會比較全面，在公司你要做的東西絕大部分都在做別人的產品服務，在功能邏輯也會比較複雜，吸收得東西也會比較多。

那在公司執行專案確保東西能如期交付，通常都會去壓時程，例如設計師要花二天設計與跟客户確認，網頁切版 N 天，後端套程式 N 天。在時間有限的情況下，才能督促自己提升程式開發能力。

常常會有學生問說：「老師我想用 XX 框架，你覺得我可以在下個專案上用用看嗎？」

我都會建議說想試試看都 **ok**，但不要一口氣在一個專案用太多新技術，以免變數太多，像我在開發新專案時，至少七成是我有把握的東西，其中三成就會是最近我吸收的新技巧，想透過專案來使用他們提升我的程式功力。在時間壓力之下，學習效果自然會被逼得更好。

當然，你要全盤投入新技術在專案上也不是不行，個人覺得在開發程式時，也是在培養一種「我就是有辦法將這個系統依照我的技術觀打造起來」的自信，在決定導入前，先把一些重要的邏輯設計先設計出一個雛形 MVP，確認 ok 就放手去幹吧。

那時程被拖延到怎麼辦呢？當然是自己加班補起來啊，如果是你自己壓得時程但自己錯估時程，本來原本技術就可以做好的事情，因爲你投入新技術導致時程延宕，那就是你自己的責任。但這也能讓你培養起「瞭解自己的學習投入狀況，並評估工作時程的能力」，這樣一來在專案跑個好幾輪樣流程後，功力絕對大增。

適時重構，重新檢視自身不足

在公司一定會遇到時間很趕，但又必須有產出的時候。以我來說，正常開發時程我就會用我自己擅長且又正規的寫法來寫，都會考慮到程式品質與未來靈活彈性的狀況。但當時間少時，你自然會以所有功能都先有，連命名都超懶、完全不管任何品質，反正可以跑就好。雖然在時間允許之下你還是會瞬間寫出正規的 CODE，但絕大部分都還是相當的骯髒。

在這過程中也是對自己能力不足的照妖鏡，你可以發現自己在哪些細節上還有待加強的空間。例如對物件導向不熟悉、或是 CSS 模組設計上的脈絡，表示你對那些東西沒有眞正內化到自己的心裡。

所以在公司沒有很忙得時候，我還是會自主研究自己已經上線的爛 CODE，去思考我哪裡還有進步的空間。唯有自主發現自身不足之處，才有辦法讓自己更好。

程式寫不好，是不是我沒天份？

相信大家在學程式的過程中，都曾經閃過這些念頭：

- 爲了一個小問題卡了兩天，但就是找不到解法
- 覺得自己學得很慢，我應該不適合？
- 感覺資工資管科系才會把程式摸得很透徹，我跑來學程式眞的適合我嗎？

這些念頭我都曾經有過，但是最終我還是排除萬難地成爲了前端工程師，提供給各位我克服後的心態轉換建議。

跟自己比較就好，勿豎立對手

常遇到學生問我：「老師，如果我花 XX 時間的話，有辦法找到前端的工作嗎？」，這點我從來不會保證學生一定會在何時可以就業。因爲每個人的吸收學習程度都不同，若是你沒達到目的，下一次豈不就更難重新出發？

你需要專注的是規劃你的學習清單，就像前面番茄鐘所提到的心法，凡事都可以切出大、中、小項目，將任務碎片化再逐一攻略。每天日月累積前端技能，自然能夠水到渠成。當然會因爲天份的關係，你總會覺得周遭的人好像都很聰明學東西都很快，但你是否有看到他背後投入多少心力，每晚夜深人靜時，面對著淡淡發光的螢幕貪婪地探索知識呢？

在學校裡，成績就是一切，在名次排行榜的枷鎖下彷彿任何東西都好像要跟人比較一樣。但以掌握一個技能來說，你的對手只會有一位，就是昨日的你。常常會有學生問我：

學生：「老師我這頁花了我一天排完，會不會太久？一般專業工程師應該很快就能排好的吧？」
我：「那上個月請你排的那頁你花了多久？」
學生：「三天啊，而且還一堆 BUG」
我：「所以再給你一個月練習不就可以更快？凡事都是日月累積來的，跟自己比較就好，你現在應該著重在觀念的應用，等到眞正有專案壓力時，自然就能培養起來。觀念不對基底不好也都是枉然。」

所以也奉勸各位，不要自己幫自己找對手，你最大的對手就是昨日的自己，你只要有贏他，就足夠了。

寫程式是不是數學要好，一定要有資工資管背景？

這個要看你寫什麼程式，如果像是大數據分析、資料演算的話當然需要，但以前端來說初期是不需要的，頂多到第二份、第三份工作時才需要瞭解一些而已。一開始學 HTML、CSS 時，程式語法都是設定樣式，實際上不需要邏輯運算。在學 JavaScript 時，便會開始培養邏輯觀念，但也不會那麼的重。像是我周遭的前端強者絕大部份都是非本科系，我輔導的學生有七八成也都是非本科系，同時原本本業是服務業、業務、銀行等相關，轉換跑道還是相當順利。

拋棄你對工程師的無限幻想

還記得我當時想成爲工程師時，都會對這職業產生幻想，例如強大的工程師應該打字速度都超快吧？他們一定都是做很炫砲的項目，不像我做得東西小不拉機又不酷。他們一定很常在 open source 上相當活躍，本身一定又是個超熱血熱情的傳教士，年薪自然都是千萬等級以上，而且一定超愛寫程式，無時無刻都在寫才對。

但其實當我成爲前端工程師跟許多人接觸後才發現，那些都是來自於我的幻想，有些人當工程師只是因爲他擅長，但並不喜歡這工作。或是只是剛好歪打正著成爲工程師而已，但是平常下班時間也不會碰程式。像是Bootstrap 開發者也直言自己很痛恨電腦，原本他是要去修社會學的。PHP 之父的經典名錄也有一句提到他覺得自己不是一個開發者，當人們打算要解決一個記憶體問題時，他反而覺得重開機就好了，幹嘛要去解決？

如果你想學程式，未來也想成爲一個開發者，奉勸你不要對這職缺有過多的幻想感到卻步，你不需要很有天份很有熱情才有辦法寫程式，就連我下班時間也有很多時間拿來耍廢追劇打 Game ，沒有無時無刻都在寫程式的。

程式卡關是家常便飯

絕大部分的人都會因爲自己卡在一個小細節卡很久，就會開始自我懷疑自己到底適不適合寫程式，但我要老實說這是必經的過程，而且不代表未來就不會發生。像是我在寫程式時也時常犯許多低級錯誤，例如語法拼錯、忘了存檔就想開啓看結果、已經會的觀念但不知道爲啥卡到陰一直沒解出來。

不過這些過程也是在幫助你 debug 除錯的能力，就像是很多學生問我問題，他還沒說完我就猜到他是哪裡出了狀況，學生當然會覺得老師很厲害，但可不要忘了我們可是經歷過數千次這樣 debug 經驗才有辦法達到這火候的，而且這段過程我也沒辦法教，你只能慢慢熬才有辦法內化到你心裡。

最後建議當你覺得卡關時不妨暫停一下，去上個洗手間或洗澡吧，有時候坐久了血液循環不好，當你在移動時大腦就會重新運算新的解法，所以我都會笑稱洗手間是我的靈感來源 (誤。

找一個好 **mentor** ，讓你少走些冤枉路

什麼是 mentor (饅頭) 呢？mentor 並非像是教室的老師能夠全程時間來「教學」，而是比較偏向「引導」的角色，也因此 mentor 也可以稱爲人生與職場導師的代名詞。

我相信你在投入前端時，時常會有不知道自己下一步到底該怎麼走的窘境，很希望有人可以教你該怎麼走下一步，讓自己不至於繞太多歪路，導致浪費太多時間。我必須老實說自己從就業到現在沒有遇到 Mentor 過，直到我反過來輔導別人，在幾年前看到 mentor 這個詞彙才發現自己都在當別人 mentor 了，所以這裡也來分享如何找一個好饅頭 (mentor)。

沒有人有義務去幫你

首先你先要有個認知，沒有人天生就必須幫你忙，我常常會看到有人去加一個前端開發者的 FB 好友，然後問他些問題，沒獲得回應就抱怨對方冷漠。但是你有想過彼此素未謀面，當你請教個程式問題裡頭很多 BUG，就算資深開發者也需要花一陣子才有辦法搞懂邏輯，那對方爲什麼有義務幫你去 Debug 呢？每個人都有自己的人生要過，光自己的事情都應付不暇，怎麼還管得著別人？

如何認識潛在 **mentor**

難聽的話雖然說在前頭，但並不代表就無望了。這裡分享一些方式給各位。

追蹤你在意的資深前端開發者

我相信在你投入前端領域時，多少也透過一些網路資源與社群文章發現一些活躍人士，進而追蹤他們的動態、部落格與演講。或是在前端聚會時，遇到些資深開發者，這些人自然會有可能是 mentor 的最佳候選人，畢竟他們在這條路走跳許久，自然會知道許多經驗談。

參與他們可能出現的場合

舉例來說，他們可能會在哪裡演講或參與什麼聚會時，這時你也就可以跟著前往。如果有見到面，也可以透過提問讓對方留下好印象，如果能私下請教問題那就更棒了。這裡分享一個小訣竅給你，請不要太過阿諛奉承，誇讚對方的技術能力。越資深的開發者也因爲投入的技術過深過廣，所以也相對覺得自己掌握的技術比較起來微乎其微。比較好的方式是你有看過對方的技術部落格、 Git repository，然後請教一些東西，如果你有用他分享出來的技術自己做一些小應用，那就更棒了。對方就會覺得你眞的有認眞閱讀他的技術內容而對你產生好感，印象分數自然大加分。

結束後，使用 **mail**、加社群好友加深對方印象

接下來你就可以試著加潛在 mentor 好友，並再次自我印象加深他對你的印象，能做到這你其實就走近一大步了。

不要把潛在 **mentor** 當做免費的顧問

在你投入前端過程中，勢必會有許多程式與職涯上的問題想詢問他人，所以強烈建議你要將提問的智慧看過一次，不要讓 mentor 認爲你是伸手牌，另外在發問頻率上也要節制，像是以我個人主觀感覺，每天都問超過十幾個問題可能會有點排斥，但如果是三四天問一兩個問題倒就覺得可以。

在發問前要讓對方認爲說你確實有花許多時間在尋找問題上，別人發現你是有備而來才會主動想要幫你。另外提醒一點，如果你是程式碼問題的話。

絕對不要截圖程式碼，也不要在聊天視窗貼程式碼。
絕對不要截圖程式碼，也不要在聊天視窗貼程式碼。
絕對不要截圖程式碼，也不要在聊天視窗貼程式碼。

因爲太重要了所以要講三遍，如果是觀念不通時，你可以把該程式抽化出來放在codepen、jsbin上，好處在對方可以即時從 code 瞭解你問題，並回改給你。還有就是絕對不要將所有的程式碼放上去，只將自己觀念有問題的放上去，這樣才可以加速檢視程式碼的時間。若是專案複雜的話，至少也要傳到 Github 上，絕對不要丟一個 **zip** 壓縮檔要請對方幫你看，如果你做這樣的事情會被莫名扣很多分數，一定要切記，說眞的沒有那個美國時間解壓縮看你的 CODE。

mentor 重「引導」、非「教導」

因爲 mentor 並非會一對一教你技能，所以許多事情有疑惑時，大部分 mentor 都是會提供個方向給你參考。有些人都會以爲有可能獲得免費的教學那眞是大錯特錯。像是以我在引導學生時，我會大量問問題來讓他反思，讓他能夠自行尋求自己內心的答案，且有時候旁觀者清，又較能看出學生在意的點在哪裡。例如我就常反問學生說：

- 「你覺得這兩間拿到 offer 的公司都很好，會不會是這兩間你都沒有很喜歡？」
- 「你是設計背景，想要轉前端工程，但如果要你做設計、前端各半的工作，你會排斥嗎？」
- 「公司如果眞如你所說的那麼亂，但聊了那麼久你怎都沒考慮辭職的方向？是有什麼顧慮嗎？」
- 「對你來說，錢比較重要，還是成就感呢？」
- 「不管去新創公司、大公司都很好，如果兩個都猶豫不決，不妨讓自己在職涯過程中都去去看，看哪個比較適合自己？」

透過這樣的問答才能夠讓學生知道自己的眞正想追求的是什麼，所以有些人會覺得我這麼忙，竟然還有辦法線上輔導學生轉職感到不可思議，但其實我除了少數回應他們程式問題並確保所學觀念沒問題，依照他們的背景適時提供資源讓他們自己去消化外，絕大部分的時間也就是陪伴他們更加瞭解自己而已。

但要切記人生是你自己的，不論是 mentor 或是其他人都無法爲你的人生做決定，你要走的路是你自己要承擔風險的。若是你把這決定權交由給他人幫你做決定，那豈不就很不負責任嗎？所以我都會跟學生說：「我沒辦法下決定，若是我說了你得到失敗的結果，那豈不就會怨恨我？況且你自己下決定失敗了也沒關係，至少你自行承擔風險而努力過了，代表你也向前一大步，在以後人生會更有智慧地下決定。」

找出適合自己的 mentor

mentor 除了技術引導外，其實更看重關係的建立。有些人本身就很忙，不會主動關心你。但你問問題時，他還是都會回覆你。有的則是會主動關心你的近況，就端看雙方的個性調性如何，所以也相當看重緣分。有些又有點像是師徒制，師傅看出你的潛在性，自然會希望你能發光發熱，而徒弟也在觀察潛在師傅是否適不適合自己。

同時也會因爲技術與個人的發展，mentor 也會只短暫一陣子，或是持續很長一段時間，其實是很正常的，但也要記得受人點滴，未來有機會也能湧泉以報。

初級前端工程師的求職門檻

這絕對能被列爲我人生當中前十大最常被問到的題目之一，就趁這次機會來說明，究竟要達到什麼程度，才有辦法達到初級前端工程師的就業水平。

HTML、CSS 網頁版型是基礎功

初級(Junior)前端工程師在網頁版型上必須駕輕就熟，通常公司會聘請到前端，自然是因爲需要有一個專人來負責跨螢幕跨瀏覽器的工程師來開發他們的產品。所以你至少必須要滿足以下條件：

- 會考慮到各螢幕解析度，設計桌面、平版、手機都能夠自適應的網頁版型
- 開發前詢問產品 TA 族群習性，瀏覽器兼容性需要滿足哪些特殊條件
- 提供給企劃、設計師更好的網頁操作流程之可行性建議，優化使用者體驗。

至於 Bootstrap 的部分也建議你掌握起來，但在投入它之前，請你至少要使用 CSS Media Queries 有實作過一個多頁式的網頁，並可兼容手機、平版、桌機的網頁尤佳。因爲你的本業是前端工程師，你必須在深入框架前要對 HTML 標籤、CSS 有足夠的開發經驗，絕對不要變成只會用 Bootstrap，但超出一點 Bootstrap 額外的東西就不會自己寫 CSS 增加網頁彈性的話，那不是一個合格的前端工程師。

設計動畫效果點綴網頁樣式

爲了要讓網頁能夠與使用者產生互動，適量的動畫效果是絕對必要的。在初期時 CSS 必須要瞭解 transition、animation 的操作方式。但是若要能做到高彈性的互動，是絕對需要使用 JavaScript 來寫動畫效果的，如果你還不會自己刻，對 JavaScript 也還不會那麼熟悉，也可以先試著學 jQuery，就算不會 JavaScript，你也能透過他簡易的 API，自己設計出簡單的效果。

學會後也能夠自己去使用別人寫好的 jQuery 套件套用在網頁上，一些網站常見的動畫效果例如跑馬燈、圖片輪播、頁籤絕對要自己準備起來自己習慣用的插件清單，以備上戰場有足夠的武器應戰，如果你對熱門動畫套件不熟悉，之前我也整理了一份詳細清單，同時也曾經錄製兩個小時的jQ 影片教學，也提供給您參考。

JavaScript、AJAX、API 整合多方服務

接下來就是紮穩 JavaScript 基礎功，從最基本的變數、函數、陣列、物件你自然必須了然於胸，學 JavaScript 的用意是在於你能透過他去接收網頁瀏覽器上所有的資訊。例如說：

- 想知道當下使用者的瀏覽器寬度與高度
- 從 JS 去接收遠端伺服器的 API 資料再渲染到網頁上
- 在瀏覽器的網頁上設計 2、3D 圖形動畫
- 將訪客資訊記錄到瀏覽器上的資料庫

這些舉例只是小小的冰山一角，現在的 JavaScript 已經不只侷限在網頁瀏覽器，甚至能透過它設計 APP、桌面應用程式以及任何你想像不到的東西。

一開始在學 JavaScript 的朋友可能會覺得他是個無底洞，彷彿學無止盡是很正常的，但無論如何基礎功還是相當重要的，如果你是初學者，目前我會推薦你這本書，並試著做一些線上題目，例如 codecademy 上面的Introduction to JavaScript、JavaScript，我也曾經設計過一個JS 百題斬的題庫，若是你能答對 85 題以上，就代表你確實已經具備 Junior 的水平了。

但是若這樣還尚缺不足，在目前網頁趨勢來說，同時必須掌握 AJAX/API 整合的技術，以白話文來說明則是說在一個網頁上面，同時會介接三、四種服務已經是家常便飯的事情。舉例來說，假使我要做一個「高雄公車即時動線服務」的網站，首先你就必須接政府提供的公車動態 API，拉回來資料後再顯示在網頁上，如果你希望用地圖呈現時，就需要即時介接 Google Map API，假使你還需要做一個讓使用者儲存最愛資訊時，就可以用firebase設計註冊登入邏輯與資料庫設計，在公司做專案跟後端工程師配合時，也會搭配他提供的 API 來設計各種服務。

在這裡你所必須掌握的是閱讀文件的能力，你如何將你已經會的 JavaScript，在獲得這些 API 資訊後，顯示在瀏覽器上便是一個相當重要的課題。如果你聽不懂這段也沒關係，在我推薦的那本書上也有提到 AJAX 的部分，當你越加瞭解 AJAX 後再回頭看我這章節，自然會更加透徹。

Git、Github 是首要必學之路

Git 是版本控制語言，可以紀錄程式碼修改了哪些東西，當你不小心刪除、覆蓋檔案時，就可使用 Git 來進行還原。而 Github 則是來存放 Git 數據庫的東西。

你可能會想說，這跟前端又沒有太大關係，我處理檔案時小心一點不就好了？這樣就讓我來仔細爲各位解釋：

多位工程師協作同個專案，不用版控簡直找死

當一個中大型專案，可能就會分派兩個前端三個後端，後端 A 去處理金流、B 處理報表、C 處理系統架構，前端 D 負責使用者端、前端 E 主要爲後端介面。但一個專案該如何讓那麼多人協作程式碼卻又不打架呢？絕對是會仰賴到版本控制語言的。

我在學校教前端時，曾經有位被我教過的學生請我寫推薦函，一看公司大有來頭，雖然下意識覺得他應該不會上，但摒著年輕人總該去磨練經驗碰壁一下的想法，那時還是盡力地協助他。最後竟然意外被錄取了，後來試用期轉爲正職時，學生好奇問了帶他的主管爲什麼那時會錄取他。主管回他說：「其實那時面試一輪下來，你的前端能力只能算差強人意。但你 Git 卻是觀念最好的，我們也不怕你搞爛我們的 CODE，以前曾經有個實習生 Git 下錯指令搞爛 project .. 害我們一直有陰影。」

從這故事就可以看得出來，你要有機會進入到大公司或是有心要培養你的公司，你有辦法觀看他們的專案或試著參與，不會 Git 你就連基本的門票都拿不到。

所以在我門下的學生，我絕對會請他學 Git 再去投履歷，機會也相對大得多。

爲你自己的 CODE 把關嚴謹性

你可能會想說，我就只有一個人，哪還需要去做版控？這種觀念大錯特錯，Git 有辦法追蹤到每個檔案的每一行程式碼的更動，所以在你新增一個功能時，就可以進行版本控制。至少在你搞爛檔案時也能進行版本還原。面試官也會從你更新功能中觀看到你對程式碼的嚴謹度，例如程式碼縮排、命名規則、版控觀念來進行初步審核，所以自己盡早培養版本控制的觀念，你才不會進入公司後才在後悔當初沒學好 Git。

Github 是最大的程式碼開放平台

Github 上擁有許多開發者發佈的專案內容，你可以從中觀看許多程式碼邏輯，而你也可以上傳自己練習的專案上去，如果有開發者對你的程式碼有興趣，並發現一些問題，他也會主動發 ISSUE (問題單) 給你，或是佛心地幫你修改，並發 Pull Request (合併請求) 給你，你一定會心裡 OS 「so what？」，讓我講一個故事給你聽。

多年前我想要做一個 D3、C3.JS 的 SVG 圖表能夠匯出 png 格式並下載的功能，我那時候找到一個 angular 國外開發者有出一個完整的 sample，但因爲我是用 jQuery 寫的，於是在 Github 問他一些原理，後來我終於搞出來也回饋給他，他也很替我高興。但故事還沒有結束，在那之後的半年一年期間我仍然會收到各個開發者詢問我這段的原理，原因他一直在幫我宣傳我有實作出來，請他們來請教我，儘管我已經完全沒在理那個功能了。

從那時開始我才開始嚐到一些甜頭，透過這些 open source 能夠與各國的開發者接軌，在相互交流技術的正能量環境上，我的開發程式能力也大幅提昇！

如果想自學 Git 的話，這裡也推薦一些資源給你：

- 連猴子都懂的 Git 入門
- 30 天精通版本控制
- 爲你自己學 Git

接下來，開始爲自己找職缺

如果以上的條件你都滿足，就能開始找工作了，我自己也有一份前端就職關卡文件，你可以依照每個關卡條件來關關闖過，在後面也有提到一些履歷寫法與面試教戰手則影片，也提供給各位參考。

曾經有學生問我，老師感覺那些公司都是要找「有經驗的」，甚至會提到一、兩年經驗，那種可以投嗎？當然可以啊，如果職缺內容提到的技能你都有涉獵的話。公司在尋找人才會有很多面向，例如資淺但有潛力拉進來培訓，或你的能力已經有滿足公司的條件等等，老實講也蠻看緣分的，所以初期就當做是磨練面試經驗也蠻 ok 的。

如果不曉得如何觀察公司職缺工作內容，也歡迎找我來詢問，更多的面試細節，在後面章節也會一一介紹到的。

如何設計吸引人的前端履歷

履歷是你與職場接軌的重要門票，許多資深 HR (人資部門) 也曾經說過，他們看履歷只要不到十秒就決定你的履歷是否有參考的價值。就算你空有實力，但不了解該如何包裝自己，那也是枉然。那麼到底該如何寫一張好履歷呢？就讓我一一爲你解說。

留下個人資訊，尤其是聯繫方式

一個履歷上面自然要留下你的個人資訊，如果你擔心個資外流，也可以不要留手機，但至少要留 email 方便公司向你聯繫，我也曾經遇到學生啥都沒寫，導致公司完全不曉得該如何聯繫到對方。

- 姓名：
- 希望求職地點：
- 科系：
- E-mail：
- 手機：

設計專屬於自己的一頁式履歷

雖然現在 104、518 的電子履歷已經有既定格式，但我還是建議你必須要有一個一頁式履歷，除非你的經歷很多，才會需要兩頁 A4。用意在於有些技術主管、前端 Leader 主動到各大技術社群找人，同時也會留下他們自己的 mail 或聯繫方式。當你主動投履歷到 mail 時，就可以用一頁式履歷來投，對方也不需要登入服務就能觀看你的履歷。

另外請注意一點，你的檔案格式絕對要是 PDF 格式，絕對不要用 office 格式例如 word、excel，有些技術開發者沒在碰那些東西的，他們也沒那個美國時間去用線上服務轉檔，而 PDF 能夠直接就網頁瀏覽器打開，同時也能跨許多作業系統 (Mac、Win、Linux)，相對來說友善多了。台灣也有CakeResume設計履歷服務，也提供給你參考。

這是一個講求實力的世界，沒作品 = 沒機會

這是很現實的問題，有些朋友都說他們前端技術都有達到初級門檻，我稍微問了下程度也都蠻 ok ，但因爲沒有自己的上線作品，導致一直無法找到屬意的職缺，我只能說眞的活該。我們換個立場來思考，如果你是老闆，你遇到一個面試者說自己能力 ok，但是都沒有任何作品，你會想冒險讓他進入到你公司嗎？你不會覺得對方是不是在騙人呢？要記住公司如果看不到你的價值，一切都是在做白工。

作品，是讓你有機會將你所會的東西淋漓盡致地展現出來，就算一開始做得成品都很差怎辦？那也沒關係啊，你至少知道自己的問題在哪裡，再繼續優化到你自己認可的程度，許多事情都是萬事起頭難，跨過去後才有辦法進到下一步。

不論你是任何時期在找職缺，你想要拿到你屬意的薪水與工作挑戰，先思考一下自己有沒有一個能爲你技術背書的代表作品，如果有，自己也能自信地去投職缺，公司在看你履歷時也能初步瞭解你的程度，面試溝通上也能夠更加順暢。

所以在找工作前，先從作品開始準備起吧。

語言不是寫越多越好，重點在於讓面試官了解你的技術深度

像是我在幫朋友看前端履歷時，他們就很空泛地寫自己會 HTML、CSS、JavaScript，但是前端工程師都會這些東西啊，但是你到底透過它們實作做了哪些項目，以及比較擅長部分是什麼當然也要寫清楚，這樣面試官才可以評估你跟其他面試者的差異在哪裡。

舉兩個例子：

不好的履歷

- HTML5、CSS3
- JavaScript、jQuery

好的履歷

- Sass+ PostCSS 規劃響應式網頁版型 (連結)
- 善用 jQuery Plugin 整合第三方插件動畫效果
- Ajax/JSON 設計 SPA 架構 (連結)
- 具有介接 FB Login、Google Service API 經驗

相信從好的履歷就可看出來，你深入的程度到哪，假使是技術面較複雜的部分，我會強烈建議你要在技能旁邊加上個 連結，當 HR 認爲你 ok，進一步轉給 技術主管看時，他也可以從作品連結知道你的技術程度是否有達到他預期。

描述你在前幾份公司負責的項目

如果你已經有工作經驗，不要單純只描述你的年資，而是要敘述你在前公司負責哪些項目，而且這也是顯示你的價值的好機會，讓潛在公司在還沒找你面試時，就已經能預期你能爲他們公司帶來多少產能，當然寫得話也有一些技巧，同樣舉個例子。

不好的描述方式

XXX公司 (2016/3~2017/6)

- 電子商務網站設計
- 會計管理系統網站設計
- 財務報表前端介面

好的描述方式

- 電子商務網站設計
  - 優化效能 - 提升網頁載入時間 30%
  - FB 廣告程式碼追蹤 - 與行銷部門協作轉換流程
  - 響應式介面 - 單日 PV 10,000，行動市佔率 82%
- 會計管理系統網站設計
  - 優化記帳流程 - 前端介面優化，大幅提昇上稿流程
  - 前後端分離 - Vue.js + RESTful API 設計
- 財務報表前端介面
  - 圖形化報表 - D3.js 設計全球式地圖訂單量
  - 即時通訊 - Node.js + Socket.IO
  - APP 雙平台 - React Native 開發 Android、IOS APP

如果是你的老闆，你會希望選哪個面試者呢？答案應該很明顯了，就算 HR 不懂技術，但至少看得懂能夠被量化的數字，以及專案應用到的技術面，你的價值才會被彰顯出來，並被公司能夠產生聯想，讓你加入公司後，能夠注入一股源源不絕的開發能量。

假使你是剛初入社會的新鮮人在寫第一份履歷也是一樣，你的作品也必須詳細敘述到你接了哪幾隻 API，投入到哪些技術，這樣才容易會有面試機會。

履歷先後順序

一般從上到下的排序權重如下：

1. 個人簡介與聯繫資訊
2. 擅長的技能
3. 公司經歷 / 個人歷練
4. 自傳(最前或最後都可)

自傳字不用多，能夠描述你投入前端的近況，以及自己期望的未來發展即可。除了CakeResume履歷外，我尤其推薦Toptal的前端人才庫，去觀察他們是如何寫出自己的個人品牌價值，上頭所列到的重點都有列到。你甚至可以觀察幾個頂尖人才，評估自己預期要花多少時間追上對方，當做一個好目標逐步投入技術。

如果你是正在找第一份工作的前端，也可參考我這連結的第九關，裡頭有一些履歷範例。

魔鬼藏在細節裡

最後，分享一個檢核列表，在準備丟履歷前也請注意是否有犯以下錯誤

- 中英夾雜時，英文左右邊要留空白，不可與中文字黏貼在一起
- 技術名詞大小寫不要寫錯，常見的錯誤是將 jQuery 寫成 jquery
- 文字不要有錯字
- 超連結務必都能夠連到
- 不要寫到與前端沒有加分的項目，例如會 office 系列

前端面試法則二三事

當你的履歷被公司挑選出來邀請面試時，你該給自己一個掌聲，代表你的能力水平有達到他們的基本門檻，才會邀請你到公司深入聊聊。面試是個照妖鏡，它能夠讓你更加瞭解自己，你技術水平還差多少、你未來希望自己可以成爲什麼樣的前端、你憧憬的企業文化與產品項目也會逐漸被挖掘出來。

大家難免都會有第一次，有時候力求表現但又因不熟練面試過程，常會錯失許多機會，所以這裡分享一些面試時你需要注意的細節。

面試前，預期會被問到什麼問題

大一點的公司會請你寫性向、程式測驗，進入口頭面試後起手式通常會是：

- 你爲什麼想走前端？
- 問你各種前端議題，想瞭解你的程度

有些面試官會習慣把你問到倒，目的是爲了想瞭解你對技術的掌握度，所以要有心理準備會被問到自己不會的東西。我模擬面試過很多人，絕大部分會直言：「這個我不會」、「不太清楚」，接著就沒下文了，其實這是很要不得的。原因是工程師的其中一個必備技能是必須探索自己不懂的問題，儘管你不會起碼也要回應：「這部分我沒有涉獵，如果公司需要的話可以再研究它的 API 文件」，如果你很喜歡特定公司，可以把問題記下來，回到家研究完後，再發個感謝信給面試官，順道將解法回傳給對方，自然會對你印象加分。

再來還有重要的一點，在討論的過程中，請不要提到類似「我不會，但我願意學習」、「自己做的東西都是很簡單的東西」。換位思考下，如果你是面試官聽到這樣的話，不會覺得你還停留在學生時代，沒自覺自己已是社會人士，會下意識覺得請到你進公司會花很多時間教你，或是你只能做簡單的東西嗎？初進職場會擔心自己能力未達到就業門檻而感到退卻，但還是會建議不要刻意示弱，會多少東西就講多少話，也不要過於誇大，面試官才能理性客觀地瞭解你的程度。

想找些前端常見題目的話也推薦這兩個連結：

- 前端工程師面試問題集
- 面試細節 (Rplus Chen 整理)

要試著雙向溝通，勿讓面試官唱獨角戲

在面試的時候，絕對不要面試官說一句，你才應一句。你應該在交談的過程中，營造出讓對方覺得彷彿你就是他同事般討論工作的氛圍。例如說：

面試官：「你會寫響應式設計的話，那 Bootstrap 也 ok 嗎？」
學生A：「BS 也可以，請問你們有屬意的框架嗎？」
面試官：「我們後台都是用 BS，但我們公司前台產品之前用 BS 很不順所以都手刻，但也有在找適合的框架。」
學生A：「我也有辦法自己手寫 RWD，最近也有在摸 XX 框架，他跟 BS 比起來有哪些優缺點..族繁不及備載」

面試官其實也只是一般人，只是因爲立場的關係必須很假掰的和你對談，如果你能夠與對方產生雙向交流，將更有助於面試媒合。

要主動提問，瞭解公司特性

每當要結束時，面試官都會詢問「你有什麼問題想問嗎？」，請一定要準備些問題來問，你要瞭解今天公司在面試你，而你也在面試公司，也有選擇要不要進入公司的權力，假使眞的錄取了，但對公司狀態一無所知，進去後才發現跟自己的認知有出入豈不就得不償失？

會推薦的問題有：
「請問公司的部門架構怎麼樣？技術部門協作流程是？」
「前端主要會負責哪些項目？」
「你們如何考量個人績效，是否會影響到福利與年終？」
「有在做版本控制嗎？」
「技術部門都用哪些技術、工具、服務？」

光這些就可以瞭解許多公司的眉角，我常遇到很多朋友說他同時錄取三四份工作，但在猶豫要去哪間。詢問他每間公司的差異竟然答不上來，那就只能怪自己沒有溝通清楚。我會建議至少也要問到「前端主要會負責哪些項目」，這樣你才會瞭解自己做的項目是一個新產品開發，還是主要時間都是維護舊專案，或者一半一半。以及預期所用到的技術是不是剛好想要學的東西，那不就剛好讓你賺到？

除了有辦法讓自己技術成長外，你可以自己設計一個量表，在薪資福利、離家裡遠近、企業文化、個人技術成長、部門組成等五大項目，整理出個比較表格來考量，這樣當你獲得複數工作時，才較好評估。

這裡也分享一個提問問題的文章，裡頭甚至有針對軟體工程師、管理階層、公司領導階層進行提問的方向。

讓面試官看到您的價值

在商言商，公司請你進來是要你解決問題，讓這個體系因爲你這個齒輪加速運轉。在對談的過程中，你就必須以公司的求才內容與面試官的對談中推敲出他們想要哪一種前端。如果你很喜歡某公司，就能主動回應所提的工作細節與合作項目都曾經經手過，讓面試官信任你可

以勝任這份工作才是最重要的。

接下來就是談薪資的部分，我都會跟要面試的朋友說：「我不會跟你說你值多少薪資，你必須爲自己的能力來下籌碼」，所以我都會習慣請他們爲自己設計一個期望薪資，模擬面試時也會刻意問期望薪資，讓他們對當下的情境有充分的心理準備來應對。

結尾

分享兩部我協助學生模擬面試的影片給各位(影片一、影片二)，假使你正準備前端面試時，你可以打開此影片來自行練習面試，當我說一個問題時，你就按暫停自己試著說看看，等到最後總結時，可以再觀察自己是否有些盲點沒留意到，再加以改進。

最後再補充一些資源給有意成爲前端工程師的你/妳：

前端求職常見平台

- F2E TW
- inside
- yourator
- 104

面試前、面試後需要注意的細節

- 若要展示作品，請自己帶筆電，並考量網路狀態，可先將網頁打開，一打開筆電就能直接展示。
- 面試時，手機請保持關機。
- 有喜歡的公司可先排在後面，先排一些自己心目排名較低的公司來練經驗值。
- 事前記得對公司做些功課，溝通時也較能進入狀態。
- 面試結束後要隨時留意手機與 mail，別漏接到錄取通知的消息

挑選前端職缺時，要知道每間公司都是凹凸不平的個體

相信你還未就業時，一定對「業界」充滿想像，希望自己能夠進入一間公司發揮長才，然而事實並非童話故事般美好，我希望藉由這篇文章讓你瞭解初級前端工程師在台灣面試時或就業的各種生態，讓你在投入產業時能夠有個基礎的認知。

爲什麼我都找不到職缺，是不是我能力不夠？

找工作除了前端能力外，一間公司評估是否要拉你成爲員工會有很多細節要考量，如果公司發你感謝函，常見的原因也有可能是：

- 人格特質不符合企業文化
- 能力不如預期，太資深或太資淺都有可能
- 希望想找XX背景轉前端的開發者，不想找無經驗的 (XX請自行代入設計、後端、業務
- 薪水雙方談不攏
- 有給你口頭錄取，轉給人資後因爲臨時組織改組，所以決定不補人
- 有給你正式錄取信，過幾天技術主管覺得沒時間培養新人，於是想找能力較好的前端而婉拒你
- 有給你正式錄取信，要你先辭職，下週來報到，但最後卻因公司倒閉而兩頭空
- 老闆算過命後，發現跟你八字不合，他特愛找姓「陳」的 (眞人眞事

看到最後一個理由有沒有覺得很扯？但眞實世界上就是有很多光怪陸離的事情，只要有人就會有糾紛，人性會影響的事情多到你無法想像。

像是一開始投入前端的新人在找工作時，因爲每個技術環節都在平均成長，導致沒有特別突出的技能。不像是資深開發者有實務專案去特別磨練一項技術，例如 WebGL、AMP 等等，以致於企業需要評估你是否能在預期時間培養到足夠的程度，在考慮的細節也會比較多些。

但是能力好的學生就會比較好找工作嗎？也不盡然，在我輔導前端生涯中，有各種案例：

- 第一間公司就被錄取，儘管程度非常差
- 連面五間通通都上，不過能力只算普通
- 能力有培養起來，三個月投了五十幾間好不容易才找到
- 能力相當不錯的，卻花了半年才找到工作

有可能是大公司想組一個大前端 team，所以想自己拉新人來培訓，而那位花了半年才找到工作的學生，其實都陸續有拿到 offer，但對產品項目都很沒有 fu，不想刻意找公司濫竽充數，那我也覺得很好，從面試的過程中知道自己不足的地方，邊找工作邊把技能補起來，最後他也找到自己心儀的公司。

至於那個找了五十幾間的學生呢？就眞的是比較衰而已，他提的薪水和能力都在平均值，也不好高騖遠，但偏偏就是沒收到錄取信件。有可能剛好是搭在畢業潮，導致企業有很多人可以選擇。而且這些學生也不在少數，也曾內心透露很想在公司門口喊說爲什麼不用我，只能說除了技術外，也相當看時機與緣分。

但我仍會鼓勵學生說：「突破的鐵則是就是不停地精進自身不足的技術，才有辦法獲得更多機會。衝破這道牆，未來就是你來選公司，而不是公司在選你。」

不是你不夠好，只是時機未到。人生雖然不如意十之八九，但至少你的技術永遠不會離開你，背叛你。

大公司好還是新創公司好？

得到複數 offer 的學生常會遇到難以抉擇的選擇題，大部分不外乎有兩種狀況，一種是福利不錯，穩定的大公司，但並非研發而是維護固有產品。另一種是新創公司，技術更新快，較容易經手研發產品核心。

在雙方薪水差不多的情況下，諮詢我該選哪間公司？

通常替學生分析完後，我就會接著說：「你不覺得這兩個優缺點剛好是互補？跑過一輪面試你也清楚目前就只值這個 薪水區間，與其如此不如思考哪間公司能夠給你未來有更好的發展」

我都會建議學生在面試時，要好好詢問會投入的「產品項目」有哪些，然後再去評估是否有興趣，這樣做起來才能享受工作的過程。

再者，不要以爲你錄取後就如步青雲，因爲你還需要面對試用期三個月的關卡，考驗其實才剛開始，公司在衡量你是否適合它們的組織，你自己也在觀察公司。

不要盡信權威人士的建議

很多人心裡早已有答案，但爲了怕自己選錯選擇，常常會問周遭前輩的意見。有些甚至會直接問我說：「老師，那你是不是也會建議我到這間公司？」、「老師如果是你你會選擇哪一間？」

靠北我怎麼可能決定你的人生呢，如果我選出一間，好的當然沒問題，壞了你豈不就會怨恨我一輩子？

不論是工程師還是其它行業，你都必須培養出「獨自解決問題的能力」，而這部分另外種講法就是你就業後，就要開始嘗試承擔「選擇的風險」，爲自己的風險負起責任。

所以以過來人的經驗，頂多是陪你聊聊，或是用很多問句問你說：「那你自己覺得怎樣？」、「你自己現在比較想走前端還是後端？」、「薪水會是你考量的重點嗎？」之類的幫助你瞭解自己內心的選擇。

所以我都會跟學生說不管你是爲了「金錢」、「個人發展」、「離家近」、「同事好相處」等等的任一理由而做了選擇，我一定都舉雙手贊同。因爲你用了自己的力量做出了「選擇」，這就是向前跨出了很大一步了。

不要太相信當下面試時對公司的第一印象

當學生拿到複數 offer，都跟我說每個都超喜歡條件超好時就會在那裡天人交戰。我都會建議不要考慮太久，你眞的都超喜歡就寫個小程式亂數選一間吧XD

因爲我遇到很多學生遇到以下事情：
1.考慮很久最後因爲主管太機掰，做沒幾天就閃人。
2.也遇過進到一間感覺氣氛很嚴肅的公司，但實際進去後才發現和同事處得很來
3.面試講得很好聽，但進去後天天瘋狂加班救火

所以才會有所謂的「試用期」，這段期間公司在觀察你，你實際上也在觀察公司，如何眞的不合就別再浪費彼此時間，也不用擔心什麼「第一間公司很重要」，說眞的你技術與能力有沒有提升，能夠展現出來才是最重要的。

拋開工作能做一輩子的刻板印象

爲了避免所謂的十年經驗其實只是一年的經驗重複十年，你必須在每個時期都問問自己，這一季你在各方面有成長嗎？沒進步的原因是出在你自己還是在公司？如果某天公司撐不住了，你在外面是否仍有高度競爭力？

不論是前端還是各個領域，都逐漸被 AI 取代的時代，不要再以爲單獨學一個技能就能混個十幾二十年，有個短短幾年就很偷笑了。

最後總結兩句話給正在尋找職缺/選擇職缺的新鮮人士：
1.持續提昇競爭力才是王道
2.自己當自己的權威人士，不要養出別人幫你做決定的壞習慣

如何培養自信成爲前端工程師？

到底要磨練到什麼程度，才可以自稱自己是名前端工程師呢？在以前的年代會有師徒制，例如要成爲一位壽司師傅時會先進壽司店當學徒，在前幾年你只能在台下準備壽司材料，直到師傅覺得你可以，才有辦法站上料理台，再過幾年後你才能稱得上是一名壽司師傅並自立門户。

但以前端工程師來說沒有具體的指標，你只能自己摸索業界需求，藉由實務開發累積足夠的經驗，我曾經輔導過一些自信心嚴重不足的學生，儘管我認爲他們的能力都已經非常不錯，但還是覺得自己離專業有非常遠的距離，就如同我前面的文章提到有些人會對工程師有無限的憧憬與幻想，以致於會讓他們認爲自己一輩子都沒辦法成爲前端工程師。

還是菜鳥的我也是這樣子過來的，那時候我在工作前幾年也不知道前端，有一天發現有「前端工程師」職稱後，花了一陣子時間去摸索，最後才決定自己要成爲前端工程師，雖然那時工作已經四五年，但認眞說我是 2012 年初的時候才決定自己要往這方向投入，而是在 2013 年 3 月時才認爲自己已經有達前端工程師的門檻。

這個門檻我是做了很多事情，信心才一點一滴的建立起來的，這點非常重要，不是別人說你 ok，而是你發自內心地認爲自己終於跨過了這道艱困的障礙，那麼我究竟做了哪些事情呢，就讓我娓娓道來。

不給自己回頭路

在我決定自己要投入這方向後，我很清楚要讓自己隨時沈浸在前端領域，我才有辦法成功，在那時候剛好我讀在職碩士，於是我就毅然決然地和我的指導教授說，我的研究題目要跟前端有關，於是我挑了「響應式網頁」當做主題來研究，響應式網頁是在 2010年時被國外開發者提出這概念，說眞的在 2012 年並沒有太多文獻可以參考，而且並不是那麼多人知道響應式網頁。

但我很清楚這將會是未來前端的風潮，於是說服了我的教授讓我做這題目，在那兩年求學過程中，我花費了相當大量的時間去研究響應式網頁與前端技術。我甚至向公司說明這項技術，讓他們有信心拉案源進來開發，晚上在學校時也不時跟教授 meeting，週末寫論文時，又能順便提昇技術能力，實在一舉兩得，最終我也寫完全台第一份 RWD論文並順利畢業。

在實務、學術與畢業時程壓力之下，我彷彿進入精神時光屋般大幅提昇自身能力，也在那時奠定了我的前端基礎。

透過寫BLOG、遠端教學提升自信心

教導別人也有助於自己觀念的提升，我們在業界有個說法，如果你沒辦法將原理講得直白易懂，那代表你仍然不熟原理，在那時雖然念得是國立大學，但學費對我來說是個不小的負擔，於是我動了腦筋在一對一教導網頁設計上面，透過 teamviewer + skype 來教台灣各地的學生，在教學的過程中我也開始擁有「原來我已經有所小成，並可以教人的程度」而產生些許自信。

所以也會推薦你寫 BLOG，你可能會覺得自己寫得東西沒什麼價值，放上去怕丟臉，這種觀念就錯了，寫技術文章的好處是在你寫的過程中，也在幫助你重新複習觀念，當你寫有小成時，也無形塑造個人品牌形象。不要以爲你寫部落格沒有人會看，一定會有跟你相同程度的人想要從 Google 上探索答案，那這樣你的文章就剛好幫助到他了，我們在業界也常會說取之於 Google、用之於 Google，你 Google 到前端資訊幫助你學習，在感恩之情上你也可以寫技術文章幫助到大家。

不停地挖坑讓自己跳

那時我還是覺得自己跟業界離得很遠，我不希望自己是閉門造車，想看看其他開發者跟自己到底差得有多遠。在 2012 年時前端社群非常得少，唯一比較具有知名度的是台北的RGBA，所以我一直很期待是否有誰能夠跳出來，也在高雄舉辦開發者社群。但一直沒有消息。後來同事問我爲什麼不自己在高雄辦一個？雖然當下覺得自己不夠格，但那時還是硬著頭皮去試試看。

因爲擔心沒有人參與，於是我拋磚引玉地運用自己會的技術開了響應式議程，因爲那時我很擔心該不會高雄根本不知道前端開發吧，這樣怎麼可能創立社群？就連企劃部的同事也譏笑我說你辦付費講座最好有人來，就連免費講座都不會有人了，但我的想法很單純，如果眞的沒有同行那就代表自己一廂情願，頂多以後跑台北參加前端活動。

但活動開放售票後不到三天就完售了，那時也給我非常大的信心，也促使我順勢創辦了高雄前端社群，終於我也能在在地找到同行交流，並更加接近業界。

在那之後我舉辦第一次的高雄前端聚會活動，我印象非常深刻，當下我的開場白是：「大家好，我是洧杰，我在公司擔任前端工程師」，說完後雖然表面不動聲色，但心裡澎湃不已，眼淚甚至差點流下來，因爲花了好多歲月，我終於可以發自內心肯定自己是一位前端工程師，直到現在我的奇幻之旅仍還在持續當中，但和以往不同的是周遭已有許多在地夥伴和我一塊並肩作戰。

從我的故事中你可以看得出來，若要投入一門技術，並達到「師」的等級，自然要找許多挑戰讓自己克服並累積信心，這樣在未來的有一天，你也能夠肯定自己。

最後我列出常見的成就里程碑，你可以從中去投入一道道關卡來提昇自己的自信心。

- 參加 IT 鐵人邦撰寫技術文 (我參加了三屆)
- 獨立寫出一個網站
- 找到第一份前端工作
- 工作經歷滿一年
- 完成一個相當複雜的前端專案
- 能夠自行研究一個新前端技術
- 別人誇耀你的前端技術不錯
- 你的前輩肯定你的實力
- 業界大牛轉貼你的前端論點
- 論壇上分享自己的前端經驗給想進入前端領域的後輩
- 曾經上台分享技術經驗
- 不需要問人，你自己就能找到前端解決方法
- 當有人問個問題時，你能夠秒解回覆，因爲這個地雷你曾經卡了一個禮拜

- 進入到一間薪水福利好，有名到大家都認識的公司
- 進入的公司 title 有幫你掛成 senior 資深前端工程師
- 寫的部落格曾經被轉 PO 分享
- 社群聚會上，聊天內容不會覺得同行說的話不是火星文
- 可以很油條的說：「想當初我們學這個東西可是花了超久時間...現在資源多好學多了」
- 看到別人張貼的職缺，掃了一下覺得自己好像都有符合條件
- 第一次下 Github PR，對方也 merge 的時候
- 有人在你 Github 寫的 open source 給你星星
- 一直接收到獵人頭信件，想要幫你尋找適合工作的時候

工程師是否有年紀背景限制？

想想我來寫這個題目還眞是不太夠格，2017 年的我今年 31 歲，科系的話大學是 3D多媒體設計、碩士念資管，所以也算一半的本科系，所以這個篇幅，至少我能向你分享我所遇到的前端們，儘管非資工、資管科系，但也憑藉著一步一腳印深耕技術，最終也如願成爲前端工程師。

各行各業轉職前端

半路轉職需要不小的決定，絕大多數是認爲現在的工作非常無聊，而且他們也能夠看出在那個行業的薪資天花板，所以想試試看工程師這行業。

我最常被問到的是：「老師，我做了三年XX業，現在轉職工程師可以嗎？有沒有跟我有一樣的例子轉職成功？」

就我輔導到現在有遇過像是大樓管理員、平向設計、企劃、服務業、業務、建築、金融、旅遊業等等你意想不到的行業，你可能會想說，那他們去面試的時候，不是履歷一看到你之前做了好幾年非相關產業，不就直接被打槍？其實並不會，這些人會有一些優勢，例如說：

滿足T型人才需求

很多人都會認爲轉職就要拋棄你前份工作的經驗，這種想法完全錯誤，現在各行各業都會有資訊化的需求，就連傳統產業也是。我曾經輔導過一位在金融產業工作三、四年的學生，當她技術有所小成開始準備投履歷，一直很擔心自己沒辦法得到面試機會，我就建議他可以先將履歷開放，自然會有人對你背景有興趣。

最後竟然有間銀行邀請她去面試，原因是他們的內部系統需要大改版，需要有一個金融背景的協助開發，而在同時的時間，有個前端同行也和我要人，做的內容也是跟會計有關的服務，只能說吸引力法則眞的很有趣。

另外的優勢在於在面試時如果競爭對手程度都跟你差不多，但你剛好曾經有做過該產業的 know how，自然加分不少，所以在尋求前端機會時，也能夠針對你的前份工作經驗再寫一份客製化的履歷，假使剛好投遞的履歷剛好是相關產業，獲得面試機會自然就大得多。

所以絕對不要嫌棄以往你的經歷，他有助於你找到適合的職缺方向，讓身上的十八般武藝都具有發揮長才的機會，讓未來的你擁有更多武器作戰。像是我就有學生原本做旅遊業的跑去雄獅，做保險的跑去 xx 人壽 擔任前端工程師XD

具有足夠的社會歷練

有工作經驗的人在溝通應對上會比較得體，這也助於面試上具有一定的優勢。最常見的就是我在幫模擬面試時，剛畢業新人沒有社會歷練，過程會比較生疏不自在。但如果你的工作時常會需要「溝通」的話，這段自然是駕輕就熟。就我協助的樣本數，有兩個職業的轉職成功率相對較高，一個是服務業、另一個是業務，很讓人意外吧？也分享給你他們的求職經驗談。

家有老小轉前端

我常會到遇到一些朋友，加社群好友後，就會急迫地詢問我各種事項：
「如果我花半年至一年的時間，有辦法成爲前端工程師？」
「工作薪水低，養不活一家老小。」
「工作十年的公司準備要收起來，想尋求新的方向。」
「在家沒辦法集中精神學新的東西」
「很久沒有學東西了，怕自己做不來」
「我不知道自己能不能成功」

絕大多數有家累的人，除了原本工作無聊想轉職外，主要都是因時代進步而逐漸萎縮，只好另尋求職管道，但又因特定行業做得太久，沒有接觸新事物的習慣而感到障礙，例如報社、DVD 書店等等。

在想要轉職又有家累需要維持家計時，行事也會特別小心謹慎，因爲對他們來說，要做就必須成功，因爲生活已經不允許他們有犯錯的機會，我自己本身現在也是一個小孩的爸爸，所以非常能體會，所以這裡我也分享一些建議給妳們。

沒有人能保證你絕對成功，除非是詐騙集團

「絕對不會有所謂的保證就業這檔事」，我在協助轉職前端的這群人當中，也只有逼近五成的學員有轉職成功。有些人中途覺得那不是他想要的，那其實我覺得也很好，至少你認眞嘗試過認爲這條路不可行，再找下條路即可，這些曾經投入過的經驗也會帶領你更順利探索自己想要的工作。

時間管理比任何事都來得重要

人越長越大後時間也會被分散得很零碎，再加上父母年長容易有病痛，又必須兼具家庭事業與瑣事，所剩下的時間也是所剩無幾，就連我也是一樣。但也因爲如此，你應該要先學得不是技術，而是在時間管理。

像是我前面介紹的番茄工作法就是一個掌握零碎時間的小幫手，有些人下意識就覺得「做一件困難的事情要花幾個小時」就會懶得動，但如果是只有單純一顆番茄 25 分鐘的時間，便自然能夠打起精神來完成。先學著將困難的東西碎片化，再將碎化的任務利用番茄鐘觀念逐一攻略即可。

另外請不要一個禮拜只抽出一整天的時間來練習，而是應該要求自己每天都要花一小段時間平均練習，這樣好處在於你能夠隨時浸淫在程式世界，投入的效果才好。

要有心理準備隨時面臨新事物

在這個產業上你要有「隨時將自己踏出舒適圈」的心理準備，每隔一陣子你又會看到許多新技術取代原本你會的東西。但如果你養成這個心態，就沒什麼事情難得倒你了。只要你遇到一個新問題不會將他當做困難，而是將他視爲一個「躍躍欲試的挑戰」。

小結

如果這段背景和你/妳相近，也歡迎你來找我聊聊，我身邊也有一些朋友最後有轉職成功，也能幫您約時間與他們聊聊，畢竟背景相同也能貼近彼此的想法，例如這位朋友便是。

無資訊背景，自認邏輯不好轉前端

我曾經跟一位學生辯論「邏輯培養」的問題，他一直嫌自己邏輯很差，好像永遠都沒辦法培養起來。後來我向他提議做一個 todolist 來練習。

學生：「老師..做這東西沒辦法找到工作吧，而且我有點放棄了，每次寫到一半就都想不出來該怎麼做...」
我說：「凡事都是都基礎打起的，而且你幹嘛小看 todolist，它能做得事情可多了」
學生：「...(一臉不相信」
我說：「這樣好了，你先做新增跟刪除 todolist 的練習，做完後我會幫你改寫，你就從我寫的內容去看我們的差異」

因爲我很瞭解她的背景，所以我都會將程式碼用七成看得懂三成她沒用過的觀念來提升他的程式視野，她再自行找資源瞭解這些程式與邏輯上的開發，之後我們又接著做：

- 增加「編輯」的功能，並顯示目前的當下時間
- 增加代辦類別功能，例如「全部顯示」、「已處理」、「待處理」
- 增加 localstorage 功能，讓代辦事項紀錄起來，打開瀏覽器也看得到
- 研究 Firebase database 資料庫功能，用 JS 去設計資料庫邏輯
- 瞭解 FB API 邏輯，整合 Firebase Auth 功能，設計一個能夠讓綁定會員的 todolist 服務
- 將程式上傳到 heroku 主機
- 瞭解 chrome 插件設計邏輯，並讓 插件服務與網頁版能同時整合
- 將程式整到 Electron ，設計 Win、Mac 雙平台 APP
- 介接第三方金流，讓服務能夠進行收款

當學生做到中間，我就說：「這樣你不會再小看 todolist 了吧？」
學生：「眞的...原來一個 todolist 可以做那麼多東西。」

所以當你在研究程式時，就像我前面所提到的，不曉得怎麼寫的時候，先去 Github 上找一個「比你會寫一點的同學」，能看懂七成，其他的三成就可以自己研究。在研究對方的範例時，你的邏輯才會逐步地培養起來。

另外在籌備前端項目時，去參考一個成熟的服務也是個很好的學習方式。例如上面的 todolist 列表，就可以參考這個服務，去使用他的功能，並思考如何從小做到大，才能讓你的技術能夠更加全面。

不惑之年轉前端

年紀是否會影響到前端轉職呢？我都會拿去年受邀參加中國 CSS 開發者大會時，在機場上遇到的賀師俊來勉勵各位，他是在中國極有知名度的前端開發者，其他開發者都稱他爲「賀老」，原因也是他的歲數超過 40 歲了。在兩天的旅途中也時常交流前端技術，更讓我驚訝的是他的前端思維並未因年紀減退，反而有遠增的趨勢。研討會上也不乏遇到許多 6、7 年生在相互交流，聊得不是自己自己歲數大了怕找不到工作。而是最近研究了什麼有用在專案上，或是覺得公司很差，準備要跳槽。

所以不要讓年紀下意識影響你的技能學習，工程師並不是體力活，你所要做的是就是持續更新你的大腦，讓它隨時跟上時代潮流，當科技有所變動時，你也能因應時勢來適應社會。

延伸閱讀

- 程序员工作只能做到 35 岁吗？之后的路是怎么走的呢?
- 日本82歲老奶奶開發app

打破工程師就必須加班爆肝的迷思

我是「不加班主義者」，你的人生已經有八小時奉獻給公司、八個小時睡覺，剩下八個小時扣除掉吃飯、睡覺、洗澡、通勤的時間，你認爲還剩下多少？或許連五小時都不到，那麼你爲什麼還要加班壓縮自己的生活品質呢？我認識很多人都會對工程師一定會加班爆肝有刻板印象，所以這裡我要告訴你如何面對加班這檔事。

我很菜，所以要自主加班將不足之處補起來

這點不對，你要清楚知道公司錄取你這新人，表示你值得投資，並在試用期內培訓你成爲戰力。而在三個月試用期的過程中，他們也在觀察你的能力，如果不如預期那就掰掰找下份適合你能力曲線的工作。

你可能會想說：「老闆給我機會，我當然要好好把握，盡早掌握公司的事情啊？」，摒除掉一些慣老闆壓榨員工加班的黑心公司外，老闆眞正看重的是你的工作能力，你應該在工作時間內展現出你的產能，什麼是產能呢？白話一點就是公司交辦給你的事情都有如期完成，那就代表你有符合他們的期待。那假使沒有呢？有兩種情況，一種就是公司的任務超出你的能力太多，另一種就是你還沒不熟悉如何控管工作時程，溝通過少。

較常見的就是希望讓一位只會基礎 JS、jQuery 的初級前端，在一個禮拜內瞭解 **Vue**、**Angular**、**React** 的框架，並完成專案。我可以告訴你這根本是不可能的事情，這些高級 SPA 框架並不只有前端那麼簡單，你要瞭解後端 API 邏輯、整合第三方 API、生命週期 (init、offline)，更別說還需要瞭解各種抽象的程式觀念。光學皮毛就很吃力，怎麼可能還能立刻用在專案上呢？

但一般許多接案公司或是非資訊背景的主管級根本不懂這些，在囫圇吞棗地狂接案子時，都會認爲跟前端有關，聘請一位前端就可以搞定一切事情根本大錯特錯，我許多同行也因此被搞得不勝其擾。明明原生 JS 就可以搞定得事情，因爲甲方要求要用 Vue.js 設計，就被逼得要在不可能的期限生出來，如果當下明講這期限沒辦法做到，要公司向甲方告知期限必須推延，有可能還會被公司質疑工作能力，甚至因此被資遣的也有，我眞想幫這些可憐的前端講他們的內心話：「眞是操你媽的**B**有理說不清，有種你自己來寫寫看！？」。

主動提出建置時程

當你接受到工作任務時，首先必須先確認這東西要何時交付，如果主管沒提出，你也要自己提問，或主動告知這大概何時會好，讓上層能掌握專案狀況。我常遇到新人因爲不了解如何控管工作時程進度，而吃了不少虧，這裡也分享幾個要點給你。

不要拖到最後關頭，才說你做不出來

很多人遇到複雜的專案，你跟主管都不曉得如何壓時程，於是主管只好先給了你一個時間點。投入幾天後才發現會 delay，但自己又抱持著這幾天加班應該有辦法的逃避心態，結果到最後一刻主管跟你要東西的時候，你交不出來不被罵到臭頭才有鬼。

你可能會想說：「我已經很努力天天加班了啊，但怎麼還是一直被公司罵？」因爲公司要得是結果，沒有依照時程產出，你就會被主管責備，主管又必須被客戶抱怨，頻率變多後客戶就不相信公司，流失客戶公司賺不到錢，大家就一起完蛋，你自己看看這循環有多嚴重？切忌不要犯這個錯。

沒把握的東西先進行初步評估

如果上層交辦的東西比較複雜，一時之間無法評估時程該怎麼辦呢？當下不要回可以或不可以，較好的方式是你跟主管說先花半天、一天的時間投入研究，就如同前面提到的將事情給碎片化，瞭解核心功能所預估的工時，再回報較準確的時程會較好。

確認事情輕重緩急

有時候你在公司可能不會只有一個長官，可能會同時會有 N 個人交辦工作或插件，但請不要因此而增加你的工時，舉例來說如果 A 主管給你一項工作，期限就在下班前，而你評估花整天的時間才會勉強趕上，這時 B 主管在中午時也交辦你一個項目，也要你下班前完成，工時大概也需要半天左右。那這時該怎麼辦呢？

- A：加班
- B：西瓜偎大邊，看誰的官威大，另外個主管就放生
- C：告知 B 主管 說 A 主管也需要交件，時間有衝突到，三方討論轉圜方式。

我想答案應該很明顯了吧？很多人都會下意識選 A，但爲了要讓公司能夠在有效時間內增加最大產能，自然必須討論出哪個代辦事項才是最重要的，讓公司能夠有效推動才是正解。拜託不要當爛好人，什麼事情都答應，最後只會造成無限負面迴圈，還會讓長官以爲壓榨你是理所當然。問問你自己，你犧牲生活品質爲得是什麼？有加薪水加比較多又有加班費嗎？你滿意你現在的生活嗎？

溝通溝通再溝通才是王道

最正確的方式應該是密切保持溝通，不要畏懼你的上司，你要讓你的公司賺錢，就是要確保自己現在做的事情是目前公司最需要的，讓公司擁有更多銀彈去作戰。你做得任何工作跟進度都要密切保持溝通，例如說：

1. 你有可能會 delay，那就提前一兩天說，讓主管有心理準備去因應
2. 被接收到過多任務，就反應詢問哪些項目是最重要的
3. 被主管壓了不可能達成的任務，即時分析哪些功能是否能暫緩，先將核心項目能如期做

完

4. 下班前告知自己的進度，並預期在何時能夠完成下一階段的工作
5. 藉由部門與上層溝通，讓週五能夠能完成階段性的目標，讓你能安心休假

絕大部分的朋友還會問我，同事跟主管是超級機八毛的人，EQ 超級低，完全無法溝通該怎麼辦？啊就辭職啊，不然怎麼辦？如果你是持續補齊你不足技術的地方，在台灣根本不愁找到工作。但如果你心裡抱怨，但又一直留在公司，反而要思考你是不是能力一直沒有自主提升，自覺能力不夠不敢跳槽？所以提昇自己的能力，選擇下一個能夠發揮你的長才的的舞台才是王道。

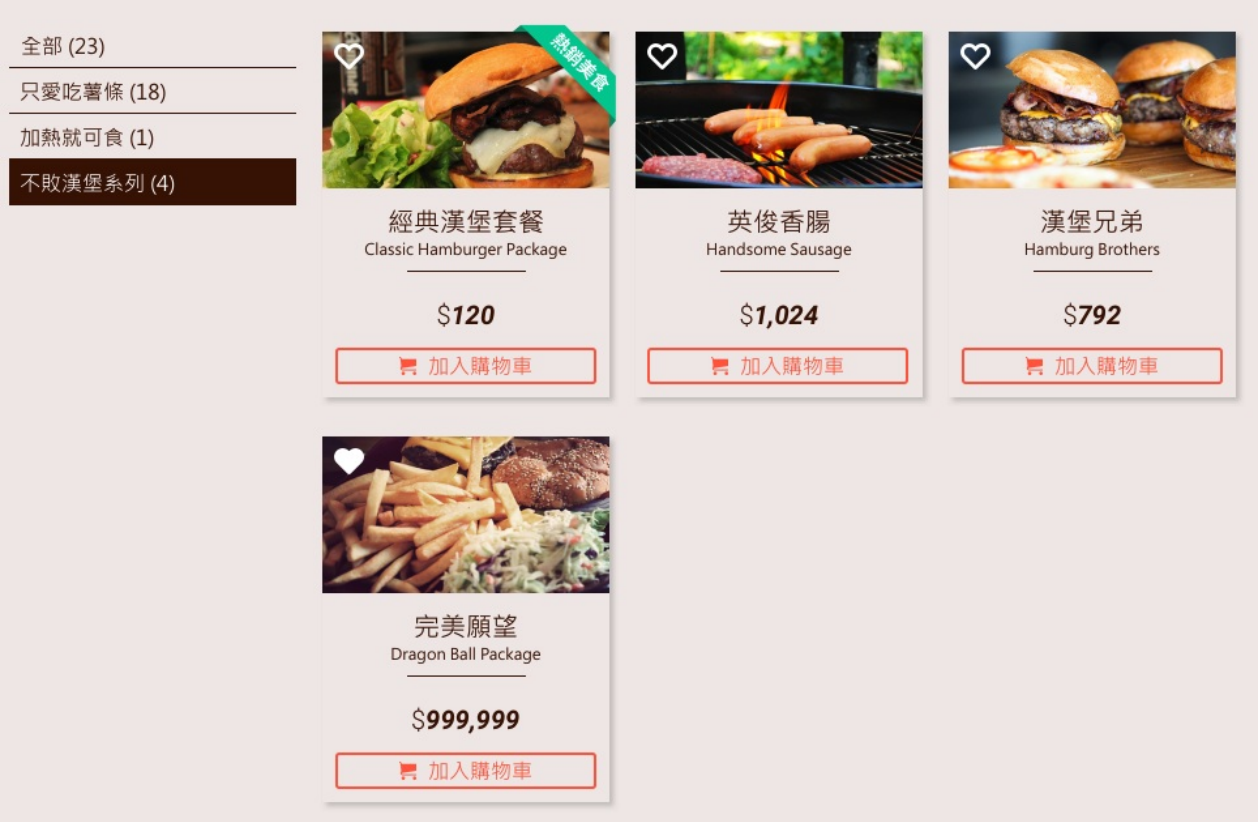
不要讓加班養成習慣

是你工作能力差才需要加班？還是你習慣性地加班，心裡總想著反正會加班，事情就慢慢做？或是老闆希望員工加班，你演戲給老闆看？問問你自己內心的答案，真的，不要加班。你人生最多活到一百歲，你沒必要花那麼多時間投入在工作上，還是你真的覺得你奴性很重？人生有很多值得投入的地方，如果你想不到現在可以開始挖掘，不要窮到只奉獻給工作。

教你開發出不會被後端吐槽的前端版型

當 UI 提供給你設計稿，你轉換爲前端介面後，下一個步驟就是轉給後端工程師開發成動態應用程式，有些剛涉入前端工作沒多久的新人在不瞭解後端邏輯時，在溝通上容易有障礙，分享幾個觀念給各位，讓你減少被吐槽的機會。

我將會以下圖的產品列表當做案例解說：



設計版型時要考量當內容變多時，也能自適應延伸

以右側產品列表爲例，很多人都會設計這樣的結構，每個 li 都有客製的 class 與對應的 CSS，請不要這樣，因爲你根本不清楚右側產品列表會有幾項產品，假使會有 20 項以上你的網頁不就掛掉了？所以你必須在 CSS 選擇器上在 `.product li` 來指定樣式，同時確保在內容變多時也能自適應從上到下排列才是。

你可能會想說，我跟企劃討論過，這裡一頁只會出現 20 個產品，那我寫客製化 20 個 class 不也可以解決問題？但這樣你會搞死後端啊，後端在從資料庫撈資料時，還必須針對你這個 class 名稱去客製化，每新增一個產品就要數字加一，無形之間也增加後端的 loading，如果不想讓後端討厭你，會跑迴圈的資料請盡量讓裡面的 class 保持固定名稱。

```
<ul class="product">
  <li class="list-1"></li>
  <li class="list-2"></li>
  <li class="list-3"></li>
  <li class="list-3"></li>
</ul>
```

語意要寫清楚，不要全部都寫 **div**

以右側產品列表的其中一個產品，如果你寫成這樣，後端一定會討厭你，或是默默將你列入黑名單。

```
<div class="product">
  <div class="box">
      <div><img src=""></div>
  <div>經典漢堡</div>
    <div>$120</div>
<div><a href="#">加入購物車</a></div>
  </div>
           <div class="box">
      <div><img src=""></div>
  <div>經典漢堡</div>
    <div>$120</div>
<div><a href="#">加入購物車</a></div>
  </div>
</div>
```

縮排不正確，看不出來誰是第一層誰是第二層，然後 class 推敲不出來前後關係性，HTML 標籤完全看不出語意，然後亂下 div，不管任何元素全部都用 div 搞定，以前的我就是這樣排版給後端被念到臭頭，較好的方式是下面這樣。

```
<ul class="product">
  <li class="pooduct-item">
    <img src="">
    <h2>經典漢堡</h2>
    <span>$120</span>
    <a href="#">加入購物車</a>
  </li>
  <li class="pooduct-item">
    <img src="">
    <h2>經典漢堡</h2>
    <span>$120</span>
    <a href="#">加入購物車</a>
  </li>
</ul>
```

自己比較看看，如果是你，會希望拿到哪份前端程式碼呢，答案應該呼之欲出了吧。

瞭解樣版語言

請去瞭解一個樣版語言(template language)，當後端拿到你的前端程式碼時，也會拆解版型，將共用的版型獨立成一個檔案，例如叫做 layout.php，其它頁面都會載入到 layout.php，舉例來說，你做了 30個網頁，如果表頭要新增一個文字，只要改那隻 layout.php 就好，不需要改30頁。

如果你不懂後端，至少要自己用 gulp 學一個樣版語言，這樣當後端問你哪些範圍要設計成 Layout，你才知道該如何回應他。叫常見的樣版語言有pug、slim、Haml。

前後端合作細節

以 API 爲例好了，有時候後端在組 JSON 時，可能因爲一些限制，導致較難從資料庫整出你想要的結構，這時就必須討論彼此都好做的方式。

但還是會有些東西是前後端都必須做的，就是「資料驗證」，例如填寫 Email 格式，當前端就已經寫錯，就必須用 JS 回饋他寫錯必須重寫。有些人都以爲前端寫完，後端就不用寫了，這是錯誤的。因爲使用者的瀏覽器五花八門，它就是有辦法繞過你的 JS 去傳送，或是用像是 Postman、curl 方式去傳資料到後端就能繞過前端，不可不防。

有些時候一定會遇到些問題是前端比較好處理，後端比較難做的時候，亦或相反，這時就相互溝通，彼此協助提昇工作效率。

投入一個後端語言

有很多朋友聽完我以上的建議，還是覺得他們跟後端的問題相當多，我個人建議你就試著學一個後端語言吧，這眞的才是一勞永逸的方法。希望用 JS 寫後端可以選 Node.js，或是 PHP + MySQL、RUBY、.NET + MSSQL。

要你投入不是要你成爲後端工程師，你是爲了以後能夠跟每位後端合作都能溝通順暢才投資的。現在學習資源相當豐富，如果看書看文章都覺得不夠，現在中文線上課程也很多，每天看個三十分鐘，不到一個月練習就能瞭解大概的輪廓，最起碼瞭解到後端語言如何與資料庫介接，並顯示到網頁上就相當足夠了。

Angular、Vue、React 框架選擇是個僞命題

常會有剛投入前端的朋友問我：
友人A：「洧杰，你覺得該不該學 XX 框架？」
我：「你說最近剛出來的那個啊，那你知道他是做什麼的嗎？」
友人A：「不知道欸...只是最近聽很多人推，感覺不學以後會找不到工作」
我：「.....」

也常被問到相同性質的框架來比較，尤其是 Angular、Vue、React 的提問率最高，但眞的問他說你自己有研究他們的用途嗎？竟然都一問三不知，眞的會讓我白眼翻到後腦杓去，這裡就講解一些身爲前端框架正確的應對觀念。

學技術是要來解決問題的

學任何技術時，先想清楚學了以後要拿來幹嘛？這目標越具體越好，例如我說 HTML、CSS 就是要自己實作一個個人履歷網站，我學後端跟資料庫的目標是要用他來實作一個留言板、部落格、電子商務。學 jQuery 是爲了能夠設計動畫效果，該如何載入別人寫好的第三方插件。學 gulp 是爲了讓自己的前端任務流能夠更加靈活。要切記，不要爲潮而學，要爲解決實作上的目標而學。你敢面試的時候跟面試官說因爲他很潮所以才學嗎？當然是要講你學這個技術是爲了要實踐某個功能，才不會讓人覺得你是來亂的。

在投入一個技術前，先找個具體的目標跟理由說服自己，再去花時間投入，心理才比較踏實。

先找成熟框架，站在前人的經驗上加速學習

例如你想瞭解 SPA (Single Page Application) 觀念，假使找一個很新的框架，你搜尋問題完全沒有人曾經踩過地雷，學一個觀念要卡非常久才有辦法進行到一步，學習成效絕對大打折扣。

那成熟框架有什麼特點呢？不是 Github 上星星比較多，更不是你周遭朋友都說他很夯。而是在你在搜尋該框架的時候有很多資源可以參考。例如打個 XX.js 在 Youtube 上就有很多系列教學影片，書城也出了很多技術書籍，在你遇到各種奇怪地雷時，關鍵字搜尋 console 裡面的錯誤就有一把抓的苦主幫你寫好解決方式，stackflow、Github 上有各種範例讓你參考。

使用成熟框架的好處就是能夠站在前人的踩雷經驗上，快速熟悉在建置一個 SPA 網站所需要的觀念，以目前 2017 年來說，你學 Angular、Vue、React 都可以，因爲都是業界挑選 SPA 框架的三大首選，每個都已經非常成熟。

如何評估框架的汰換

看到一個新框架時，我會思考以下問題：

1. 已經會一個類似功能的框架
2. 已經會了，它跟新框架優劣差在哪？
3. 它是否能解決目前開發上時常困擾自己的問題？

我們再以 SPA 爲例，我自己是先從 NG1 開始，再來投入 Vue.js，我 NG1 用了很長一段時間，中間雖然也出了不少框架，但功能都是大同小異，沒有太突出的功能，而最後會讓我換新的 SPA 框架的主要原因是在於 virtual dom 能夠大幅提昇效能，實際的 DOM 操作很耗效能的，更別說當資料量一大時更是讓我非常頭痛，必須東刪西壓縮才有辦法滿足客户的要求。以致於那時我才決定使用有支援 virtual dom 的框架，以滿足我當下開發時所遇到的困難。

在公司裡可以使用 **SPA** 框架嗎？

前端只有你一位嗎？如果是倒也沒關係，在工作時程沒那麼緊的狀況下嘗試用新框架很 ok，但如果是有一個前端團隊，甚至有技術主管呢？那你就必須要掌握到有一定程度的熟悉，或者是寫一些 side project 提升經驗。當同事或主管問你爲什麼要汰換框架時，你如果有足夠的自信說服他們，不論是開發效率、程式碼維護、框架彈性上都能說出贏過你們目前用的框架才會有機會。假使你自己都不熟也沒把握，想推動新框架難如登天。

當你是新手在找工作時，要特別去看公司工作內容與他們有用到的框架，如果你到一間 JavaScript 只會用 jQ 接一些動畫，後端也不是用 API 協作，工作內容反而比較常投入在 web layout 上，那你就根本沒有用 SPA 的機會。想要透過公司加強自己能力時，務必將公司所投入到的技術問清楚，不要不好意思，才能幫助你能否在公司有效提升前端能力。

先學深，再學精

我常遇到很多人在初期挑選框架就花了很多時間，彷彿是人生中最重大的決定，一直很擔心選了但之後框架不流行會後悔萬分。只能規勸你醒醒吧，沒有任何框架可以讓你吃一輩子的，你需要的是盡快投入一個新框架，並把它摸到很深，有實際做出幾個 side project，將這些開發經驗累積起來做爲你未來的籌碼。最要不得的是每個都只摸皮毛就放棄，你勢必會遇到很多問題，但解決未知的問題不就是工程師所需要必備的技能之一嗎？就如同佛法上常會講的：「面對它、接受它、處理它、放下它」，重要的是不要逃避它。

雖然前端框架會因時代趨勢而式微，但我唯一能向你保證的是你所學的任何東西絕對不會是白費的，像是我學 jQ 讓我在操控 DOM 元素上變得更加熟悉，學 NG1 讓我掌握到 two way binding、MVC 的抽象觀念，學 D3 讓我對 SVG 底層更加透徹，學 WebVR 使我對 WebGL、

three.js 有更深入的瞭解，在不停地學框架讓我更加瞭解底層，也幫助我掌握一個框架速度也越來越快了。

最後還是用個老話來結尾，唯有不停升級自己的腦袋思維才是王道。

要認清公司能帶給你的「技術功力」有限，你得自己額外補起來

你如何看待你的技術成長？你在公司一兩年，但總覺得越到後期成長幅度低到不行？或是都認爲工作內容都在做重複性的事物？你最後一次很快樂地寫程式是什麼時候？如果上面的問句有讓你產生漣漪，就必須小心自己的程式能量正在逐步萎縮中，爲什麼會這樣呢？就讓我一一道來。

你是不是變成「有**N**年經驗，但根本是重複第一年工作？」

當你開始第一份工作，任何事情都讓你覺得十分新鮮，以我來說來了，當時我開始在幫企業設計網頁時，就覺得「哇，我在幫公司設計他的形象網頁耶，這東西是會上線的，不是自己練習的，好有成就感~」、「公司有好多 CODE 可以參考哦，看完我一定可以變得超強！」、「工作要用到的程式還不是很熟悉，趁這機會補起來」。前幾次你會覺得很新鮮又超有成就感，但是如果是超過十次、五十次？甚至一年兩年，你眞的還會覺得有趣嗎？

答案自然是否定的，所以工作到最後當然會乏味，原因是每間公司都會有自己的生存法則，當你前一年已經習慣他們的 SOP 後，到後面就會覺得無趣，且技能也培養到能夠應付公司業務的程度。假使靠這套流程就能養起一整間公司，它們自然不會想要更動，除非有重大改版會影響到營收，才會認眞思考改進。如果剛進社會的前端沒意識到這點，就很容易在一間公司待了兩三年，但還是會覺得自己沒什麼成長，就算要找工作，也會覺得沒什麼可以拿來當做籌碼的能力而退卻。

所謂的專業就是瞭解自身不足，並刻意補起來

究竟什麼是「達人」呢？要怎麼樣才可稱自己具有「匠人」等級？所謂的專家就是在各種細節上都會有病態般地自我要求，而前端開發更是如此。

你認爲前端開發是什麼呢？新增個 HTML 裡面寫一寫用瀏覽器打開就會出現，就以爲是自己是前端嗎？那麼以下數點你掌握幾點？

- 瀏覽器原理 ：你知道瀏覽器的網頁渲染機制嗎？
- 資訊安全 ：你知道前端要如何寫才能避免 XSS 攻擊？
- 效能優化 ：當資料量多到無法負荷時，你該如何優化效能呢？你
- 程式趨勢 ：是否有掌握到目前最新趨勢的寫法，是否有目前雖然是測試階段，但你能預測到他未來有機會成爲一個產品服務？

- 網站追蹤 ：你知道 SEO 嗎？有辦法與行銷部門協作，共同開發 Growth Hack 精神的 Funnel 追蹤呢？
- **WebGL** 動態設計 ：你知道網頁甚至能夠在瀏覽器呈現 3D 介面，要你設計一個互動性線上遊戲你思考得環節夠全面嗎？
- 狀態設計 ：當網站離線狀態時，你的網站仍然有辦法運作，直到連線又能夠連接到伺服器去嗎？
- 協同開發 ：你有辦法和其它前端討論模組化概念，讓多位前端加倍開發產能在同一個專案上嗎？
- 無障礙設計 ：你有想過你的網站是否能讓身障人士都能輕鬆地瀏覽網頁嗎？
- 後端邏輯 ：你知道後端是如何傳送網頁資訊到瀏覽器的呢？知道什麼是 Header、狀態碼、Session、Cookie 嗎？

每一個層面都值得花上大半年的時間去投入，如果公司沒有提供這個養分，爲了讓自己成爲專業的前端工程師，和前端有關係的細節你當然都需要深入去瞭解。

尋找要投入的目標，並思考要花幾年追上

承上所提，也不是要你像無頭蒼蠅一樣學東學西，如果你希望自己學得更有成就感，最容易的方式就是尋求更有挑戰性的工作。所以我都會貼Toptal上面的前端人才履歷給新手前端參考，Toptal 其中的優點是在每個前端人才裡面都會寫到自己的工作經驗，以及在每間公司上負責的工作項目，也可因此拓展自己的前端視野，不會以爲自己待得公司就是全世界，這樣就太過井底之蛙了。

如果國外的職缺很遙遠，其實你也可以在 FB 前端社群或 104 尋找資深前端工程師的職缺，思考一下自己還欠缺什麼，再來去評估自己要花上幾年才能配得上這份職缺，那它就會是一份很棒的目標。

有哪些方法能夠補起技術起來呢？**side project** 是你的好朋友

我都會建議朋友可以多做一些 side project，什麼是 side project 呢？就是工作之外，你自己也去搞一些東西來實作。譬如來說公司本身已經有一套很成熟的開發流程，直到有一天你發現目前有幾項新技術能加速工作流程，但因爲自己還不熟悉，所以自己找了些專案來實作，等自己玩得比較熟以後，覺得這流程可行再推薦公司使用。

那有沒有一些 side project 的方向呢？這裡也條列一些讓你參考，也是我幫自己挖坑的項目：

1. 學會新技術，拿舊專案重寫優化
2. 臨時需要設計一個平台協助他人時，例如 八仙塵爆 ，那時我也設計一個查詢介面，順便練習從 jQ 轉 NG 。

3. 從 OPEN DATA 獲得寶貴數據資料，拿 JSON API 自行介接地圖、圖表資訊
4. 思考大眾常遇到的問題，設計一個小服務來解決，當初的 求職天演通 其實也是從 side project 變成公司創業方向
5. 從自己生活周遭不便來想靈感，例如寫一個 Node.js 爬蟲去抓最便宜的機票，若有釋出則發 Mail 通知
6. 跟朋友組隊參加技術黑客松，將想練的技術在比賽過程中來發揮得淋漓盡致
7. 做自己想做的服務，讓自己具有產品思維來運營它，未來的有一天說不定有機會靠它創業（好大的坑

最後我也來分享自己學習技術的方向，當我想學一個新技術時我會先自己研究個大概輪廓，然後主動和上面的人說：「我會 XX 技術了，它可以幫業務帶來一些不同的方向，可以去推推看」，可能是吸引力法則吧，就真的會有幾個專案跑進來讓我實際應用技術在專案上。能幫公司賺錢，技術也可以跟著升級，實在一舉兩得。所以如果發覺公司沒辦法給你成長時，另外個方向就是你自己推坑新技術給大家，至於要如何在公司推坑新技術又是另外一個坑了，有機會我會再到下篇來分享。

工程師打造高效工作的秘訣

有沒有常常覺得一整天都無精打采，明明工作進度差一大截，但就是提不起勁呢？或者工作常常做到一半就被打斷，本來高度集中的狀態因此消失得無影無蹤，那麼究竟該如何一直保持高效率的工作狀態呢？

作息正常，隨時保持思緒靈活

簡單說就是「要睡飽」，如果你一直持續在睡不飽的狀態，腦袋就很容易混沌。我遇到很多人在工作的時候會惡性加班，以致於讓自己的作息變得極不正常，更常見的是早上來到公司後，人雖然來了，但你的思緒並還沒有到公司，往往要再過好幾個小時才會回神。但是寫程式是極度需要高度思考的工作，如果你在上班時間一直處於精神不佳的狀態，你認爲寫得出好程式嗎？所以在前面的章節才會提到不要加班，讓自己的生活品質正常，在工作時間保持最佳狀態，工作產能才會越來越好。

自主經營不被打擾的環境

接下來就是工作環境了，以前我剛進公司時常會被同事密集性地打擾，寫前端寫到一半，其它部門都會不時來找我討論事情，原本在很集中的狀態因此被打斷眞的氣得咬牙切齒。直到工作幾年後才發覺環境要自己營造出來的，那麼究竟該如何做呢？

讓同事養成不會一直想要密集性找你的習慣

有些同事可能會因爲你就坐在旁邊，想到雞毛蒜事的事情就會想找你討論一下，也因此被打斷的頻率也會跟著升高，因爲他並不知道你目前的工作狀態，所以較好的方式是說，你主動跟對方說目前你的狀況，請他有事情時先整理列表起來，等到你忙到個段落時再回頭找他討論。當每位同事都瞭解你的工作狀態，久了以後他們也會習慣和你討論的流程。

但是你要記得並不能因爲這樣就忽略同事交辦給你的工作，如果你都在忙自己的工作，並沒有依照時程提供給他們要的東西，想當然爾他們就會對你產生不信任感，以致於就會想不時去盯你的進度。要打造適合自己的工作環境，首先你必須取得同事與主管的信任，否則一切都是空談。

透過線上專案管理工具來交代中小型任務

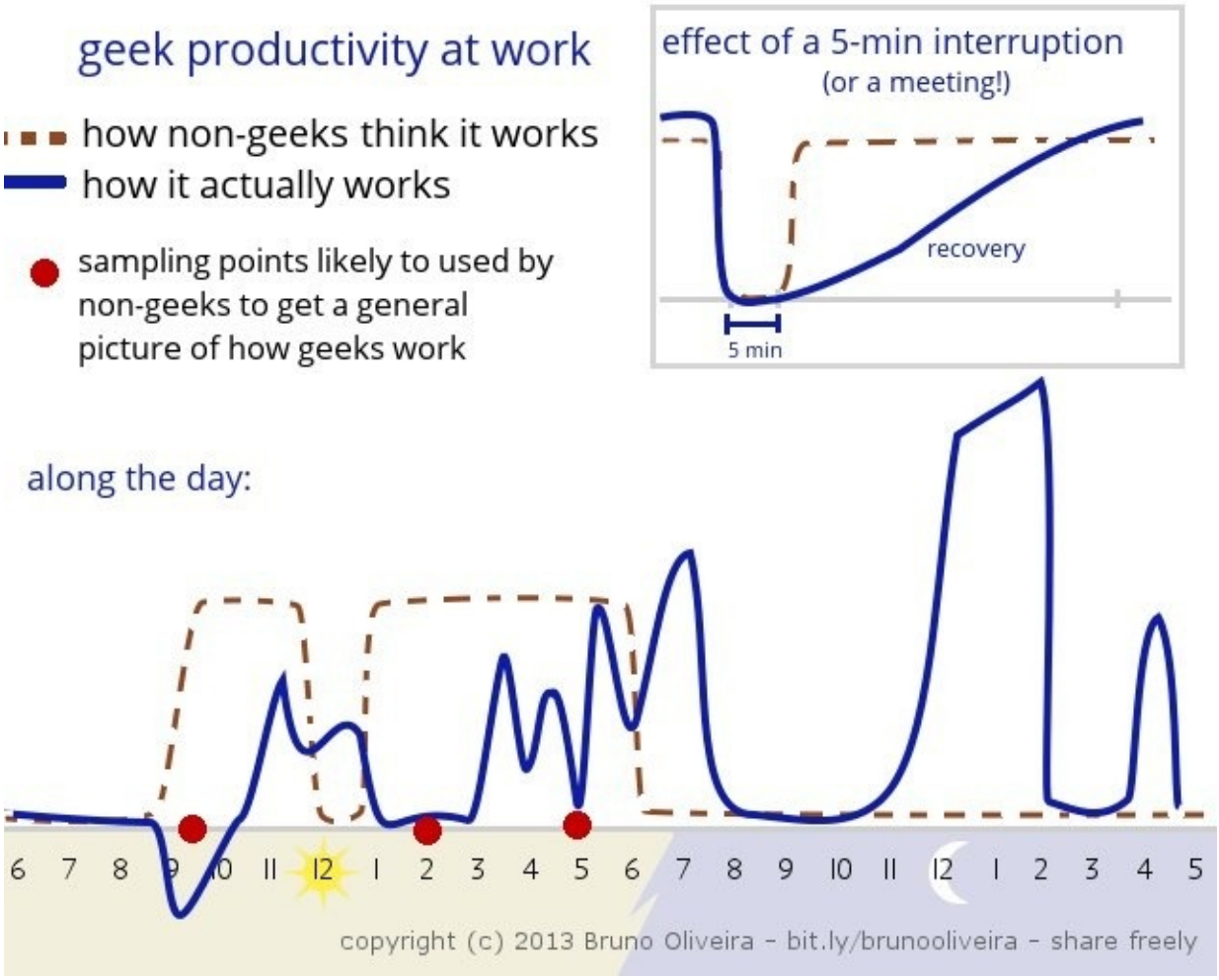
除了面對面溝通外，另外一個不會被打擾的方式就是推坑公司用專案管理服務，例如Trello、Asana，當有工作任務或發現網站有 BUG 時，假使並非緊急狀況，其實都能用類似這樣的服務，請同事先發 ISSUE (代辦事項) 出來，等到你有空時，再到服務上來看有沒有問題。像是我們公司目前就是使用 Trello 搭配 Git 來討論程式 ISSUE 的。

先主動完成「溝通性質」的工作

當我工作到個段落時，我會定期去找其它部門溝通工作項目，有任何細節都一次講完，例如說早上一來上班時就相當適合，大家都還在暖機的狀態，爲了讓一天工作有個好的開始，10~15分鐘密集溝通要做的項目，通常就能將當天 80% 的討論細節都說得差不多，其它就用 Trello 來輔助即可。

瞭解自己的生理狀態

你知道自己一天當中的生理狀態嗎？我們先來看一張圖。



紅色虛線是一般人認爲工程師的產能，而藍線才是你的工作狀態。其實還蠻合理的，例如你剛上班時還沒進入狀態，然後才慢慢進入狀態，那因爲中午吃飯後血液都跑到腸胃，大腦就不輪轉。下午五點血糖變低但是過了那段飢餓期後精神反而會越來越好。

所以有些軟體公司甚至會十二點半~一點才吃午餐，或是自己抽出一小時去吃飯，用意也是希望能讓工程師保持高效，有趣的地方是工程師晚上十一至凌晨兩點產能更高不是沒有原因的，因爲這段時間不會有人會打擾你寫程式，所以效率自然會高到爆炸。

而右上角則是大家對工程師的誤解，他們都會認爲說被打擾一下，工程師會很快恢復狀態，實則不然，其實是會緩慢上升才對，所以不要小看被打斷的力量。

當然每個人的生理狀態都不一樣，但只要掌握到這個觀念，你就能知道自己什麼時間點適合去溝通，什麼時間適合寫程式。以我來說我很習慣早上九點跟中午吃飯過後溝通事情，3~5點就專心寫程式。當我評估有一個複雜程式在公司裡會需要一個禮拜才寫得出來時，有時候我會帶回家寫，往往兩個小時就戰勝我預期要花三個工作天的 Coding 量。

第三方軟體輔助

以前剛工作時我的心情很容易浮躁，沒辦法定下心來工作，再加上在網路上搜尋資料時，很容易被其它資訊牽著鼻子走，無形之間就變成時間殺手，最後再來分享幾個 Chrome 插件與軟體給各位：

- StayFocus：能設定一天你看特定網站的時間限制，例如你一天看 FB、twiiter 只能限制 25 分鐘，超過時服務就會婊你應該要工作了吧？
- Life Is Too Short：打開 Chrome 後就會倒數你的人生，下意識提醒自己不要再耍廢。
- RescueTime：裝在自己電腦的程式，會記錄你用每個軟體的時間，每週會發一封 Mail，讓你知道你在用各個軟體的總時間，便可藉此觀察自己在「社群網站」、「商業」、「設計軟體」、「開發類軟體」上花了多少時間，好讓自己評估產能成效。
- 關掉所有聲音通知：在寫程式期間我會關掉手機與 Mac 上面任何會發出聲音的通知，以避免有提醒導致讓我分心，例如 Line、Mail，在 iPhone 預設設定上也有個勿擾模式可以使用

適時充電擺脫工作低潮

不論各行各業，都一定會有工作低潮，以我的前端生涯中就有幾次低潮期，例如說：

- 突然對寫前端產生厭倦
- 對公司交辦的工作一直覺得無趣
- 覺得自己除了寫程式以外沒有任何長處感到自卑
- 一直追技術覺得非常累
- 已經預期自己沒辦法進入 Google、FaceBook，是不是該早點放棄？
- 回到家就一直耍廢，感覺自己就是個廢人

當你有這些想法時，也可以當做是一個警訊，有可能是你生活重心都在寫程式，生活中沒有其它調劑，導致容易進入到負面漩渦中。這裡也分享幾個充電的方式給各位。

耍廢不是罪

辛苦上班一整天，回家想要休息玩場電動，刷手機是很理所當然的事情，有些人會不想要放過自己，認爲自己天份已經很差，應該要隨時都要吸收新技術，才可以堪得上叫做工程師。

這跟調琴上的弦一樣，如果你調得太緊，琴弦就會斷掉，音色也會過硬，調得太鬆時也完全沒辦法發出聲音，最好的方式就是調得適中，才能彈出好音色。就如同生活般，你如果二十四小時無時無刻都在寫程式，不理會生活品質，身體遲早會向你抗議，但如果你一直沒有前進，也不會有成果。最好的方式就是適當安排 Coding 的時間即可。

像是我不想寫程式時就會索性一個禮拜甚至一個月都不寫，除了工作外，其它時間就盡情耍廢，直到我內心會有個聲音會和我說：「嗯，好像充電充得差不多，最近看到一個新技術蠻有趣，可以來玩玩看」，我才會開始回到我的程式路上。

要記得，休息是爲了走更長遠的路，人生是一趟馬拉松，不是百米快跑。

讓旅遊暫時忘卻煩惱，沈澱心靈後再次出發

我們生活在都市中，科技進步也伴隨著我們也必須隨之成長，國中、高中、大學一直讀書，出社會後爲了養家餬口而拼命賺錢，在這種高度緊張步調快速的生活中，身心靈進入到低潮是很常見的，就連我也不例外。

我曾經也有段低潮得非常誇張，每天都不想寫程式，那時剛好和我老婆去日本玩，我老婆就問我說：「那你要帶 Mac 一起去嗎？」，我猶豫了一下後決定試試看不帶電腦的日子會怎麼樣，於是那時就沒帶電腦出國。

警覺自己被電腦制約

第一天其實非常難受，其實寫程式久了的工程師通常都會有一種被制約的感覺，只要一小時不看電腦就渾身不對勁，但過幾天後發現原來沒電腦也不會怎麼樣，原本肩膀以往都很酸，也因如此而輕鬆不少。我自己也才意識到原來以前的我那麼依賴電腦。

有空閒的時間思考人生方向

有時候人在低潮時，總會想要爲這低潮找出一個理由，但如果你身在局中，很難看清問題的本質。像是那時人處在國外，任何的俗事都暫時不用攬在身上，我可以好好思考究竟自己是本來就不喜歡寫程式硬逼自己寫，還是只是近期內挫折感太重導致無法負荷，亦或是覺得薪水太低壓縮到生活品質？在這旅途中也可以試著找出自身內心的答案。

回到正軌重新出發

當我回國後，回到家接觸電腦後，發現自己對它超級陌生，連打字都變得不熟悉了，同時也覺得非常新鮮，感覺自己又能再戰十年，到最後我也找出自己爲什麼對程式低潮，原因是重心太過倚賴在寫程式上，導致生活品質也因此受影響，身體狀況也變得非常差才感到厭世，找出癥結點後過沒多久我又繼續回到正軌了。

現在台灣到日本拜廉價航空所賜，機票加上酒店一、二萬內就可搞定，上網找下別人幾天幾夜的行程，不會說日文也沒差，那裡也有許多會講中文的服務員。如果覺得太貴，其實到花蓮、台東住個幾天也不錯。

培養第二興趣或專長

這也是一種讓你的生活重心能夠有多方發展的做法，當你在自有領域不順遂，或是一直很在意某件事情時，其實就能找一些方法來轉移注意力。以興趣來說我自己就很喜歡看書，甚至能到書局待上一整天，週末安排密室逃脫，和朋友家人一塊去玩。或是玩線上遊戲，不瞞各位我從學生時代就非常迷線上遊戲，甚至一度想要成爲電競選手，也曾經當過六年遊戲論壇板務，每個月管理數千篇遊戲攻略文章。

培養專長的話我自己是找前端有相關的領域，像是這兩三年我都在研究 Growth Hacker，去年開始研究區塊鏈，當我寫程式很不順的時候，就會改去投入其它領域的東西，有時候我光是看自家服務的廣告投放數據都覺得時間過得很快，一天又過去了。或是看下智能合約是如何被設計出來，像是最近也用了乙太幣買了幾隻貓貓。

不要認爲說這些東西都跟自己專業無關，許多成功的服務絕大部分是身兼多個領域找出商業模式。尤其現在也有許多傳統產業也開始在轉型，例如與果農上下游合作，搭配各種商業整合的果物配，或是讓自己的房子也能共享出去的Airbnb。

最後，從各種興趣與專長來去探索「自己熱愛的東西」、「自己擅長的天賦」、「自己想投入的方向」，再從中找出這三個有高度密合重疊的東西，那就很有可能會成爲未來你想邁進的目標。

認識自己的技術天花板，並穩定成長

你可能知道學某些技能可以讓自己快速找到工作，但學了以後才發現學習曲線過高，學不會而導致出現低潮，甚至會懷疑自己沒有寫程式的天份，但這根本是錯誤的觀念。正所謂欲速則不達，不要連走路都還不會就想學飛。例如連 PC 版的網站都沒做過，就去挑戰響應式網頁，沒用過基礎 JS 寫過一些應用，就想直攻SPA 框架 (react、Vue、Angular)，學習挫折感也會加倍，對本身成長反而會有負面的效果。

不要勉強你自己去挑戰高難度的技能，這觀念就有點像是玩線上遊戲一下，在新手村還沒累積足夠的等級與裝備，硬是挑戰魔王只是找死。所以你首當要做的事情是逐漸的瞭解自己，因爲你的頭上並非像是遊戲一樣，會有個儀表版寫你目前 9 等，顯示攻擊與防禦的素質。我相信你在投入前端時，也可以慢慢意識到自己掌握一個技能所需要耗費的學習時間成本，以及自身的能力目前在哪個階段，再藉由這些開發經驗，找到下個足夠你應付的關卡逐一攻破，日月累積下來自然會水到渠成。

教你解讀前端徵才廠商職缺內容的背後眞相

你是否有想找前端工作但被徵才內容嚇到不敢應徵？有時候徵才廠商在撰寫履歷時，也有可能不會是工程師協助撰寫履歷，但又爲了希望新前端和其它同事的溝通能夠契合，所以時常會看到徵才條件多到溢出來的必備技能，所以當你看到以下職缺必備項目時，不要慌張，去解讀求才廠商背後的用意。同時也寫給求才廠商一個參考方向。

熟悉 JS Framework，jQuery、Vue、Angular、React 尤佳

有些廠商可能剛接觸前端職業，
不了解目前 JavaScript 趨勢，
所以就熱門的全寫上，
這樣很容易造成求職者的惶恐，
就連我自己也只能說對 JQ、NG 算稍微精通，Vue 開始有摸些皮毛的階段，
其他也頂多算是玩到建立個 todolist、寫些玩具的程度，
根本還沒成熟到敢用在專案上。

web 廠商要前端會 JavaScript Framework，
主要也是希望會下面兩項技能：
1.設計網頁互動性動畫效果
2.串接 Ajax、Restful API，具系統性地規劃前端架構

那就建議廠商寫明工作任務，
再從履歷、面試來去評斷他是否勝任就好，
現在前端框架多到爆炸，
我也認識不少純用 Native、冷門框架但強到爆炸，
要記住框架只是工具，你就算有關刀，但不會耍也是白搭

工程師、設計師職稱傻傻分不清楚

我在輔導學生就業前端的時候，
常常遇到職稱寫「前端工程師」，
但去面試時卻要求需設計畫面(Mockup)，
這眞的是比 HR 要找 JAVA 但卻找 JavaScript 工程師還要更雷！

所以每次高雄前端社群聚會要結束時，
我都會不厭其煩播這部影片，
主要希望讓會眾了解前端設計師、前端工程師的差異。

所以這裡建議廠商，
如果工作項目需要設計畫面，
我自己是建議職稱加上「設計」，
例如「網頁設計師」、「前端設計師」。

如果完全不用設計畫面，那就加個「工程」，
例如「網頁工程師」、「前端工程師」。
至少能讓求職者有個基本判斷，點閱轉換率也比較高。

至於有些朋友提說如果只單純切版，
不會接 AJAX、SPA 的職稱要叫什麼，
我個人覺得叫「前端工程師」也沒什麼不妥，
也有 在104 看過「網頁切版工程師」、「切版工程師」也OK的。

懂後端語言尤佳

有幾次幫前端朋友討論職缺的時候，
常發現對方濾掉些我認爲他們可以勝任且薪水也給到位的公司，
細問後才得知他們會對工作項目裡面有寫到「懂某某後端語言尤佳」的條件而卻步，
儘管那是加分條件而不是必備條件。
我問了幾次徵才廠商，實際狀況只是：
(1)擔心與公司後端工程師合作不順 (佔80%)
(2)只是現階段大案需要，未來「有可能」需要負責後端(佔20%)

老實講如果你是一個：
1.已經會 Template Language(jade、slim、ejs)，了解 partial、Layout 的觀念
2.曾經有跟後端工程師合作經驗

那其實就已經可以跟全部的後端工程師合作了，
很多都是大同小異的，
每次我跟不同後端工程師合作，
只要了解 View 放在哪裡，
CSS、JS 的路徑在哪就直接上工改 Code了。

不過會建議徵才廠商要和面試者主動表達要找的對象是哪種，
有些廠商會希望先騙進來，等前端需求沒那麼高再誘導轉後端，
不過爲了建立長久關係，建議還是事先說清楚會比較好。

懂 **UI/UX** 尤佳

老實說除非是 UI Designer轉前端，
或者本身對這塊領域有興趣，
否則眞的是有點強人所難。

但經由我和許多徵才廠商詢問過後，
他們要的大多是「能夠與美術、後端合作，提出可操作性的前端介面建議」，
其實這才是徵才廠商爲什麼要找前端工程師的最主要原因。

每當 UI Designer設計 Mockup 出來後，
我都會提出四、五種可行性操作流程，
同時瀏覽器兼容性也相當成熟的建議，
彼此再激盪出更佳的使用者介面，
所以 UI/UX 並非是 UI Designer的工作，
我個人認爲是全部團隊都該參與的事項，包含PM與後端，

如果你看到徵才內容條件有「懂 UI/UX 尤佳」，
我建議就自己腦補成「能夠與美術、後端合作，提出可操作性的前端介面建議」就好了。

你可曾思考離職後尋求更好的自己？

馬雲曾經說過一句話，通常會辭職的絕大部分原因是「錢沒給到位，心委屈了」，那麼除此之外呢？你還可以思考下面因素。

金錢：薪水不符自身能力，爲什麼跳槽會比加薪的成長幅度還高？

當一個產業形成時，什麼樣的能力配上什麼樣的薪水問幾個業界人也能略知一二。

而業界也非常缺靠譜的工程師，所以新手前端在一間公司培養出足夠的開發經驗，薪水在初期也會也跟著三級跳，以至於跳槽也遠比一年平均公司幫你加薪 5～10%來說還吸引人。

沒有金錢，你就沒有生活品質，沒有生活品質更別說你想要實踐什麼目標與理想，一切都會淪爲空談。如果你自認開發程度有到一定的水平，就可以主動向公司提加薪，若是公司相準你的成長幅度有超乎他們期待，認爲你有可能是未來他們的主力之一，自然會投資你。但仍會因爲許多面向，導致公司沒辦法調薪到符合你的期待，例如說：

- 他們還停留在你剛進來的菜鳥樣，認爲你不符加薪的門檻
- 公司所需要的技術能量並不要求太高，就算你技術提升，在公司也無用武之地
- 公司的資源權重都在其它部門，並無過多預算培養前端人才
- 公司有調薪制度，沒辦法一口氣加薪太多
- 部門本身有無形的薪資天花板，沒辦法將你加薪到超過主管等級薪水

從上面你可以看出，原來加薪不只是人的主觀問題，同時也會關係到公司對前端的重視，以及公司內部本身的制度，當你瞭解這些後，就可以觀察出你現在的公司薪資天花板在哪個區間。

技術：想跳槽，但你有配得上你能力的代表作了嗎？

有很多前端朋友都和我說在公司薪水少想要跳槽，我都會再細問說那公司的技術能量如何，你都掌握起來了嗎？如果「是」，自然去尋覓更好的職缺，但如果「不是」，我都會建議朋友先將技能累積起來，這樣你能談判的技術籌碼才多。

較常見的是有些剛投入前端領域的新手進入一間公司，什麼經驗都沒學到就想換個環境，但明明公司正已經投入新前端技術，而且也是業界常用到的前端框架，你當然要把握這大好機會將該技術內化成你自己的東西，提升前端視野。直到你認爲這間公司沒有任何你能夠學習

的東西，或是你主動向公司提想投入新技術在下個產品上，但公司打槍你時，就必須好好思考去留的問題。

我也曾經被回說：「這樣不是很卑鄙嗎？通通學完就繞跑？」，那我又不禁想說：「那公司對你的未來有任何的技術成長規劃嗎？」，要把好人才留在身邊，你不提供給員工一個具有挑戰性的工作任務讓他成長，留得住嗎？

所以有些面試官在面試工程師時，也會問到：「你希望未來幾年想鑽研哪些技術？」、「若你進入到公司，你希望我們提供你哪些資源讓你成長？」，再從中瞭解對方的期望，雙方各取所需，你幫公司賺錢並提升技能，而公司也提供給你作戰環境讓你放手一搏，才能夠達到雙贏局面。

說到這裡，你不妨也想想，未來兩三年，你希望自己的前端領域達到何種境界？你想找下份更好的工作，你是否有一個能配得上你能力的代表作當做籌碼了嗎？

成就：你做的事情有帶給你滿滿的成就感嗎？

我曾經和一位朋友聊過，他們待得公司福利很好，技術主管也會丟資源讓他們消化升級，但總覺得缺乏了些什麼，後來細問一下才得知他做的東西都是公司內部系統，或是幫其它公司做的 B2B 服務。那因爲所做的系統都有鎖 IP，以致於他做的大作沒辦法讓其它人可以看到，覺得缺乏成就感。

會想投入前端工程師的族群當中，絕大部分都會希望自己的東西是可以給使用者進行操作，設計出來的服務可以給很多人去使用的。尤其當別人問你說公司最近在玩什麼時，你可以很自豪地打一個網址，展現出酷炫的成果，都是前端工程師憧憬的事情。

所以在進行前端面試時，務必要向面試官詢問部門協作、開發項目、舊案維護、技術能量等問題，你才能從中找出這份工作是否能帶給你足夠的成就感。

部門氣氛：機車主管與同事容易成爲導火線

有學生曾經問我：「公司的小主管一直找他麻煩，又愛挑撥離間」、「主管不會寫程式，很容易疑神疑鬼怕別人搶了她的位置會故意找麻煩」、「最近又說她的 CODE 老闆看過覺得不好，要延長試用期，但老闆根本沒看過」，也因爲如此每天壓力都很大，甚至會開始發燒頭痛，希望問我可以提供一些方向給他。

我就和她說：「聊了這麼久，但都沒聽你提到辭職的選項？你已經是位資深的網頁設計師了，本來就不愁找不到工作，你應該是朝向這間公司現在是否能讓自己的技術成長爲方向來思考，不要將心思放在公司小團體的經營上才對。」

如果你還是位前端新手，要知道在公司上會有機率遇到這種機掰人，如果他碰巧又是你的主管，爲了求表現刻意數落你的不是，讓長官認爲自己很會管理員工。或是共事的員工 EQ 很低，彼此磁場不合，就必須考慮去留問題。建議不要浪費試用期前三個月的時間，好好觀察你的同事與技術主管，再好好確認是否適合繼續待下去。

結尾

你必須要有個認知，在台灣還是有許多中小企業爲了求生存，所以沒辦法每個層面都有辦法面面俱到，例如說：

- 公司剛起步，準備要開發一個新服務，團隊有很多前端強者，但仍想找個沒經驗的前端來輔助，但因爲新創公司所以短時間沒現金流無法幫你大幅調薪
- 已經有一個很成熟的服務在賺錢，想找一位前端來維護，雖然福利很好，但缺點是沒有機會讓你開發新項目
- 薪水、技術能量、成就感都非常棒，但看不慣另一位同事的作風，無奈辭職
- 進入上市上櫃的大公司，因爲有歷史包袱，所以工作內容有一半的時間會維護舊專案，但另一半時間仍有開發新產品

尤其你還是前端新手在找公司時，因爲技術仍在差強人意的階段，自然沒什麼選擇機會，但要瞭解每間公司都會有優缺，再從中思考「你想在這間公司獲得什麼？」、「雖然錢少，但公司的豐沛開發能量是否爲當下最佳投入捷徑？」，最後，再讓這個理由成爲你的養分變得更強才是正解。

善用工具提昇開發效率

在石器時代，原始人必須很辛苦用鑽木取火的方式來生火，而古代的人開始會用打火石。隨著時代的演進，也開始有打火機、火柴等工具，而在現代，只要有瓦斯充足，你的火源想燃燒多久都可以，且不費吹灰之力。

從此可以看出任何工具都會隨著科技進步而跟著演化，寫程式也是一樣的道理，我們隨時都必須觀察是否有更實用的工具來提升寫 CODE 效率。

程式編輯器的演化史

以編輯器來說好了，它就是一個很好觀察出科技在進步的過程，在十幾年前大家寫 HTML 都是用記事本來撰寫，接著開始有「可視化」的編輯器，像是 FrontPage、Dreamweaver，讓想學網頁的人不會因爲看到密密麻麻的程式碼而卻步。直到近幾年才又開始有 Sublime Text 3、ATOM、Visual Studio Code 這類的文字編輯器，與記事本相比，這種專門寫程式的編輯器可是強大的多了，每位開發者都可以爲編輯器寫套件，讓其它人都可以安裝套件，你可以自己打造適合自己開發習慣的編輯器介面、熱鍵、顏色高亮、coding style、自動除錯等等。

它幫助開發者能夠聚焦在程式開發上，甚至能用自動化檢查的方式幫助你自動除錯，就以我自己來說 我的網頁編輯器操作歷史則是 FrontPage > Dreamweaver > Sublime Text 2、3 > Visual Studio Code，幾乎每隔兩三年我就會換編輯器來讓自己變得更強。

持續優化自己的模版架構

一開始學前端的時候，HTML 結構非常單純，頂多有插入一隻 CSS，網頁也只有一兩頁，但隨著你會的東西越來越多，架構也會跟著變大，如果每次在開發時，都要從零開始慢慢建構不是很辛苦嗎？首先要新增 HTML、CSS 檔，去 JS 框架官網下載最新版本，CSS 又必須重新寫過，每個細節如果不小心漏掉就又很麻煩。

所以你可以思考建構自己的boilerplate，什麼是 boilerplate 呢？就是設計一個乾淨的開發環境，每當有新專案時，你就複製那個開發環境下來，裡頭就有你慣用的 JS 框架、前端任務工具 (Gulp、Webpack)、以及常用的 Sass Mixin 結構等等，讓你能夠專心在開發新程式上面，不會覺得每次開發都像是重新開始一樣。觀念又很像是你在大學上繳交報告給教授時被退回，因爲你沒 follow 他的 word 標準格式，例如標題要 18px，行距要 1.5倍，所以教授提供給你一個 word 範例，裡面的各個設定都設好了，不要自己慢慢建立，只要遵守規範就有符合教授作業的規則是一樣的。

像是在 Github 上你關鍵字搜尋 boilerplate ，也可以看到許多開發者的建構環境，而我也會建議你在 Github 上面試著放自己的 boilerplate，當你有新專案時便透過 Git 指令 clone 下來就方便多了。

不要有鄙視心態，自己用的順手最重要

業界會有個鄙視鍊的玩笑，就以編輯器來說，連結裡也附註一段：「用 Vim 的工程師鄙視用 Emacs 的工程師，用 Emacs 的工程師鄙視用 Vim 的工程師，無論是用 Vim 或 Emacs 的工程師都鄙視所有用其他編輯器的工程師；用 Atom、Notepad++、Sublime Text 的工程師鄙視用 Windows 記事本的工程師。」。

這裡會建議不要鄙視別人用的工具，有些時候工程師用起來就是很不順手，像是以前我用 Dreamweaver 時，那時我就在想該選 notepad++ 還是 Sublime Text 2，後來我用了一陣子覺得 Sublime 用得比較順手，但周遭也有人選擇了 notepad，這中間其實又關係到自己的人格特質、喜好偏愛、操作習慣等等。

我自己也常遇到這種問題，例如有時候別人推薦的工具我聽他講完也覺得非常讚，但是我實際去用總是覺得哪裡卡卡的，最後硬用幾天宣告放棄，過幾天後發現另外個類似服務，一使用發現喜歡得不得了，甚至就直接註冊付費版，但眞要我說兩者的差異我自己也說不上來。所以眞的自己用得順手最重要。

工具用得好，工作效率自然高

不要小看這些工具運用的累積效用，就以 HTML 標籤來說，目前的熱門編輯器都有支援一個強大的套件叫做emmet，你可以思考看看自己打以下程式碼需要多少時間，如果自己手打起碼也要一分鐘以上吧？但我用 emmet 只要 10 秒不到就可以打出來了。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" href="all.css">
  <script src="all.js"></script>
</head>
<body>
  <div class="wrap">
    <div class="header">
      <ul>
        <li><a href=""></a></li>
        <li><a href=""></a></li>
        <li><a href=""></a></li>
      </ul>
    </div>
  </div>
</body>
</html>
```

如果一整年我跟你每天都會打一次這個語法，我就比你還多了整整五個小時做其它的事情，你說驚不驚人？專業的工程師爲了讓自己更懶惰，自然會去找各種工具來提升寫 CODE 效率，讓自己擁有更多的時間專注在他想投入的事情上。

那麼到底該如何觀察自己的哪些環節可以用工具來輔助呢？最後也丟出一些檢核列表給你參考：

- 每個專案都會寫到的功能，但手寫又很花時間的就可以設計成 snippet (程式碼片段)
- 思考日常生活中會花你超過五分鐘以上的工作，就思考如何讓它自動化完成(推薦連結)
- 如果每次在寫一個程式都會卡在一個抽象觀念上時，你就必須花時間瞭解它整體脈絡
- 思考一個禮拜或一個月的頻率上都固定會去做一件 30~60 分鐘的工作時，就有優化的空間(申請文件範本、自動化銀行帳單繳款取代臨櫃)

前端工程師要不要接案？如何報價？

在江湖上混，遲早會遇到這個議題，有可能是你的朋友與親戚知道你是工程師，希望你幫他做個網站，亦或是朋友介紹廠商，詢問你是否有接外包意願都還蠻常見的，假使你從來沒接過外包，相信你也會覺得躍躍欲試，但是又因爲是自己不瞭解的合作方式又感到有些退卻。

我很常被朋友問：「洧杰有個朋友有案子要外包給我，他是要做一個形象網站，我該報多少比較合理？」這問法眞的很籠統，就像是你問人說你想買台車，多少錢才合理有87分像，在你不瞭解前端的規格時，胡亂報價只是找死。所以這裡我也向你分享接案外包需要瞭解的一些心法。

依照頁面複雜度來計算

有很多廠商都很習慣用「頁數」來訂價，例如一頁五百，十頁五千的算法，如果你眞的用這樣來報，肯定會吃大虧，如果廠商不瞭解前端介面，你也必須有足夠的 sense 告知他以下項目：

瀏覽器與解析度兼容性

如果網頁需要支援舊版 IE 時，報價我會提高到1.5~2倍，所以當你報價給對方時，務必要主動提出來瀏覽器規格，否則如果你的介面都大量用到熱門瀏覽器才可以用的新語法，但 IE 系列完全不支援。當你交稿給廠商時，對方就會打臉說 IE 整個破版，對他們來講你做的東西一點價值都沒有。

再來就是螢幕解析度部分是否哪幾個版型會需要客製化，尤其是當廠商請我報響應式網頁價格時，我會依照對方設計師給的稿件去評估時程。講一些很誇張的吧，例如說只給你一個 **PC** 版，要你變出響應式網頁根本他媽的天才，前端不是通靈王啊，或是設計有丟出各解析度介面，但是呈現部分超級困難，沒有網頁 UI 觀念也很麻煩。

所以光是瀏覽器跟解析度兼容性這兩個部分，你就有很多細節需要溝通的了，不可不防。

以功能性來報價

曾經有個廠商外包給我前端開發介面，第一個網站有十頁，我報二萬，第二個網站我卻報十萬。廠商一整個沒辦法理解爲什麼我突然獅子大開口，經由我解釋後才釋懷。

第一個網站的每個頁面都大致雷同，我將 Layout 開發好後，其它頁面只有中間的文字與圖片進行替換，所以做好一頁後，我再把文案貼上去就收工。

但是第二個網站每個頁面都需要客製化動畫效果，有些甚至必須得用 JS+CSS 輔助才有辦法呈現，而且還必須考慮熱門解析度上呈現也必須 ok，而且需要不時地溝通網站細節，確保動態效果符合期待。而且部分頁面還必須接後端 API 。

說到這裡，相信你再也不敢用「頁面數量」來報價了，請以「功能性的數量」來計算才較爲準確。

報價沒有所謂的市場價格，爲自己的能力訂下價格

這種東西是沒有公定價可言的，你完全無法參考其它前端的意見，該怎麼說呢？在我剛成爲前端沒多久，一個前端頁面我收一千，過幾年後我開始報一頁一萬，爲什麼可以差到那麼多呢？一方面是我的能力提升，所以接的案子的頁面複雜度也比較高，另一方面是我認爲自己有值這個價值，切忌不要你報了個價格，但對方說：「你的報價高於其它前端同行」而退卻，你值多少是只有你自己能幫自己評價，其它人都沒有理由對你說三道四。

如果你對報價完全沒有概念，也還沒建立自己對專業度的信心，我也提供兩個方向給你參考：

依照工作天來推算

所謂的工作天就是代表你一天八個小時拿來投入工作，你再換算你的月薪除以工作天即可，或是你標出自己的價值，假設一天的工作天價值 3000 元，同時要記住這個工作天是你休息進修時間抽出來再額外做的，所以價值高一些也 ok。

那這裡考考各位，如果業主問你時程，一月一號時你回報要五個工作天，也就是工作40個小時可以做完，假如隔天開始算，代表你一月六號就可以把東西做出來？完、全、不、可、能。

你下班後的時間到你上床睡覺的期間，除非你熬夜否則不可能一天做滿八個小時，所以一個工作天你實際上要花兩三天的晚上平均消耗掉，所以你報給廠商應該是將近兩個禮拜且不含假日的時間才有辦法做完。

若廠商的東西是急件，希望你六日也幫他工作時，請記得假日的工作天要再加乘上去，畢竟你是用你娛樂的時間來幫廠商趕件，加錢是很正常的。最後再用個案例能夠更加明白。

假使廠商A在禮拜四丟給你設計稿要你報價，同時和你說這東西是急件，必須下週三一早必須交件，希望明天週五就開始動工，而你評估禮拜六、日必須趕工，才有辦法趕出來給他。平日平均投入三個小時

接下來你就必須評估自己的時程與價值：

1. 你認爲自己一個工作天的價值 是 $5,000
2. 你評估平常日下班後一天能投入 4 小時

3. 週末兩天你都要花上整整兩個工作天才有辦法在週五如期交件

時程就會如下，總計就會是 $27,500：

- 週五(4hr) = $2,500
- 週六(8hr) = $10,000 (假日加成二倍)
- 週六(8hr) = $10,000 (假日加成二倍)
- 下週一(4hr) = $2,500
- 下週二(4hr) = $2,500

不要報低，要高到符合你的期待

你工作下班已經很辛苦了，你還要額外去做外包接案的東西，如果又是急件的話，說實在的你也算燃燒自己的生命在賺錢，所以請務必報出去的價格，是你認爲你是可以很高興地去執行這個案子。而且通常你會花你預期外的時間進行溝通與改寫，所以報高是有必要的。

不要錢少、案子趕、廠商又機車，你 M 屬性？上面的工作天算法也只是個參考，你可以最後再自己調權重加成上去，像是我就有個廠商機車度的欄位，非常機車我就會算出報價後再乘上 1.75 倍，普通機車但設計師很沒 sense 就是 1.5 倍，但假使是超級好廠商就維持報價。

所以想要開始嘗試接案，先自己試著估出你一個工作天值多少錢吧，沒有自信可以先從 $1000~1500 抓起，等到你技術逐漸累積，這個工作天也可以依照自己的自信程度而水漲船高。

務必量力而爲，勿忽視本業

一開始嚐鮮想接案，想試看看自己的能耐，想賺到額外的收入等等會有這樣想法是很正常的，初期都去接觸我也覺得很 ok。但還是要呼籲不要忘記自己的技術累積。如果外包的案子都是在做重複性高的工作，雖然表面薪水變多了，但長遠來說你也跟技術逐漸脫節。因爲你是拿你投入新技術的時間來做你早就擅長的技術，久了自然追不上趨勢，要找下份前端工作時，反而沒有足夠的技術能量去談更高的薪水，那就本末倒置了。

與其你燃燒生命一個月 6~10萬 持續幾年，最後身體狀況越來越差，不如讓自己技術提昇到一定水平，去拿年薪 80~100 的 offer，同時又兼顧生活品質，您說是不？

最後記得要簽訂合約，廠商沒範本你就自己訂一個，傳送 mail 過去給廠商，一式兩份雙方簽名蓋章後各執一式，不會很麻煩且也能保障自己權益。也附上我曾經錄過的影片，讓你瞭解撰寫合約與款項等相關細節。

- 報價與合約設計細節
- 網頁切版報價細節

經營人脈發展，讓自己永遠不缺下份前端工作

工作環境很差，想離職找下間公司重新出發，但又擔心辭職後找不到下份工作，導致讓自己躊躇不前了嗎？現在的年代已經很難讓你一個工作可以直接幹到退休，那除了讓自己的技術永遠保持競爭力外，還有沒有其它的方式可以幫助自己離職後可以快速找到職缺的呢？或者是讓自己能保持足夠的自信心，就算明天就要離開公司，也不怕找不到下份工作？

多認識同行交換職缺情報

正所謂「在家靠父母，出外靠朋友」，出來混當然要多認識一些夥伴，有危急時互相幫助，有好處時互相分享，那麼要如何認識「前端朋友」呢？當然是去他們時常會逛的場子啦。

就以我來說，平常在 FB 前端社團討論時，時常都會碰到一些熟面孔，聊久了也開始有些小熟，雖然我住在高雄，對方住在台北，但台灣就這麼小，時常會舉辦各種技術研討會，例如 JSDC、MOPCON、Modern Web 等等，就是一個很好相認抬槓的好時機，我們甚至會稱這樣的研討會叫做「大拜拜」，原本幾年沒見的前端同行，也因爲這個機會有相遇的機會。

如果覺得研討會人太多也很難找到認識的人，也可以從在地社群來著手，人少的好處也代表你跟每個人聊天的時間也會變多。如果自己並不是那麼會找人聊天的個性，那該怎麼辦？其實也可以從 Line 群組、telegram、slack 的方式來抬槓，尤其是像這種通訊群組每隔幾天就會有人在上頭張貼職缺，讓你有更多的小道消息來接觸業界也是很不錯的方式。

當然也不是要你刻意去認識人，你可以用自己的方式來接觸即可，以我來說我也是會找「感覺磁場跟我比較合的人」來聊，也並不是任何人都可以聊得起來。

打造個人品牌，讓全世界都認識你

你不妨思考看看，當你褪去大公司的光環後，你還剩下什麼？一般人都認爲是偏向公司單位才會想要塑造一個品牌形象，但以目前網際網路發達的盛況，每個人都有可能會因爲自己的壯舉而瞬間出名，從目前的 YouTuber 與網紅的追蹤數，動輒都有數十數百萬，他們只要好好經營自己，就可以爲他們帶來收入，那麼工程師有什麼方式經營個人的品牌呢？

分享專精技術領域

前端領域時常會出新技術，當你發現該技術有辦法取代你目前工作流程，且在國外也非常熱門時，你就可以思考將它的學習資源整理起來，放在網路上提供給其它人能夠深入學習。

像是以前我參加 IT 鐵人邦時，就曾經寫過 90 天 Sass 教學並宣傳到FB前端社團，也獲得熱烈的迴響，原本以爲這樣就沒事了，但沒想到開始有一連串的演講、講課邀約，最後我自己也心血來潮開課也是座無虛席。直到現在還是會在研討會、聚會上遇到一些朋友說曾經看過我的 Sass 教學。

與其你學一百個技術，但都不精，不如將一門技術摸到非常專精，成爲該技術的專家，讓別人只要一提到某技術時，就會聯想到你，那你就成功了。

撰寫技術部落格

寫部落格其實並不用將他當做一件很嚴肅的事情，你可以當做自己在記錄寫 CODE 的過程，如果今天一個觀念還不是很貫通，那就很適合寫成部落格，在你寫作的過程中，無形之間也幫你整理程式脈絡，又能讓其它正在學習的開發者受益，實爲一舉兩者。以目前的部落格趨勢來說，medium就相當適合，或是自己造一個部落格Hexo也 ok。

業界其實比你想像的還小，當你去軟體公司面試時，也會因爲你長期耕耘前端技術圈讓大家對你有足夠的瞭解，跟其它面試者比較起來，你就又多了一項加分條件，豈不妙哉？

持續更新自己的履歷到 **Linkedin**

除了 104 外，你還能到哪裡刊登自己的履歷呢？首推LinkedIn給你，有個單位專門會協助廠商找適合的人才，我們會稱他們叫做「獵頭公司」、「Headhunter」，如果你的能力相當足夠，Headhunter 便會透過 LinkedIn 聯繫到你，瞭解你的人格特質與想要進入的公司屬性，並媒合適合的公司給你，媒合成功的話他們會再從徵才廠商抽取費用。

所以在前端職涯當中，若能認識幾位靠譜的 Headhunter 也是很 ok 的，但話說在前頭，樹大必有枯枝，也是會有很多沒 sense 的獵頭，例如上班時間狂 call 電話給你，或是不瞭解你的技術背景亂推職缺。不過人脈培養本來就是尋找彼此都契合的合作對象，所以仍要務必多愼選觀察。

身爲前端，你瞭解公司的專案性質嗎？

我們在前面的章節有提到面試如何評估公司的工作，但沒有仔細講解各種軟體公司的屬性，以及背後你需要注意到哪些細節，就讓我娓娓道來，好讓各位準備踏入前端的朋友提供一個參考依據。

接案公司 - 以量取勝型

什麼是以量取勝呢？也就是當你在 Google 打網頁設計這個關鍵字時，前一兩頁的網頁設計公司便是，這些公司的共通點就是替不是資訊背景的廠商設計網站與系統開發。

如果你進入到的是網站設計公司，你能夠磨練到的就是開發速度與實務經驗，通常這類型的公司會有許多業務去承接大量案源，再丟回到公司讓網頁開發部門去消化。你身上背的案子同時會有兩三件以上，甚至數十件都是有可能的，在案件如此密集的情況下，你自然必須加緊跟上專案的腳步去達成專案，好處就是能讓你在短時間大幅度提升。

但這種公司會有一個缺點就是到最後你會發現業務、設計、前端爲了快速消耗一個案子，很多東西都會 SOP 流程化，設計出來的東西都很雷同，你爲了要滿足公司的業務量，CSS 大部分也都會沿用，甚至把背景圖片、色系、文案改一改，新網站就出來了，幾乎不到半年你就會覺得自己做得工作相當的重複化。而且專案的生命週期又短，你根本沒時間投入新的技術在新的網站開發上。

另外也分享些現實面的故事，有幾間業界公認的血汗網頁設計公司，，工作量也大到超乎你的想像，你的長官會隨時盯著你要進度。要怎麼形容工作量呢？就好比說你回到家以後，已經累到都沒力氣洗澡，而且也累到睡不著，或是洗澡洗到睡著，瞇一下發現已經天亮又要上班了。

我的一些學生有面試過這類性質的公司，會問我說是不是就不建議去了，但其實我就會反問對方：「你能從那裡獲得什麼？」，以剛接觸頁面的學生最欠缺的就是「經驗與實戰技術」，你若能在那麼高壓性的工作環境撐個一年，就像是進入精神時光屋般獲得好幾年的開發經驗，等到你獲得你想要的東西後，自然可以再換到更好的環境，豈不是更好？

我就曾經輔導過一位雙親皆有疾病，急需提升開發能力的學生，聽了我的建議後進了火坑一年，年薪只有35萬，每天都會加班到晚上 10、11 點，旺季時甚至好幾天熬夜都有，後來第二份公司到了外商，薪水福利與生活品質彷彿有天與地的差別。

所以這類公司，我會將他歸類成「可以去練功，但不宜久待，若有覺得重複性質工作開始感到無聊，便可考慮去留」，除非你的主管是會視你的成長提供適當養分，會找具有挑戰性的專案讓你練功，且也注重你的生活品質，薪水也有加到符合期待，倒就另當別論。

接案公司 - 評估專案相性

在前面我有提到所謂的「專案生命週期」，它是什麼呢？就是你開發的時間需要多久，以及它會延續維護多少時間，例如幫一個公司作形象網頁來說好了，你做個幾頁 HTML 也不接動態程式就交件，經手開發的時間兩三天都不到，寫完後你也不必維護他，對方會再自己處理後續上架事宜，這樣子的流程你自然沒辦法從裡面獲得過多的經驗。

以致於你想用個比較進階的前端框架，但是架構就超簡單，簡直就是殺雞用牛刀，反而無法摸到框架的精髓。而且你也沒辦法思考到當今天這個服務在營運過程中，你寫得這段語法是否靈活，要加新功能時容不容易。或者是說在維護期間你體驗到了以前的寫法害自己某個環節花了很多時間在還債，讓你培養出深厚的開發思維，讓你能夠將這份經驗帶往到下份專案上。

而這種接案公司不論是 SI 還是網頁設計公司都會思考營運方向，同時承接具有挑戰性的案子讓團隊來練功。以公司來說我們都會講一年有四季，一季有三個月。公司都會視情況去承接兩、三個會 run 兩季以上的大案子，期間再去接幾個中案子與小案子。而且通常案子結案後，後續都會簽以年度爲單位的維護合約。像是這種案子，就很適合你來投入一個大型前端框架在大、中案子上面，新技術也才會有用武之地。

而這些公司也會從各種案源中思考公司方向，例如說：

- 某間大公司因爲長期配的軟體公司不錯，索性直接談併購成自己的軟體部門
- 從專案中挖出商機開發產品服務，公司除了接案外，部分的現金流也來自於自己的產品
- 初期接案爲目標，但接得案源實際上都是爲了要開發自己的產品，並會去申請 SBIR 政府輔助讓他們的產品更具優勢，並轉職成功
- 曾思考過開發產品方向，最後覺得公司還是擅長接案，客源案量充足，往後也持續穩定十幾年。

這些類型的公司就比較多練功的機會，不過凡事都沒有這麼美好，公司爲了養活員工勢必得接大型專案，但是大專案代表時程的掌握性也會跟著難以控制，畢竟架構也大的關係，所以通常一個時程 delay 又狀況多的大專案結束時，往往都會造成許多員工辭職。保哥曾經也寫過類似的文章，也歡迎你去看看。

產品服務設計 - 初期

有些剛起步的新創公司目標就很明確，就是自己想要設計幾個服務產品出來，因爲產品也正在設計階段，所以充滿了許多變數，但優點在於在公司草創時期，很多東西都還沒有訂定標準，你想用任何技術都具有非常大的彈性，不受歷史包袱限制。

但是你要記住一件事情，新創公司有個特點就是會不停地建立最小 MVP，盡快地丟到市場上實驗成果，收到用户回饋後再立即調整，以符合其商業模式。我們甚至會用一句話來描述創業成功的公司一般都會是「原來以爲要做**A**服務，做到一半時後來發現市場眞的要的是**B**服

務，最後轉型成**C**服務才成功」，絕大部分新創都會在 B服務就死了 90% 錢就燒光了。所以如果你在這類公司發現需求一直在變，架構一直在改那眞的是很正常的事情，請不要覺得太奇怪，爲了要一直探究市場眞正需要的服務，好讓他們能夠在極有限的時間內先建立出一個能支撐的現金流，初期目標如果沒快速達到損益兩平，幾乎都撐不到一年。所以一開始的 CODE 都有可能髒到不行還蠻常見的，若眞的有做起來，在產品中後期階段就必須再將技術債給補起來。

所以你有機會待到這種公司的話，能有助於提升你的產品思維，公司小小的好處就是你的同事有可能就是老闆與創辦人，通常會自己鼓起勇氣創業代表他們在社會上擁有足夠的社會歷練，他們看到機會了也想要拼一把試試。在日積月累的相處下，你也能從他身上學到一招半式，好讓自己的職涯中能多點武器在身上。再加上你也跟著公司一起從零打磨產品，箇中細節也能整個跑過一次流程，日後當當你自己也想到許多好點子時，自己也能倚賴這些經驗自己嘗試開發出一些產品，不是很棒嗎？

有些人可能會對什麼叫做「產品服務」有些疑惑，例如公司設計一些線上服務，讓潛在使用者因爲有需求而買單，就算是老點子也會因爲科技的進步有更多的機會，來舉一些例子：

- 傳統都是打電話叫師傅來修水電與裝潢，現在有PULO、呼叫師傅用APP幫你線上媒合附近師傅
- 以前吃水果都必須自己到市場買，現在有果物配定期幫你配送你想吃的高品質水果
- 受夠傳統計程車參差不齊的素質水平，有UberAPP 幫你媒合司機，用線上評價系統來審核司機素質
- 想要找一個顧問，但求助無門，有鐘點大師讓你能和顧問們有接軌機會

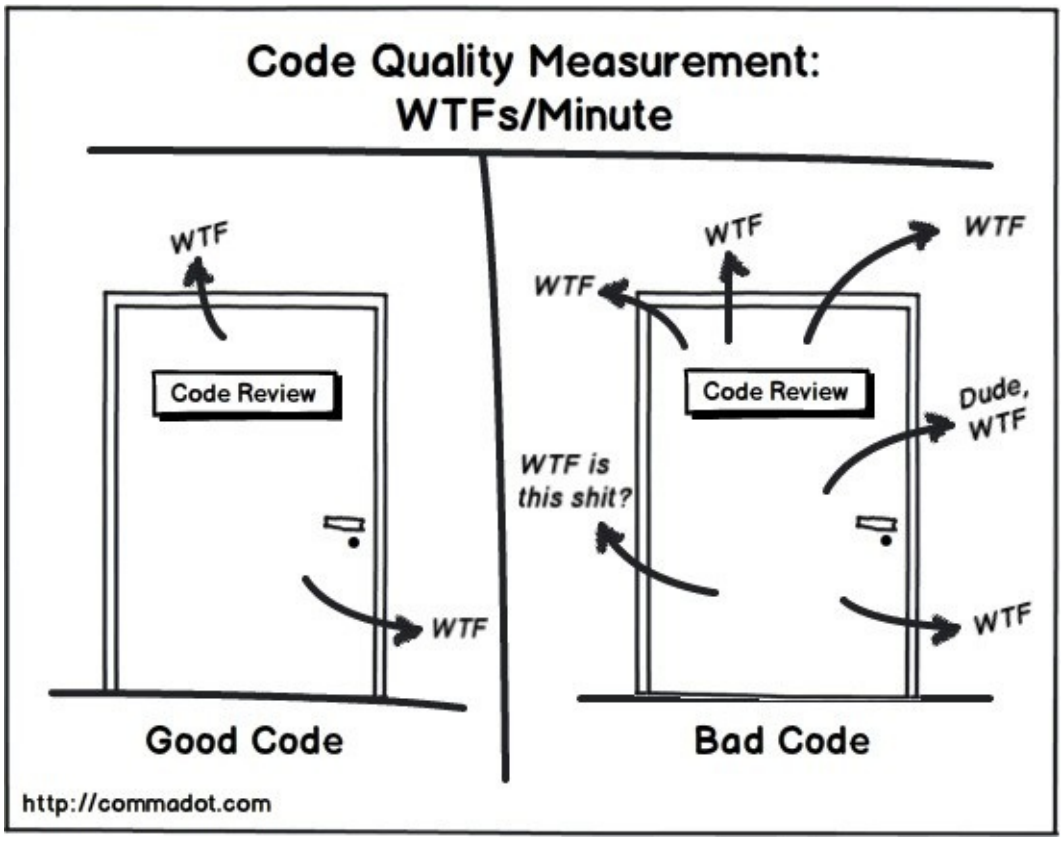
產品服務設計 - 中期

中期是什麼意思呢？就是公司因爲這服務有站穩腳步，現金流有可能即將損益打平，接下來要持續擴增人力來進行開發與重構，重構的意思就是程式碼因爲初期更新頻繁導致 CODE 毫無邏輯性可言，所以必須重新調整網站架構，以符合未來擴增時能夠更加具有彈性。

所以當你進入公司後看到核心程式碼時，有可能會遇到下圖的右邊情形，髒話滿天飛，因爲 CODE 太髒了你超無言。但我必須和你說，這也是寫程式必經的過程，運營一陣子的公司或多或少都會維護一些專案與產品，自然會需要聘請前端工程師去做程式上的擴充維護，去改別人寫的程式碼，又稱做改屍體。

這段期間你會學到許多奇淫技巧，例如：

- 臨時有個重大 BUG，你用了個小技巧先擋著，雖然很不正規但反正問題有先解決了。
- 資料庫邏輯你改不動，情急下你只好用 JS+CSS 障眼法的方式先處理掉
- 有個邏輯會造成網頁 loading，經由你重構後變得載入速度超快，超有成就感
- 大量接觸到具有一定架構等級的 CODE，浸淫之下能判斷出程式碼品質優劣



產品服務設計 - 後期

如果公司已經有一個持續幫助他們賺取利潤的服務時，程式架構就已經定型了，你頂多是增加些小功能或改 BUG。在這種專案下，你可以看到很沈重的歷史包袱，試圖想改哪裡的 CODE 但又很擔心他會爆炸，會覺得自己是在拆炸彈。有可能你會花好長一陣子才有辦法理解整個程式脈絡，短則三個月長則一年。

我會建議如果公司只讓你負責這份工作項目，沒有其它新案開發時，不要久待。因爲沒辦法讓自己有經手過完整的開發流程經驗，我就曾經遇到幾位前端朋友因爲初期相中他的薪水福利後，過了幾年才驚覺自己技術不增反退，但又因爲成家立業現金流開銷大不敢辭職，最後服務消失被資遣，就又必須花更多的技術債補起來，眞的得不償失。

我必須再次強調，公司跟員工彼此是各取所需的關係，你需要「金錢、成就、技術能量」，公司需要你「爲他們建立商業模式」，如果公司在這三種少給了你其中一種，你就要有警覺性了。因爲並不是每間公司都會看重員工成長，做人還是要多靠自己，讓自己保持競爭力。

支援性質部門

這種工作內容也相當有趣，這是我自己取得類別，短時間我也不曉得要怎麼描述。舉大型電商來說好了，每個部門各司其職，有數位行銷、國際行銷、業務、品管、出貨、客服、會計、軟體部門等等。你要做的事情就是協助各個部門的需求，開發出能加速他們工作效率的

技術工具。

來說一些例子讓大家有更多的聯想空間：

- 客服寫 mail 覺得一直回重複的話很麻煩，開發出罐頭訊息服務讓它們將常見訊息進行分類管理
- 行銷與業務在訂定價格時常會去看競爭對手網站賣多少，寫一隻網頁爬蟲，訂時去撈各大賣場的價格整合在同一頁面上
- 出貨部門每次都要用難用的第三方貨運公司的系統查資料，你寫爬蟲幫他把資料都整理起來，方便他出貨
- 部分同仁都會在網頁做許多重複性質的工作，你寫 chrome 插件提升他們的上稿速度
- 行銷要打一系列的活動，需要客製化活動網頁
- 幫業務埋追蹤 CODE，觀察使用者轉換行爲，並再優化 Landing Page

除了電商外還有很多單位都需要一個軟體部門，例如醫院、銀行、學校、上市上櫃公司等等都很需要，畢竟優化工作效率絕對脫離不了軟體開發。

結尾

希望我的各種舉例讓你能夠對公司的商業模式有一定的輪廓印象，你可以從公司裡面觀察出專案生命週期，這樣才能夠更深入瞭解能從中獲得什麼，凡事絕對沒有對或錯，好或不好，你需要做的事情就是不停地反思「你想在這間公司獲得什麼？」，是想先追求大量的金錢嗎？還是想快速磨練開發經驗？或是每天都能有滿滿的成就感？初期投入前端你是不太可能三者兼備的，魚與熊掌不可兼得，想清楚後，就把時間花下去仔細打磨自己吧。

如何規劃前端職涯，成爲百萬前端

我自己是在決定成爲前端工程師的第五年才達到百萬年薪的，這個時間點僅供參考，有時候選對公司或是有遇到貴人的話可以加速很多，像是初期我根本沒有任何資源，也沒遇到mentor，在前端路上有種瞎子摸黑路難上加難，所以這裡我將分享給你如何規劃自己的前端職涯，你會發現要掌握高薪並不是你要會很多很潮的技術，而是你是否有足夠的能力獨當一面。

請以年薪爲成長幅度單位

很多人都會習慣以月薪當做談薪條件，但我在指導學生時都會以年薪來討論，但公司只會跟你講月薪會是多少，那麼你在面試時，就必須主動詢問以下項目：

月薪是否有扣勞健保

假設來說人資跟你談薪水說是 3 萬，結果當你拿到薪資單時就變成 28,990，相信你一定會黑人問號，因爲公司替你保勞健保時，企業會替你負擔大部分費用外，你個人也需要負擔費用，可以參考這個勞健保級距表。12 個月累積下來就將近一萬以上，所以在談薪水時，也務必將此細節詢問清楚，以避免拿到薪水跟預期會有落差

保**N**個月

有些企業都會寫類似 保 13、保 14 個月，這是什麼意思呢？意思就是保證你的薪水可以拿到 13 個月，例如你薪水 3 萬的話，13個月就是 39 萬，那這第 13 個月會怎麼發呢？一年不是只有 12 個月？其它的部分就會挪到 年終、三節獎金來平均分攤。所以你也可以不用提到是否有年終獎金，跟公司談好確定會有 保 N 個月就好。

三節獎金、紅利**(bonus)**

三節就是春節、中秋、端午都會給額外的禮金，但是若是一些中大型公司會保 14~16個月時，有可能也會在三節發給你一個月的獎金，而有些公司會和你說保13，不含在三節，會額外發送。

至於紅利則是公司有賺錢的話，會定期提供紅利分給所有員工，發送的頻率有一季也有一年，端看公司屬性。

最後來考考各位，假使人資跟你談薪水，說月薪 4萬(實領)，14個月，三節固定各發送1200，沒有紅利的話，年薪會是多少呢？
也就是 4_14(月)+3_1200(三節)= 563,600。

瞭解薪水運作流程後，接下來我們就可以再繼續往後面看下去。

年薪 **30~55**萬：讓自己成爲一塊海綿吸收各種知識

如果這是你初入社會的第一份工作，很多事情你都會覺得很新鮮，這裡我會給你幾個建議：

跨部門的知識掌握起來

一間公司的運作通常都是多個部門合作，你需要瞭解自己在跟其它部門協作時，和他們建立起共通語言，而且也要懂得他們的常見術語，不要不懂裝懂，眞的不會就直接問個清楚。公司爲了要完成一個項目時，每個部門都會有他們的需要達成的目的。舉例來說每個部門就會類似這樣問你問題：

- 企劃：「你可不可以先壓個 deadline 給我，因爲依照我們的 KPI，這東西必須下個月就必須上線吸收到足夠會員數才不會違約」
- UI：「我這樣設計的動畫效果你開發得出來嗎？需要我先出圖給你？還是檔案丟給你自己切？還是要用 keynote 拉個時間序給你？」
- 行銷：「這次專案要用 GTM 整，客户說想看 funnel 看哪裡需要加強，你可能要問下 A 同事這段要怎麼埋 CODE」
- 後端：「測試用的 API 我先丟 POSTMAN 給你哦，repo 我開好放到 bitbucket 上你可以再 clone 下。」
- QA：「我剛測試這頁面有些問題，有開 ISSUE 上去，你再看看」

有沒有覺得上面有很多外星語言？各種完全聽不懂的術語，什麼是 funnel，爲啥要把 CODE「埋」起來？POSTMAN 又是啥？KPI 又是什麼單字的縮寫？

所以在初期你必須和各部門同事建立出溝通默契，必要時也必須瞭解該領域的基本觀念，好讓討論能夠更加順暢。這樣你在找下份工作時，面試官也可以從中瞭解你有跟各部門合作的，不用擔心協作時還得增加過多溝通成本。

瞭解你的公司屬性

如同前面章節所提，接下來你要好好看清這間公司的屬性，你能從他身上學到哪些東西？有沒有誰針對某領域特別強，可以從他身上學到不少東西。在一間公司裡每個人都有可能是你的老師，好的我們吸收起來，壞的我們引以爲戒。讓自己身上可以擁有更多武器。

像是以前我的老闆周姐就是一個溝通整合能力很強的人，雖然她完全不懂程式，但卻能跟工程師討論程式邏輯。在我還是菜鳥時，她也常帶我去開會見見世面，往往會議桌上都是董事等級的人，有時候大家七嘴八舌討論不出個共事來是，他卻能夠邏輯很清楚地將大家的意見整理引導出一個共識，周姐身上就是有種像是海賊王身上擁有的霸氣，一出手就能將問題搞定。

而過幾年有一次我跟一位老客户討論專案時，客户就和我說：「你越來越厲害了，剛有一度我甚至以爲自己在跟周姐討論。」我才發現自己在耳濡目染之下也從周姐身上學到了一招半式。

所以除了前端技能外，如果有大前輩在公司，就好好請教他讓自己升級吧。

設計出自己的代表作

進入公司後，當你認爲這間公司已經沒辦法再讓自己成長，也覺得自己今非昔比，想要跳下一間福利更好，更具挑戰性的公司時，記得要先有自己的代表作，什麼代表作呢？就是今天從你做的前端介面時，就能看出你的實力在哪。如果你的實力明明很好，但是作品卻搬不了檯面，你自己都不認可自己了，你有什麼自信能夠讓面試官覺得你 ok 呢？

假使這間公司無法讓你自己做出代表作，那就自己花額外時間做 side project，或是跳到其它公司，但薪水水平有略高一些的公司，而其中投入的技術是你想精進的再繼續努力。

年薪 **55 ~70** 萬：有接觸過完整專案週期

很多人都會卡在第二階段，有些前端都會覺得就算再怎麼找薪水都遊走在第一個級距，到底該滿足哪些項目才可以到達這個級距呢？你至少在那之前先具備以下事項，若沒有，你還是必須再找適合的公司補足這段落差：

有經手過中、大型專案

所謂的中、大型專案是以專案複雜程度而訂，時程是要以一季、年爲單位，且建置完後續也有在營運維護的階段，較常見的里程碑就像是電子商務網站、B2B 系統等等。

如果開發週期都是以幾天幾週爲單位，那表示專案內容相對來說也較小，面試官在看你的歷年專案時也看不出你的經驗深度，所謂開發並不單純寫 CODE 完就沒事，更包含重構、測試、維護、架構、團隊分工等事宜，如果你通通都沒經歷到，那你怎麼能讓公司相信你將一個千萬等級的系統與你共事共同開發呢？

前後端部門團隊協作

是否有團隊協作的經驗，例如一個系統由幾個前端與後端共同完成，並實際有用到版本控制系統進行分工。所以在面試這份級距時，你表現出來的內涵是否能夠讓面試官相信讓你加入的話，不用花太多溝通陣痛期，就能快速進入團隊狀況提升產能才是最重要的。畢竟在這個階段要得就不是必須花時間培養你，而是你必須盡快能成爲戰力才是重點。

年薪 **70 ~100** 萬：有機會主導整個前端架構

到這個階段後，通常要的都是 領袖 (leader)、資深(senior) 等級了，在年薪 55~70 萬時，你可能還會跟資深資淺同事討論到某某環節要用什麼框架會比較適合，但是身爲最資深的你勢必要成爲「能夠做決定的人」，而且你就是能依照你的思維建構出整個程式世界觀出來。

當準備要開發一個大型系統時，菜鳥工程師所想的都只是部分程式碼邏輯是否可能，但你則能看得更遠，甚至能預期這專案成敗的關鍵點在哪些細節。這有個俗語就是，菜鳥只會看見一棵樹就自以爲看到全世界，但資深開發者則能看到整個樹林，又俗稱「見樹又見林」。

你不會拘泥於框架的限制，在你的宏觀世界中，任何語言都萬法不離其宗回歸原始，原本都還會在專案上用各種奇淫技巧，最後慢慢反璞歸眞用最直白的語法一刀切中核心完成任務。

在這個階段你開始有機會自己負責一個專案，並試圖帶幾個資淺前端一起將整個專案運作起來，並瞭解該如何分工，如何模組化設計以達到最大的產能。

年薪 **100~150**萬：全面性地由你主導整個產品線的前端

到這個等級，就是你已能設計出一個成功的服務，在歷經不停地失敗成功的數百數千次開發輪迴中，你早已知道要設計出一個成功的項目，要具備哪些關鍵因素(Keystone)，除了帶領自己的前端團隊外，同時又能跨部門激盪討論出更成功的方案。

到達這個階段，除了你仍在公司幫忙開發產品外，又有可能你早已有創業的打算，並試圖自己做出心目中想做的產品。或者是出技術股共同創立公司。

技術已是僞議題

我相信你已經發覺在上面的篇幅，我完全沒提到「任何你需要具備的程式語言」，其實在我們業界都是會用下面的方式來判斷你是否靠譜：

- 實習生：自己也不知道在公司要做什麼，別人要你做還必須教你才有辦法達成
- 初級(Junior)：不知道怎麼做，但給你方向的話還是做得出來
- 中階：大部分都知道怎麼做，但遇到不確定的還是會詢問他人意見
- 資深(Senior)：知道自己要做什麼，擁有自主解決問題能力

英文重不重要？

在第一份薪水只要你有國中等級程度，其實是絕對可以勝任的。但如果你想要達到 55~70 萬的級距可以比較順利的話，外商也會是選擇之一。就以我輔導到現在的學生，有五分之一都是在外商工作。

當然不是要你完全精通，至少能夠與外國人溝通即可，就算不用那麼標準也沒關係，只要你能夠讓對方聽得懂就 ok，外國人也並不是那麼注重語法的。另外一點就是閱讀能力，至少要看得懂英文 API 文件，並實作出功能出來。當國內沒資料時，你至少要能看得懂 stackflow 的隻字片語，而英文閱讀能力至少要達到「能看出這討論就是你要的答案」才算 ok。我也曾經看過有工程師的面試條件寫「會從 stackflow 尋求答案」。

所以會建議你在第一份工作找到後，再依序規劃將英文能力補起來，讓以後自己的管道也會更多些。

遠距工作(Remote)正夯

我常會跟學生說全世界都很缺工程師，所以如果你英文溝通能力還算 ok，也可以在台灣 remote 國外的工作。現在在台灣也越來越盛行了。

我常常會遇到老家在其它縣市，但在台北打拼的學生，有部分都會吐露自己想要回家鄉，我都會勉勵對方先磨練好經驗，等到技能足夠後再回家鄉 remote 也 ok。像是我就認識很多工程師回高雄、台南 remote 北部公司。現在這年代所有東西都是雲端，只要你知道 Git repo 位置、伺服器放在哪個空間上，就能遠端與其它工程師協作，任何溝通都可以用 slack、trolle 等軟體來補充溝通。

我就曾經輔導過一位住在花蓮的學生，並也在當地置產結婚並買房，他那時也很猶豫到底要不要去北部磨練一陣子再回家鄉，於是好巧不巧有廠商敲我希望找 remote，在經過層層關卡面試後也如願獲得前端職缺。

未來遠端工作會越來越普及，若你心動的話，也能提早開始準備，同時也推薦你 remote 台灣社團。

看清薪資天花板，下一步還能怎麼走

在 2017 年，我看到現在如果純做前端的話，薪資我看到現在開最高是 150 萬，這是目前的前端薪資天花板，詳細也可到mit.Jobs觀看，但未來就不一定了，伴隨著前端能實作的事情越來越多，例如：

- WebGL 建置 3D 環境
- React Native 建置雙平台 APP
- AR、VR
- PWA
- AMP

在前端領域中，你也必須投入幾個更專精的方向，好讓你與其它資深開發者相比更具競爭力。除此之外的話就是考慮讓自己成爲 T型人才，例如後端、數位行銷、營運等等，尋找自己有興趣的項目來精進，同時結合本業擁有更好的發展。

不只本業收入，讓自己的綜合能力超過百萬年薪

以我自己來說，我除了本業工作外，我也用了我會的技能幫我賺到許多額外收入，以前的我也曾經一對一遠端教學、技術演講、業界授課、自辦課程、學校授課、線上課程等等，你也可以思考例如外包、與朋友合作一個小項目等等，都是很好的方向，希望這些建議能夠有助於你能夠更有效率的規劃自己的前端職涯。