編程

- 只要競技程式設計能激勵你，那就沒問題。
- 程式設計就像寫作，是一種創造性的活動。
- 做自己的專案，寫你的技術博客。為一個你會維護多年的專案編程，就像維護一個長達十年的博客一樣。
- 通常，你不需要追求當前的技術熱潮，因為許多趨勢幾年後就會消退。
- 追求你的好奇心，為自己而編程。
- 嘗試為自己創造程式，它們不是工作任務。
- 如果你在編程時經常感到不快樂，那麼你的方法可能錯了。
- iOS、Android、後端、前端、AI 都是好的。至少可以嘗試用它們做個小專案或學習幾個月。
- 調試是關於懷疑的。不要相信你的每一行代碼；你可以想到更好的方法來做。
- 在編程中，即使是一個字符或一行日誌也很重要。它們告訴你一些事情。
- 使用編程為他人製作產品。有使用者是很有趣的。
- 你不需要苛刻。幾百個真正愛你產品的用戶比幾萬個只是喜歡你產品的用戶要好。
- 記住你為什麼開始編程，永遠不要忘記。
- 將編程中的知識應用到生活的各個方面。它們是一樣的。批量做事或一個一個來。如何將工作分為單元。每個應用程序背後的底層技術。網絡請求背後的細微差別。
- 抽象和邏輯思維。注重細節的思維。想出每一個解決方案的思維。
- 真理就是真理。通常，電腦不會錯。電路不會錯。編譯器不會錯。當有 bug 時不要感到沮喪。
- 追求優雅和簡單的解決方案。簡單是最終的複雜。你需要努力思考，留下必要的，去掉多餘的。
- 對於編程語言，能完成工作的語言就可以了。我個人推薦 Java 和 Python。
- 關注尹旺在https://www.yinwang.org。他是編程界少數的天才之一，儘管他說天才從不存在。
- 編程的知識和原則可以很容易地應用到語言學習、硬件修理、生活黑客和科學研究中。
- 對於大多數編程任務，除了高中數學外，你不需要花哨的數學。
- 幾年後反思你的舊代碼或長期維護一個代碼專案。它會教你很多。
- 如果你失去了對編程的熱情，就暫時做其他事情。
- 測試的時機很重要。自然地去做。你通常不需要為你的專案寫測試。盡量不寫測試，寫單元測試，寫集成測試，寫 API 測試。明智地比較它們。
- 嘗試 AI 代碼編輯器。經常使用 ChatGPT 或其他聊天機器人。由於 AI 工具現在很容易使用，你可以專注於更具創造性或重要的部分。