模倣

私は Android と iOS を学んだ経験を振り返り、非常に価値のある知識の多くが模倣を通じて習得されたものであることに気づきました。

どうやって模倣するか？二つのプロジェクトを開く。例えば、Jack のフレンドサークルの実装を模倣する場合、彼のコードを 2 行見て、それから自分のプロジェクトで彼のコードを書く。理解できない場合はまた戻って見て、理解できたら自分の考えで書く。一度にたくさん書く。

この方法を通じて、Android のプルダウンリフレッシュとウォーターフォールリスト、iOS の手書き UI などを自分で書けるようになったことを深く覚えています。

Android のプルダウンリフレッシュ機能を実装したとき、初めて XML を使わずに直接 View を制御し、ジェスチャーについて理解しました。これは私にとって大きな進歩でした。時には、オリジナルのコードを写しているうちに、突然理解が深まることもあります。自分だけでは気づけないことがあるのです。どんなに考えても、見落としている部分が多く、重要なポイントが見えないことがあります。しかし、コードを打ち込み、手を動かすことで、より没頭し、なぜそうなるのかを頻繁に考え、より多くの細部に気づくようになります。その結果、その仕組みを理解できるようになるのです。例えば、前回の視差効果の記事のように、最初は背景画像が横や縦に移動することを知りませんでしたが、コードを模写するうちに多くのことに気づき、毛ガラス効果の作り方も学びました。

iOS では以前、私は xib をよく使っていました。手書きのコードを試したことも何度かありましたが、そのたびに諦めてしまいました。例えば、友達の輪（WeChat の Moments のような機能）のプロジェクトを模倣する際、まるでコピーしているかのようでした。名前が時々異なる以外は、ほとんど同じでした。しかし、それでも手書きコードの重要な点に気づくことができました。例えば、リストのセルについて、高さが内容に応じて変化する場合、描画する前にその高さを計算し、それをテーブルビューに渡す必要があります。では、どうやって高さを計算するのでしょうか？一つの方法は、固定された幅を使って高さを計算することです。こういったことは、コードを書き写しているうちに自然と気づくものです。

模倣するときは、往々にしてより没頭しやすいものです。私がコードを見ていると、すぐに眠くなってしまいます。一つは頭が疲れるからで、もう一つは体全体が静かで、休息状態に入りやすいからです。私の友達はコードを見るとき、メモを取ります。Zhihu では、issue を解決するためにコードを見て、プロジェクトに慣れるという人もいます。これらはどれも良い方法で、要は参加すること、ただ見ているだけではいけません。もちろん、ただ見て考えるだけで長く続けられる人もいるかもしれません。

模倣するときと、自分で創造するときは異なります。模倣の方が簡単で、なぜなら模倣なので、同じコードを書いて、そのままコピーすればいいからです。創造のプロセスはそれよりもずっと

難しいです。他人のアプリを模倣するのは比較的簡単で、他人のレイアウトを見れば、frame を計算しやすいです。模倣は少し簡単なので、長続きしやすいです。難しすぎて諦めることはありません。しかし、模倣から得られる知識は、創造から得られるものよりも多いことがあります。

模倣は、試行錯誤を減らし、正しい知識をより早く習得する助けとなります。プログラミングを学ぶ際には、プロジェクトを行いながら Google で検索することができます。しかし、この方法では多くの試行錯誤が発生し、あれこれと探求することになりますが、特に初心者にとっては必ずしも適切な方法ではありません。また、多くの場合、試行錯誤は無意味なものです。時には、正しい知識を理解し、それがなぜ正しいのかを知るだけで十分です。試行錯誤の際には、例えば HTML のフォーマットが崩れた場合、あれこれと修正を試み、正しく修正できたとしても、次回同じ問題に遭遇する可能性があります。模倣する場合、正しい答えは他人のコードの中にあり、それを参考にすることができます。正しい答えは手の届くところにあります。そのため、正しい知識を迅速に習得することができます。例えば、HTML のフォーマットを学ぶために、いくつかの HTML フォーマットのプロジェクトを見つけ、そのコードを一行ずつ書き写しながら、自分で試行錯誤するよりも、より速く、より良く学ぶことができるかもしれません。

模倣しましょう！