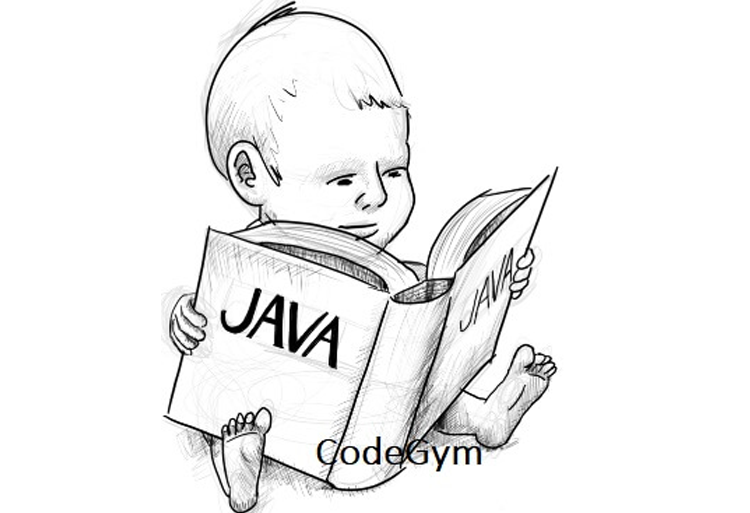
# [[Học nghề] Ngôn ngữ đầu tiên của bạn](https://codegym.vn/blog/2018/07/27/hoc-nghe-ngon-ngu-dau-tien-cua-ban/#respond)



*“Bằng cách giải phóng bộ nhớ khỏi những việc không cần thiết, con người có thể tập trung vào các vấn đề khó hơn, sức mạnh trí óc từ đó được cải thiện. Thuật ngữ chuyên ngành của bất cứ ngành nghề nào hay trong lĩnh vực thương mại đều rất khó hiểu đối với những người chưa từng biết đến chúng. Nhưng không phải là vì bản thân chúng quá khó. Ngược lại chúng được đưa ra để con người có thể làm mọi việc một cách dễ dàng hơn.”*

-Alfred North Whitehead, cuốn “*An Introduction to Mathematics*”.

## **Bối cảnh**

Bạn mới bắt đầu và chỉ có một lượng kiến thức khá hạn hẹp về một hoặc hai ngôn ngữ lập trình.

## **Vấn đề**

Bạn cảm thấy công việc của mình phụ thuộc vào việc bạn phải đưa ra một giải pháp viết bằng một ngôn ngữ lập trình cụ thể và phải cùng một tiêu chuẩn chất lượng so với các đồng nghiệp. Hoặc, việc đạt được công việc ngay từ đầu phụ thuộc vào năng lực của bạn trong một ngôn ngữ lập trình cụ thể nào đó.

## **Giải pháp**

Hãy chọn một ngôn ngữ. Sử dụng thành thạo ngôn ngữ đó. Trong vài năm tới, đây sẽ là ngôn ngữ chính mà bạn sử dụng để xử lý các vấn đề, cũng như là ngôn ngữ mặc định để bạn trau dồi và luyện tập. Sự lựa chọn này là một thử thách. Quan trọng là bạn phải cẩn thận đánh giá cân nhắc sự lựa chọn này, vì nó sẽ là nền tảng để bạn xây dựng sự nghiệp.

Nếu bạn được đề nghị giải quyết một vấn đề bằng một ngôn ngữ lập trình nào đó, hãy để sự thúc đẩy muốn tìm ra giải pháp dẫn đường cho bạn. Nếu bạn đang theo đuổi một công việc mà yêu cầu cần có một ngôn ngữ cụ thể, hãy xây dựng một ứng dụng nhỏ nhỏ xinh xinh sử dụng ngôn ngữ đó, tốt nhất là dùng một dự án nguồn mở vì nó sẽ giúp cho nhà tuyển dụng tương lai của bạn có thể dễ dàng xem qua sản phẩm mà bạn làm. Hoặc không thì bạn có thể nhờ một người có nhiều kinh nghiệm hơn và sẵn sàng để xin giúp đỡ. Có một người giúp bạn ngay lập tức có thể tạo nên khác biệt giữa việc sẽ mất vài phút hoặc sẽ mất cả ngày. Tuy nhiên, hãy nhớ rằng bạn không muốn phụ thuộc hoàn toàn vào anh bạn dày dặn kinh nghiệm này để xử lý toàn bộ các vấn đề bạn gặp.

Một trong những cách căn bản để nâng cao kinh nghiệm học ngôn ngữ đầu tiên là bạn phải có vấn đề thực sự để giải quyết. Điều này giúp việc học của bạn gắn liền với thực tế và cung cấp cho bạn một vòng lặp phản hồi đầu tiên tương đối lớn. Việc học với những ví dụ nhỏ, đã trình bày trong sách vở và các bài báo hạn chế khả năng của bạn, bạn sẽ không còn cơ hội tự áp dụng phương pháp của chính mình vào vấn đề, thứ mà chính bạn sẽ làm trong công việc sau này. Cách căn bản để gia tăng trải nghiệm này là tìm mọi cơ hội để tạo ra những vòng lặp phản hồi. Cụ thể hơn, việc liên tục tạo ra những phản hồi ngắn giúp bạn đo được tiến bộ của bản thân. Một vài ngôn ngữ có công cụ hỗ trợ tốt hơn để có được phản hồi, nhưng dù là ngôn ngữ nào, bạn có thể thực hiện một số bước để tạo ra một môi trường sandbox để trải nghiệm.

Ruby có công cụ tương tác command-line là irb. Trong Rails thì có script/console. Tương tự, Erlang có erb, Firebug cung cấp rất nhiều cách hữu dụng để khám phá Javascript runtime trên trình duyệt Firefox, bao gồm cả một trình bao tương tác. Nhiều ngôn ngữ cũng hỗ trợ các công cụ tương đương.

Thỉnh thoảng, những công cụ này là không đủ, bạn cần một môi trường sandbox lớn hơn. Dave thích giữ một lớp (class) Java trống để mở trong IDE của anh ấy khi muốn nghịch một API mới lạ hoặc tính năng nào đó của ngôn ngữ:

| 1  2  3  4  5 | public class Main {  public static void main(String[] args) throws Exception {  System.out.println(/\*play with stuff here\*/);  }  } |
| --- | --- |

Một khi bạn đã học đủ để có thể bắt đầu viết code, kỹ thuật vừa code vừa test sẽ giúp bạn tập trung vào việc thực hiện những bước nhỏ và đảm bảo rằng bạn sẽ kiểm tra những giả định của mình. Nhờ vào sự phổ biến của kỹ thuật này, hiếm thấy một ngôn ngữ nào mà không có framework để test. Đừng ngại ngần viết những test đơn giản để kiểm tra độ hiểu biết của bạn về ngôn ngữ đó, hoặc chỉ để làm quen với framework dùng để test.

Bắt đầu với việc thực hiện những bước nhỏ có phần vô nghĩa, khi bạn tiếp tục luyện tập, những bước nhỏ này phát triển tương ứng. Ví dụ, ngôn ngữ Ruby có chức năng giúp bạn áp dụng một khối code vào tất cả các phần tử của một danh sách và thu thập kết quả vào một danh sách mới. Bạn có thể viết code như sau để có thể hiểu được chức năng này:

| 1  2  3  4  5  6  7  8  9 | require "test/unit"  class LearningTest < Test::Unit::TestCase  def test\_my\_understanding\_of\_blocks\_and\_procs  original = [1, 2, 3]  expected = [2, 3, 4]  p = Proc.new { |n| n + 1 }  assert\_equal expected, original.map(&p)  end  end |
| --- | --- |

Không chỉ dùng learning tests cho việc học một ngôn ngữ mới, bạn có thể dùng cách này để tìm hiểu thư viện của mọi người hoạt động như thế nào. Dần dần những vendor test này (như Ade đã đặt tên chúng trong một cuộc phỏng vấn nhỏ tại Hội nghị London Test Automation) có thể dùng để kiểm tra xem bản cập nhật mới nhất của một thư viện có khiến hệ thống của bạn sập hay không. Khi hệ thống sập, những bộ test trỏ tới thư viện mới, coi đây là nguyên nhân gây ra vấn đề bởi các thư viện đó chỉ thực hiện những chức năng mà bạn cung cấp chúng để thực hiện. Trong những hệ thống ổn định, chúng có thể được dùng để kiểm chứng xem một bản cài đặt khác của một thư viện có cung cấp cho chức năng làm mọi thứ bạn cần hay không.

Cuối cùng, bạn sẽ tiến bộ từ việc chỉ viết learning test đến viết test để kiểm tra code thật sự của bạn hơn là việc bạn hiểu về cấu trúc và các API. Dần dần, bạn sẽ thấy rằng có nhiều kĩ thuật hơn cách kiểm tra unit đơn giản mà dùng máy tính kiểm chứng công việc của bạn cũng như giao tiếp với các thành viên trong nhóm.

Dưới đây là một đoạn thảo luận về việc học cách nghĩ khác đi qua việc học một ngôn ngữ mới, nhưng lời khuyên của Ralph Johnson (đồng tác giả của Design Patterns) cũng có thể áp dụng cho việc học ngôn ngữ đầu tiên.

*Hỏi: Vậy giả sử ai đó muốn học để nghĩ khác đi, bạn sẽ đề nghị ngôn ngữ nào? Ruby, Python, Smalltalk?*

*Trả lời: Tôi thích Smalltalk hơn. Nhưng không quan trọng tôi thích ngôn ngữ nào. Bạn nên chọn ngôn ngữ dựa trên người song hành cùng bạn. Bạn có biết ai là người yêu thích một trong ba ngôn ngữ này? Bạn có thể nói chuyện thường xuyên với người này không? Thậm chí tốt hơn, liệu bạn có thể làm một dự án với người này không.*

*Cho đến nay cách tốt nhất để học một ngôn ngữ là bạn làm việc với một người là chuyên gia về nó. Bạn nên chọn một ngôn ngữ trên những người bạn biết. Một người chuyên gia là tất cả những gì cần có, nhưng bạn cần một người như vậy.*

*Trường hợp tốt nhất là bạn làm việc với một chuyên gia trong một dự án sử dụng ngôn ngữ đó, dù chỉ mỗi thứ năm hàng tuần. Sẽ tốt nhất nếu bạn tự làm dự án và mang theo code mẫu tới chuyên gia vào bữa trưa, hai lần một tuần.*

*Bạn có thể tự học một ngôn ngữ nhưng sẽ mất rất nhiều thời gian để học được cái hồn của một ngôn ngữ trừ khi bạn tương tác với những chuyên gia.*

-Ralph Johnson nói về học một ngôn ngữ-

Lời khuyên của Ralph liên quan trực tiếp tới Find Mentors pattern và chỉ ra rằng việc học của bạn nên có các cố vấn. Do đó, việc một chuyên gia từ một ngôn ngữ tương tự có thể feedback nên là nhân tố chính khi bạn chọn ngôn ngữ đầu tiên. Chúng ta cũng nên chú ý rằng bằng việc chọn một ngôn ngữ, bạn cũng chọn tham gia một cộng đồng mạng thực hành với các mệnh đề có trước, các buổi tụ họp và các cơ chế để giao tiếp. Bạn nên tận dụng lợi ích của mạng lưới hỗ trợ đó để bạn không chỉ học một ngôn ngữ mà còn tham gia cộng đồng đầu tiên của Kindred Spirits. Công việc ưa thích, những ranh giới, những định kiến, những niềm tin của cộng đồng này sẽ là tất cả những gì bạn có ban đầu. Khi chọn học một ngôn nữa, bạn nên tham dự các buổi họp trực tiếp của những người theo đuổi ngôn ngữ đó (hoặc xem một trong các diễn đàn của họ) và xem liệu bạn có muốn tham gia cộng đồng đó không.

Một trong những lợi ích của việc tham gia một cộng đồng là ở đó bạn có thể tìm được những code nâng cao hơn cú pháp thông thường và bắt đầu thực hiện những gì bạn nghĩ. Nhưng đó mới chỉ là bắt đầu. Mỗi ngôn ngữ có những cái tinh tế của riêng chúng mà khó có thể nhận ra bằng cách đọc code của người khác.

Ví dụ, trong XSLT có phương thức Muenchian, trong khi Perl có Schwartzian Transform và C có Duff’s Device. Những kĩ thuật này có thể học bằng cách đọc code, nhưng việc học để hiểu tại sao chúng quan trọng và khi nào dùng chúng cần kho kinh nghiệm của cả một cộng đồng. Đôi khi kho kinh nghiệm này chỉ tồn tại theo dạng truyền miệng, bạn phải nói chuyện với người cụ thể để có được kiến thức. Có lúc kiến thức chỉ tồn tại trong một danh sách mail được lưu trữ hoặc trong một cookbook trên mạng, ở đó câu chữ thiếu rõ ràng khiến nó khó có thể tiếp cận. Trong những trường hợp này, người mới học ngôn ngữ đó phải tham gia cộng đồng hàng năm trời để có thể chạm đến kho kiến thức. Ngày nay, những điều tinh tế thường được bắt gặp trong những quyển sách như Effective Perl Programming (Addison-Wesley), Effective Java (Prentice Hall), and Effective C++ (Addison-Wesley). Đọc những cuốn sách như thế này ngay khi bạn làm chủ những cú pháp cơ bản có thể đẩy nhan việc học của bạn vô cùng và giúp bạn tránh những lỗi thường gặp.

Tất cả những điều này sẽ giúp bạn đào sâu vào ngôn ngữ đầu tiên của mình. Qua nhiều năm, ngôn ngữ đầu tiên sẽ là nền tảng để bạn học các ngôn ngữ khác. Bạn càng nắm rõ ngôn ngữ đầu tiên bao nhiêu, bạn sẽ học ngôn ngữ tiếp theo dễ dàng hơn. Mặc dù bạn sẽ chủ yếu sử dụng ngôn ngữ này để giải quyết các vấn đề hằng ngày và phát triển chức năng, theo thời gian sẽ mất thời gian để mở rộng nó vượt qua mức bạn sẽ làm trong công việc bình thường. Mở rộng ngôn ngữ theo các hướng khác thường sẽ giúp bạn khám phá ra ngôn ngữ nào vượt trội và một ngôn ngữ phải vật lộn.

Bài đăng trên blog của Eric Merritt “The Shape of Your Mind” lướt qua về cách các ngôn ngữ lập trình có thể có tác động sâu sắc đến các kỹ năng giải quyết vấn đề của bạn:

*Theo nhiều cách ngôn ngữ lập trình khá giống với các thiết bị dùng để tạo hình hộp sọ của trẻ sơ sinh ở Paracas, bàn chân của phụ nữ Hán hoặc cổ của phụ nữ Karen Paduang. Trong trường hợp của các ngôn ngữ này, thay vì định hình hộp sọ chúng có xu hướng định hình cách chúng ta suy nghĩ về các vấn đề, cách thức chúng ta hình thành ý tưởng và cách áp dụng những ý tưởng đó vào một vấn đề cụ thể. Ví dụ, nếu bạn chỉ viết code trong phiên bản đầu của Fortran (Fortran 77 và trước đó) thì có lẽ bạn thậm chí không biết đệ quy tồn tại. Ngoài ra, nếu bạn đã từng code Haskell, có lẽ bạn sẽ biết rất ít về các kiểu vòng lặp.*

-Eric Merritt, “The Shape of Your Mind”-

Một nguy cơ khi đào sâu vào ngôn ngữ đầu tiên của bạn là bị mắc kẹt. Nó có thể sẽ dính với bạn trong suốt sự nghiệp của bạn như là tiếng mẹ đẻ của bạn. Nhưng không cho phép bạn thông thạo nó để ngăn cản bạn học và sử dụng các ngôn ngữ khác. Một sự nghiệp vững mạnh nên giới thiệu cho bạn về toàn cảnh đa dạng của ngôn ngữ phát triển phần mềm. Mỗi ngôn ngữ cung cấp một cơ hội để giải quyết các vấn đề bằng cách sử dụng các mô hình khác nhau. Khi bạn vượt ra khỏi ngôn ngữ đầu tiên của mình, hãy tìm kiếm cơ hội để học các ngôn ngữ có cách tiếp cận hoàn toàn khác so với những ngôn ngữ bạn đã biết. Học viên thoải mái với một ngôn ngữ hướng đối tượng nên khám phá một ngôn ngữ lập trình chức năng. Người học việc thoải mái với dynamic typing nên khám phá cách static typing. Học viên thoải mái với lập trình phía server nên khám phá thiết kế giao diện người dùng. Suốt chặng đường, bạn chắc chắn sẽ phát triển sự ưa thích với các ngôn ngữ và các phương pháp tiếp cận giải quyết vấn đề, nhưng tránh máy móc, điều đó sẽ đẩy bạn tới việc tiếp cận một chiều. Sự mở rộng này là bước đi đầu tiên hướng tới việc yêu cầu làm chủ rộng rãi của một chuyên gia.

*Bạn không nên gắn bó với bất kỳ công nghệ cụ thể, nhưng có nền tảng rộng rãi và kinh nghiệm cơ sở để cho phép bạn lựa chọn giải pháp tốt trong các tình huống cụ thể.*

-Dave Thomas và Andy Hunt, *The Pragmatic Programmer*-

## **Hành động**

Tìm đặc trưng của ngôn ngữ bạn. Đọc nó. Đối với một số ngôn ngữ, điều này có thể dễ dàng như việc chọn một cuốn sách đã xuất bản. Đối với những người khác, chỉ có thể có một cú pháp sẵn có. Đối với những người khác, đặc trưng chỉ có thể tồn tại trong quá trình triển khai của ngôn ngữ. Hãy xem xét việc tiếp nhận thử thách viết các đặc trưng..

Nếu thư viện chuẩn của ngôn ngữ là mã nguồn mở, hãy đọc qua nó bằng cách sử dụng các kỹ thuật được mô tả trong Use the Source pattern. Bạn không thể bị ấn tượng bởi chất lượng của code mà bạn nhìn thấy ở đó, nhưng hãy nhớ rằng các người viết code đó không có cộng đồng để học hỏi và phải làm mọi thứ khi họ bắt đầu. Hãy xem xét gửi cho họ một bản sửa lỗi để khắc phục bất kỳ lỗi nào bạn phát hiện.

Cách tiếp cận khác mà bạn có thể làm để xây dựng kiến ​​thức về một ngôn ngữ là hỏi những người bạn làm việc cách họ chọn ngôn ngữ đầu tiên mà họ nắm vững như thế nào. Sau đó thêm các tiêu chí mà họ đưa cho bạn vào tập hợp bạn đã sử dụng khi bạn chọn ngôn ngữ đầu tiên. Chúng sẽ trở nên tiện dụng khi bạn chọn ngôn ngữ tiếp theo.

Cuối cùng, bạn có thể tìm hiểu thêm về các khái niệm chúng tôi đã đề cập trước đó: phương pháp Muenchian, Schwartzian Transform, và Duff’s Device. Tất cả đều được đặt tên theo các lập trình viên đã có một vấn đề thiết thực để giải quyết. Hãy cố gắng tìm kiếm những vấn đề mà những khái niệm này trực tiếp giải quyết, và tự hỏi bản thân bạn sẽ giải quyết những vấn đề tương tự bằng ngôn ngữ chính của bạn như thế nào.