CHAPTER 1

What Is Scope?

Biên dịch lý thuyết

Điều này có thể là bình thường hoặc ngạc nhiên, phụ thuộc vào những ngôn ngữ khác nhau bạn đã học, nhưng mặc dù Javascript được liệt kê vào ngôn ngữ biên dịch~~. Nó không được biên dịch ở bậc cao hơn so với nhiều ngôn ngữ biên dịch truyền thống khác~~.

Tuy nhiên, javascript engine thi hành nhiều bước giống nhau, mặc dù cách thức phức tạp hơn những gì chúng ta thấy thông thường so với ngôn ngữ biên dịch truyền thống khác.

Trong tiến trình biên dịch ngôn ngữ truyền thống, 1 khúc dữ liệu của source code trải qua 3 bước điển hình trước khi được thực thi, nói thẳng ra là “sự biên dịch”:

***Tokenizing/Lexing***

Tách 1 đoạn string từ những kí tự có nghĩa của 1 đoạn dữ liệu được gọi là tokens. Chẳng hạn như var a = 2; , đoạn code này sẽ dc tách nhỏ thành những đoạn tokens như sau: var, a, =, 2, ;. Khoảng trắng sẽ không dc tính.

 Sự khác biệt tokenizing và lexing là khó hình dung và có tính hàn lâm, nhưng nó tập trung vào những tokens đó có được định nghĩa theo hướng phi trạng thái hay có trạng thái hay không. ~~Để đơn giản, nếu thuật toán tách từ (tokenizer) .. để tính toán có hay không nên được xem xét....~~

***Parsing***

Nhận vào 1 chuỗi tokens và trả về 1 cấu trúc cây các phần tử lồng nhau miêu tả cấu trúc ngữ pháp của chương trình. Cái cây này được gọi là “AST” (abstract syntax tree).

Cấu trúc cây của var a = 2; có thể bắt đầu ở nút cao nhất được gọi là *VariableDeclaration*, với nút con tiếp theo được gọi là Indentifer (giá trị của nó là a), nút con ở giai đoạn tiếp theo nữa được gọi là *AssignmentExpresion* , bản thân nó có 1 đứa con được gọi là *NumericLiteral* (nó có giá trị bằng 2).

***Code-Generation***

Tiến trình nhận 1 AST và trả về bên trong đoạn code có thể chạy được.

Javascript engine rộng lớn , phức tạp hơn 3 bước trên như những ngôn ngữ biên dịch khác. Chẳng hạn , trong quá trình parsing và code-generation, chắc chắn là những bước để tối ưu hóa hiệu suât chương trình

Một điều nữa, Javascript engines không có????

Understanding Scope

Khi bạn thấy chương trình var a = 2; , hầu như bạn nghĩ đây là 1 câu lệnh. Nhưng thực tế đây là là 2 câu lệnh rõ ràng, 1 cái là cho sự biên dịch, 1 cái cho việc thực thi.

Phân tích var a = 2;

Việc đầu tiên biên dịch sẽ làm với chương trình này sẽ là tiến hành chia nhỏ thành các tokens.

Một giả thiết hợp lý là trình biên dịch sẽ tạo ra code bằng việc sử dụng pseudocode: “ phân chia bộ nhớ cho 1 biến, gán nó là a, sau đó gắn value là 2 cho biến đó”.

Tóm lại: có 2 hành động rõ ràng cho viến gán 1 biến. Đầu tiên, trình biên dịch sẽ khai báo 1 biến (nếu trước đó chưa khai báo) ở trong scope hiện tại. Thứ 2 khi thi hành, Engine sẽ tìm kiếm biến trong scope và gán giá trị cho nó nếu tìm thấy.

**Compiler Speak**

Chúng ta cần 1 ít dữ liệu bit để biên dịch nhanh hơn ..

Khi Engine thi thực thi code mà trình biên dịch tạo ra ở bước 2, nó phải tìm kiếm biến a để xem nếu nó dc khai báo, bước tìm kiếm này là hỏi ý kiến scope . Nhưng kiểu tìm kiếm của Engine mà thi hành sẽ ảnh hưởng kết quả của việc tìm kiếm.

Trong trường hợp của chúng ta, điều này nói rằng Engine đang thực hiện 1 sự tìm kiếm LHS (lefthand side). Loại tìm kiếm khác được gọi là RHS(righthand side).

Side ... của cái gì? Của toán tử gán

Hay nói cách khác, 1 LHS look-up hoàn thành là khi 1 biến xuất hiện ở phía bên trái của 1 toán tử gán, RHS look-up hoàn thành khi 1 biến xuất hiện ở phía bên phải của 1 toán tử gán.

Cụ thể hơn, 1 RHS look-up là không thể phân biệt được, đơn giản là nó tìm kiếm giá trị cho biến , còn LHS là tìm 1 biến gán lại cho chính biến đó.