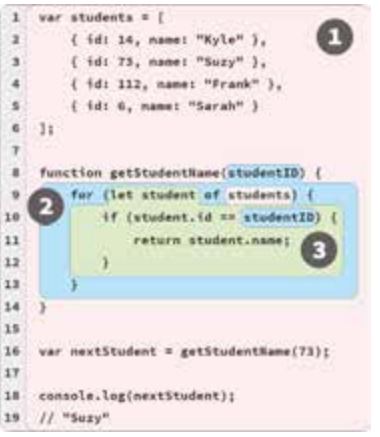
Rozdział 3
Łańcuch zakresów

W rozdziałach 1 i 2 przedstawiliśmy konkretną definicję *zakresu leksykalnego* (i jego części) oraz zilustrowaliśmy metafory pomagające w zrozumieniu podstaw konceptualnych. Zanim przejdziesz do dalszej części tego rozdziału, znajdź kogoś, komu możesz wytłumaczyć własnymi słowami (pisemnie lub ustnie), czym jest zakres leksykalny i dlaczego warto go zrozumieć.

Może pojawić się pokusa, by pominąć ten krok, ale naszym zdaniem naprawdę warto poświęcić czas na przemyślenie idei jako wyjaśnienia dla innych osób. W ten sposób pomagamy naszym mózgom przetrawić to, czego się uczymy!

Czas zagłębić się w szczegóły, dlatego w dalszej części spodziewaj się dużo więcej detali. Jednak nie zniechęcaj się, ponieważ te omówienia naprawdę pokazują, jak *mało* wiemy o zakresach. Przeznacz odpowiednią ilość czasu na przestudiowanie tego tekstu i wszystkich przedstawionych fragmentów kodu.

Aby odświeżyć kontekst naszego przykładu, przypomnijmy ilustrację zagnieżdżonych zakresów oznaczonych za pomocą kolorowych ramek (rysunek 2 w rozdziale 2):



Rysunek 2 (rozdział 2). Kolorowe ramki zakresów

Połączenia między zakresami, które są zagnieżdżone w innych zakresach, nazywamy łańcuchem zakresów (*scope chain*). Łańcuch zakresów określa ścieżkę, zgodnie z którą można uzyskiwać dostęp do zmiennych. Ten łańcuch jest skierowany, co oznacza, że wyszukiwanie można przeprowadzać tylko w górę/na zewnętrz.

„Wyszukiwanie" jest (głównie) konceptualne

Zwróć uwagę na kolor odwołania do zmiennej `students` w pętli `for` na rysunku 2. Skąd wiadomo, że jest ono kulką CZERWONY(1)?

W rozdziale 2 opisaliśmy uzyskiwanie dostępu do zmiennej jako „wyszukiwanie", które *Silnik* rozpoczyna od spytania *Menedżera bieżącego zakresu*, czy słyszał o identyfikatorze/zmiennej. A następnie wraca w górę/na zewnątrz przez łańcuch zagnieżdżonych zakresów (w kierunku zakresu globalnego), aż (potencjalnie) znajdzie deklarację. Wyszukiwanie zostaje zatrzymane, gdy odnaleziona zostanie pierwsza pasująca nazwana deklaracja w kubełku zakresu.

W procesie wyszukiwania ustaliśmy, że `students` jest kulką CZERWONY(1), ponieważ przechodząc po łańcuchu zakresów nie znaleźliśmy pasującej nazwy zmiennej, dopóki nie dotarliśmy do zakresu globalnego CZERWONY(1).

Analogicznie ustaliliśmy, że `studentID` w instrukcji `if` to kulka NIEBIESKI(2).

Ten opis procesu wyszukiwania w czasie uruchomienia pomaga w zrozumieniu ogólnej koncepcji, ale w praktyce sytuacja wygląda zwykle nieco inaczej.

Kolor kubełka kulki (czyli metainformacja o tym, z którego zakresu pochodzi dana zmienna) jest *zwykle ustalany* w procesie kompilacji w trakcie wstępnego przetwarzania. Ponieważ zakres leksykalny jest w tym momencie prawie ukończony, nic, co zachodzi później w czasie uruchomienia, nie zmienia koloru kulki.

Ponieważ kolor kulki jest znany od czasu kompilacji i nie zmienia się, ta informacja jest najprawdopodobniej przechowywana we wpisie każdej zmiennej w drzewie AST (a przynajmniej jest tam dostępna). Ta informacja jest następnie jawnie wykorzystywana przez instrukcje wykonywalne realizowane w czasie uruchomienia programu.

Innymi słowy, *Silnik* (patrz s. 41 w rozdziale 2) nie musi przeszukiwać wielu zakresów w celu sprawdzenia, z którego kubełka zakresu pochodzi dana zmienna. Ta informacja jest już znana! Wyeliminowanie konieczności wyszukiwania w czasie wykonania programu jest kluczową zaletą zakresu leksykalnego. Wszystko przebiega wydajniej, gdy nie trzeba poświęcać czasu na te wszystkie wyszukania.

Opisując przed chwilą proces ustalania koloru kulki w czasie kompilacji, użyłem sformułowania „...zwykle ustalany...". Czy może się zatem zdarzyć, że *nie* będzie on znany w czasie kompilacji?

Weźmy pod uwagę odwołanie do zmiennej, która nie jest zadeklarowana w żadnym leksykalnie dostępnym zakresie w aktualnym pliku – w rozdziale 1 książki *Na dobry początek* pokazujemy, że z perspektywy kompilacji JS każdy plik jest osobnym programem. Jeśli żadna deklaracja nie zostanie znaleziona, *niekoniecznie* jest to błąd. Niewykluczone, że w czasie uruchomienia inny plik (program) deklaruje tę zmienną we wspólnym zakresie globalnym.

Dlatego ostateczne ustalenie, czy zmienna została prawidłowo zadeklarowana w jakimś dostępnym kubełku, trzeba będzie odroczyć do czasu wykonania.

Wszelkie odwołania do zmiennych, które są wstępnie *niezadeklarowane* pozostają niepokolorowanymi kulkami w czasie kompilacji pliku. Ich koloru nie można ustalić do czasu skompilowania odpowiedniego innego pliku (-ów) i uruchomienia aplikacji. To odroczone wyszukiwanie pozwoli w końcu ustalić, w jakim zakresie znaleźć można zmienną (najprawdopodobniej w zakresie globalnym).

Jednak to wyszukanie trzeba byłoby przeprowadzić maksymalnie jeden raz dla zmiennej, ponieważ później w czasie wykonania nic innego nie może zmienić koloru kulki.

W punkcie „Niepowodzenia wyszukiwania" w rozdziale 2 wyjaśniliśmy, co dzieje się, gdy kulka pozostaje niepokolorowana, dopóki jej odwołanie nie zostaje wykonane w czasie uruchomienia.