Separate Compilation for the MVP?

Ross Tate

No Separate Compilation

Type Imports

- We can remove the dependence on type imports

V-Tables and I-Tables

- Rather than v-tables and i-tables, one can use a numeric class identifier
  - For each class/interface method, have a func that switches on the identifier
  - Just uses direct calls
- Consequences:
  - Remove dependency on typed function references and call_ref
  - Focus implementation/optimization effort on direct calls (generally easier)

Module Splitting

Program compiled whole and then split into smaller modules

Required Capabilities (for Java)

- Ability to access fields from known layouts of objects allocated in other modules
- Ability to defer loading of functionality (e.g. asyncify or promise integration)

Coarse Separate Compilation

e.g. Java Modules

Required Capabilities (for Java modules)

- Ability to represent and efficiently use v-tables/i-tables/im-tables
- Ability to import types
- Ability to access individual fields within unknown layout of separate module
- Ability to extend a struct with unknown definition in a different module
  - Both for object layouts and for v-table layouts
- Ability to initialize an object/v-table whose layout is defined across modules
  - What to do about immutable fields and (mutable) fields with non-defaultable types?
- (Guarantee that non-exported fields cannot be directly accessed)