Jbuilder design discussion

Agenda

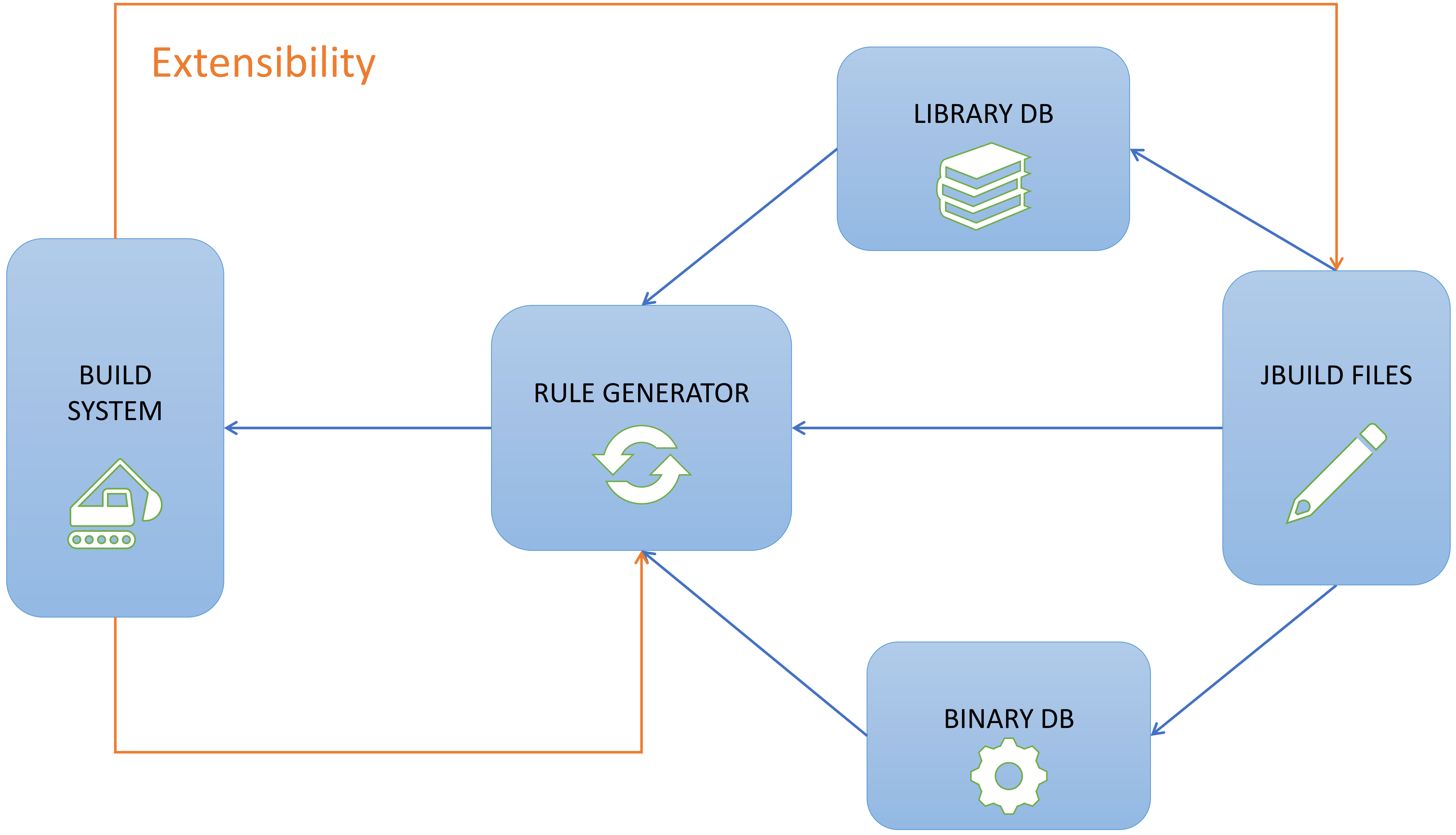
- Main topic: Extensibility
  - Overview of jbuilder, what can be done, difficulties and tradeoffs
  - Discuss what we want:
    - Make custom code generator first class (ctypes, atd, …)
    - Support other languages/backends (bucklescript, ocsigen, …)
    - …

- Other design choices and priorities
  - Alternative implementations (variants)
  - Setting default release/dev flags
  - OS-based dispatching
  - Cross-compilation
  - Multi-directories libraries
  - …

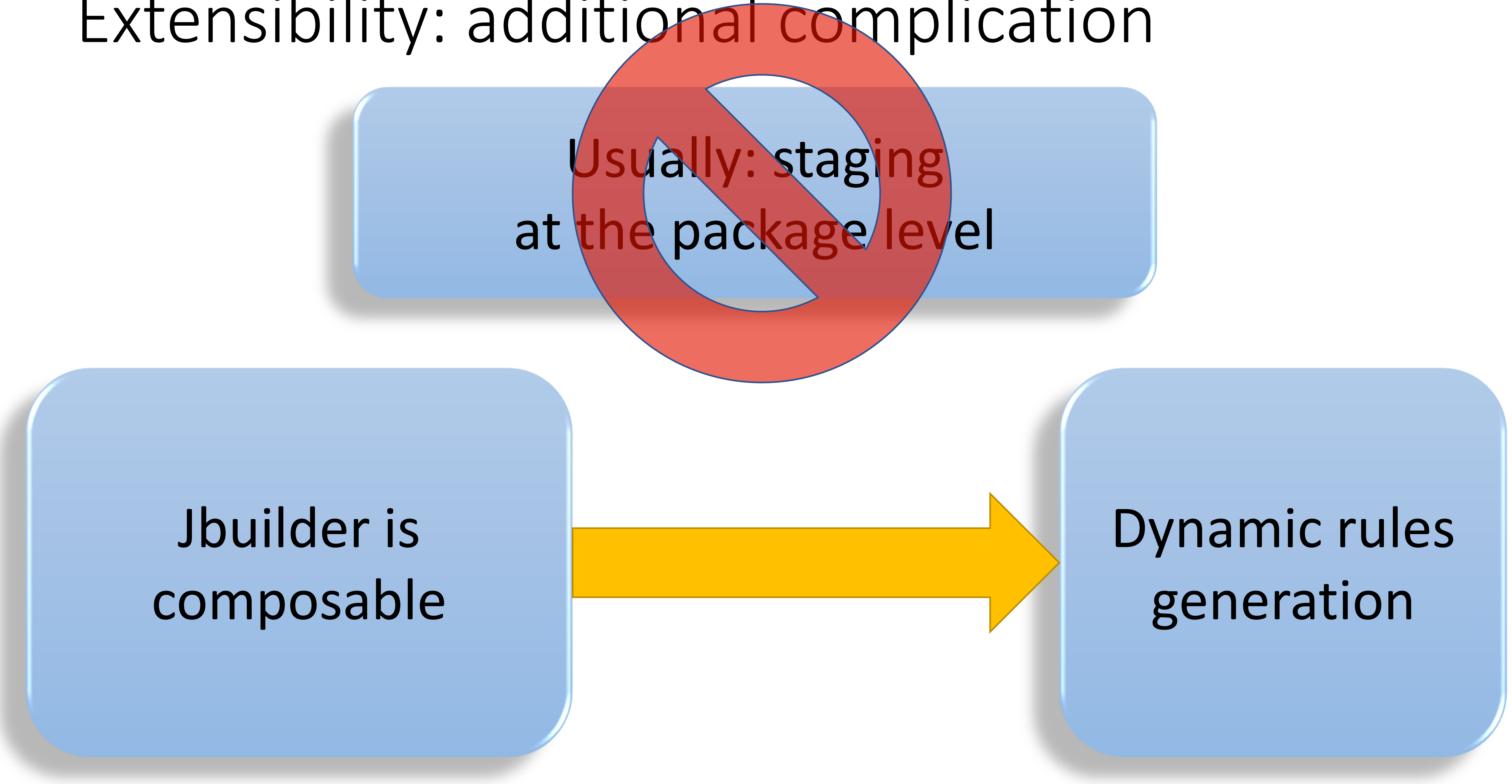
Current state

- ~250 projects in opam using jbuilder (~350 packages)
- ~1000 jbuild files (25 in OCaml syntax)
- 94 PRs, 154 issues

Overview of Jbuilder and extensibility questions


Extensibility
LIBRARY DB
BUILD SYSTEM
RULE GENERATOR
JBUILD FILES
BINARY DB

Extensibility: additional complication



Dynamic rules: prototype

```
(executable ((name foo)))

(rule (with-output-to rules.jbuild (run foo.exe)))

(include rules.jbuild)
```

Dynamic rules: limitation

- Cannot generate libraries or public executables
  - Cannot resolve library/executable names if some parts are not yet know
  - Could partition the workspace by package but not great :
    - Reduce parallelism
    - The contents of opam files becomes relevant for the build
    - Might be needed even inside a single package
    - Complicate things

~~OCaml syntax~~

Possible design for extensibility

Plugins can add new stanzas

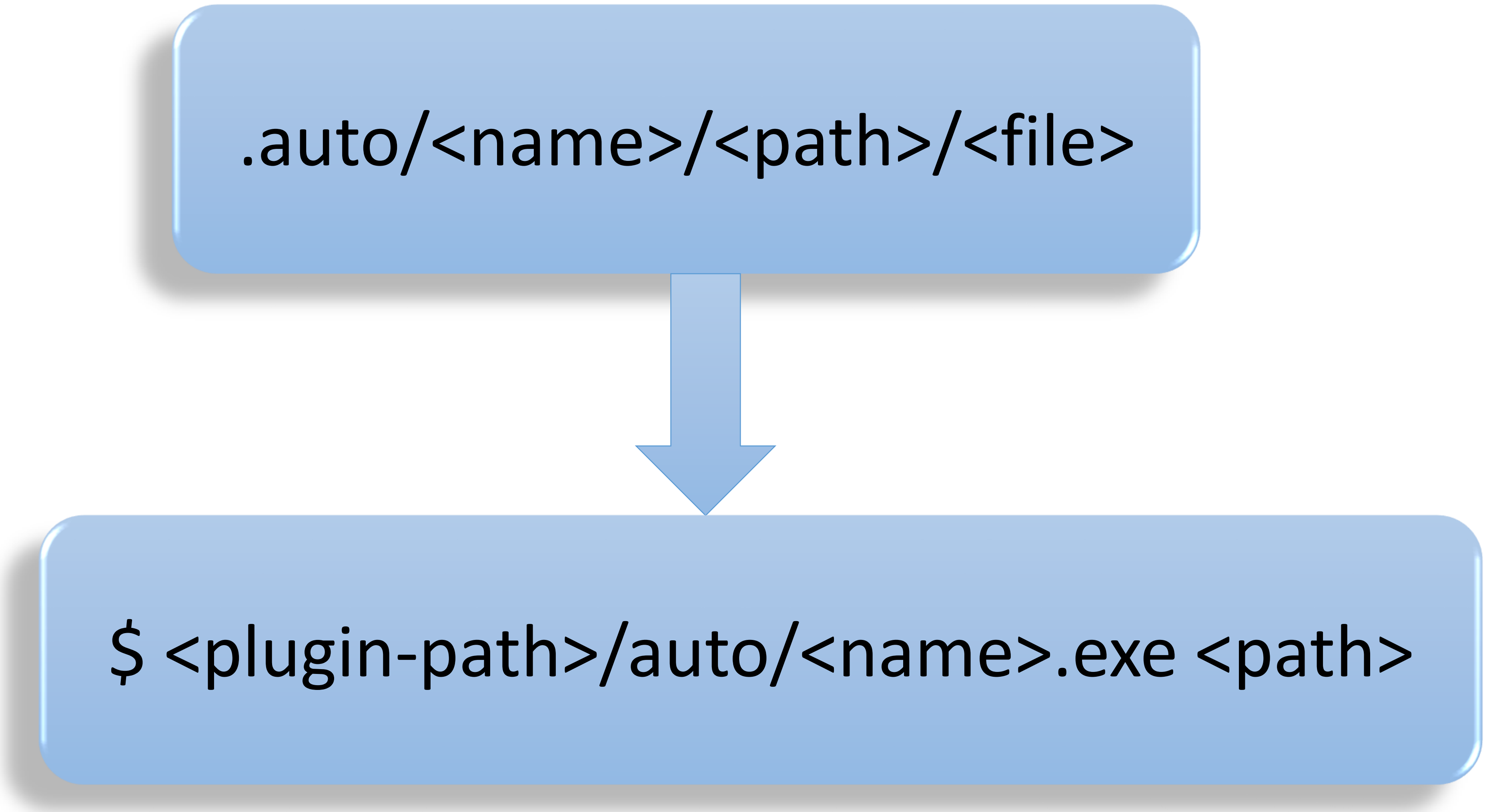
```
(ctypes.stubs
 ((names (foo bar))
  …)
```

Implementation:

- Dynlink library “ctypes”
- Execute:

```
  $ <plugin-path>/ctypes.stubs.exe “((names …) …)”
```

Global rules (ppx, js_of_ocaml, …)


.auto/<name>/<path>/<file>
$ <plugin-path>/auto/<name>.exe <path>

mk-jbuilder

- Make it easy to create custom jbuilder executables
- Full access to internal APIs