HITACHI
Inspire the Next

Current status of Flow Linter

2/25/2019

Kunihiko Toumura

Hitachi, Ltd.

1. Discussion and Results at Hursley (last Nov./Dec.)

HITACHI
Inspire the Next

- agreed about usefulness of flow verification tool.

- start discussion involving Node-RED developer community
  - Publish design notes for Flow manipulation API and Flow linter
    - finished
  - Publish code on somewhere in github.com/node-red/
    - not yet

- Verification tool should support both CLI and GUI (Flow Editor) at first design.
  - Use Bundler (e.g. Browserify) to convert node.js codes to browser code.
  - “Write once, run on both CLI and GUI” is desirable for verification rule developer.

2. Discussions and Results on Slack

HITACHI
Inspire the Next

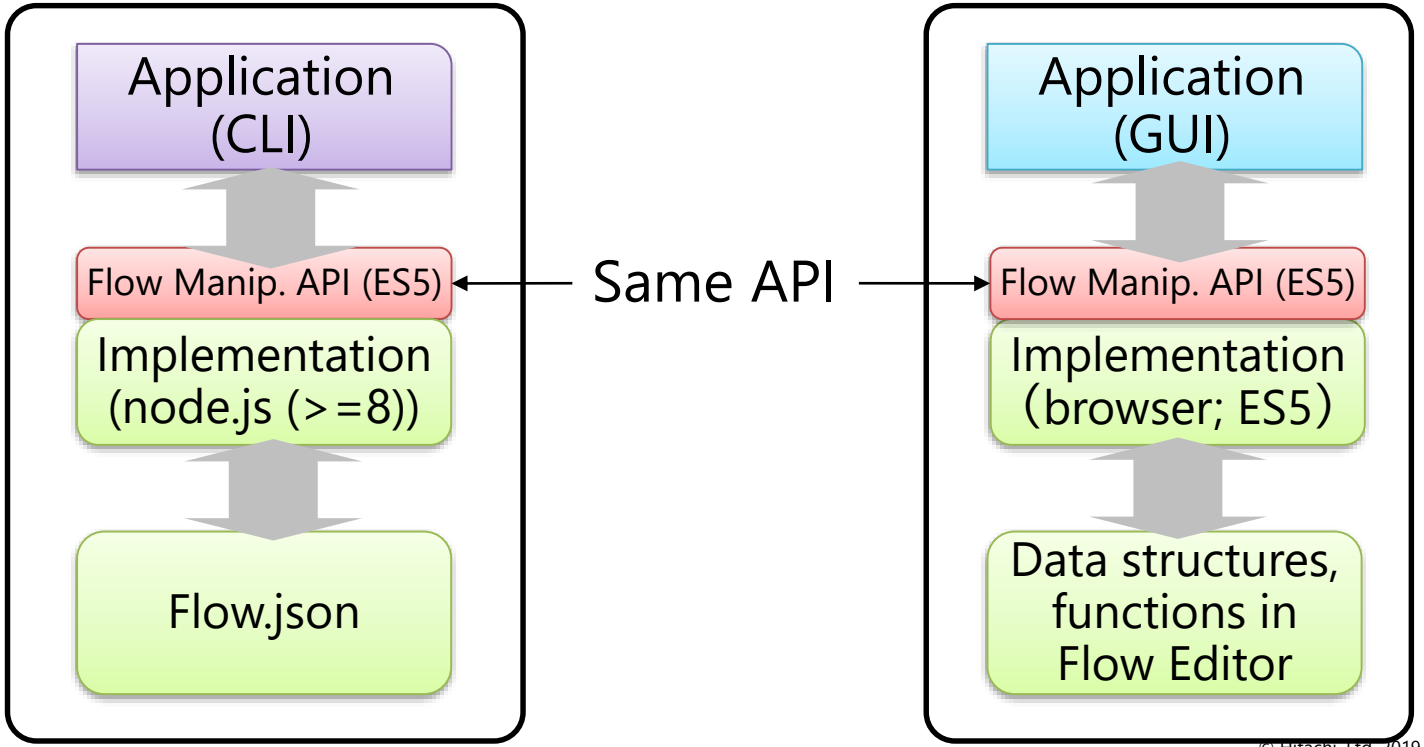
- Idea: Run verification code only on node.js side, and call it from Flow Editor when needed.
  - No. It may cause bad response on UI and increase traffic between browser and server.
    - Resolution: use bundler to generate code on browser-side.

- Issue: Current Flow Manipulation API use a lot of ES6 functions. Can we use Babel to transpile from ES6 code to ES5 code?
  - No. It is undesirable to depend Babel for building Node-RED core.
    - Resolution: Modify Flow Manipulation API not to use ES6 (and later) functions.

3. Current design of Flow Manipulation API

HITACHI
Inspire the Next

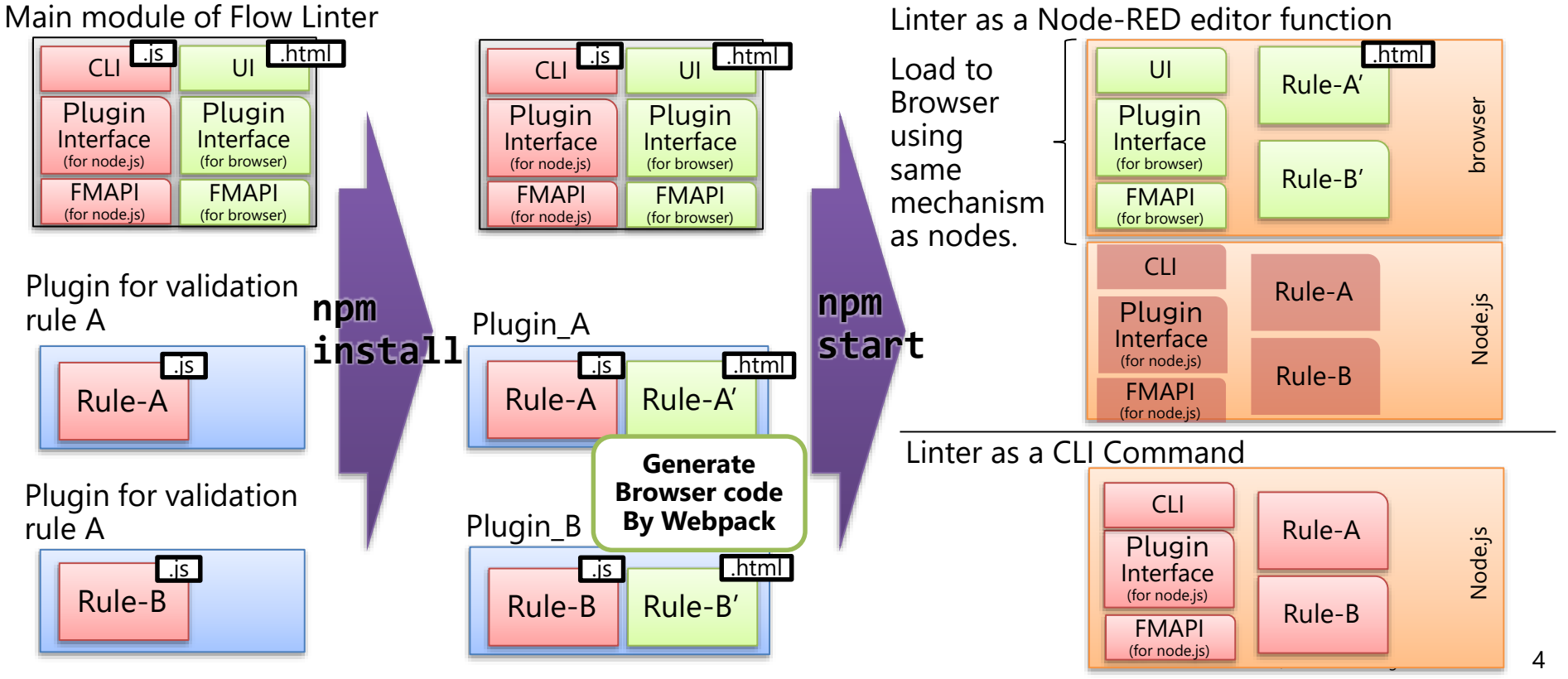
- Provide same API for Application on CLI and GUI
- Different implementations, because they handle different data structure



3. Current design of Flow linter

HITACHI
Inspire the Next

- Use 'Webpack' to bundle and generate rule code for browsers.
- Each module are loaded to browser using node-loading mechanisms.



4. Discussion

HITACHI
Inspire the Next

- Is it OK for using node-loading mechanism for Flow Linter
  - Somewhat awkward, but no additional function is needed.

- Issue: Some verification rule might be very large.
  - For example, Eslint is needed to verify codes in function node, but its size bloat up to 1Mbytes...