HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Lively Projectional Editor

David Rauch

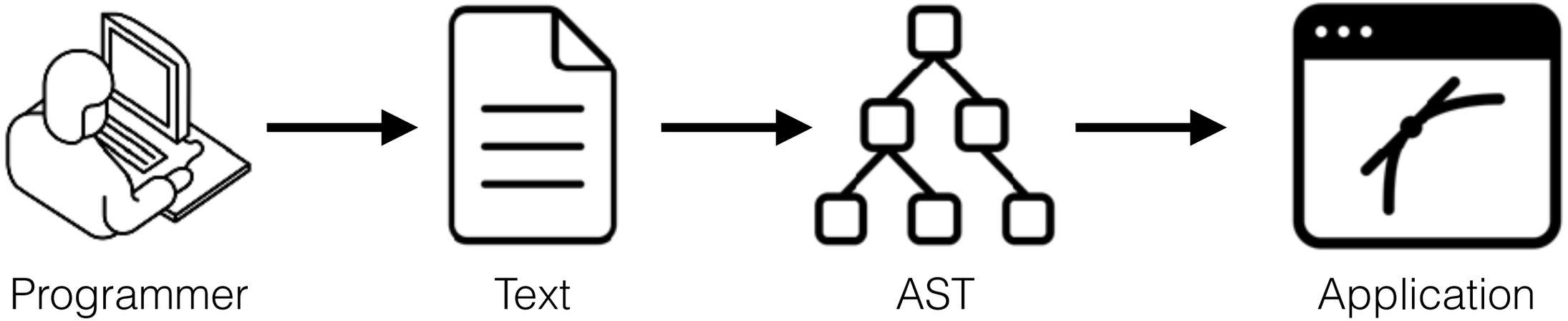
31.1.2017

Hasso Plattner Institut
Softwaredesign Seminar
Wintersemester 2016/17

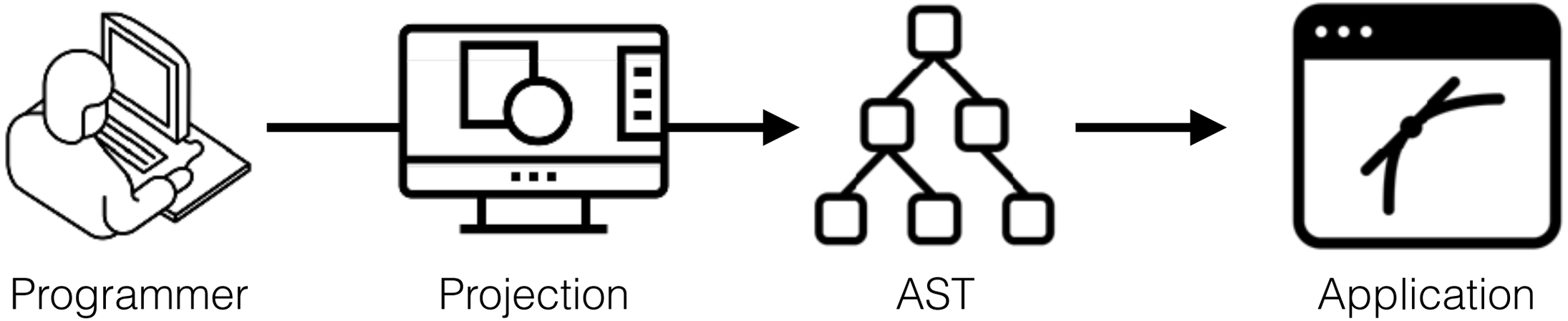
Contents

- Projectional Editors
- Lively Projectional Editor
- Demo
- Summary, Future Work

Typical Programming Workflow



Projectional Editing Workflow



Projectional Editors

Advantages / Goals

- Diverse notations
- Flexible representation
- Faster learning
- Fewer syntax errors

Disadvantages

- Unfamiliar editing
- Requires knowledge of structure
- Inefficient entering of code

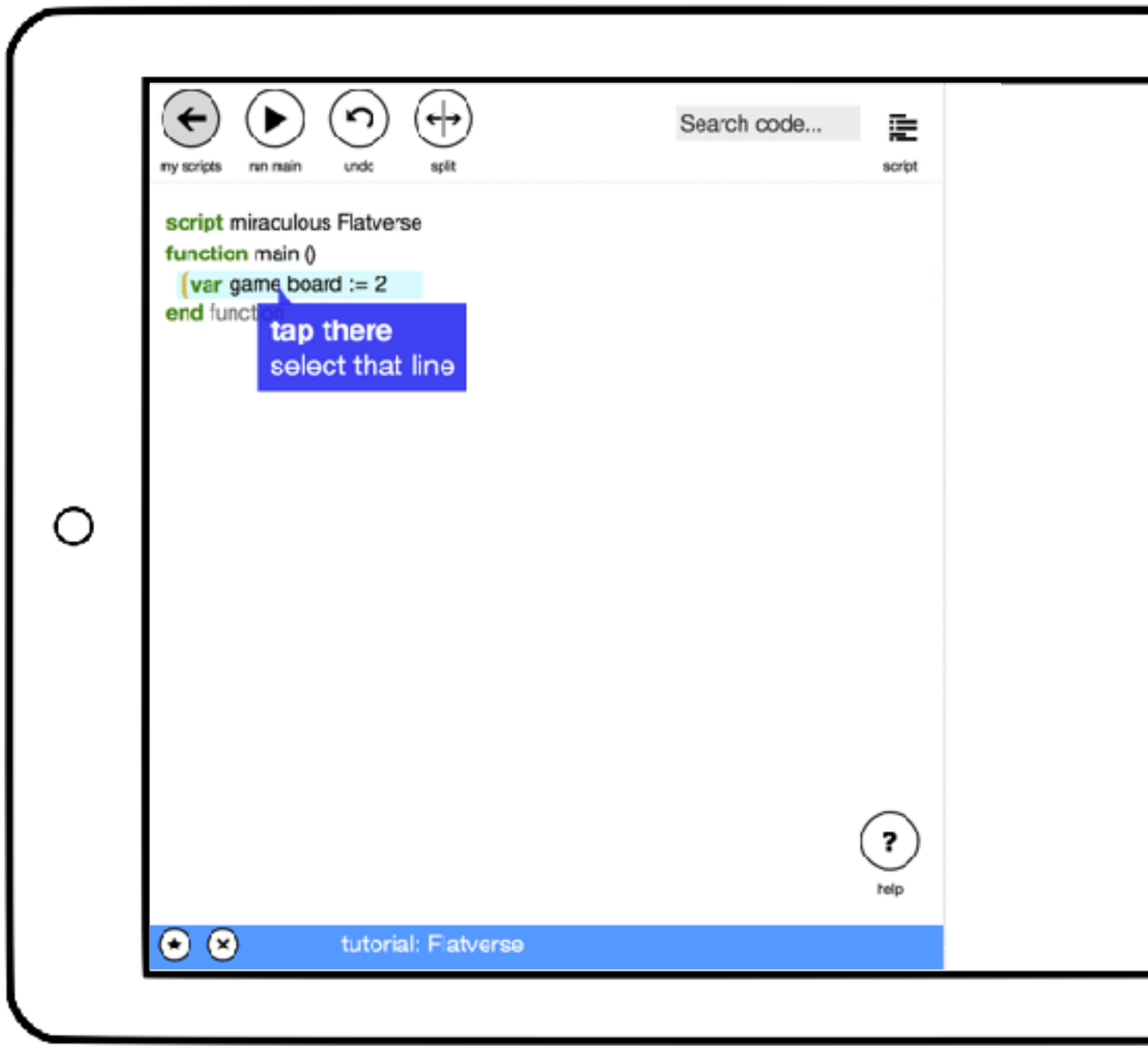
See also: Voelter, Sigmund, Berger, Kolb: **Towards User-Friendly Projectional Editors**, 2014

Microsoft TouchDevelop

Microsoft TouchDevelop, https://www.touchdevelop.com

Microsoft TouchDevelop

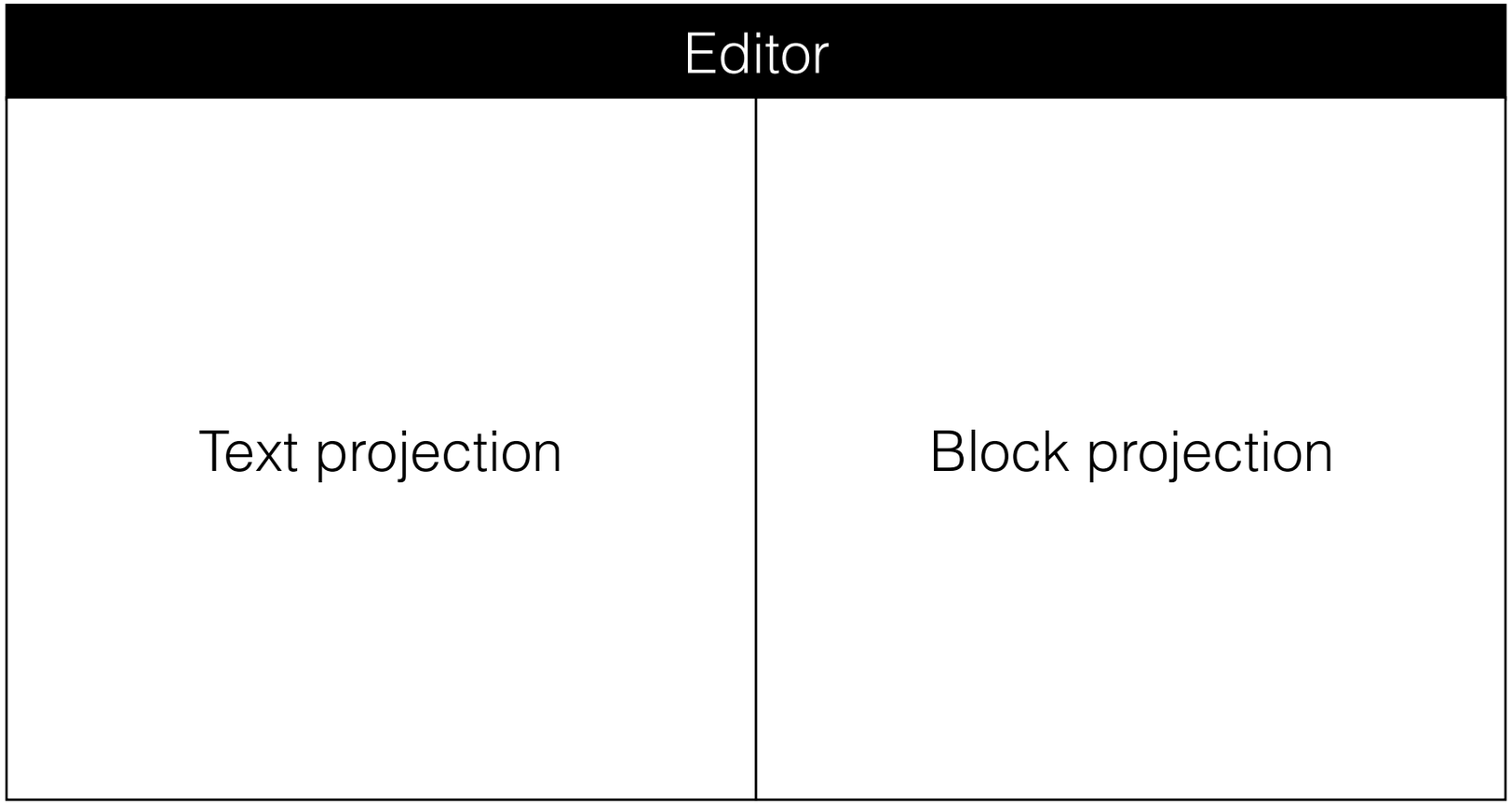
- Taylor-made language
  vs. JavaScript
- Limited use case
  vs. general editing
- Designed for mobile
  vs. Desktop



Microsoft TouchDevelop, https://www.touchdevelop

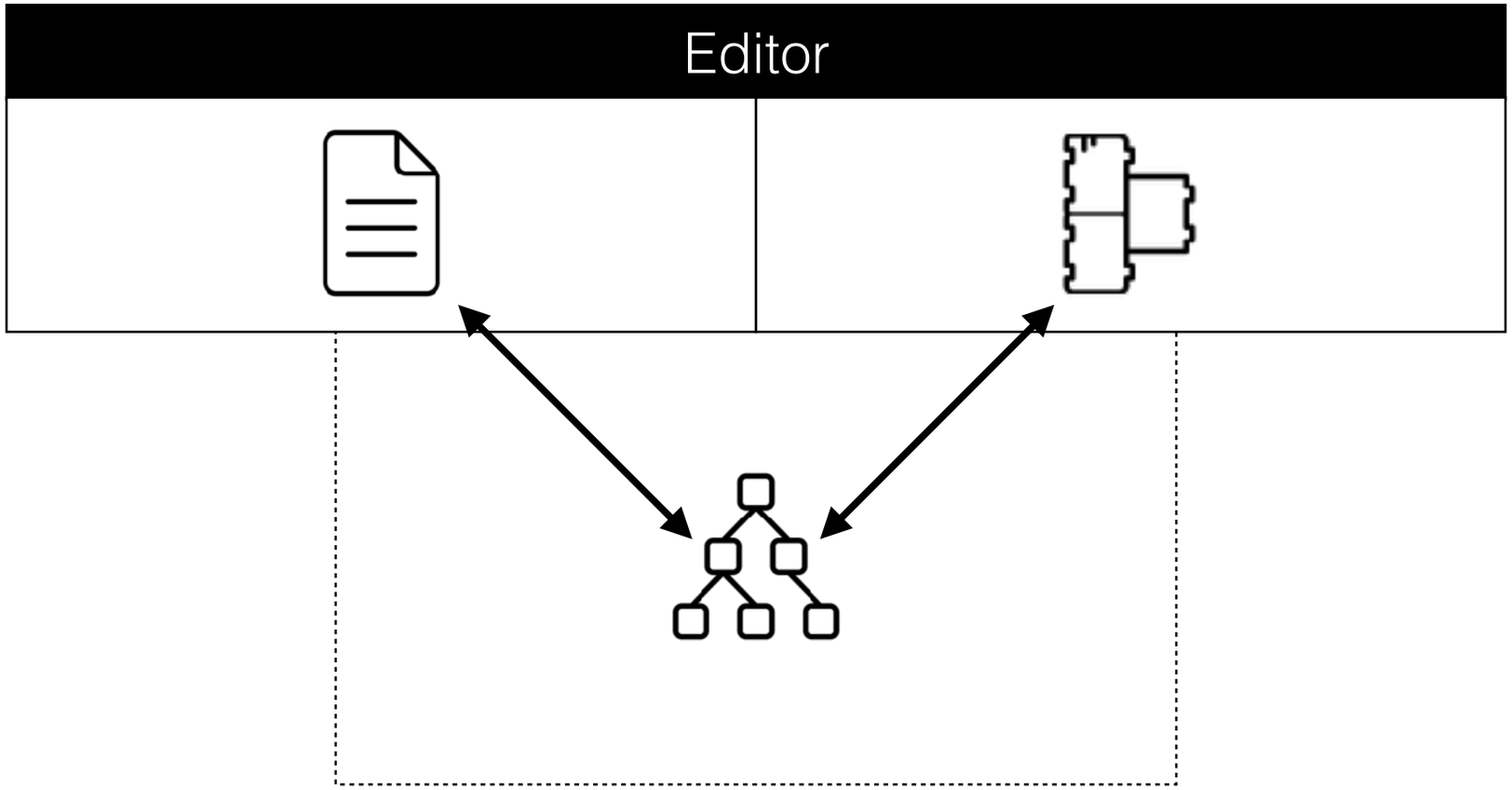
Lively Projectional Editor

- Two projections: Text- and Block-based
  - Fast entry in text projection
  - Advanced insight and editing in block projection

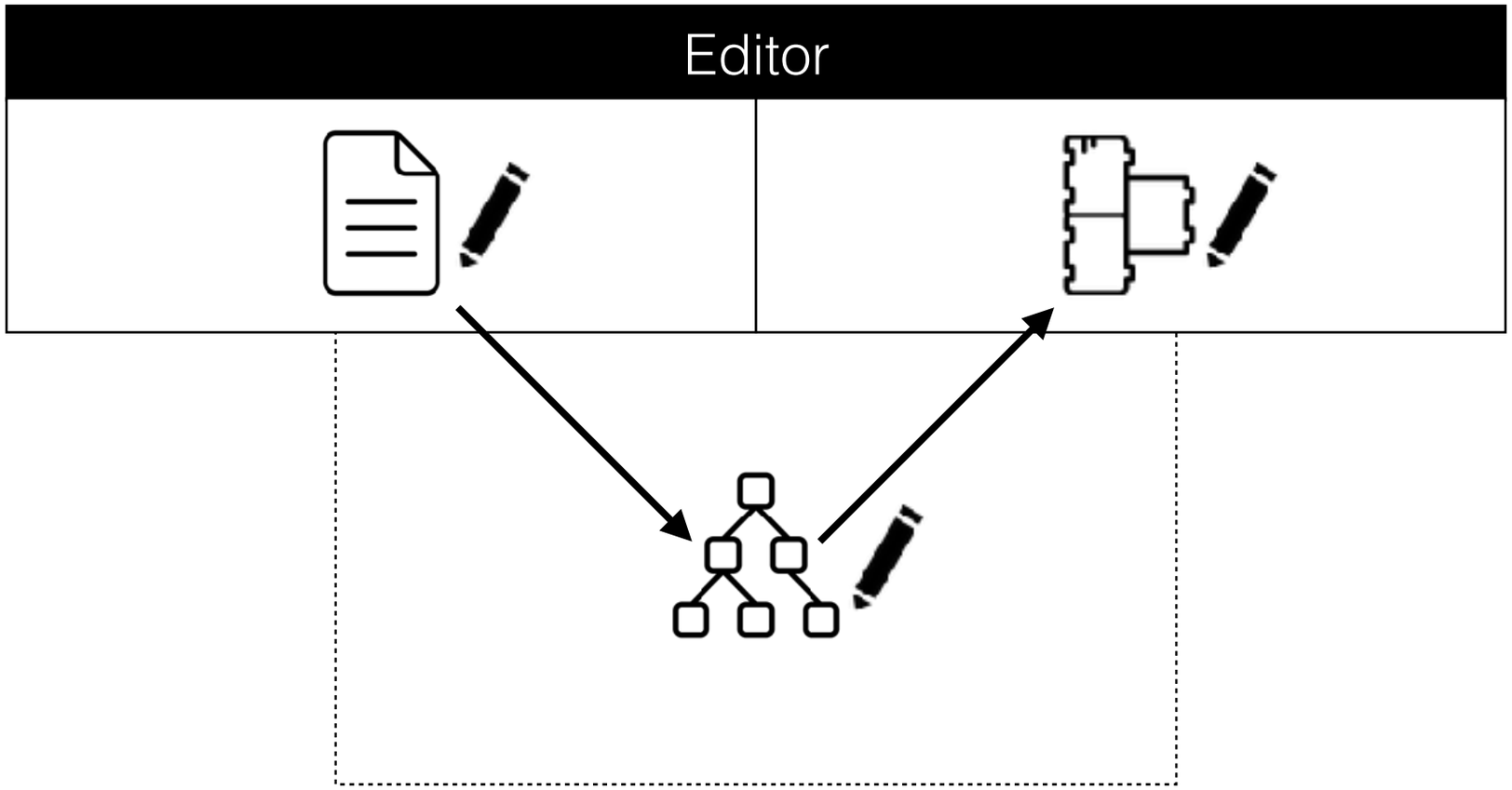


Implementation

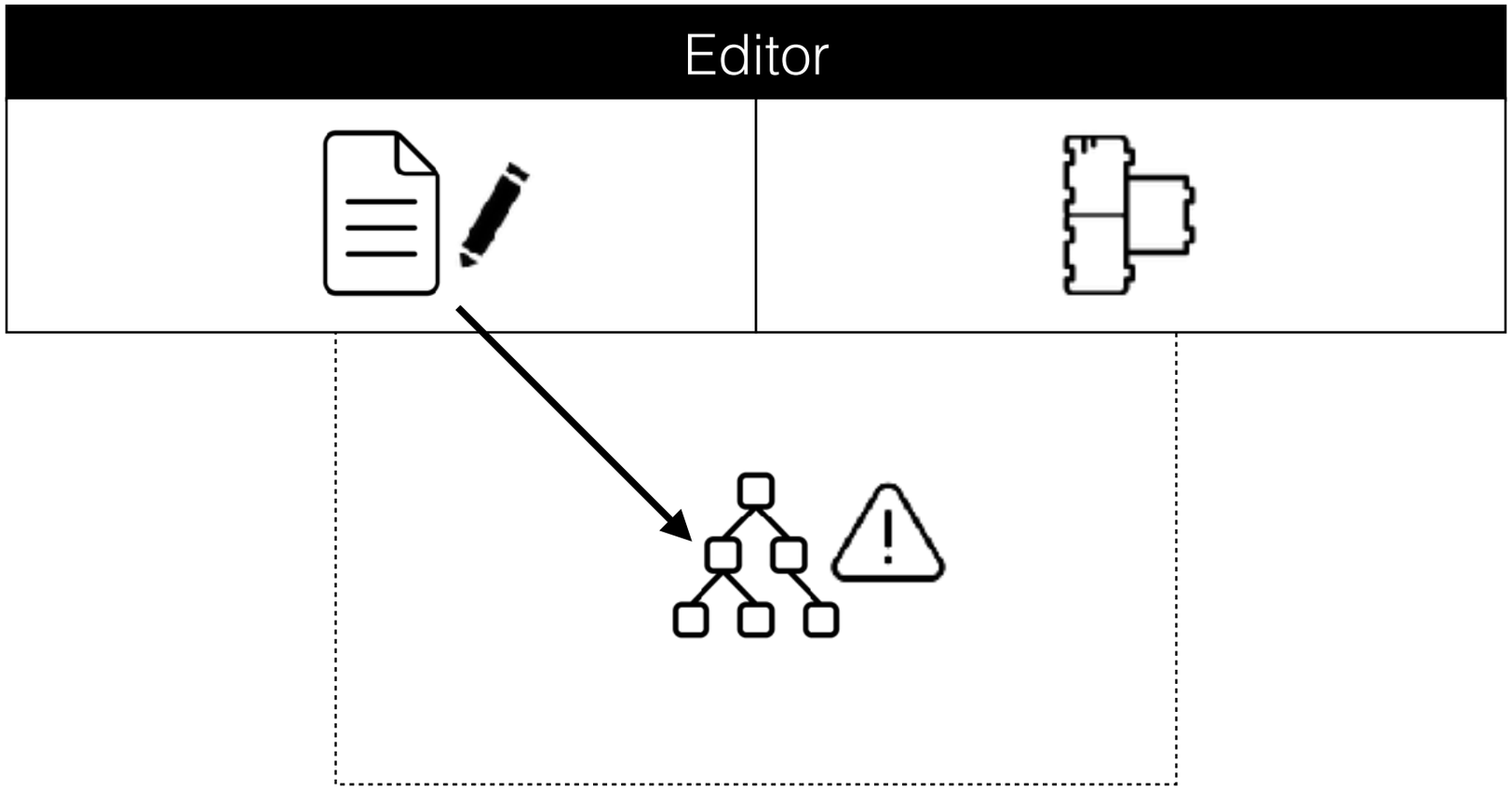
- AST as “document”
  - Text- and Block-view on the AST
  - Editor keeps views in sync



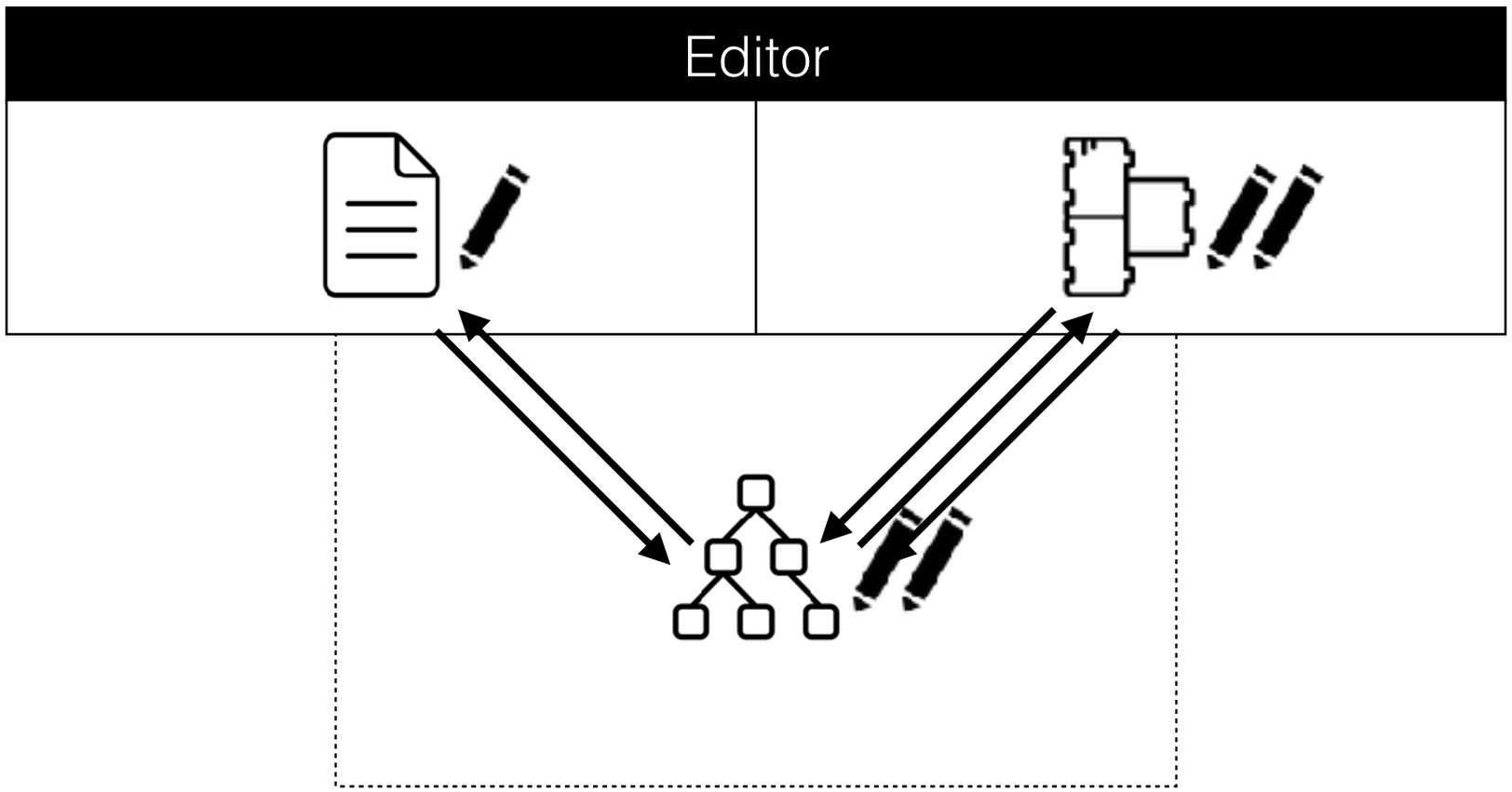
Sync - Success



AST corruption



Projection update cycle



Demo

Lively Projectional Editor

- Edit JavaScript and gain additional insight
  - Structure of the AST
  - Syntactic possibilities
- Babel
  - `babylon`[1] to parse code
  - `babel-types`[2] to get AST information
- Google Blockly[3] as block editor

[1] https://github.com/babel/babylon
[2] https://github.com/babel/babel/tree/master/packages/babel-types
[3] https://developers.google.com/blockly/

Future Work

- Advanced, context-aware editing
- Different projections of the AST
- Projectional error highlighting

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Thank you for your attention