Migrations

the tale of upgrading libraries in a massive monorepo

http://www.strawpoll.me/45862988

Albert Groothedde

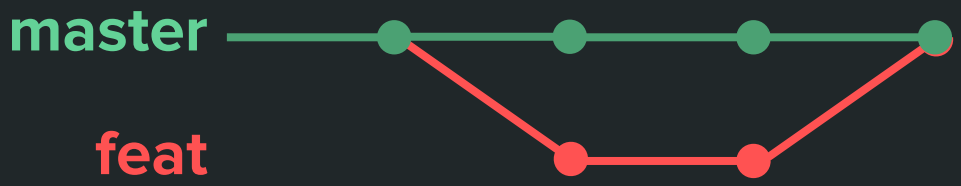
💰 Frontmen (part of iO)
🏠 Gorinchem
📱 06 5439 2021
📪 albertgroothedde [] gmail.com
💻 github.com/alber70g

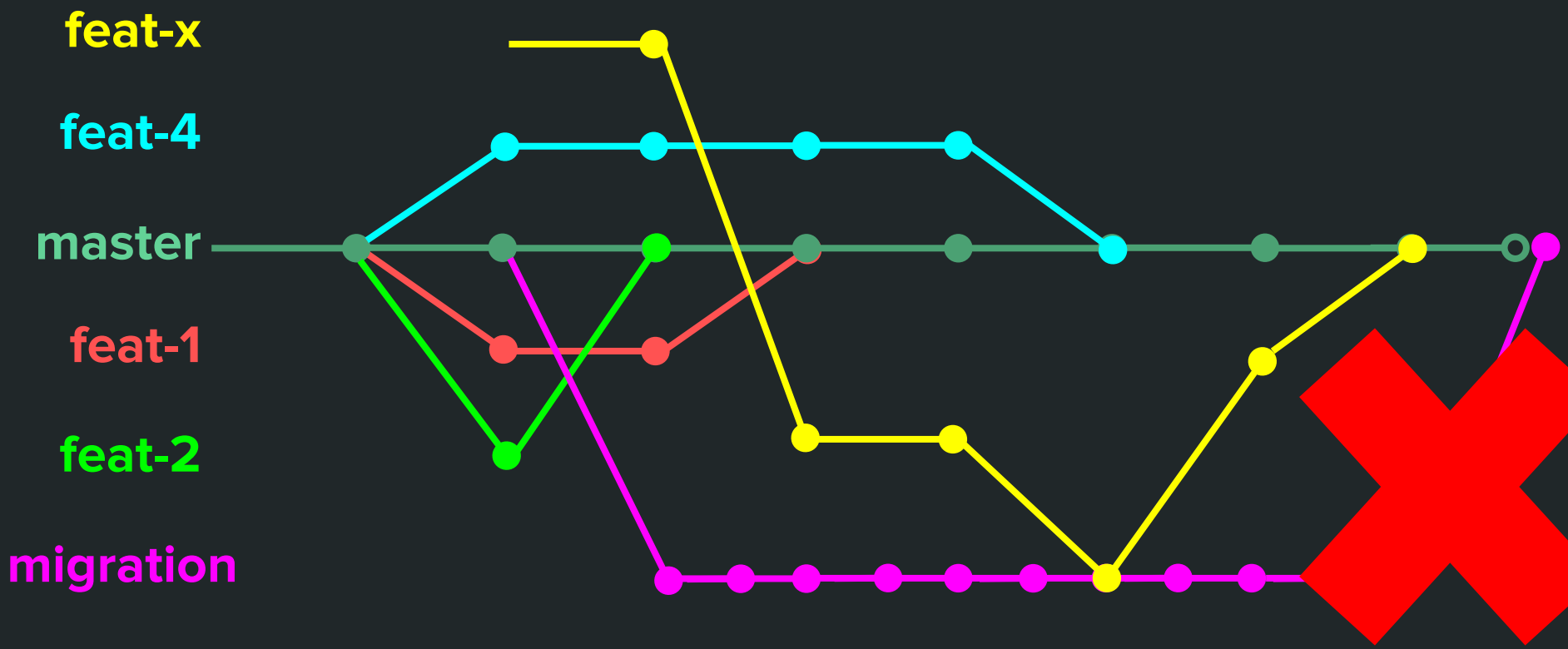
http://www.strawpoll.me/45862988

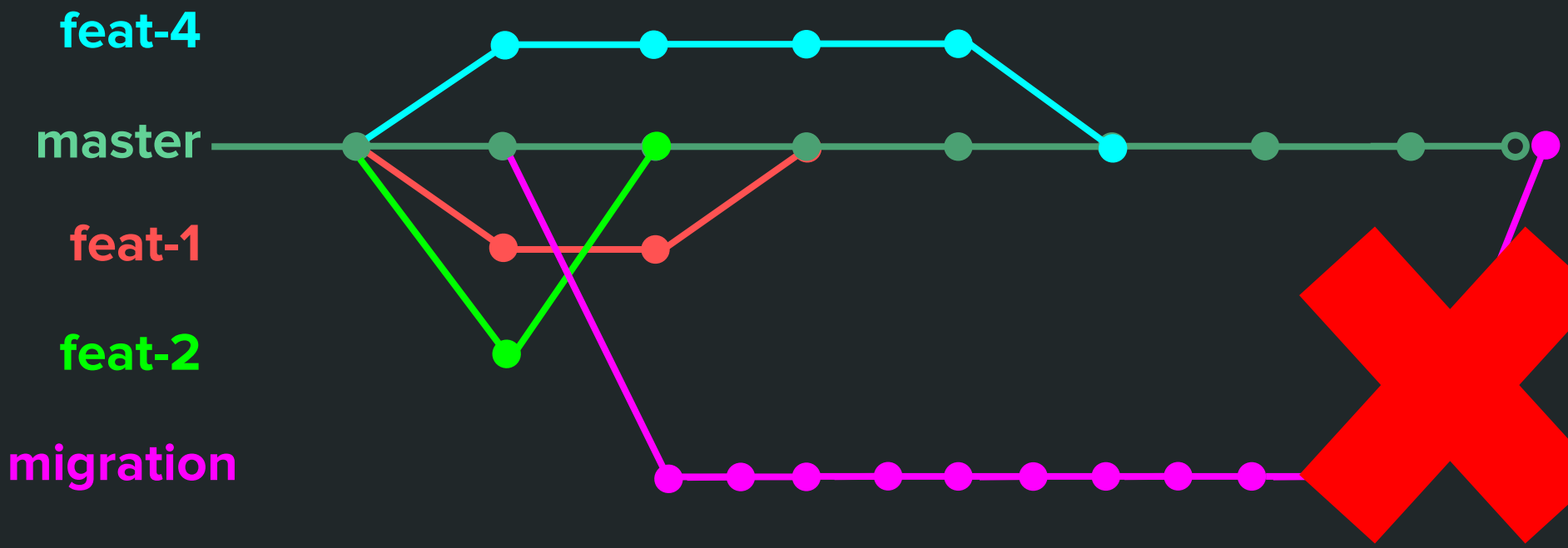
In the beginning...

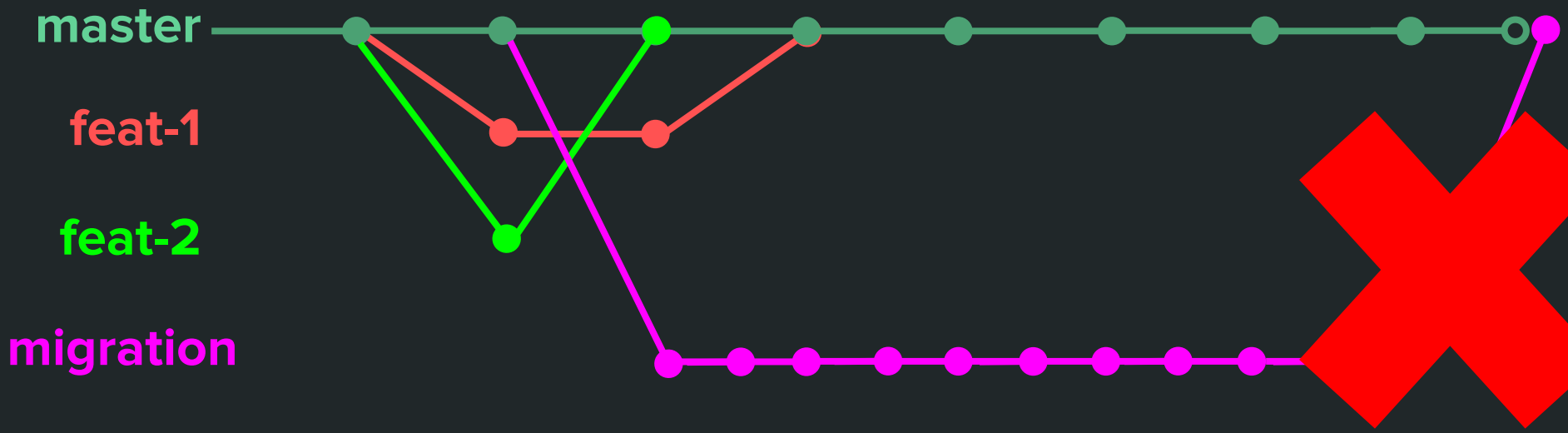
...we had ~~version 1~~ version 2

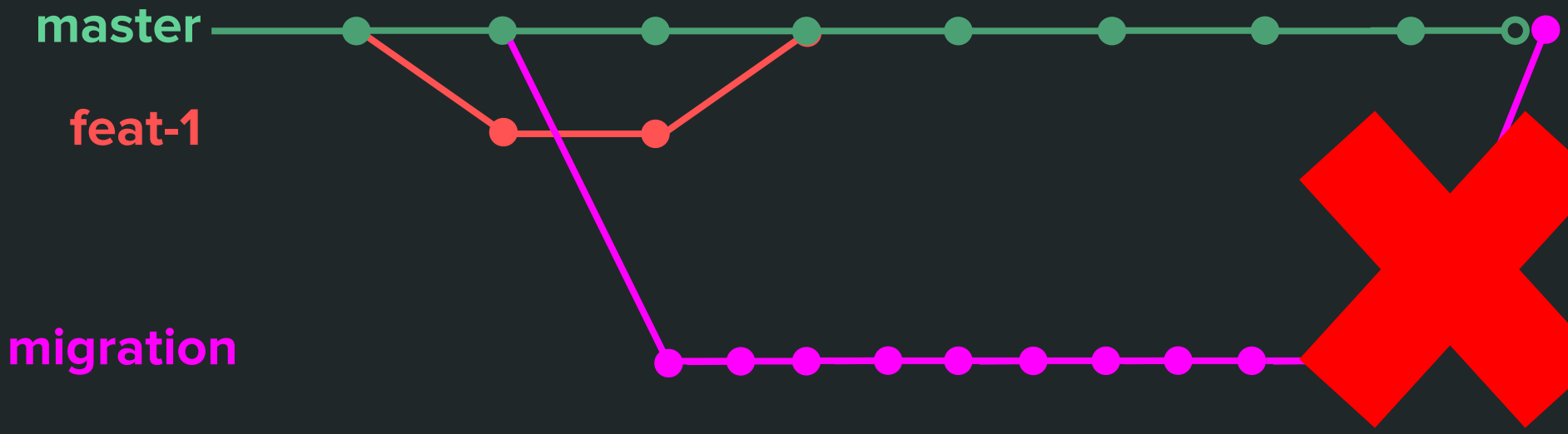
- jest upgrade
- ~~/angular/~~libs and ~~/angular/~~apps to root
- angular 8
- nx-to-root
- ionic 5
- nx standardisation
- angular 9
- and more...

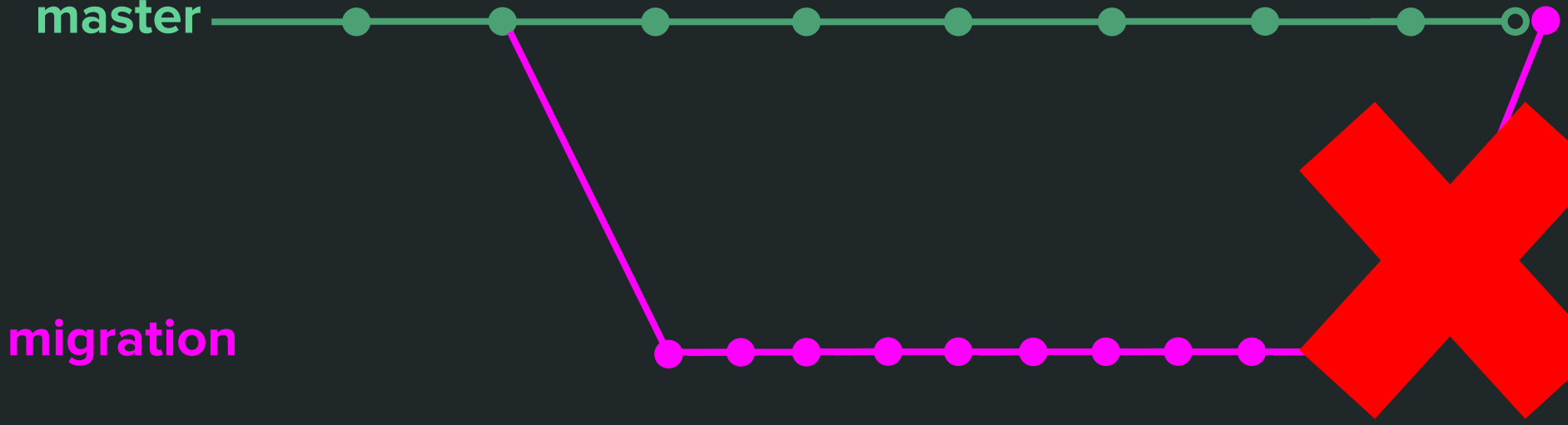

master
feat

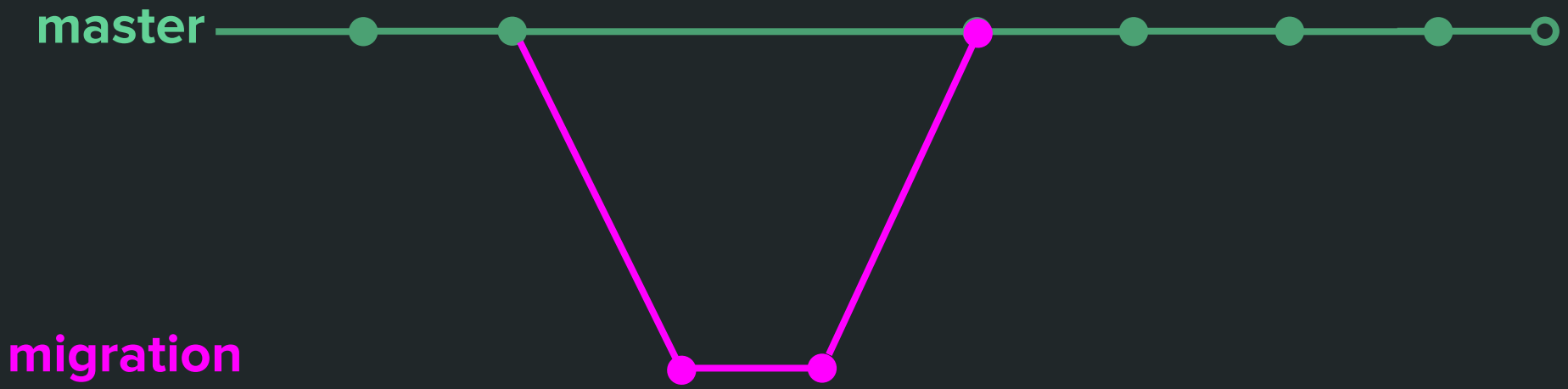

feat-x
feat-4
master
feat-1
feat-2
migration

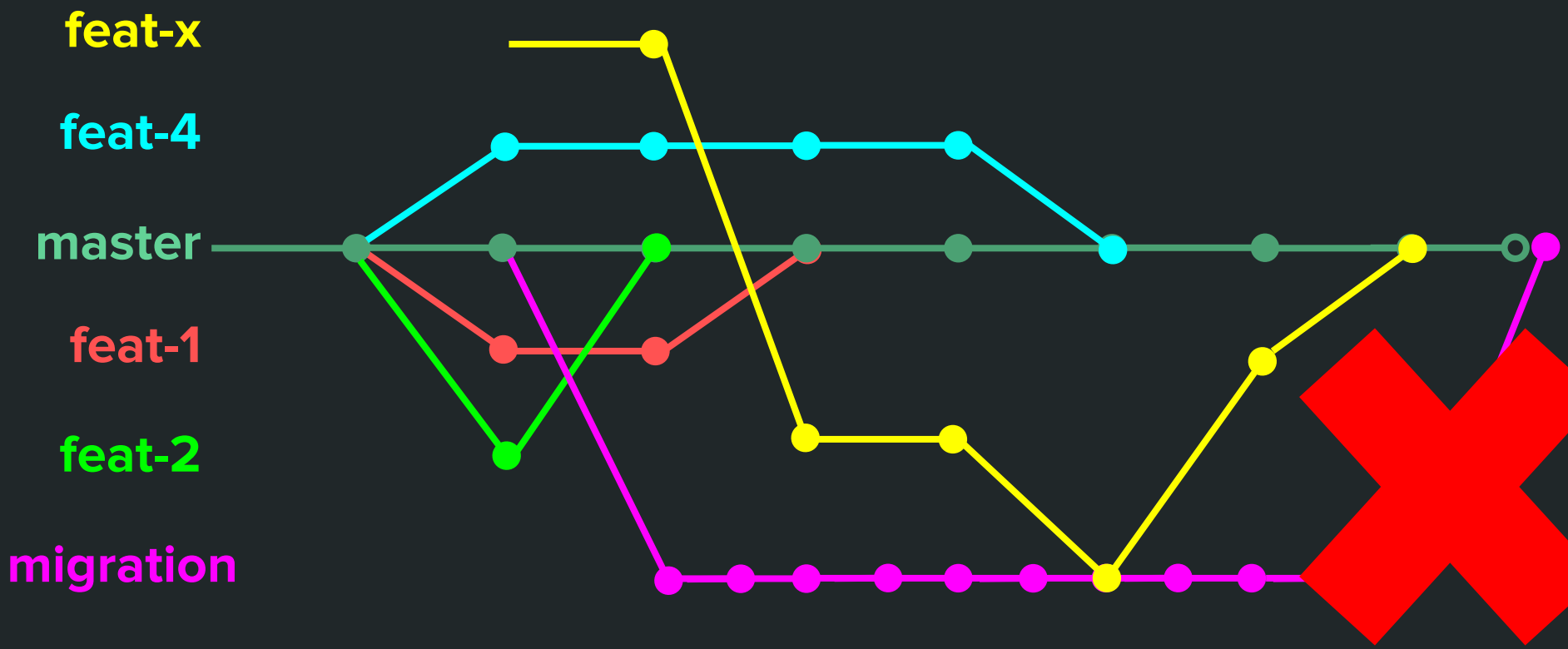

feat-4
master
feat-1
feat-2
migration

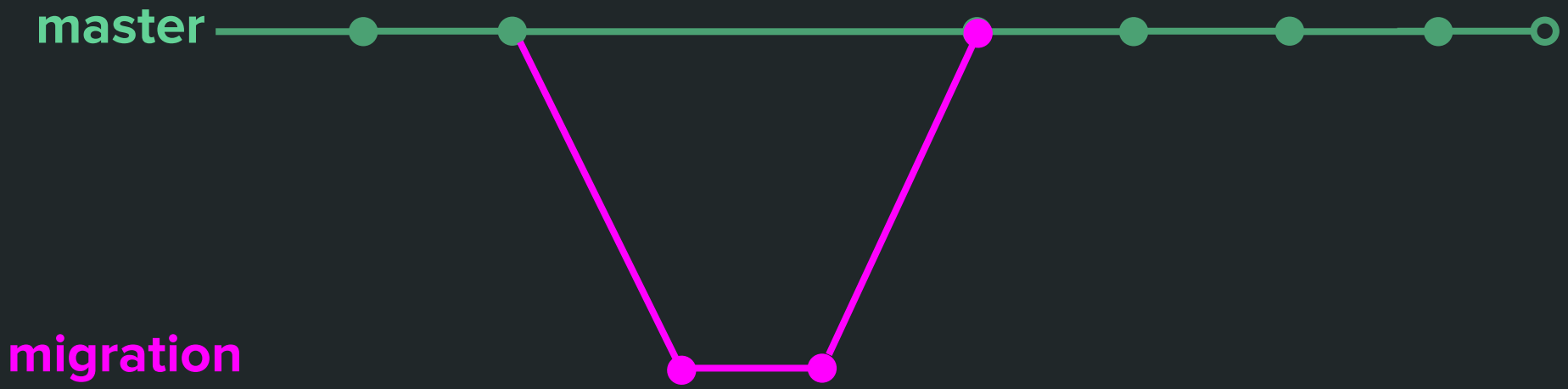

master
feat-1
feat-2
migration


master
feat-1
migration


master
migration

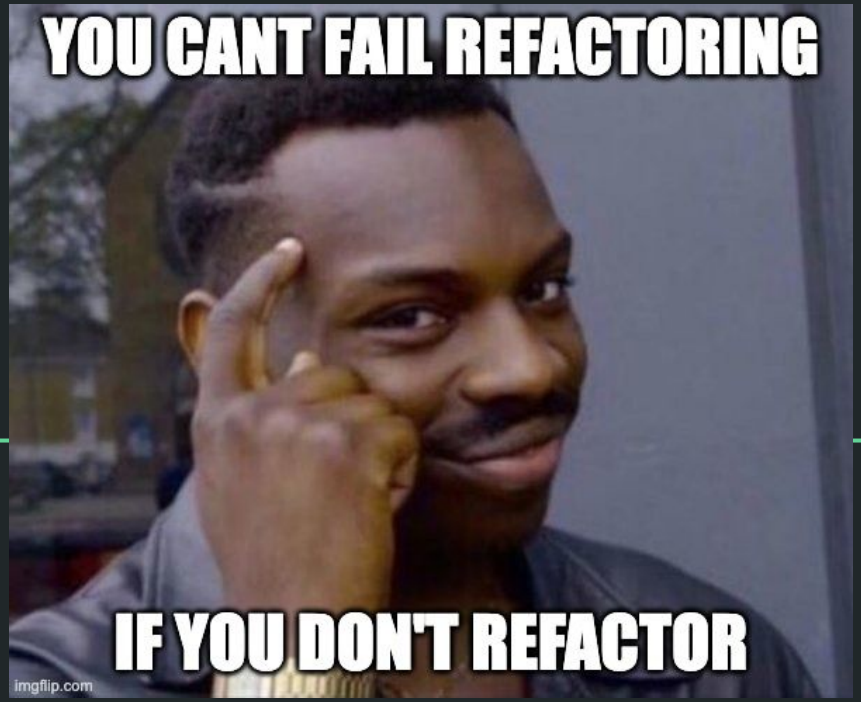

master
migration


feat-x
feat-4
master
feat-1
feat-2
migration


master
migration

How to perform large-scale refactoring in a busy code base

or you don’t...

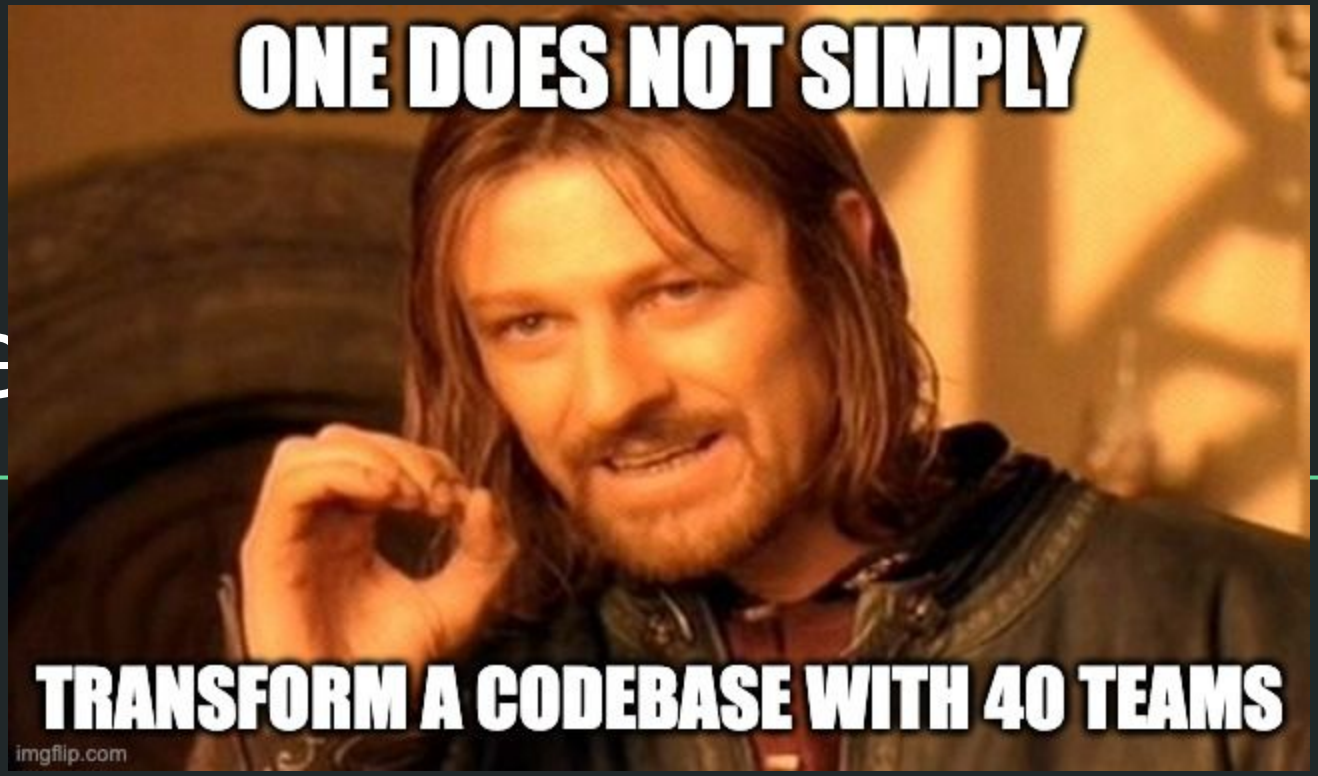


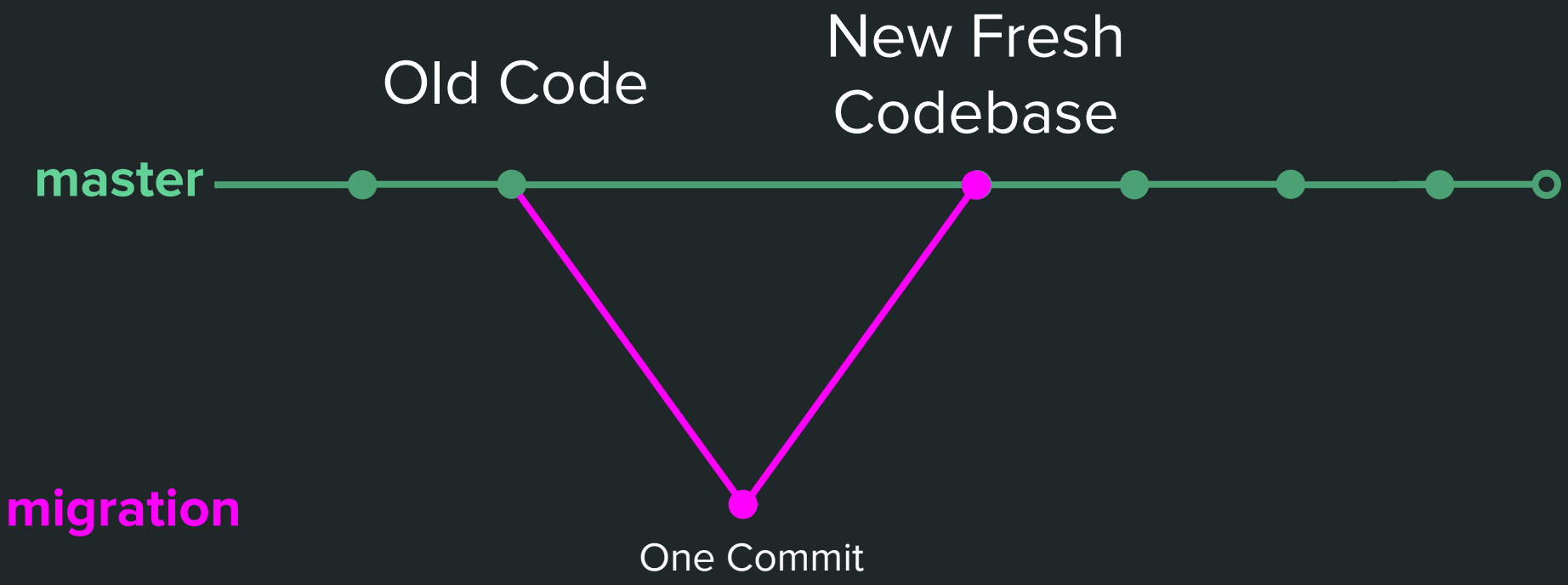
Everyb g

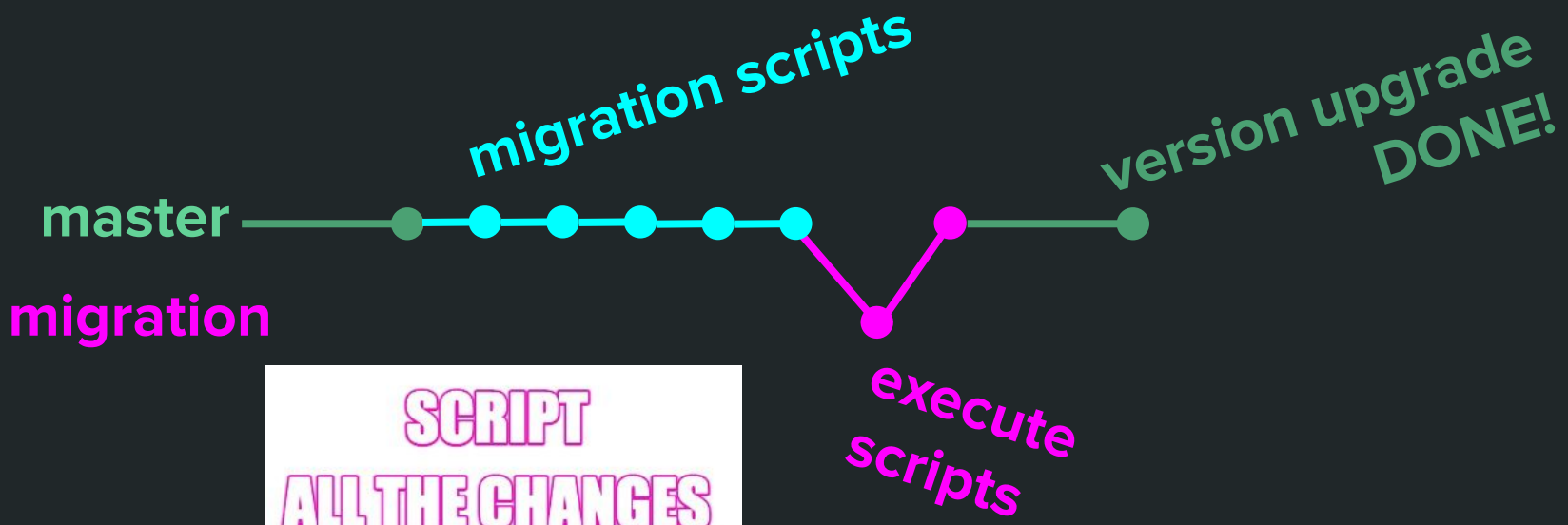
specialized t

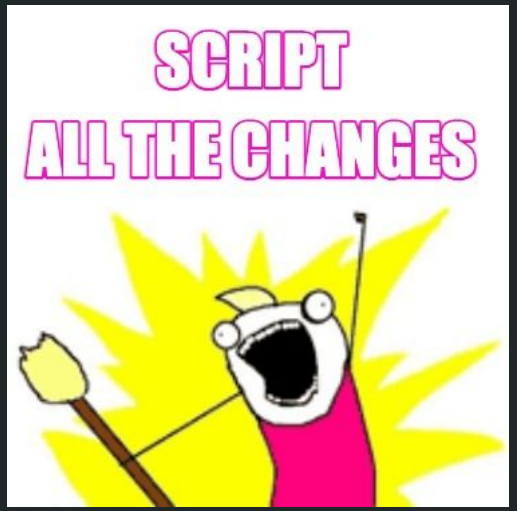
Eve

40 tea


ONE DOES NOT SIMPLY
TRANSFORM A CODEBASE WITH 40 TEAMS
imgflip.com


Old Code
New Fresh
Codebase
master
migration
One Commit


migration scripts
master
version upgrade
DONE!
migration
execute
scripts


SCRIPT
ALL THE CHANGES

Migration scripts

- execute an upgrade of angular 9 > angular 10
  - `$ npm install angular@10`
- run react-codemods
  - `$ npx react-codemod pure-component <path>`
- move files around and update those who depend on it
- change how home-grown libs are being called
- ...

Code

```
function x() {}

const y = () => {}
```

AST

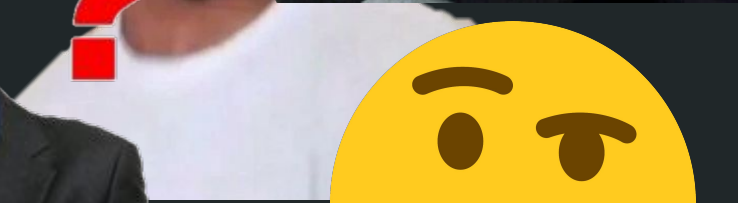
Abstract Syntax Tree

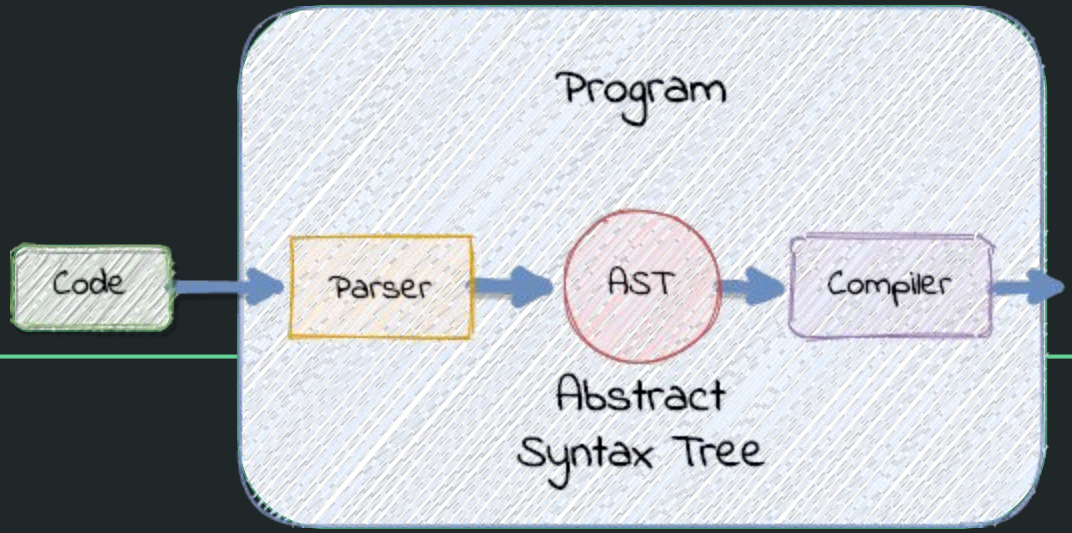
AST

Abstract Syntax Tree

AT

Abstrac



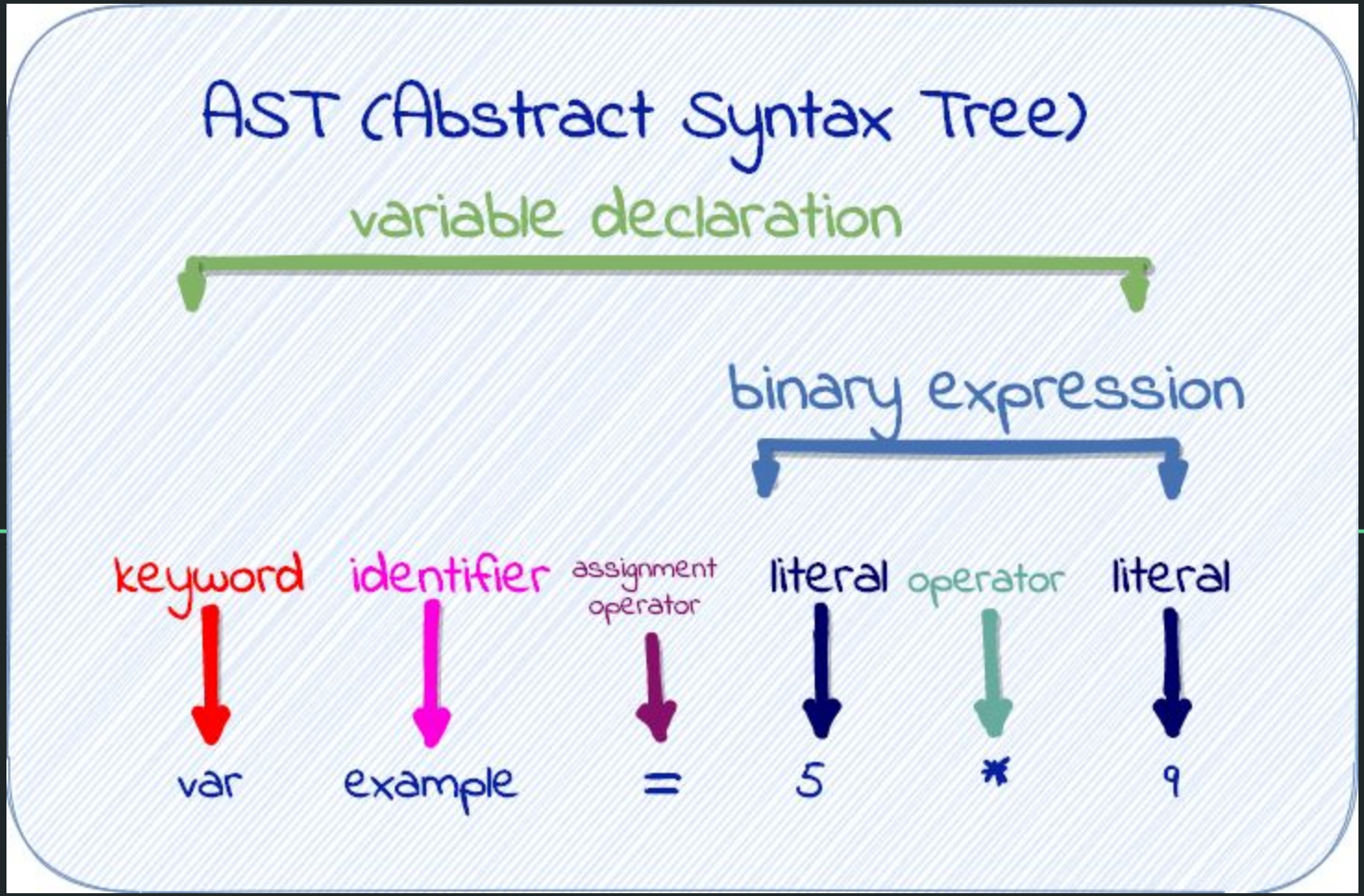


https://medium.com/bluecore-engineering/asts-codemods-effective-and-automated-codebase-refactoring-3ef93ff975a1

Only structural and **content-related** details of the source code **are preserved**

semi-colons, spaces, newlines, comments, etc

depending on parser

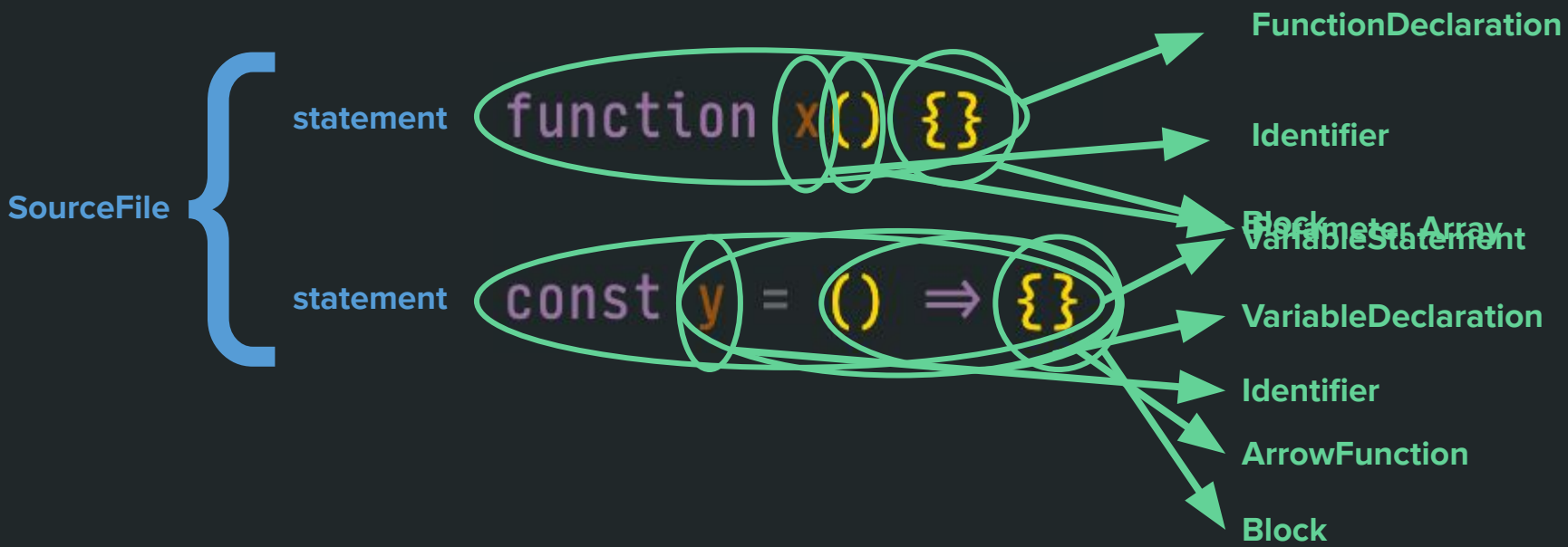

AST (Abstract Syntax Tree)
variable declaration
binary expression
keyword
identifier
assignment operator
literal
operator
literal
var
example
=
5
*
9

AST Example

```
function x() {}

const y = () => {}
```


```
{
 "type": "Program",
 "body": [
   {
     "type": "FunctionDeclaration",
     "id": { "type": "Identifier", "name": "x", "
     "generator": false, "expression": false, "as
     "body": {
       "type": "BlockStatement",
       "body": [],
       "range": [14, 16]
     },
     "range": [1, 16]
   },
   {
     "type": "VariableDeclaration",
     "declarations": [
       { "type": "VariableDeclarator", "id": { "t
         "init": { "type": "ArrowFunctionExpressi
           "body": { "type": "BlockStatement", "b
           "async": false, "expression": false, "
         },
         "range": [24, 36]
       }
     ],
     "kind": "const",
     "range": [18, 36]
```


SourceFile
statement
function x() {}
statement
const y = () => {}
FunctionDeclaration
Identifier
ParameterArray
VariableStatement
VariableDeclaration
Identifier
ArrowFunction
Block

TypeScript AST Viewer

```
1
2  function x() {}
3
4
5  const y = () => {}
```

Pos 0, Ln 1, Col 1

```
1  factory.createVariableDeclaration(
2    factory.createIdentifier("y"),
3    undefined,
4    undefined,
5    factory.createArrowFunction(
6      undefined,
7      undefined,
8      [],
9      undefined,
10     factory.createToken(ts.SyntaxKind.EqualsGreaterThanT
11     factory.createBlock(
12       [],
13       false
14     )
15   )
16 )
17
```

- SourceFile
  - FunctionDeclaration
    - Identifier
    - Block
  - VariableStatement
    - VariableDeclarationList
      - VariableDeclaration
        - Identifier
        - ArrowFunction
          - EqualsGreaterThanToken
          - Block
  - EndOfFileToken

https://ts-ast-viewer.com/

Node

- VariableDeclaration
  - pos: 24
  - end: 37
  - flags: 0
  - kind: 253 (SyntaxKind.VariableDeclaration)
  - decorators: undefined
  - modifiers: undefined
  - name: ▸ Identifier
  - initializer: ▸ ArrowFunction
  - type: undefined
  - exclamationToken: undefined
  - getChildCount(): 3
  - getFullStart(): 24
  - getStart(): 25
  - getStart(sourceFile, true): 25
  - getFullWidth(): 13
  - getWidth(): 12
  - getLeadingTriviaWidth(): 1
  - getFullText(): y = () => {}
  - getText(): y = () => {}
  - ts.getLeadingCommentRanges(fileFullText, 24): undefined
  - ts.getTrailingCommentRanges(fileFullText, 37): undefined

Type

- () => void
  - flags: 524288
  - objectFlags: 16

Symbol

- y
  - flags: 2
  - escapedName: "y"
  - declarations: [
    - ▸ VariableDeclaration (y)
  -]
  - exports: {}
  - valueDeclaration: ▸ VariableDeclaration (y)

Signature

[None]

FlowNode

[None]

ts-ast-viewer.com

- can select version of TypeScript
- select script kind: `JS(X), TS(X), JSON`
- factory functions

astexplorer.net

- lots of languages
  - JS, Go, GraphQL, HTML, Java, OCaml, PHP, Lua, Rust, SQL, etc, etc...
- lot more parser options
- run transform functions
- *keyboard modes: vim, emacs, sublime*

Code Walkthrough

Recap

- for big code refactor work, use scripts
- anything that can be done before the update/upgrade/refactor, do it on master

- for changes that are only syntax related, use @angular-devkit/schematics

- for changes that need to update references, use ts-morph

Tools

Use these as boilerplate/guidance

- https://alber70g/refactor-test-schematics: example on angular schematics
- https://github.com/alber70g/presentation-refactoring-ast: project from the code-walkthrough

NPM tools/packages

- ts-morph: convert code using ts-compiler
- zx: bunch of tools to use with refactor (execute shell scripts, etc)
- @angular-devkit/schematics: modify tree with dry run etc
- comment-json: parse and write json *including* comments
- glob-tester-cli: use to test globs on you cli