Fast and Precise Type Checking for JavaScript

Avik Chaudhuri

Type Inference

Flow uses type inference to find bugs even without type annotations. It precisely tracks the types of variables as they flow through your program.

Idiomatic JS

Flow is designed for JavaScript programmers. It understands common JavaScript idioms and very dynamic code.

Realtime Feedback

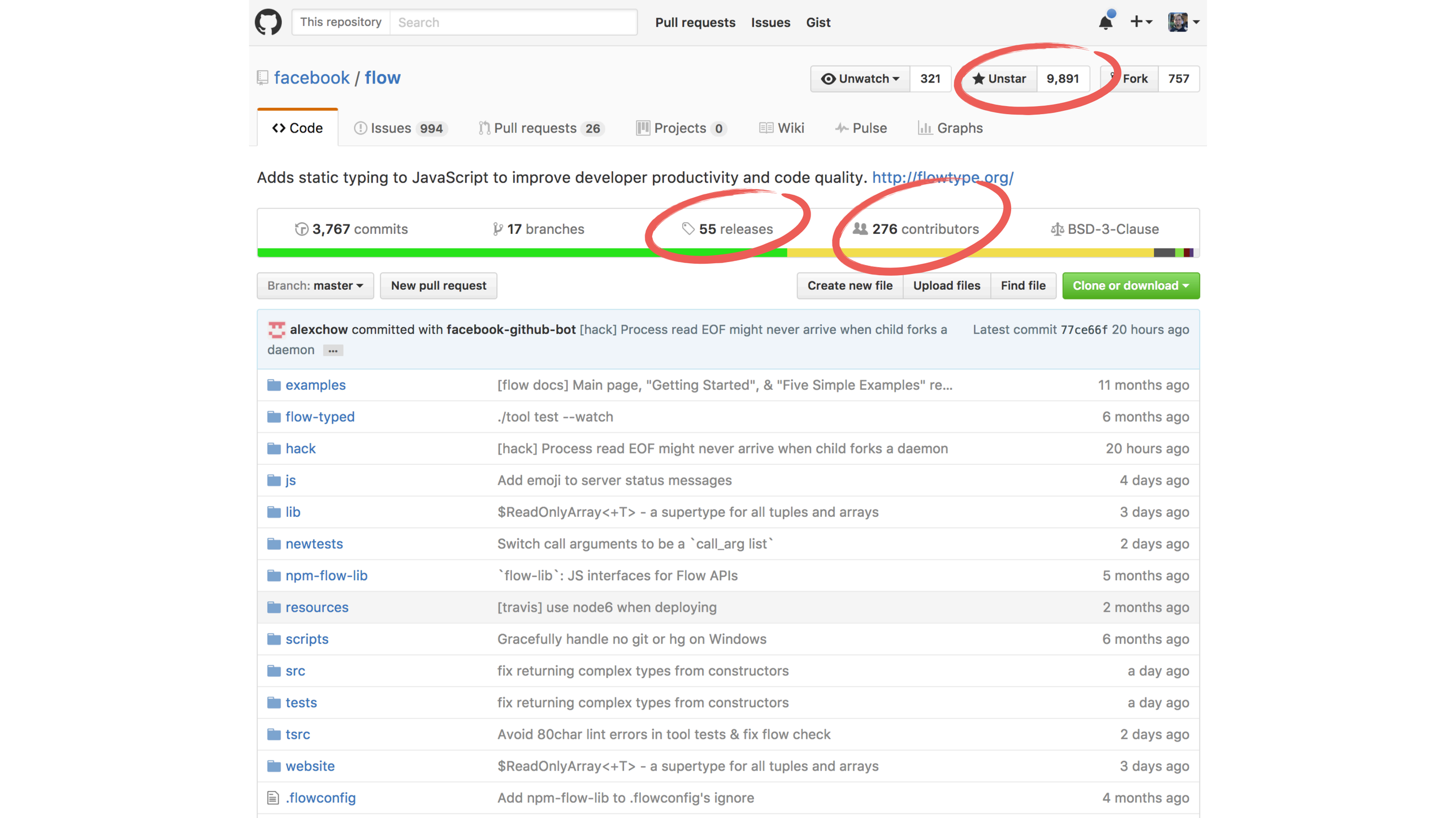
Flow incrementally rechecks your changes as you work, preserving the fast feedback cycle of developing plain JavaScript.

Flow can catch common bugs in JavaScript programs before they run, including:

- silent type conversions,
- `null` dereferences,
- and the dreaded `undefined is not a function`.

```
1  // @flow
2  function foo(x) {
3    return x * 10;
4  }
5  foo('Hello, world!');
```

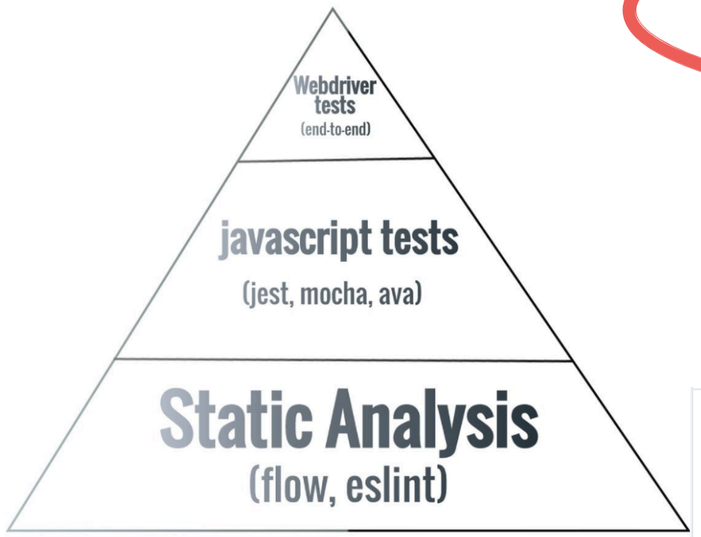
show Flow output


This repository
Search
Pull requests
Issues
Gist
facebook / flow
Unwatch 321
Unstar 9,891
Fork 757
Code
Issues 994
Pull requests 26
Projects 0
Wiki
Pulse
Graphs
Adds static typing to JavaScript to improve developer productivity and code quality. http://flowtype.org/
3,767 commits
17 branches
55 releases
276 contributors
BSD-3-Clause
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
alexchow committed with facebook-github-bot [hack] Process read EOF might never arrive when child forks a daemon ...
Latest commit 77ce66f 20 hours ago
examples [flow docs] Main page, "Getting Started", & "Five Simple Examples" re... 11 months ago
flow-typed ./tool test --watch 6 months ago
hack [hack] Process read EOF might never arrive when child forks a daemon 20 hours ago
js Add emoji to server status messages 4 days ago
lib $ReadOnlyArray<+T> - a supertype for all tuples and arrays 3 days ago
newtests Switch call arguments to be a `call_arg list` 2 days ago
npm-flow-lib `flow-lib`: JS interfaces for Flow APIs 5 months ago
resources [travis] use node6 when deploying 2 months ago
scripts Gracefully handle no git or hg on Windows 6 months ago
src fix returning complex types from constructors a day ago
tests fix returning complex types from constructors a day ago
tsrc Avoid 80char lint errors in tool tests & fix flow check 2 days ago
website $ReadOnlyArray<+T> - a supertype for all tuples and arrays 3 days ago
.flowconfig Add npm-flow-lib to .flowconfig's ignore 4 months ago

Flow Retweeted

Dmitrii Abramov @abramov_dmitrii · 5 Dec 2016

testing pyramid in 2016



11 198 313 •••

Shane Cavaliere @shanecav Follow

The better I get with @flowtype, the less of it I end up having to write. I'm loving how good it is at type inference.

RETWEETS 8 19

Flow Retweeted

codemix @codemixers · Jan 4

Just released the first version of flow-runtime, @flowtype compatible run time type system for JS codemix.github.io/flow-runtime/

6 57 100 •••

Brendan @irvinebroque Follow

Exact object types in @flowtype are such an amazing refactoring tool for big apps. Worth any pain of upgrading in order to start using them.

Flow Retweeted

Alden Daniels @alden_daniels · 11 Nov 2016

Been using @flowtype heavily in a recent project. It's a game-changer. After a while you forget that JS isn't typed natively. So good.

1 6 37 •••

Flow Retweeted

Dan Abramov @dan_abramov · 11 Nov 2016

I like how @flowtype can save me from null reference errors ever before I add any annotations.

```
let index = -1;
const stack = [];
const emptyObject = require('emptyObject');

function getMaskedContext(contextTypes : any) {
  if (index === -1) {
    return emptyObject;
  }
  const currentContext = stack[index];
  const maskedContext = {};
  for (let key in contextTypes) {
    maskedContext[key] = currentContext[key];
  }
  return maskedContext;
};

exports.popContext = function() {
  stack[index] = null;
  index--;
};
```

Copy

access of computed property/element

Computed property/element cannot be accessed on possibly null: src/renderers/shared/fiber/ReactFiberContext.js:26

null

GIF

9 50 166 •••

Mark Dalgleish @markdalgleish Follow

New team member opened a PR adding @flowtype to our established React + Webpack + Babel app. Really surprised how straightforward it was.

RETWEETS 12 LIKES 75

9:06 PM - 29 Sep 2016

12 75 •••

Flow @flowtype Following

New blog post! Introducing flow-typed: Tested, versioned, community-managed libdefs & libdef tooling for Flow. flowtype.org/blog/2016/10/1 …

RETWEETS 73 LIKES 112

Flow Retweeted

Jose @jacortinas · 11 Nov 2016

I'm becoming a bigger and bigger fan of @flowtype the more I use it. Seriously useful tool.

```
type Props = {
  subreddit?: ?string;
  filter: 'hot' | 'new' | 'controversial';
}

export default class SubredditListingScreen extends Component {
  props: Props;

  static defaultProps: Props = {
    filter: 'blah'
  };

  render() {
    return (
      <View style={styles.container}>
        <Text>{this.props.filter}</Text>
      </View>
    );
  }
}
```

Flow Warning string This type is incompatible with string enum

Flow Trace string enum at line 17 col 11 in js/components/SubredditListingScreen.js

1 3 18 •••

Mihaly Csikszentmihalyi:

Flow, the secret to happiness

TED2004 · 18:55 · **Filmed** Feb 2004

32 subtitle languages

View interactive transcript

Share this idea

Facebook LinkedIn Twitter Link Email Embed

3,467,268 Total views

Mihaly Csikszentmihalyi asks, "What makes a life worth living?" Noting that money cannot make us happy, he looks to those who find pleasure and lasting satisfaction in activities that bring about a state of "flow."

Are you in flow?

(*If not, how can Flow help?*)


MOVE
FAST AND
BREAK
THINGS


MOVE
FAST WITH
STABLE
INFRA

At Facebook (and most companies)…

- **Outages are costly** (money, trust)
- **Challenge:** *Enable fast evolution without breaking stuff*
- As # engineers (and LOC) grow quickly…

  How do they **understand** how things fit together?

  How do they **fix** bugs / evolve things?

  How do they **build** new things without being afraid?

Hello, types!

Design goal 1:

Precision

Precise type checking

- *Follow along the code as much as possible*
- **Reduce noise**

  Give useful info about types, reaching definitions, …

  Don’t hide real errors in a sea of spurious errors
- **Soundness is important**

  Types should be trustworthy!

  (… except in a few justifiably difficult cases)

Design goal 2:

Speed

Fast type checking

- **Important not to introduce** any **latency** in workflow

  Preserve illusion of quick edit-refresh cycle

  Work in background, utilize idle time

- **Parallelize** as much as possible!

- **Do** as **little** work as possible!

  Type check incrementally, reusing most of the work

Let’s see some examples!

example1.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  length("");
8
```

4 0 example1.js 7:12 100% LF UTF-8 Babel ES6 JavaScript 5 updates

example2.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  length("");
8  length([1,2,3]);
9
```

4 0 example2.js 8:17 100% LF UTF-8 Babel ES6 JavaScript 5 updates

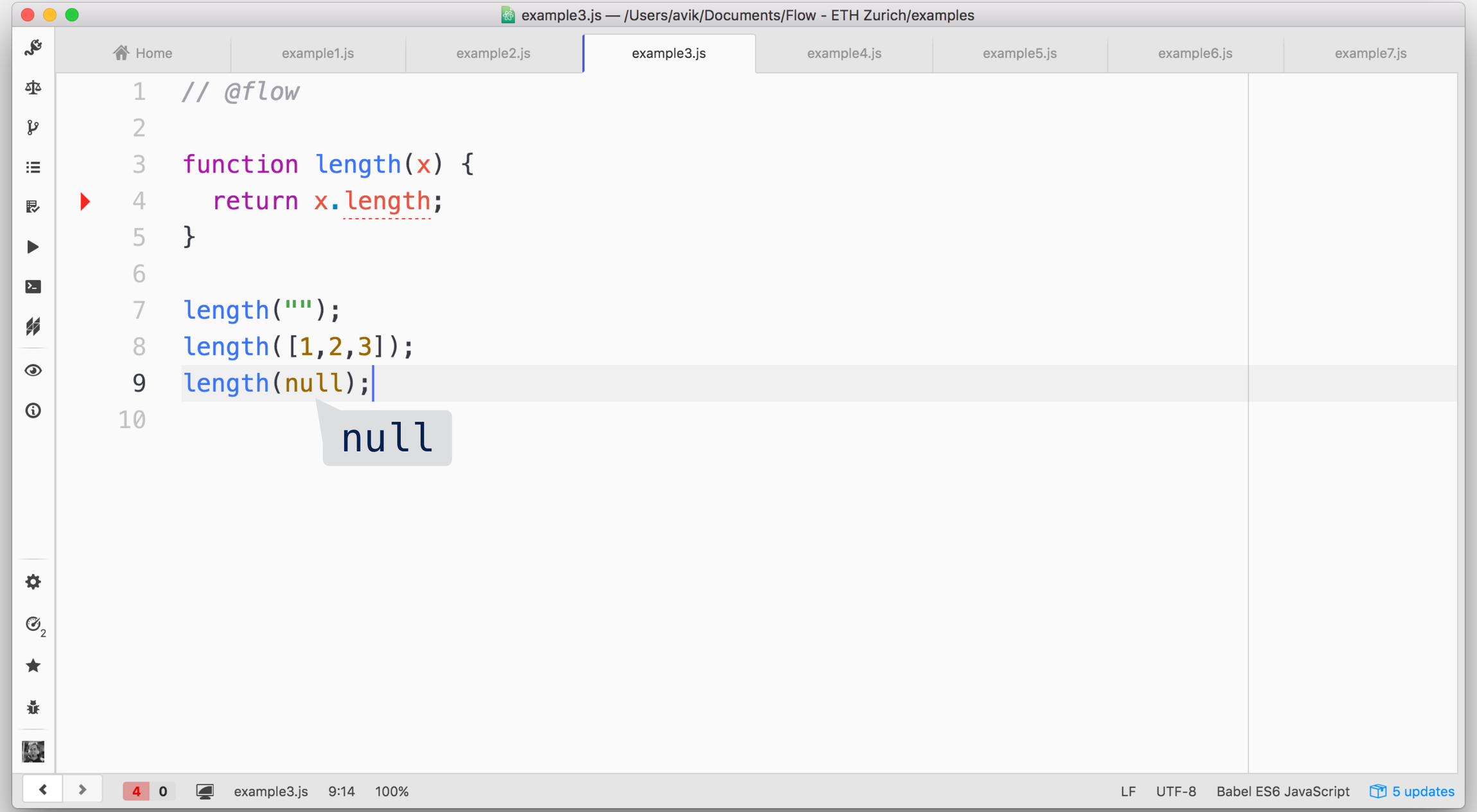
example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples

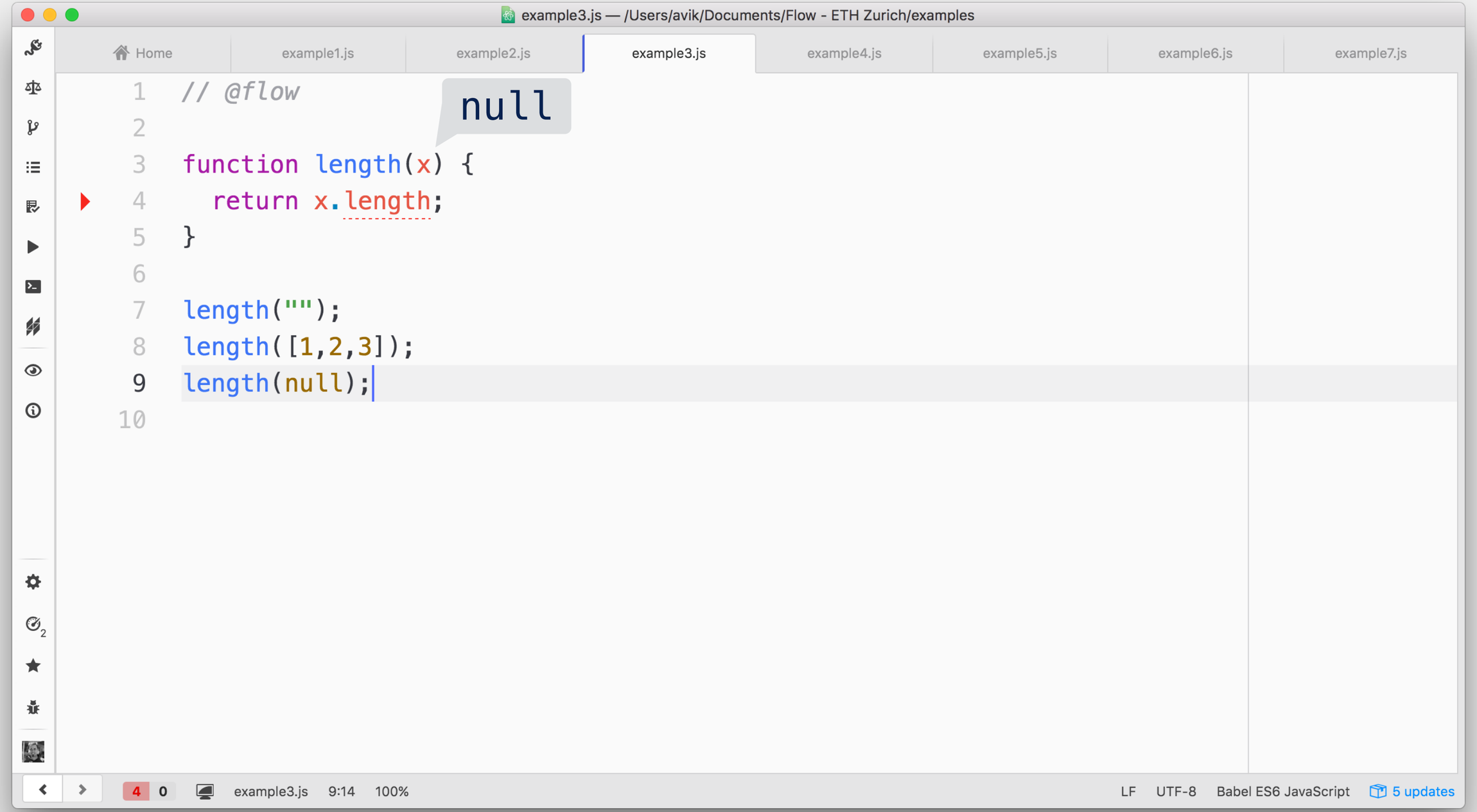
Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

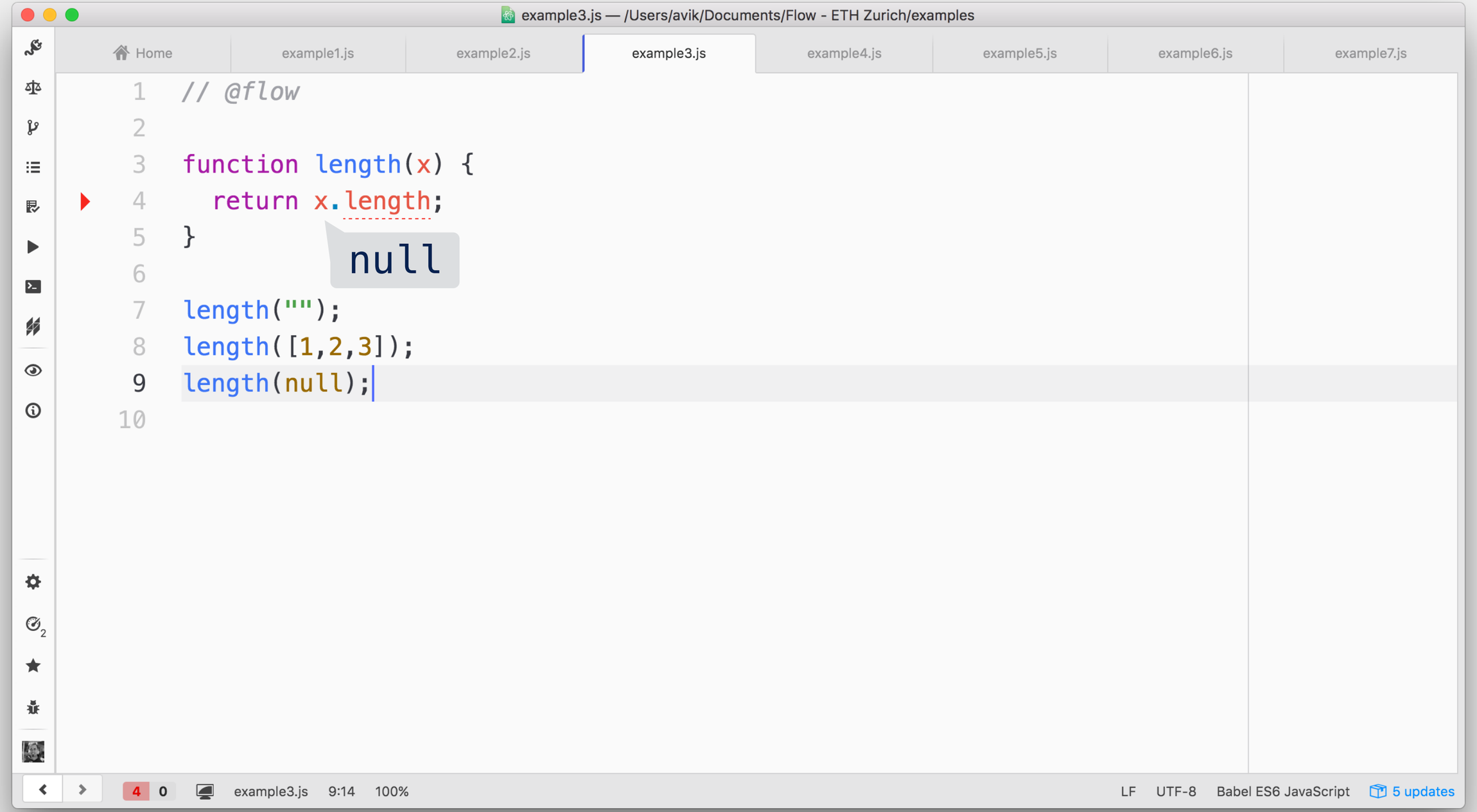
```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  length("");
8  length([1,2,3]);
9  length(null);
10
```

4 0 example3.js 9:14 100% LF UTF-8 Babel ES6 JavaScript 5 updates

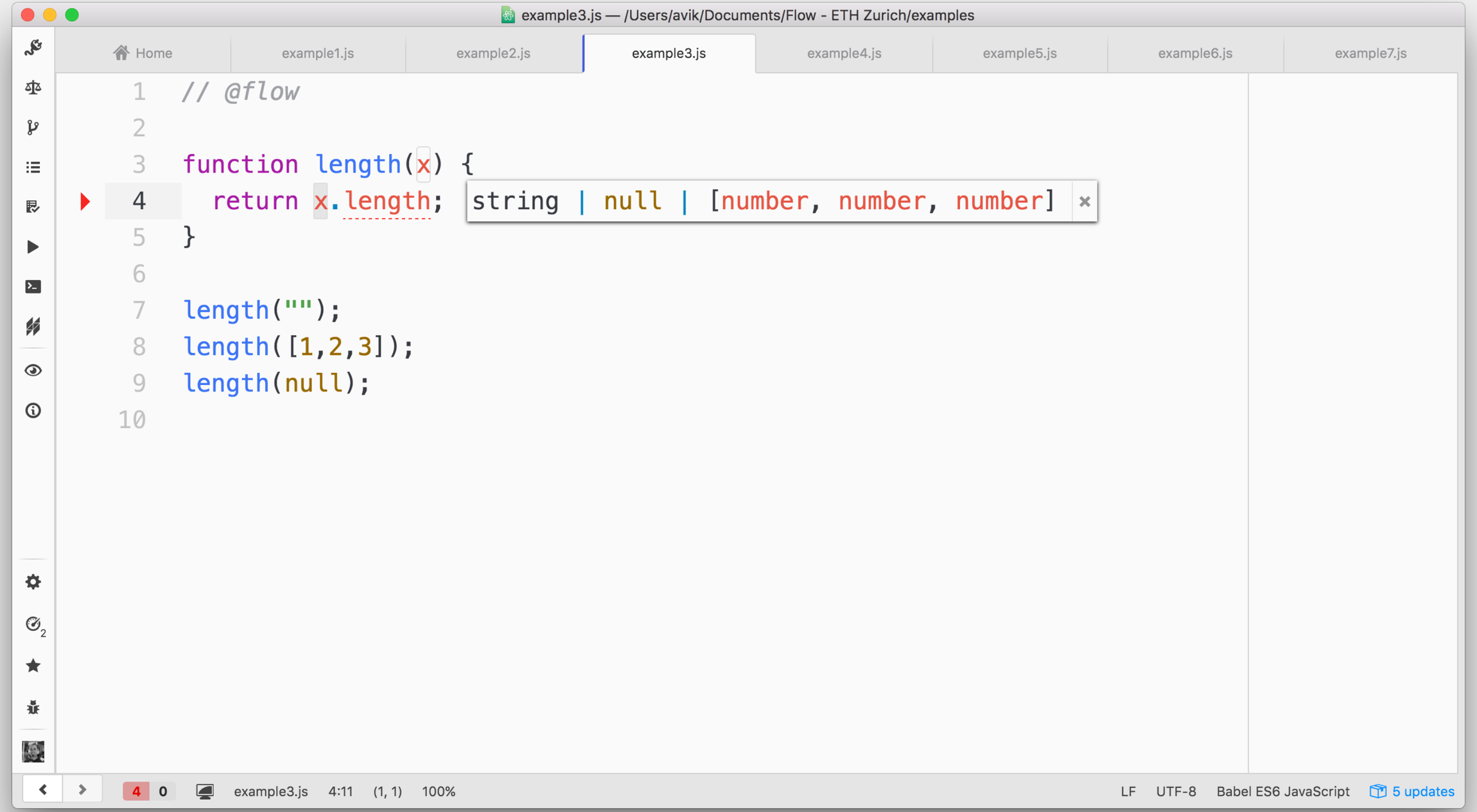
What happens at run time?

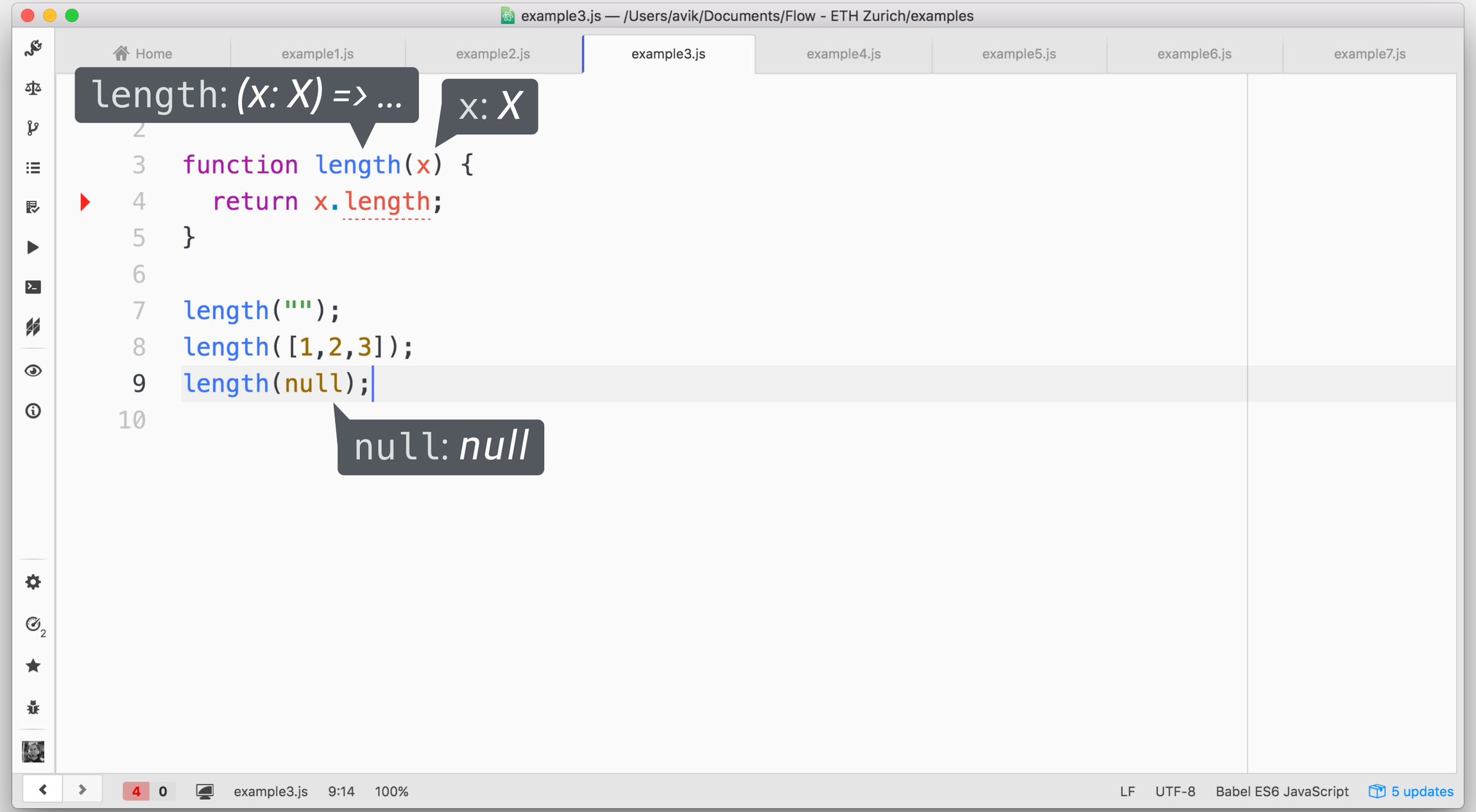

example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 return x.length;
5 }
6
7 length("");
8 length([1,2,3]);
9 length(null);
10
null
4 0
example3.js
9:14
100%
LF
UTF-8
Babel ES6 JavaScript
5 updates

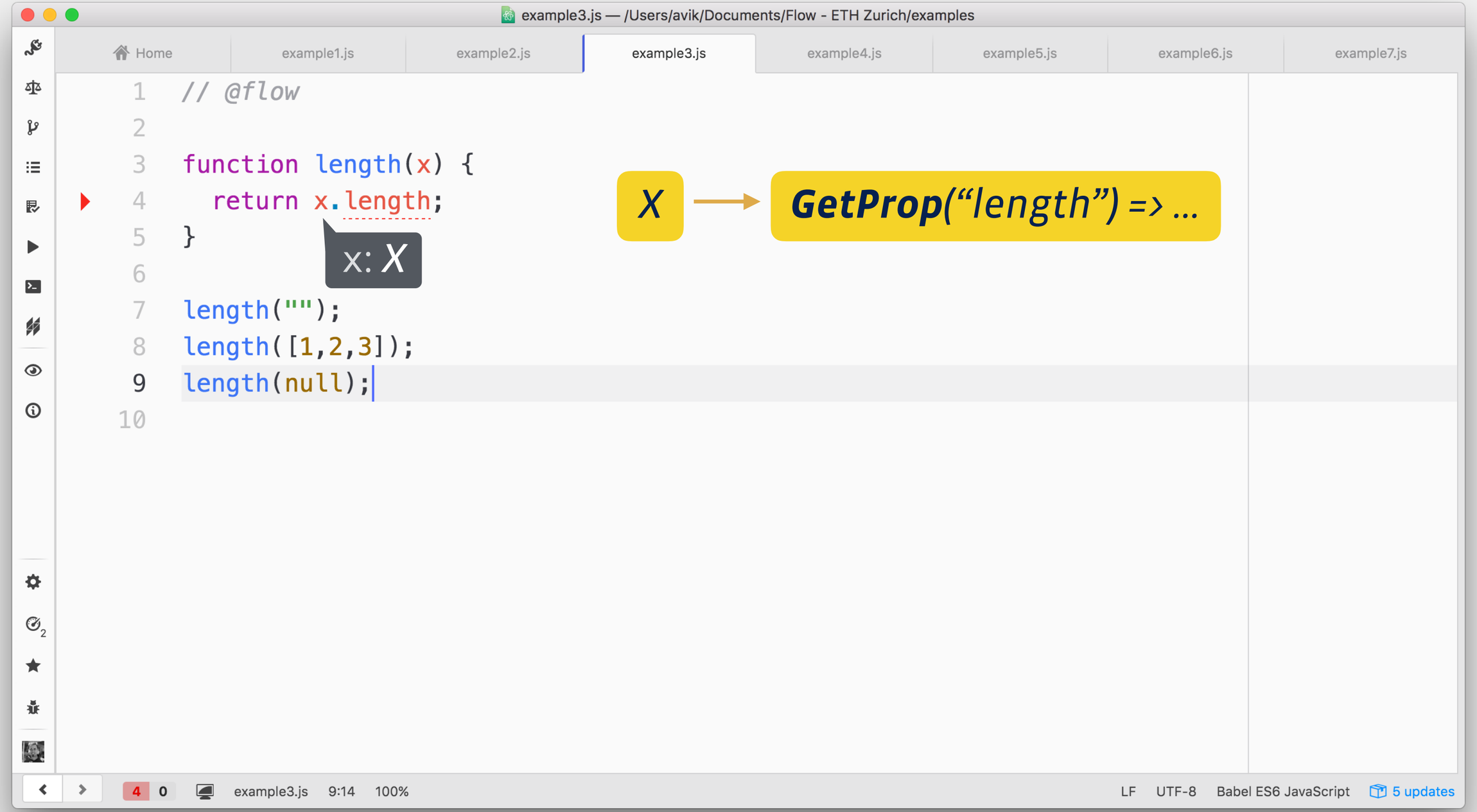

example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 return x.length;
5 }
6
7 length("");
8 length([1,2,3]);
9 length(null);
10
null
4 0
example3.js
9:14
100%
LF
UTF-8
Babel ES6 JavaScript
5 updates

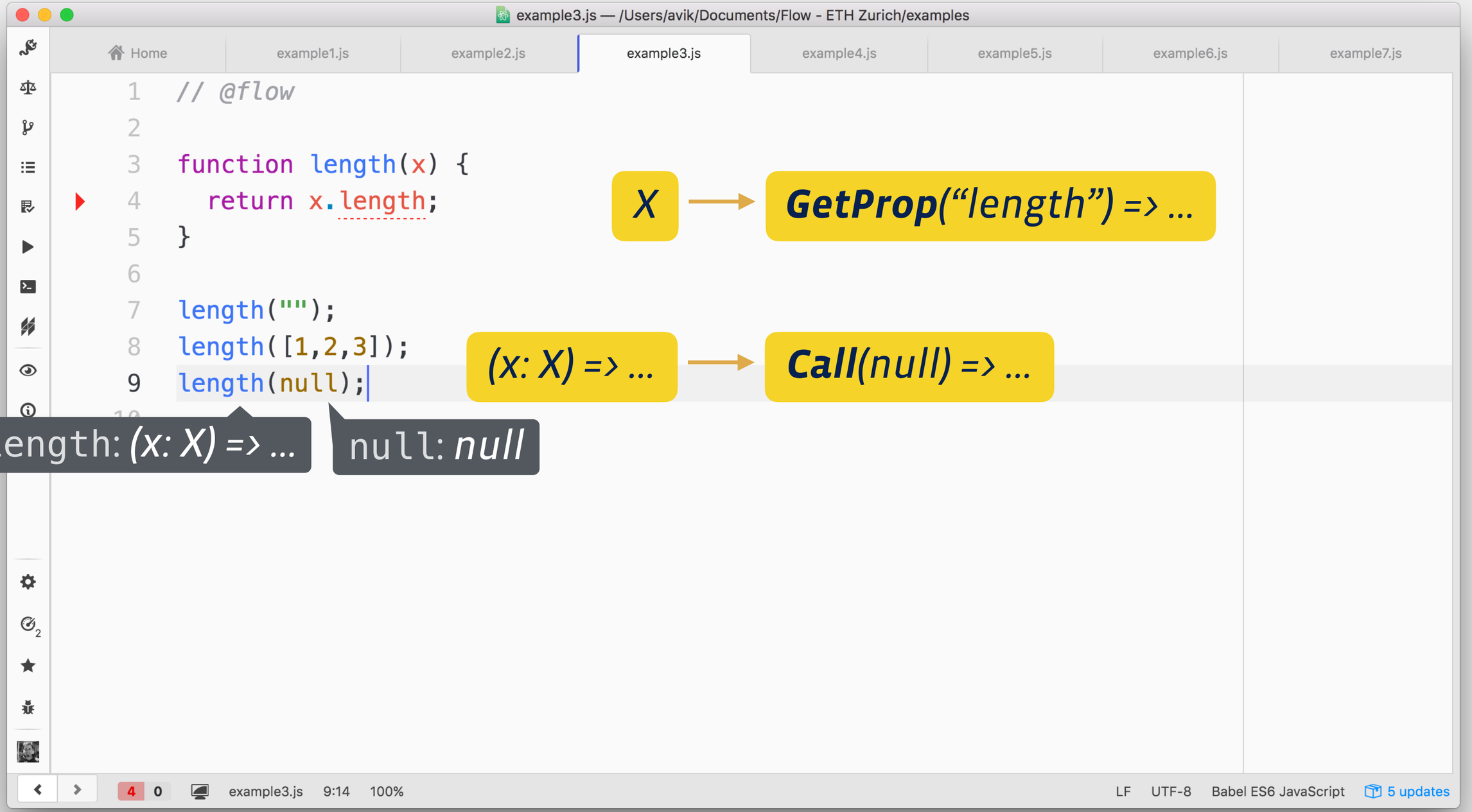

example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 return x.length;
5 }
null
6
7 length("");
8 length([1,2,3]);
9 length(null);
10
4 0
example3.js
9:14
100%
LF
UTF-8
Babel ES6 JavaScript
5 updates

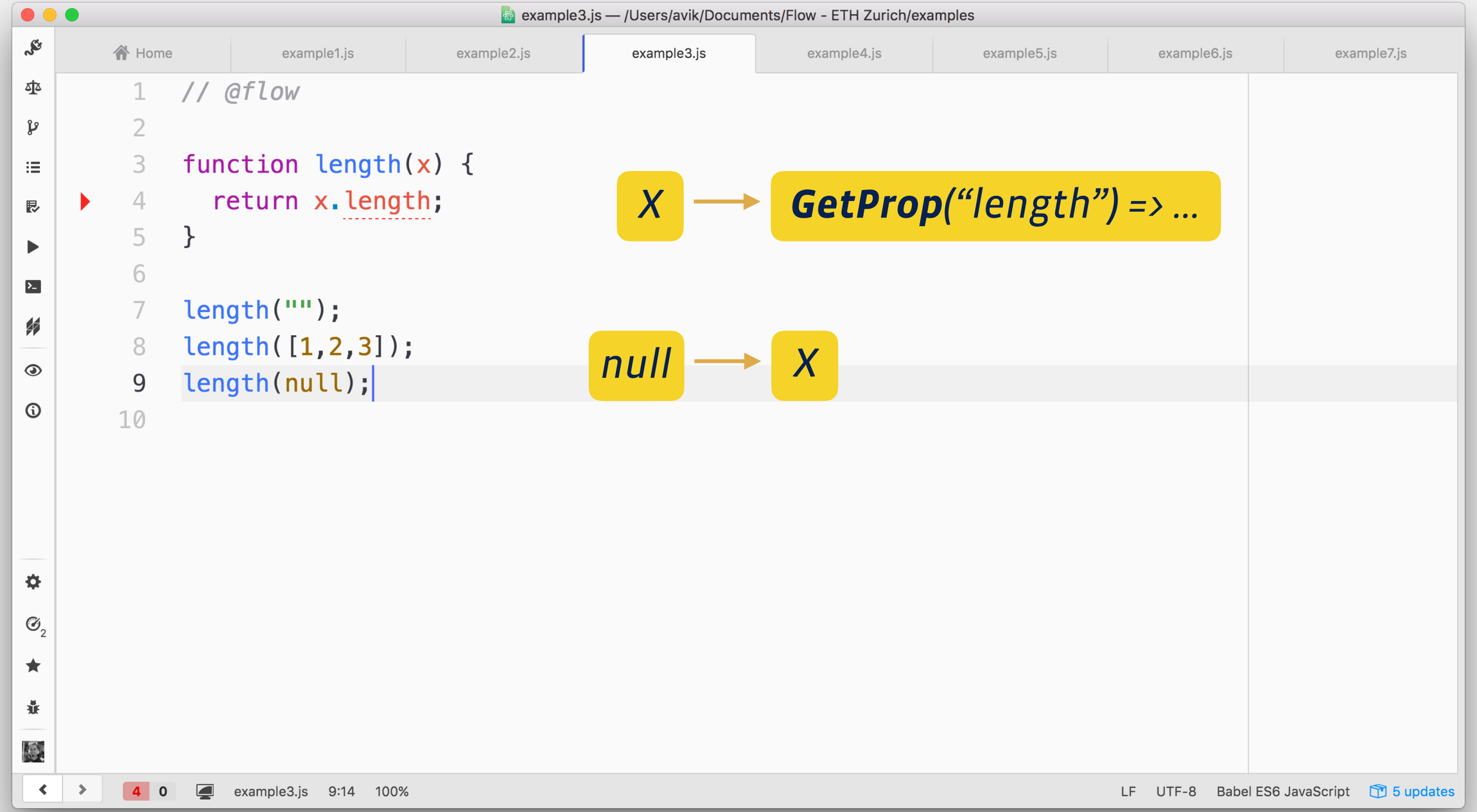
Flow “runs” the same code *statically*...

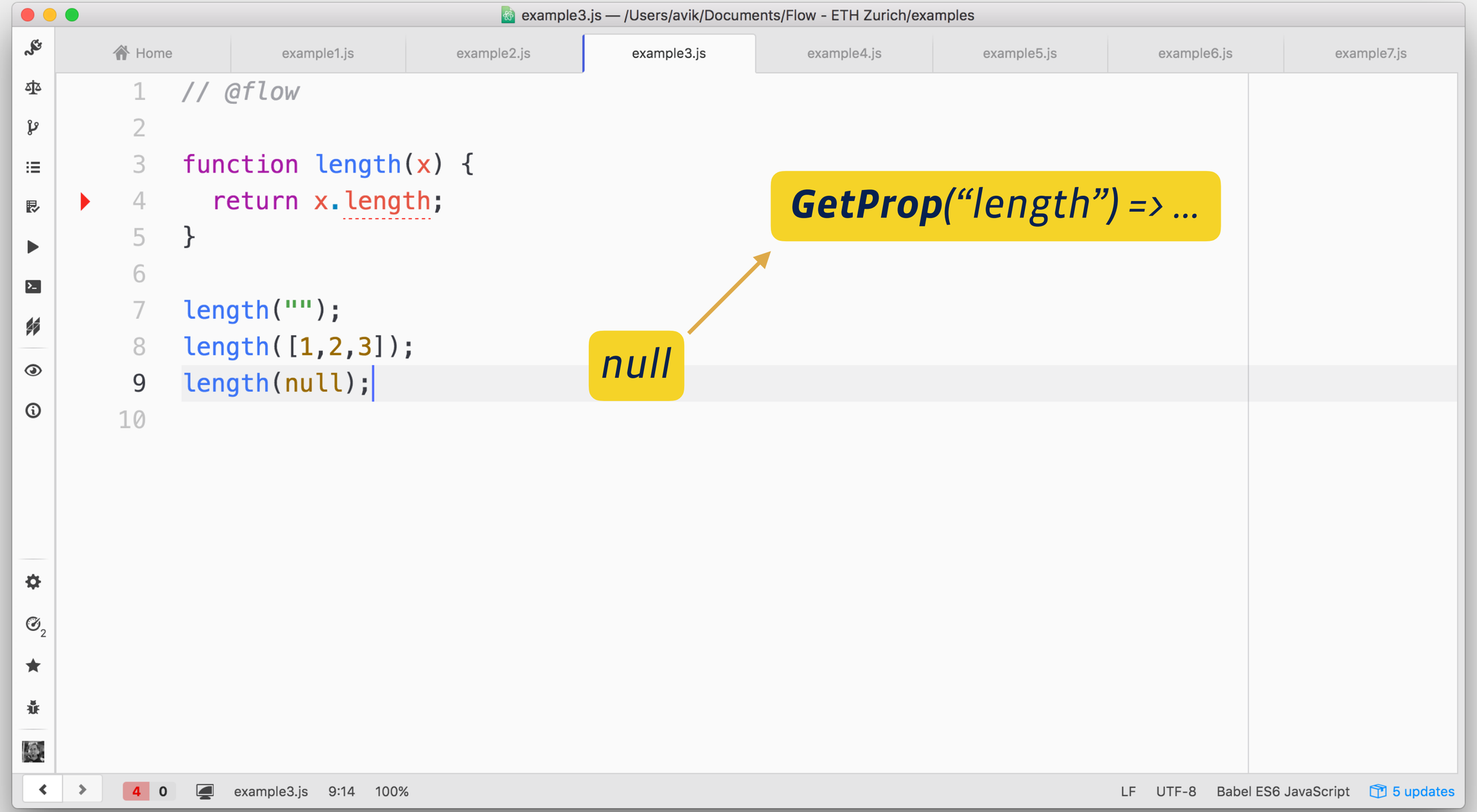

example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 return x.length;
5 }
6
7 length("");
8 length([1,2,3]);
9 length(null);
10
string | null | [number, number, number]
4 0
example3.js
4:11
(1, 1)
100%
LF
UTF-8
Babel ES6 JavaScript
5 updates

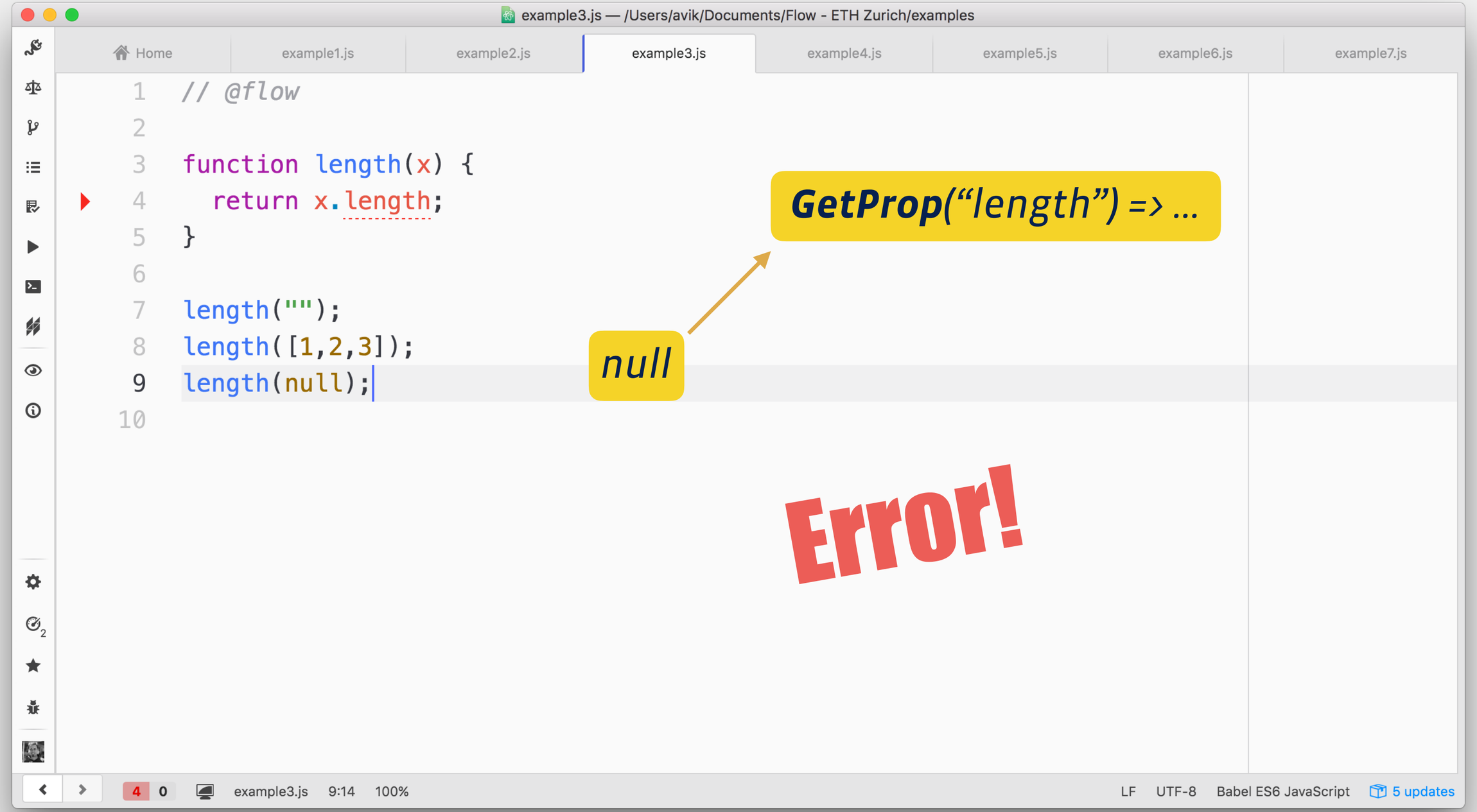

example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
length: (x: X) => ...
x: X
function length(x) {
return x.length;
}
length("");
length([1,2,3]);
length(null);
null: null
4 0
example3.js
9:14
100%
LF
UTF-8
Babel ES6 JavaScript
5 updates


example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 return x.length;
5 }
6
7 length("");
8 length([1,2,3]);
9 length(null);
10
x: X
X
GetProp("length") => ...
4 0
example3.js 9:14 100%
LF UTF-8 Babel ES6 JavaScript
5 updates


example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 return x.length;
5 }
6
7 length("");
8 length([1,2,3]);
9 length(null);
X → GetProp("length") => ...
(x: X) => ... → Call(null) => ...
length: (x: X) => ...
null: null
4 0
example3.js 9:14 100%
LF UTF-8 Babel ES6 JavaScript
5 updates


example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 return x.length;
5 }
6
7 length("");
8 length([1,2,3]);
9 length(null);
10
X
GetProp("length") => ...
null
X
4 0
example3.js 9:14 100%
LF UTF-8 Babel ES6 JavaScript
5 updates


example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 return x.length;
5 }
6
7 length("");
8 length([1,2,3]);
9 length(null);
10
GetProp("length") => ...
null
4 0
example3.js 9:14 100%
LF UTF-8 Babel ES6 JavaScript
5 updates


example3.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 return x.length;
5 }
6
7 length("");
8 length([1,2,3]);
9 length(null);
10
GetProp("length") => ...
null
Error!
4 0
example3.js
9:14
100%
LF
UTF-8
Babel ES6 JavaScript
5 updates

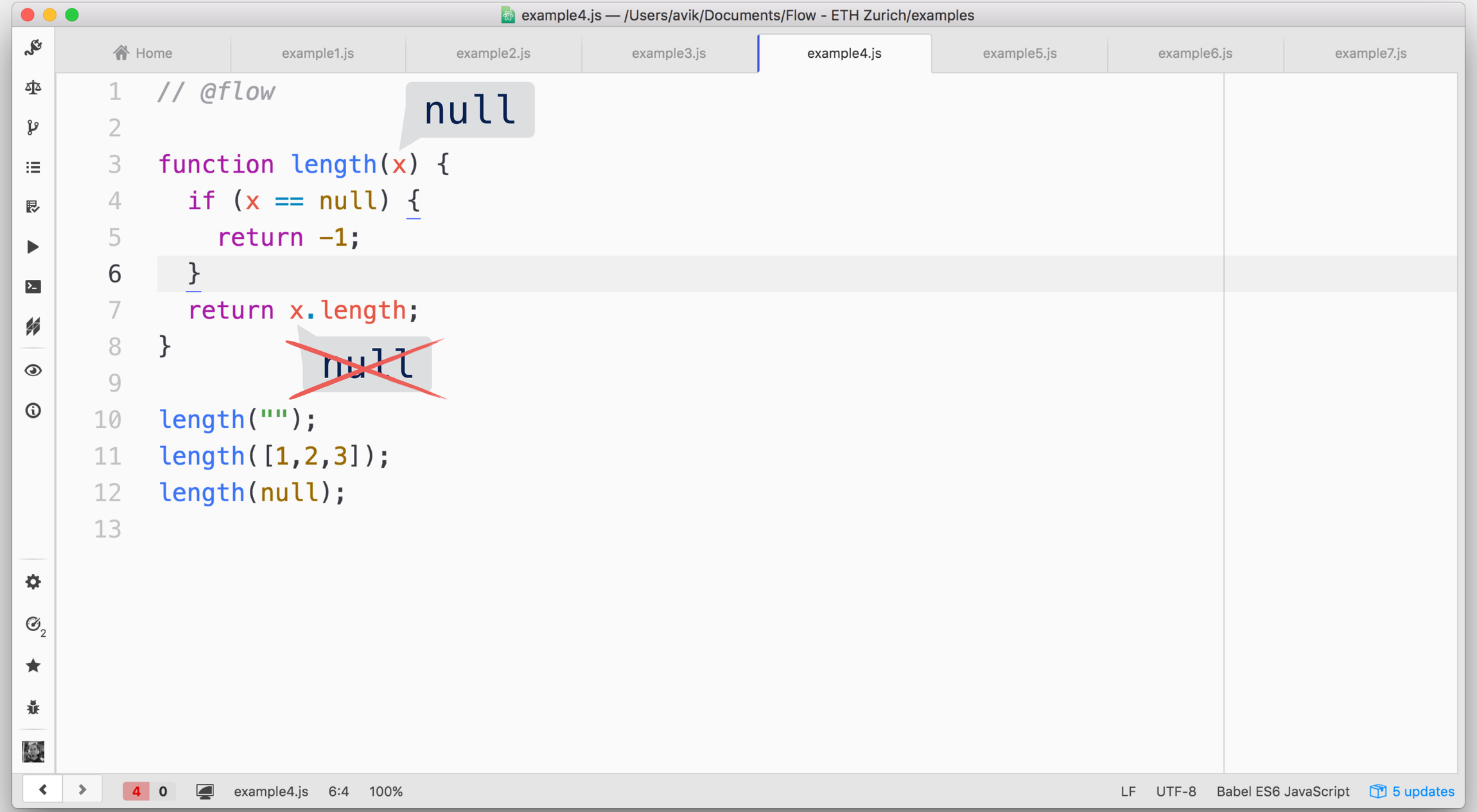
Let’s fix this code!

example4.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    if (x == null) {
5      return -1;
6    }
7    return x.length;
8  }
9
10 length("");
11 length([1,2,3]);
12 length(null);
13
```

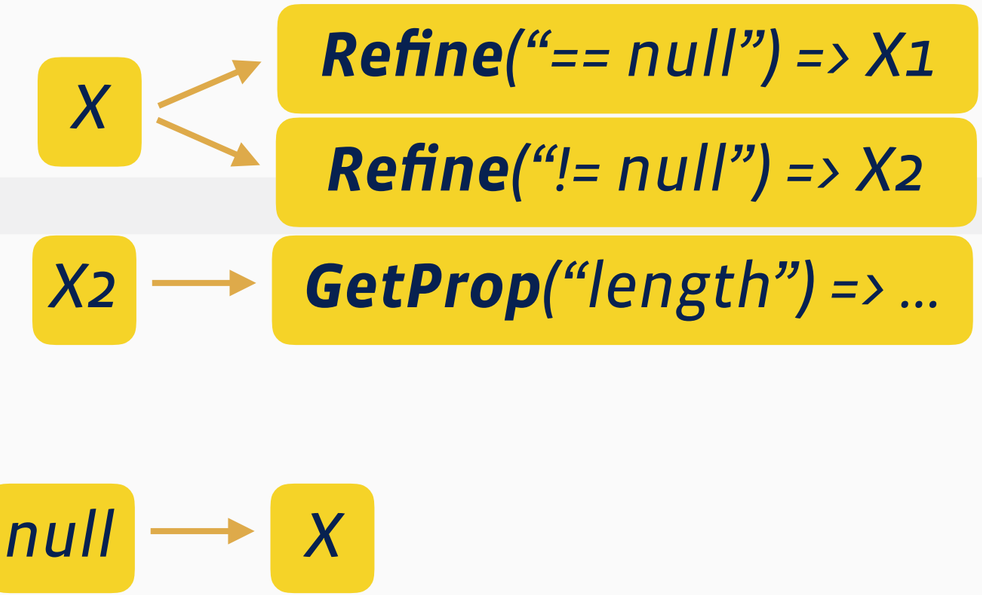
4 0 example4.js 6:4 100% LF UTF-8 Babel ES6 JavaScript 5 updates


example4.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 length("");
11 length([1,2,3]);
12 length(null);
13
null
null
4 0
example4.js 6:4 100%
LF UTF-8 Babel ES6 JavaScript
5 updates

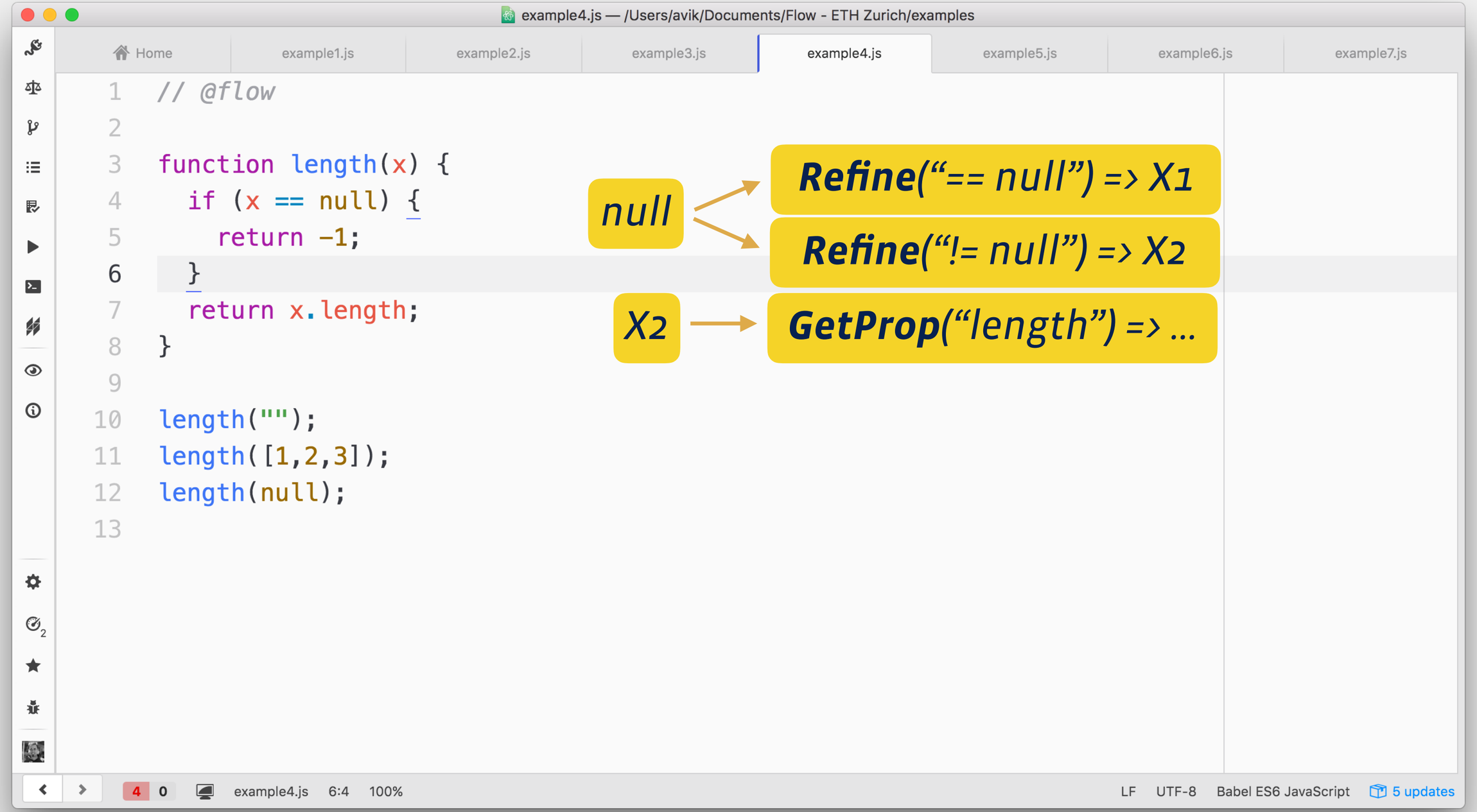
example4.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    if (x == null) {
5      return -1;
6    }
7    return x.length;
8  }
9
10 length("");
11 length([1,2,3]);
12 length(null);
13
```



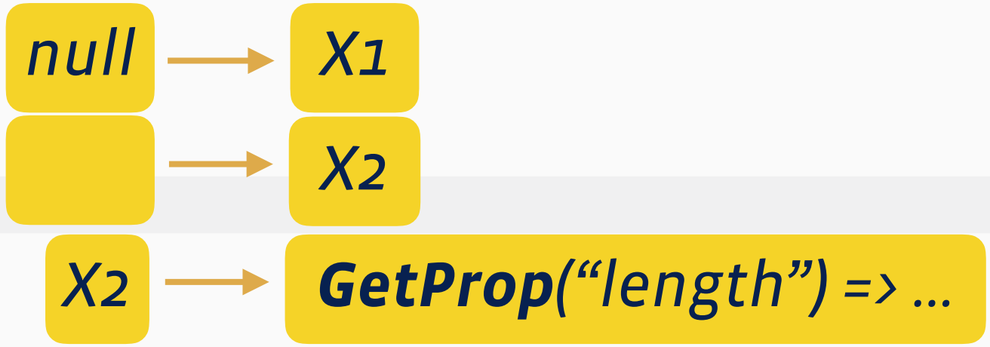
4 0 example4.js 6:4 100% LF UTF-8 Babel ES6 JavaScript 5 updates


example4.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 length("");
11 length([1,2,3]);
12 length(null);
13
null
Refine("== null") => X1
Refine("!= null") => X2
X2
GetProp("length") => ...
4 0
example4.js 6:4 100%
LF UTF-8 Babel ES6 JavaScript
5 updates

example4.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    if (x == null) {
5      return -1;
6    }
7    return x.length;
8  }
9
10 length("");
11 length([1,2,3]);
12 length(null);
13
```



4 0 example4.js 6:4 100% LF UTF-8 Babel ES6 JavaScript 5 updates

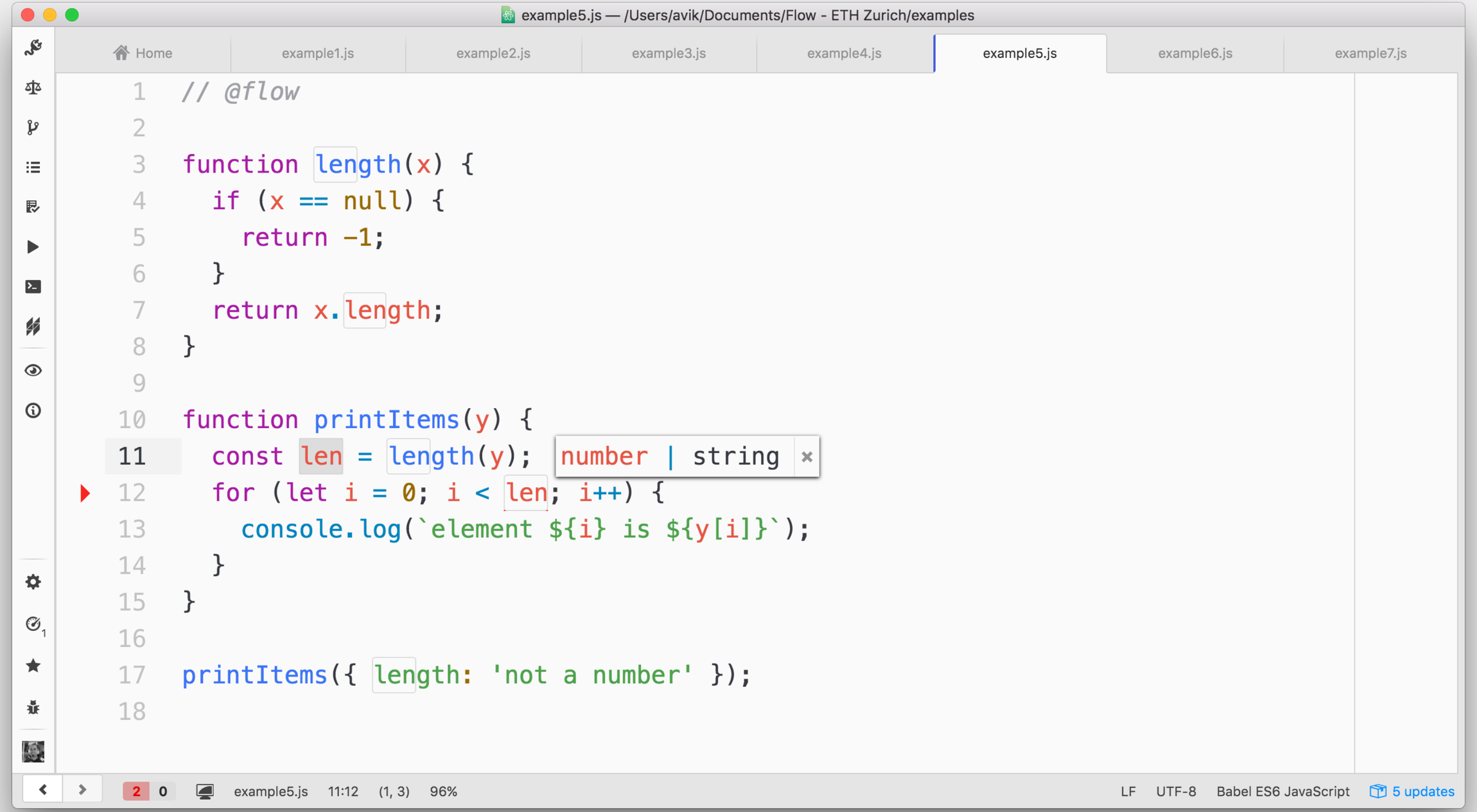
Ready for some more code?

example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    if (x == null) {
5      return -1;
6    }
7    return x.length;
8  }
9
10 function printItems(y) {
11   const len = length(y);
12   for (let i = 0; i < len; i++) {
13     console.log(`element ${i} is ${y[i]}`);
14   }
15 }
16
17 printItems({ length: 'not a number' });
18
```

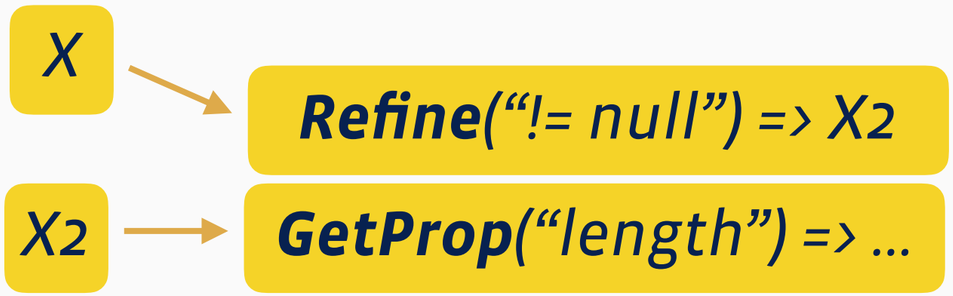
2 0 example5.js 17:40 96% LF UTF-8 Babel ES6 JavaScript 5 updates


example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
number | string
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
2 0
example5.js
11:12
(1, 3)
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates

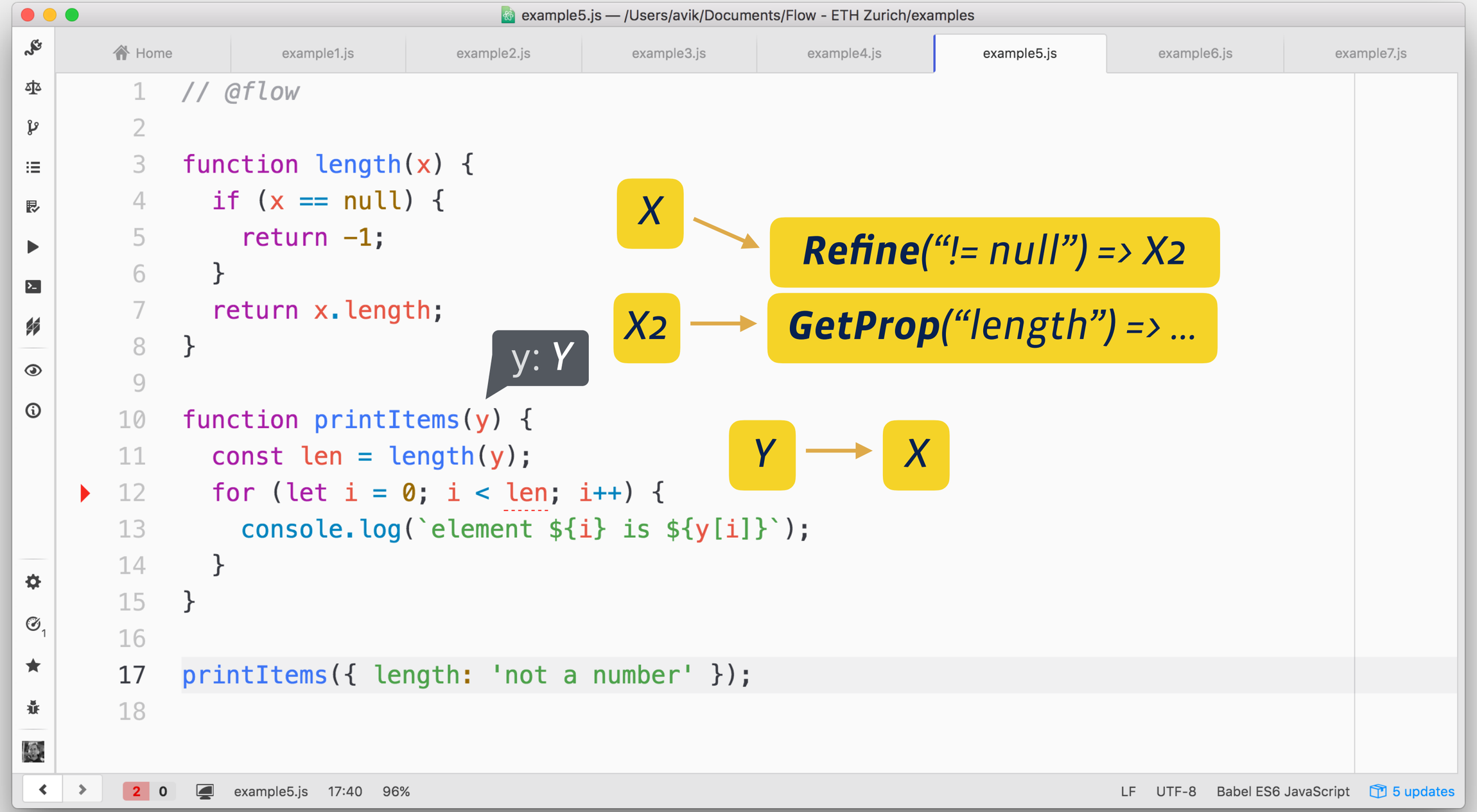
example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples

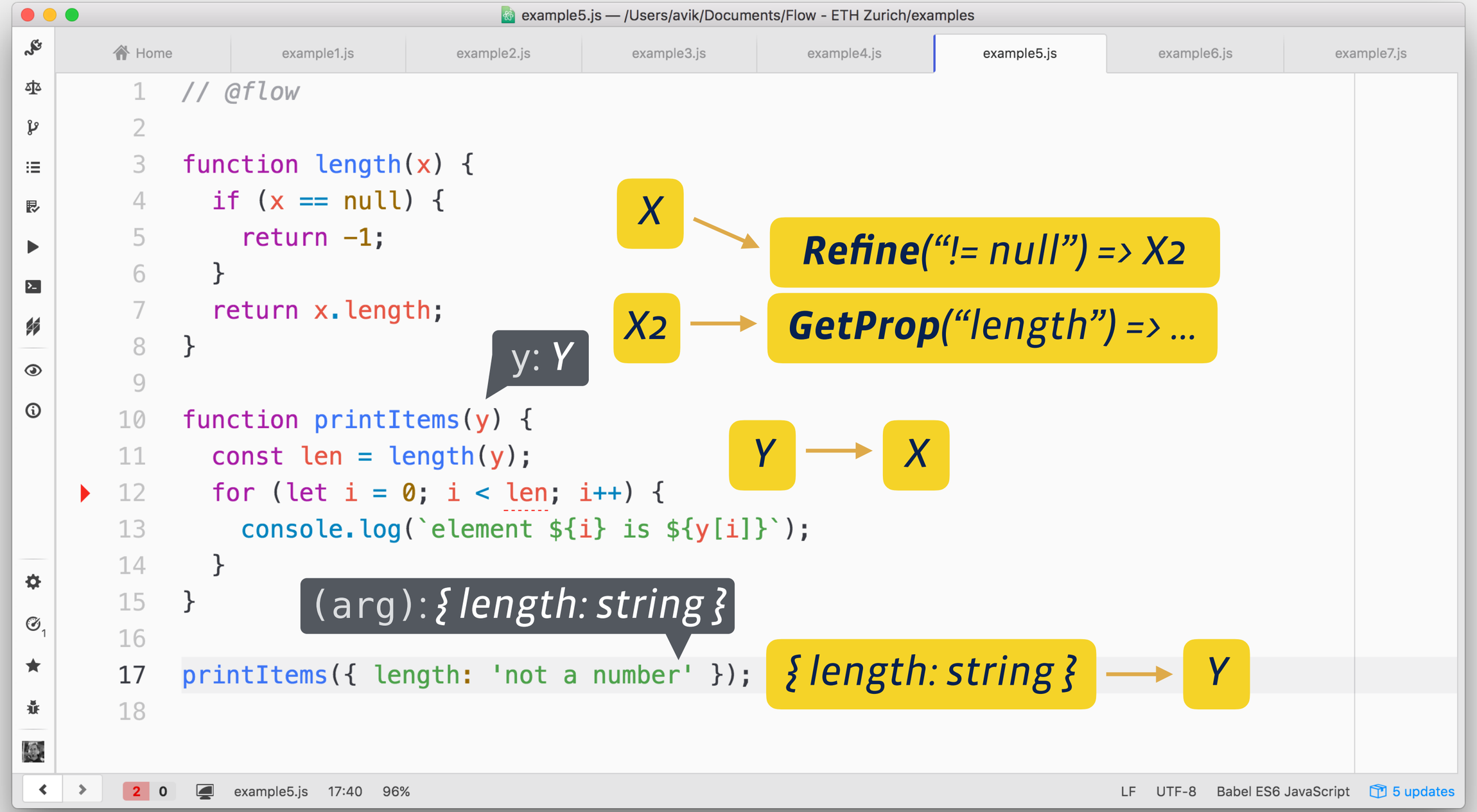
Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

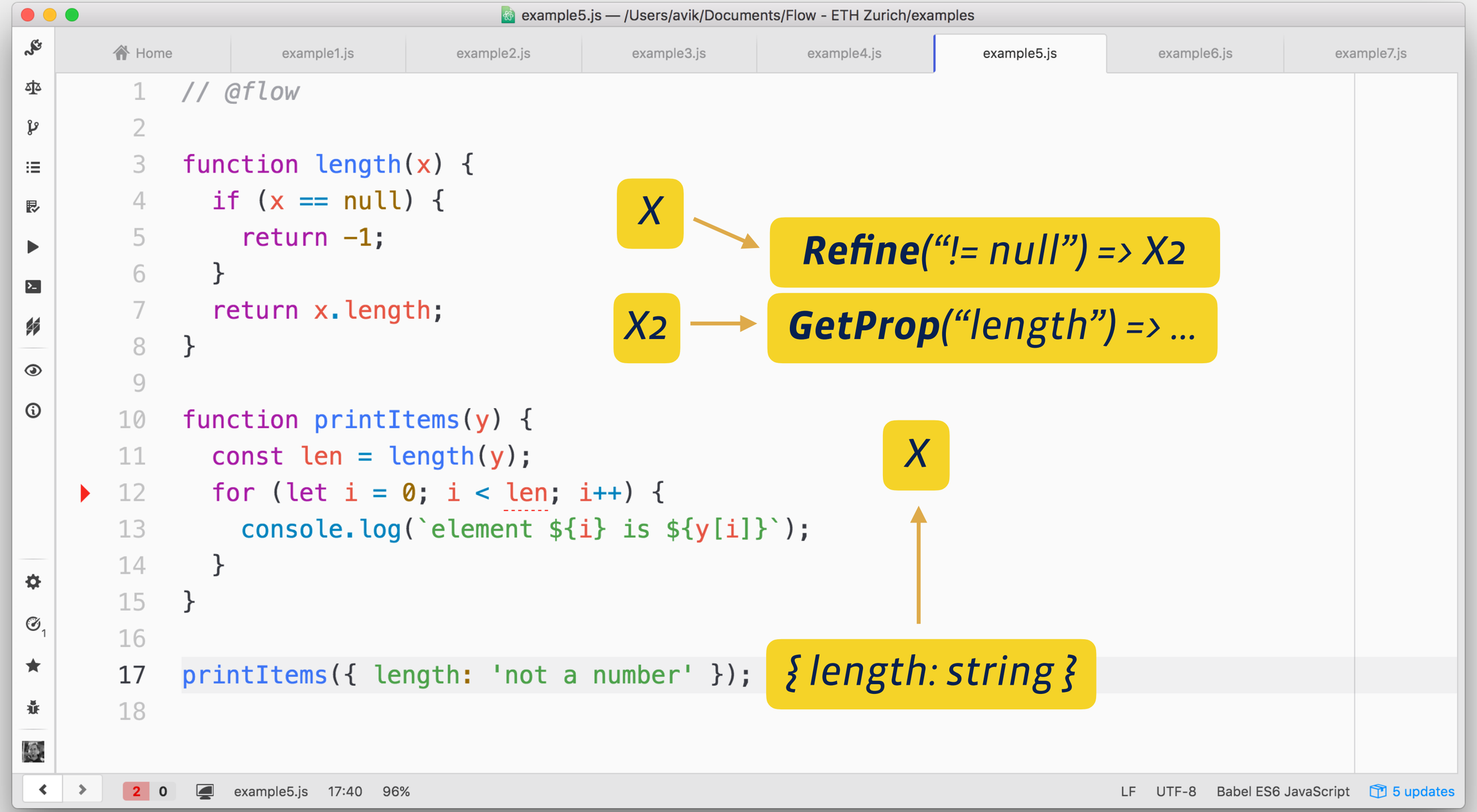
```
1  // @flow
2
3  function length(x) {
4    if (x == null) {
5      return -1;
6    }
7    return x.length;
8  }
9
10 function printItems(y) {
11   const len = length(y);
12   for (let i = 0; i < len; i++) {
13     console.log(`element ${i} is ${y[i]}`);
14   }
15 }
16
17 printItems({ length: 'not a number' });
18
```

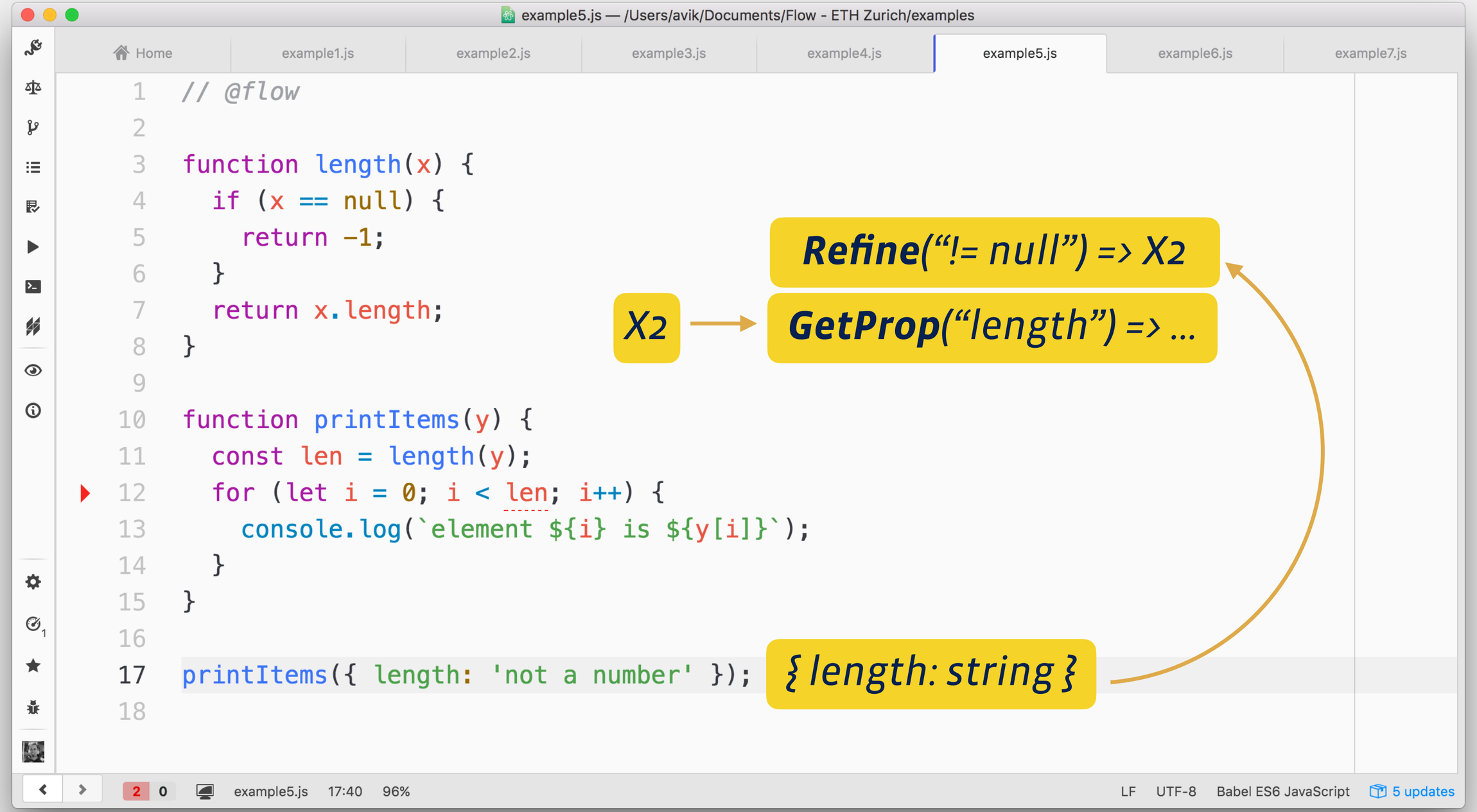


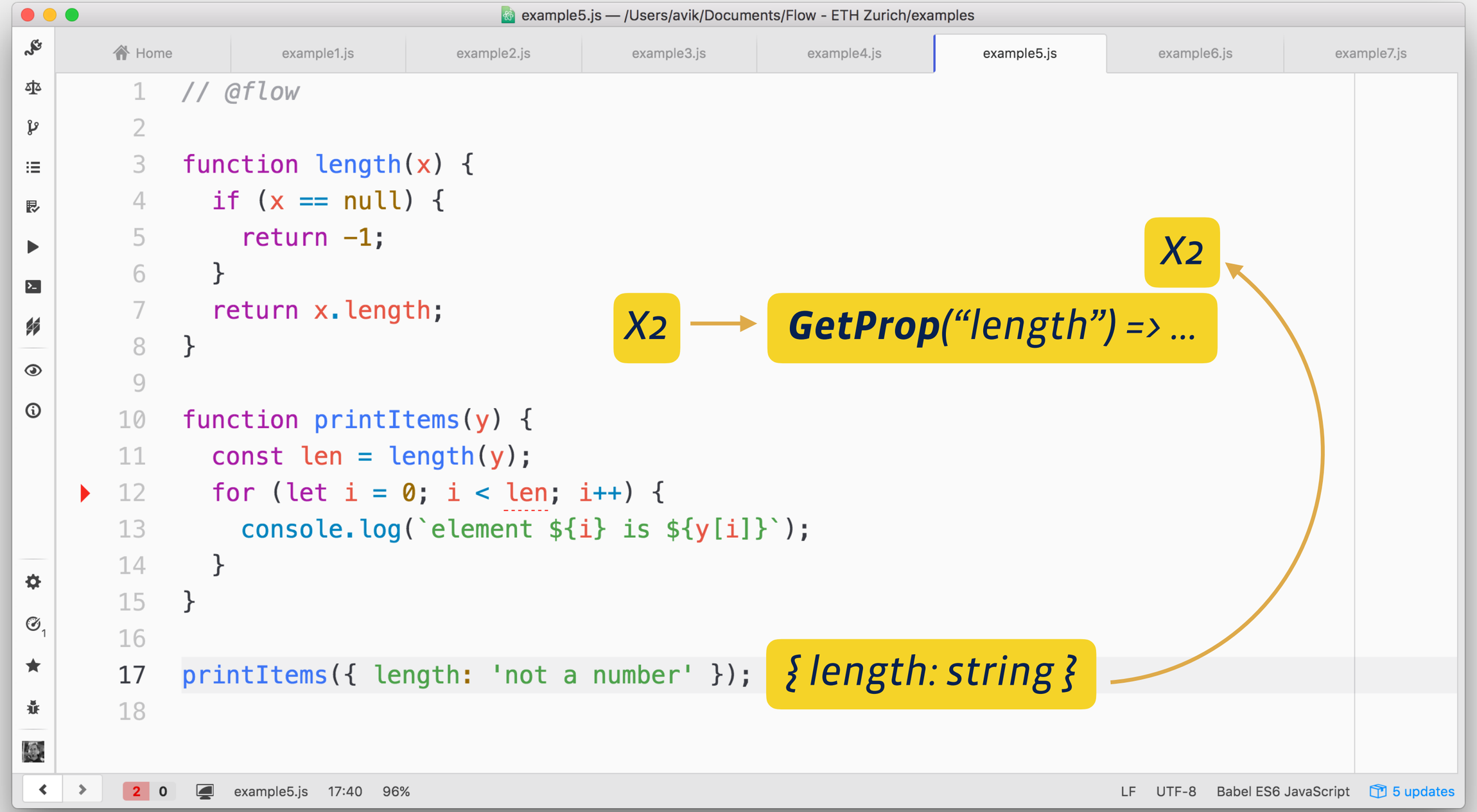
2 0 example5.js 17:40 96% LF UTF-8 Babel ES6 JavaScript 5 updates

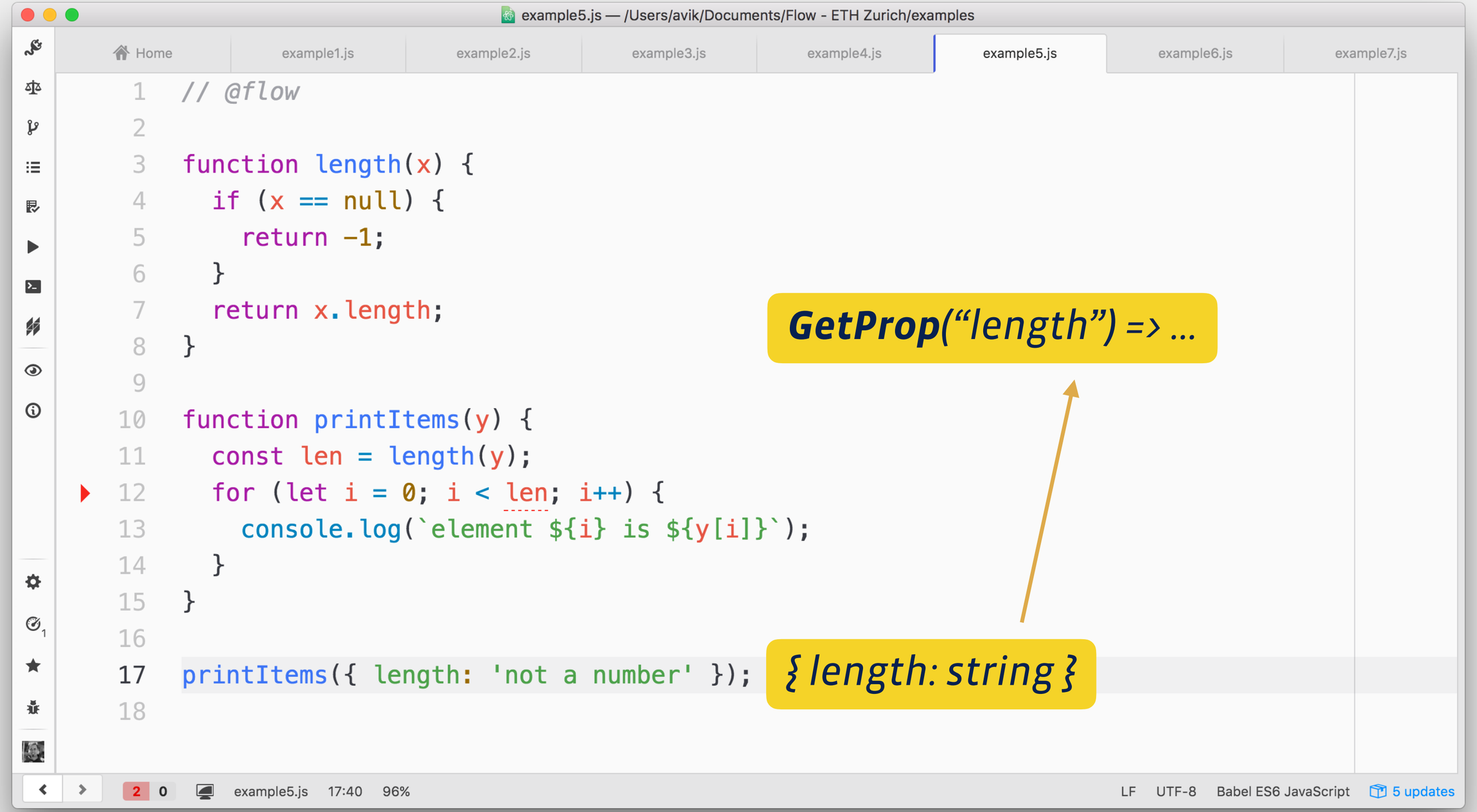

example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
y: Y
X
Refine("!= null") => X2
X2
GetProp("length") => ...
Y
X
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates

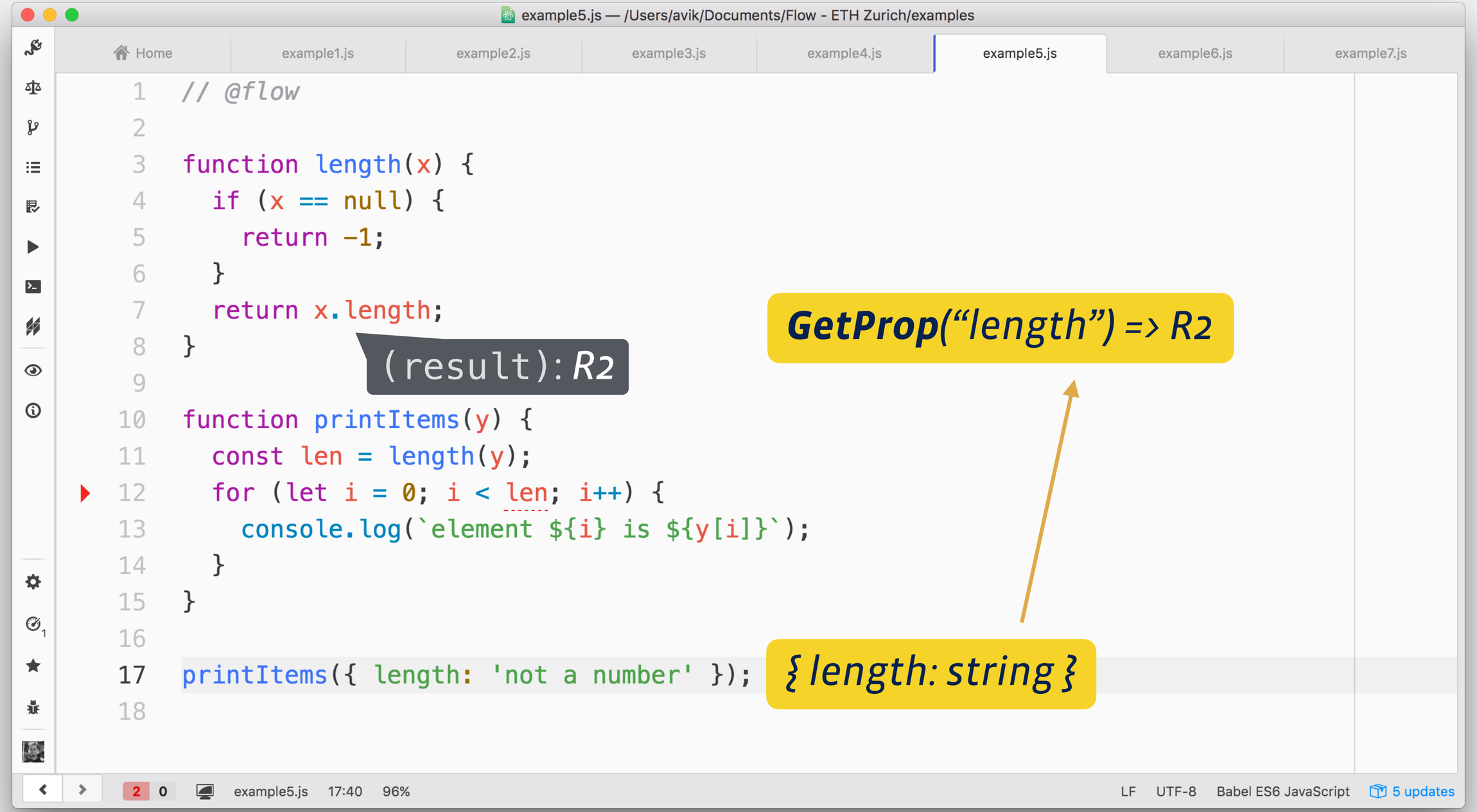

example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
X
Refine("!= null") => X2
X2
GetProp("length") => ...
y: Y
Y
X
(arg): { length: string }
{ length: string }
Y
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates

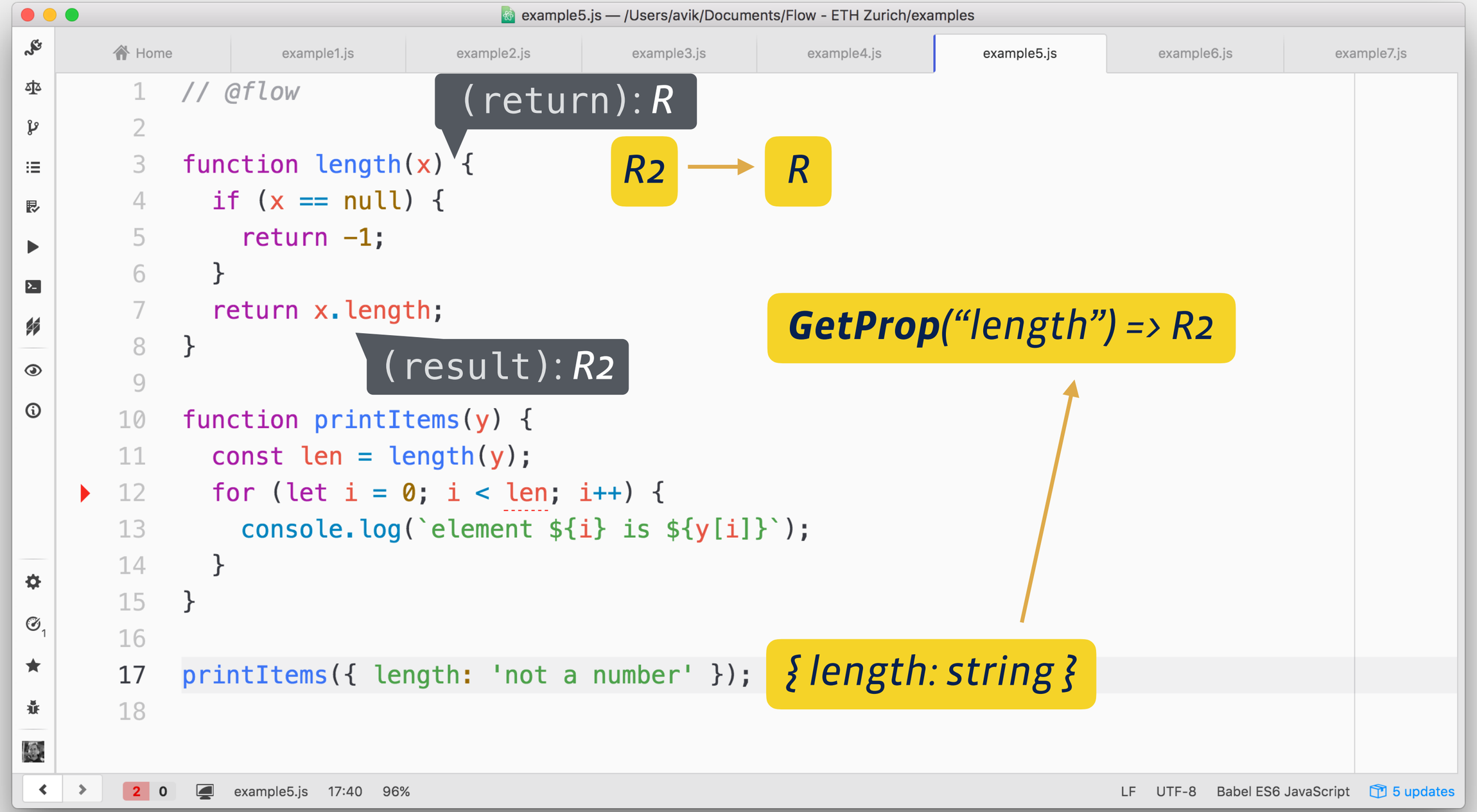

example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
X
Refine("!= null") => X2
X2
GetProp("length") => ...
X
{ length: string }
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates

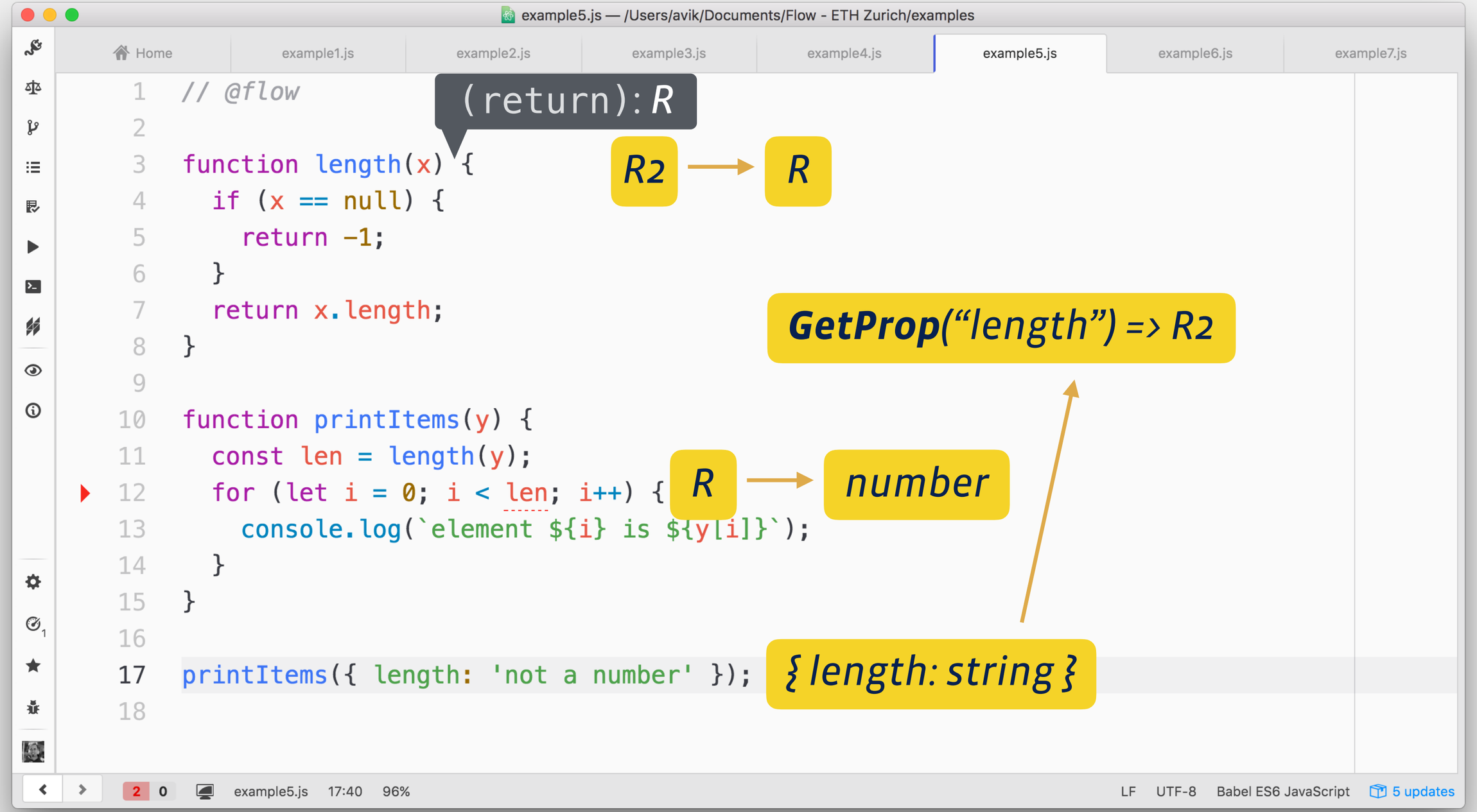

example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
Refine("!= null") => X2
X2
GetProp("length") => ...
{ length: string }
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates

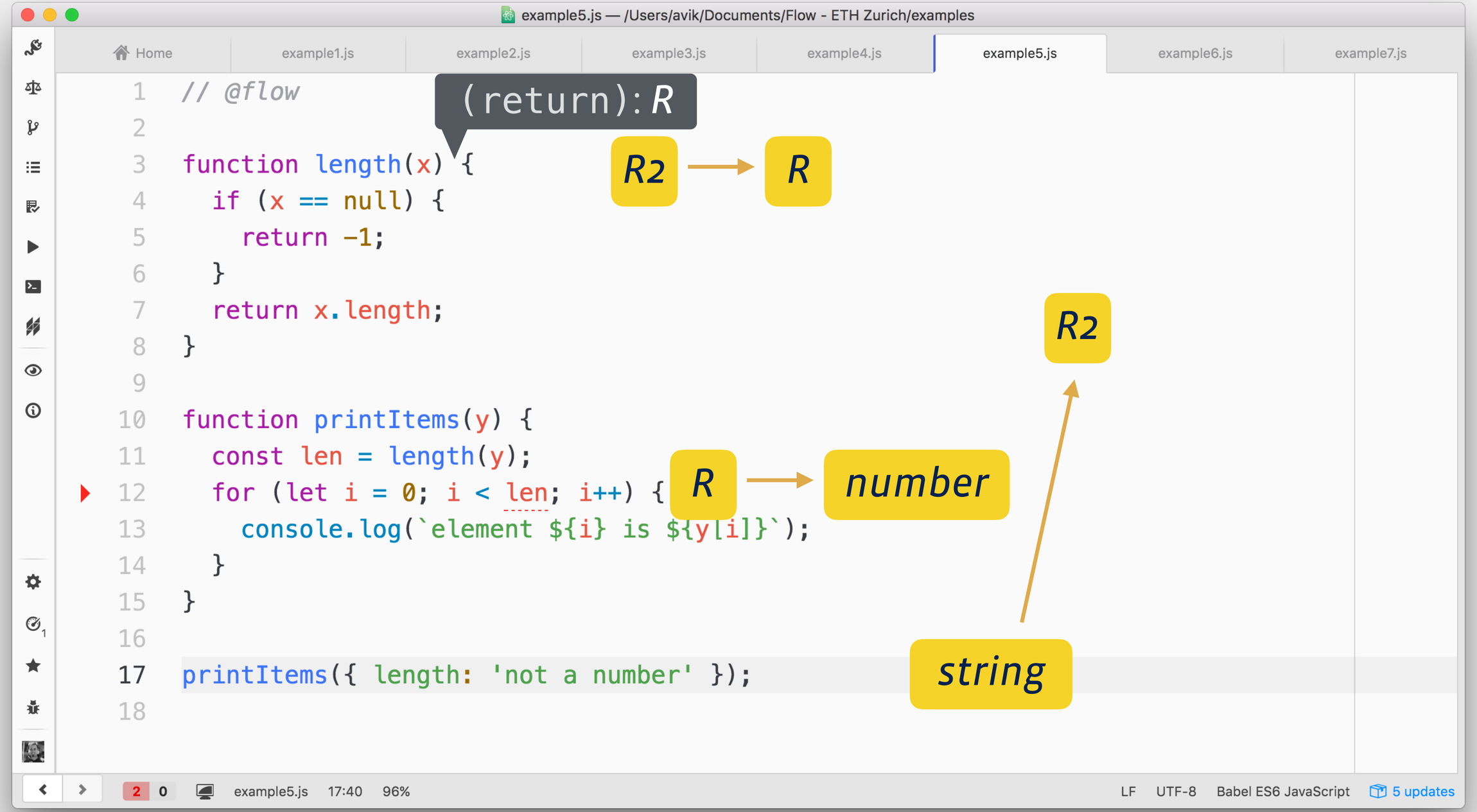

example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
X2
X2
GetProp("length") => ...
{ length: string }
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates

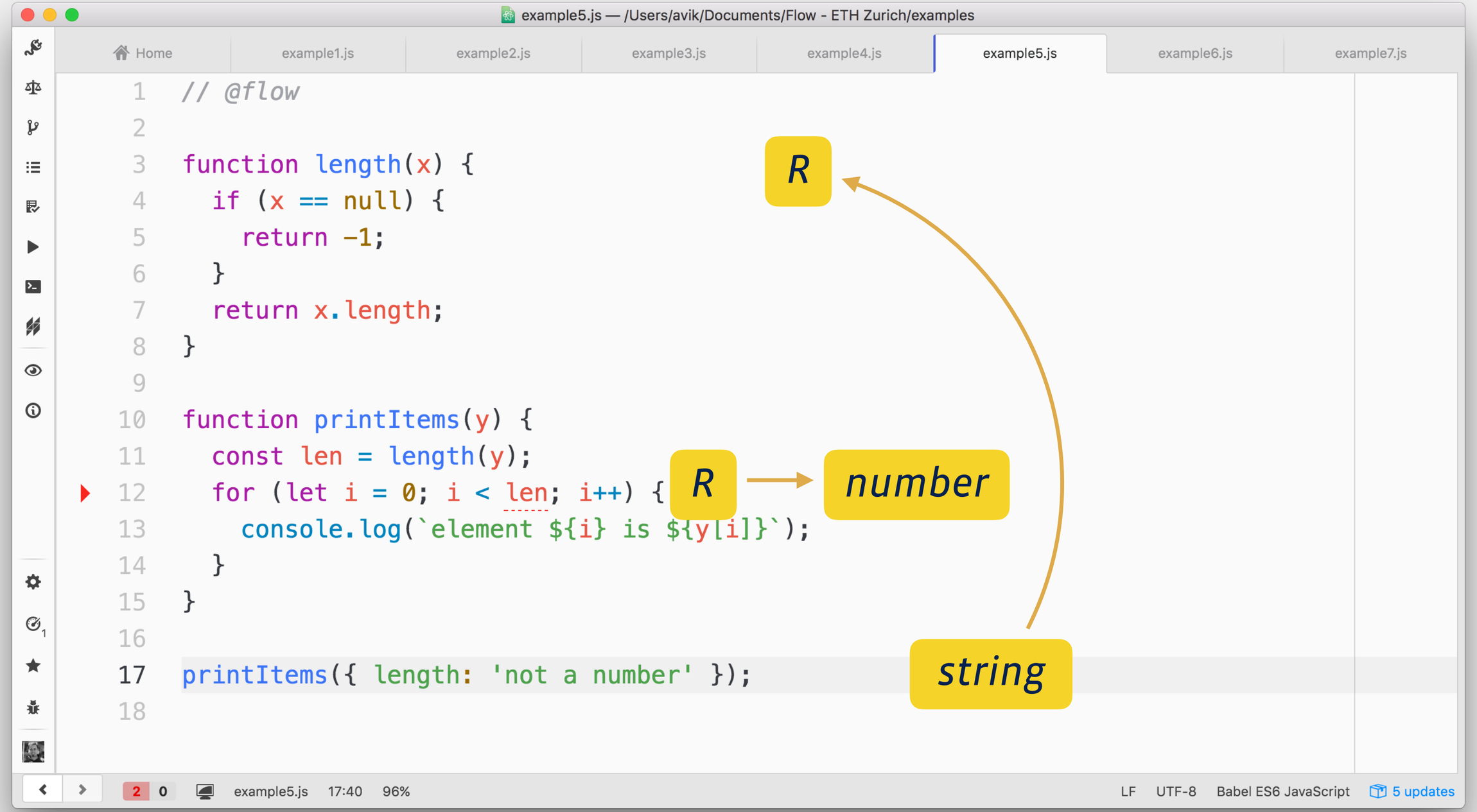

example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
GetProp("length") => ...
{ length: string }
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates

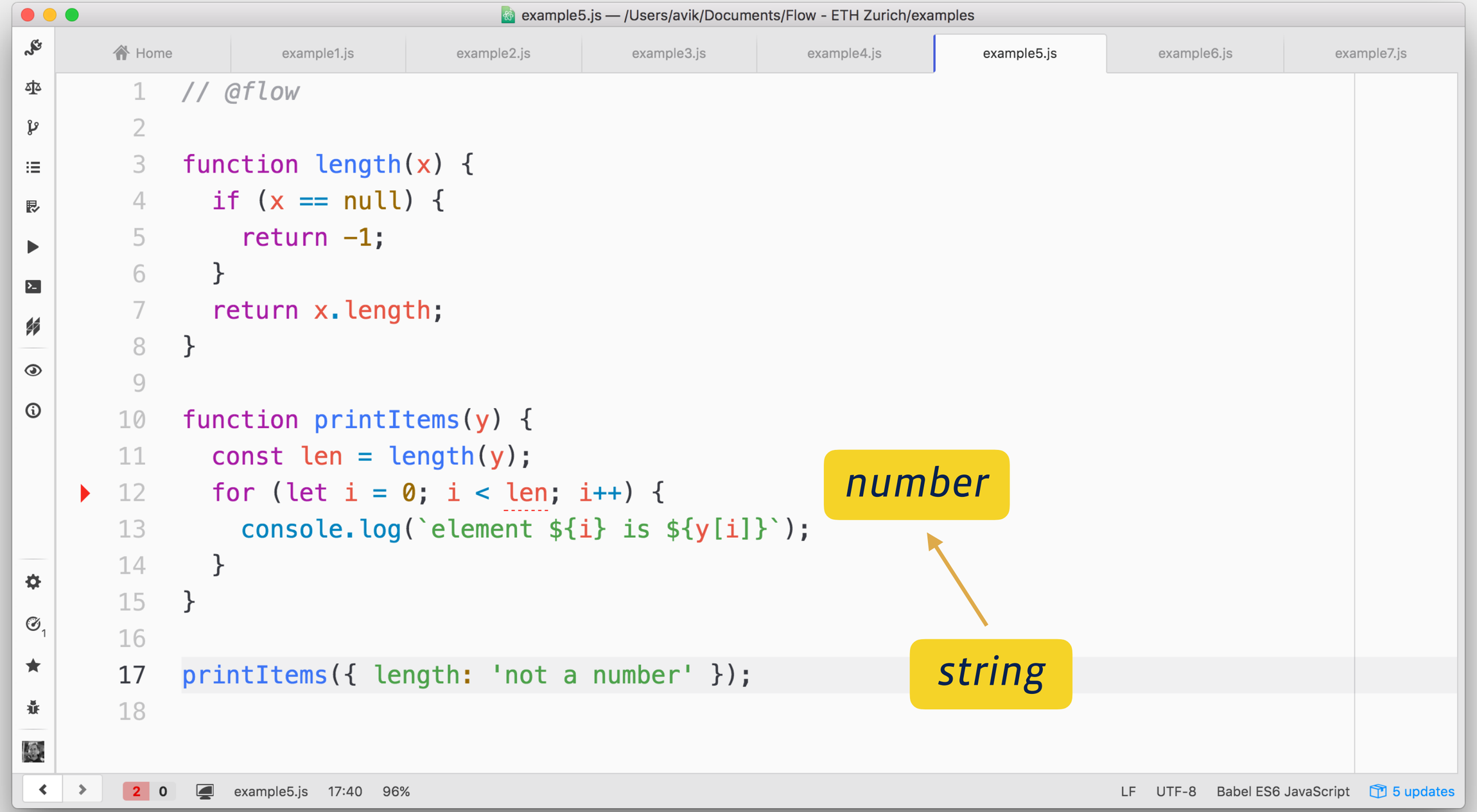

example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
(result): R2
GetProp("length") => R2
{ length: string }
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates

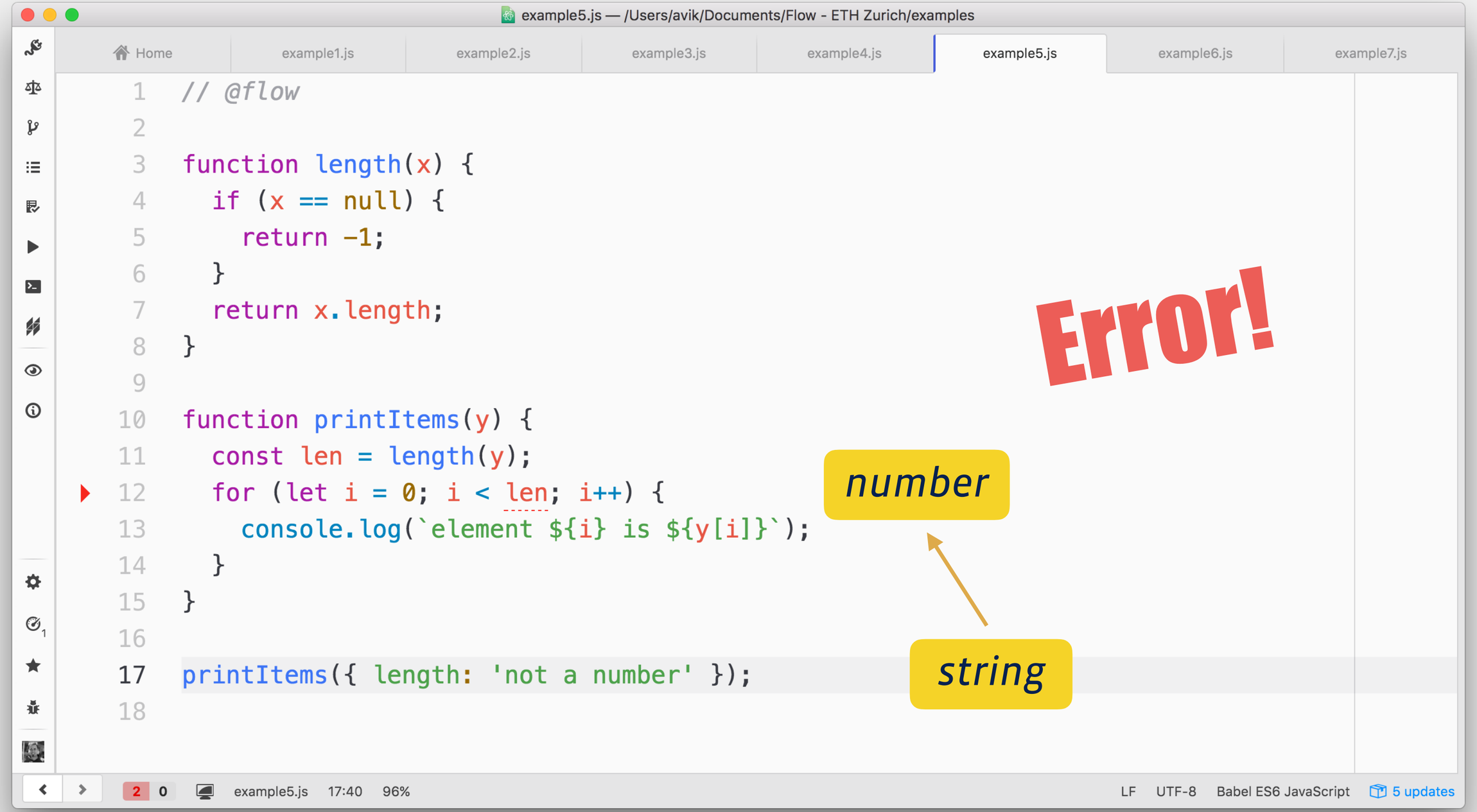

example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
(return): R
R2
R
(result): R2
GetProp("length") => R2
{ length: string }
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates


example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
(return): R
R2
R
GetProp("length") => R2
R
number
{ length: string }
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates


example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
(return): R
R2
R
R2
R
number
string
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates


example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
R
R
number
string
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates


example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
number
string
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates


example5.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 function length(x) {
4 if (x == null) {
5 return -1;
6 }
7 return x.length;
8 }
9
10 function printItems(y) {
11 const len = length(y);
12 for (let i = 0; i < len; i++) {
13 console.log(`element ${i} is ${y[i]}`);
14 }
15 }
16
17 printItems({ length: 'not a number' });
18
Error!
number
string
2 0
example5.js
17:40
96%
LF
UTF-8
Babel ES6 JavaScript
5 updates

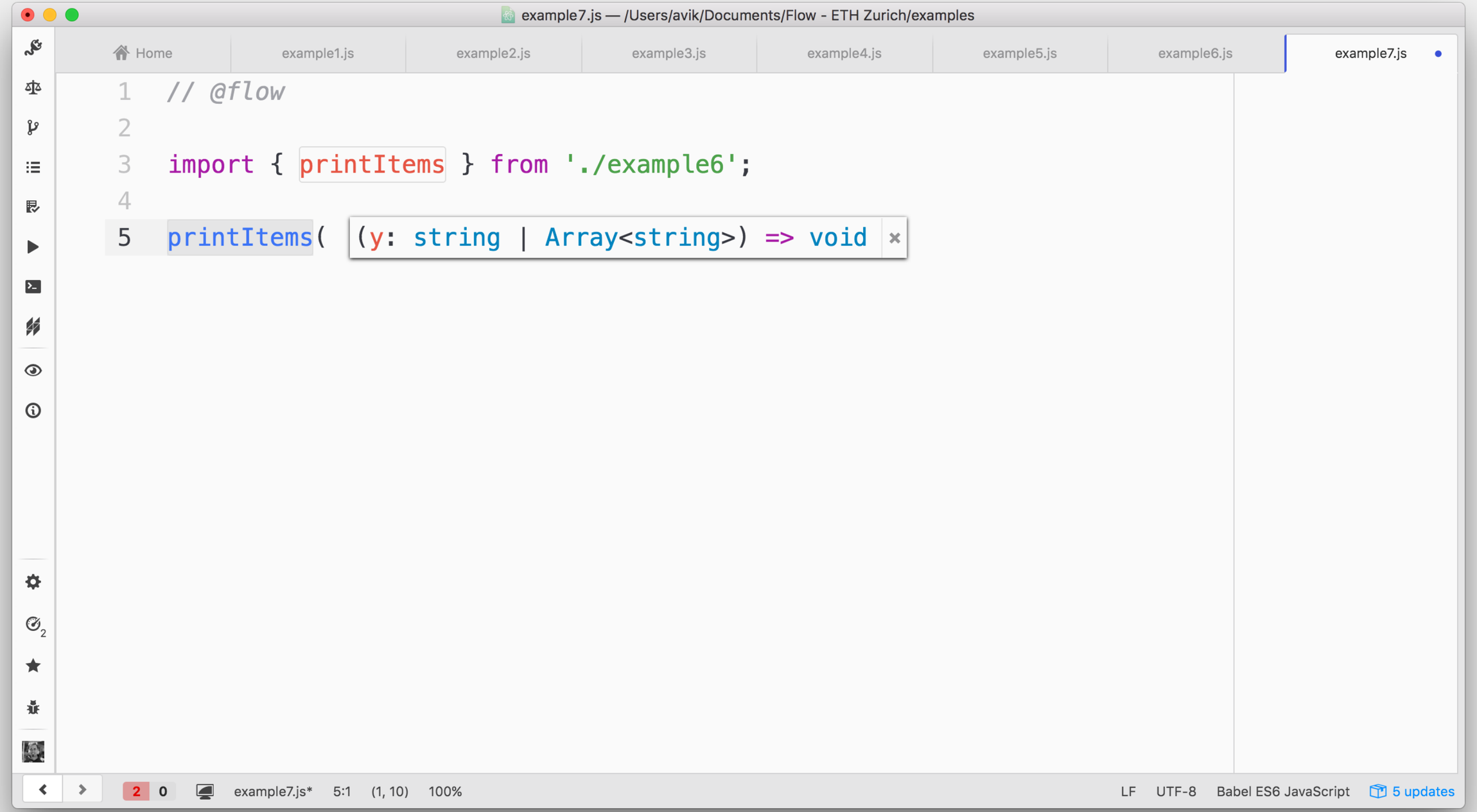
Gah, let’s pin down the API...

example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log(`element ${i} is ${y[i]}`);
11   }
12 }
13
```

2 0 example6.js 7:53 100% LF UTF-8 Babel ES6 JavaScript 5 updates


example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples
Home
example1.js
example2.js
example3.js
example4.js
example5.js
example6.js
example7.js
1 // @flow
2
3 import { printItems } from './example6';
4
5 printItems((y: string | Array<string>) => void
example7.js* 5:1 (1, 10) 100%
LF UTF-8 Babel ES6 JavaScript 5 updates

property variance

object spreads

optional properties

functions as predicates

extensible objects

exact objects

open methods

Wide support for JavaScript idioms

disjoint unions

type parameter variance

tuples

literal types

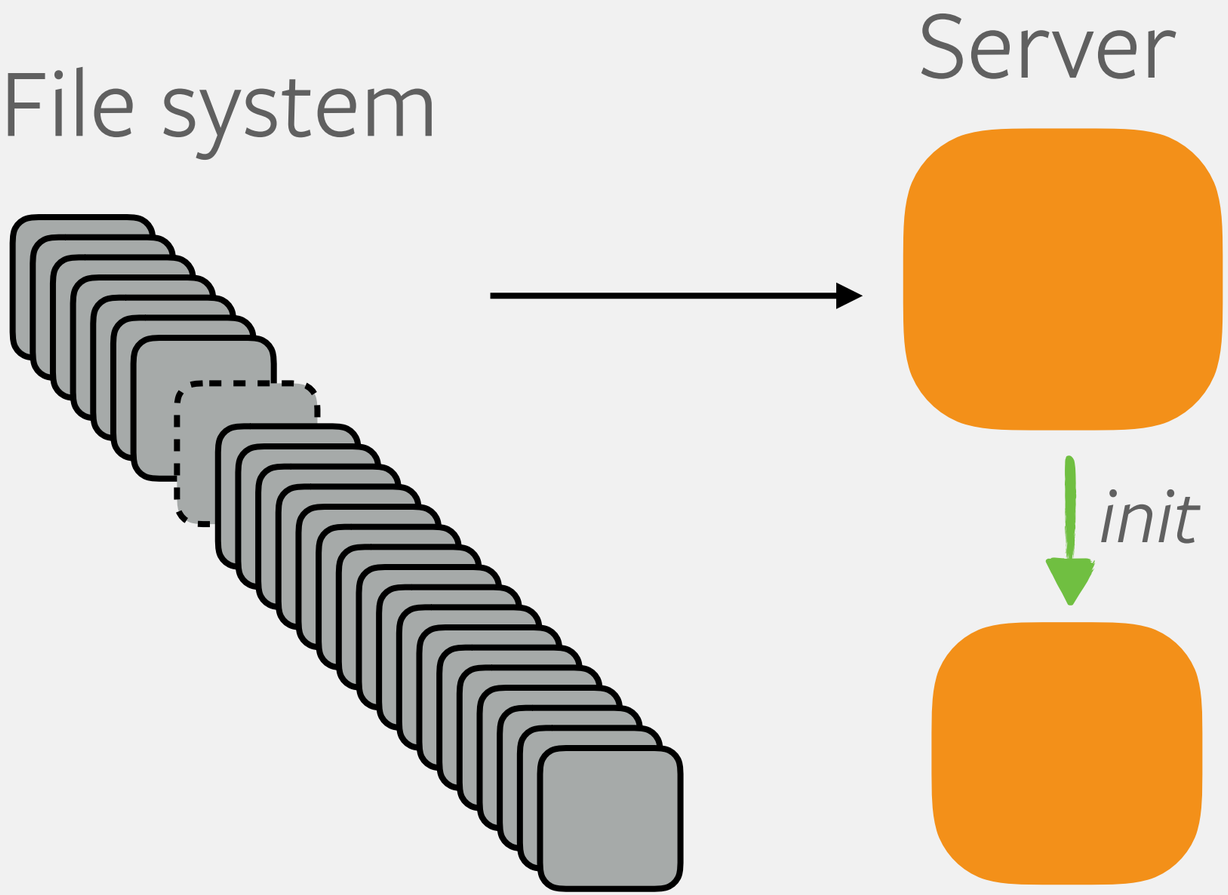
bounded generics

optional parameters

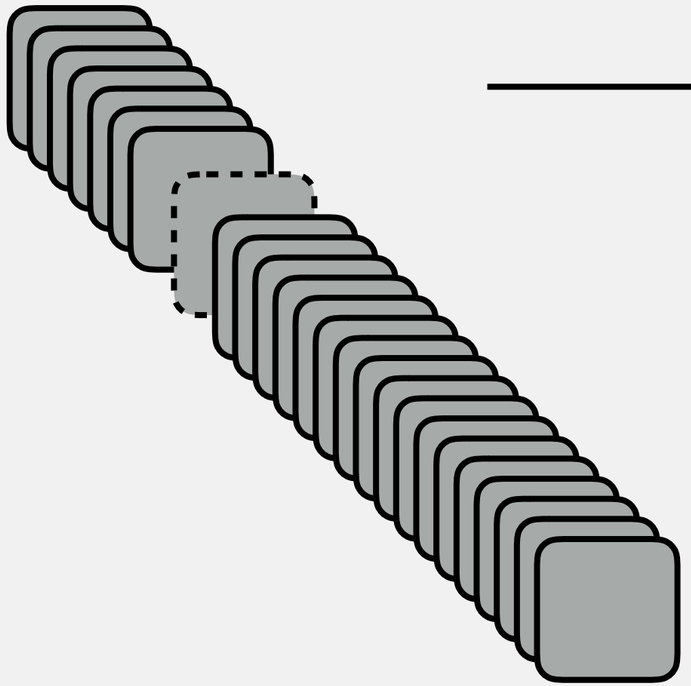
type transforms

function overloading

Design for responsiveness


File system
Server
init

File system

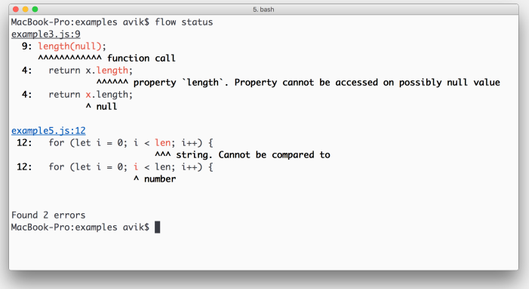


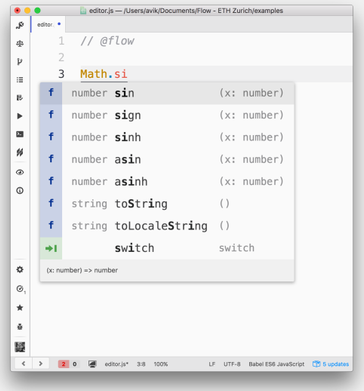
Server

init

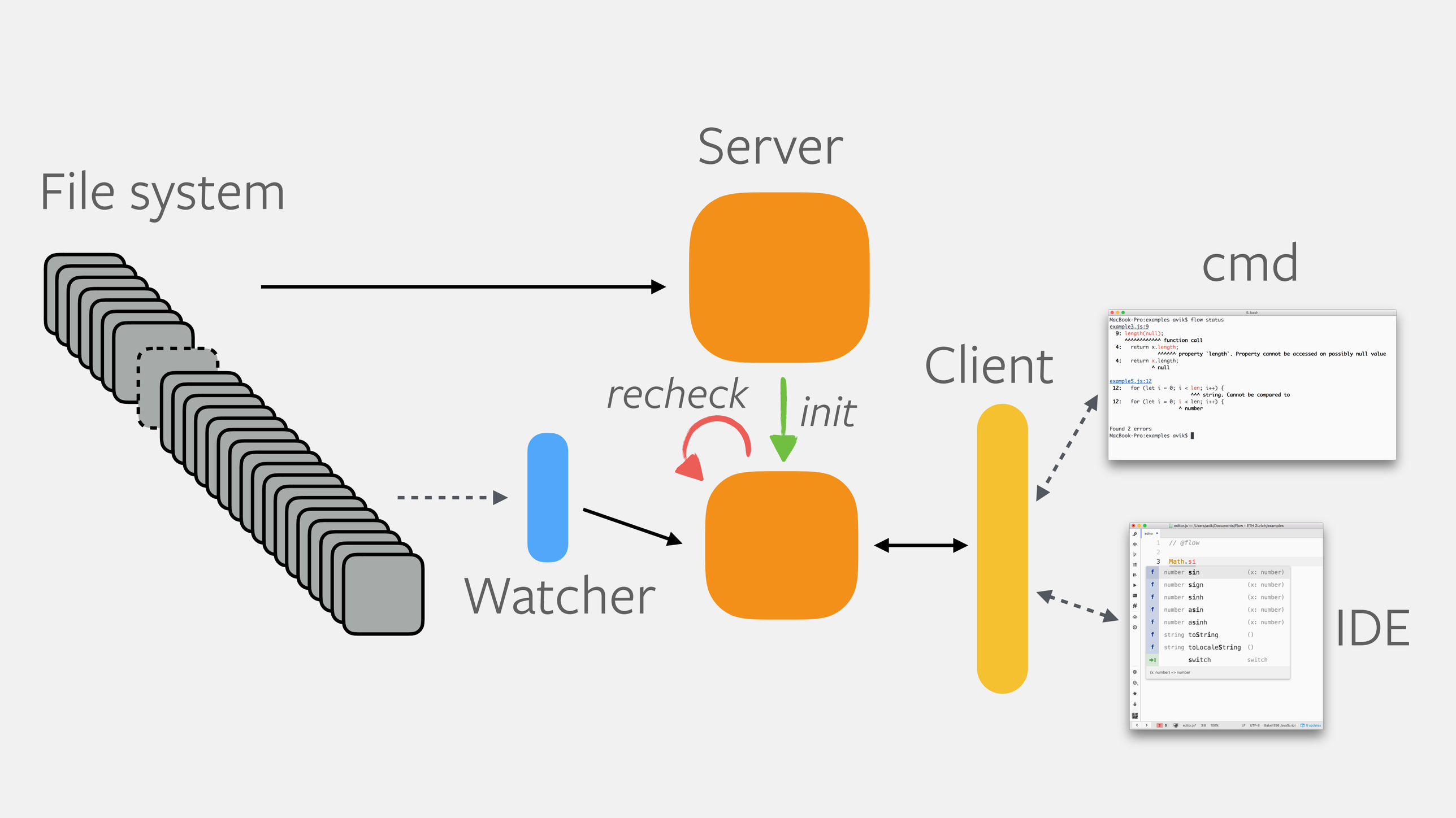
Client

cmd





IDE


File system
Server
recheck
init
Client
cmd
Watcher
IDE
5. bash
MacBook-Pro:examples avik$ flow status
example3.js:9
9: length(null);
^^^^^^^^^^^^ function call
4: return x.length;
^^^^^^ property `length`. Property cannot be accessed on possibly null value
4: return x.length;
^ null
example5.js:12
12: for (let i = 0; i < len; i++) {
^^^ string. Cannot be compared to
12: for (let i = 0; i < len; i++) {
^ number
Found 2 errors
MacBook-Pro:examples avik$
editor.js — /Users/avik/Documents/Flow - ETH Zurich/examples
// @flow
Math.si
number sin (x: number)
number sign (x: number)
number sinh (x: number)
number asin (x: number)
number asinh (x: number)
string toString ()
string toLocaleString ()
switch switch
(x: number) => number

Initializing...

example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

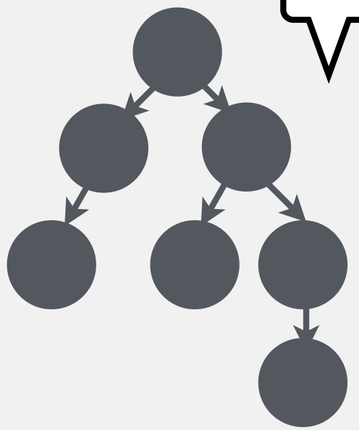
```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

2 0 example6.js 7:53 100% LF UTF-8 Babel ES6 JavaScript 5 updates

example6.js

AST

Parse



example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

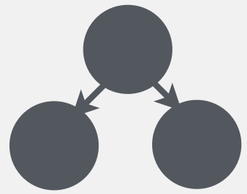
```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems(
```

(y: string | Array<string>) => void

2 0 example7.js* 5:1 (1, 10) 100% LF UTF-8 Babel ES6 JavaScript 5 updates

example7.js

Parse



example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

2 0 example6.js 7:53 100% LF UTF-8 Babel ES6 JavaScript 5 updates

example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

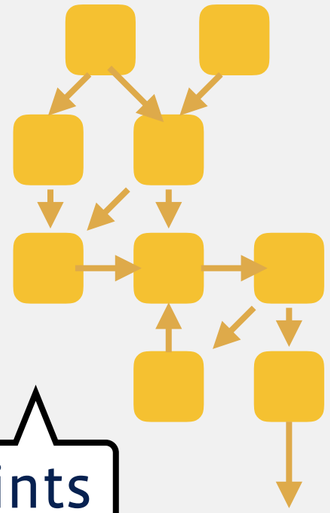
Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems(  (y: string | Array<string>) => void
```

2 0 example7.js* 5:1 (1, 10) 100% LF UTF-8 Babel ES6 JavaScript 5 updates

example6.js

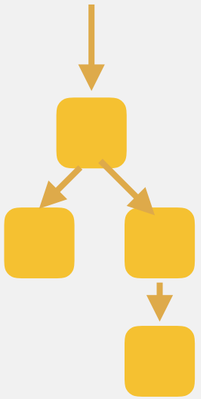
Compile



Constraints

example7.js

Compile



example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

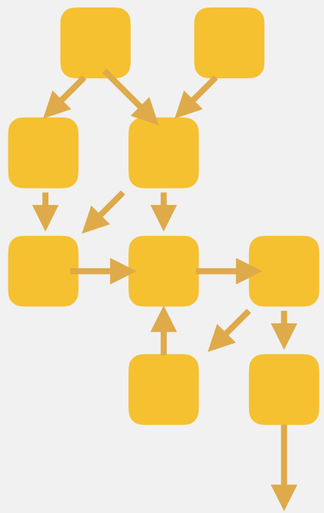
```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems(  (y: string | Array<string>) => void
```

example7.js* 5:1 (1, 10) 100% LF UTF-8 Babel ES6 JavaScript 5 updates

example6.js

Link

Constraints

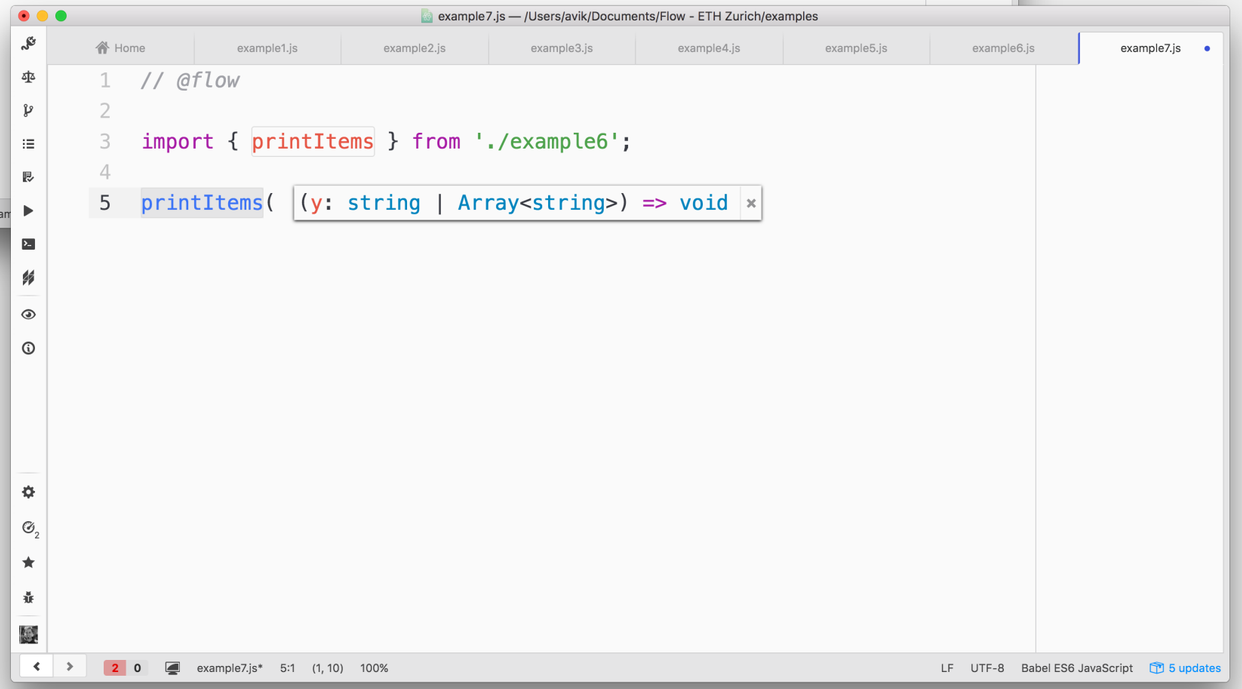


example7.js

example6.js

Link

Signature



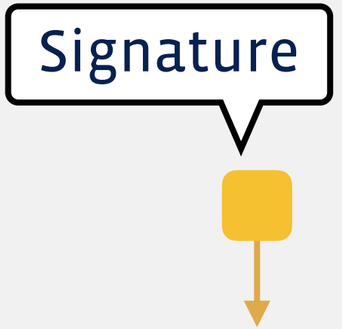
example7.js

example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

example6.js



example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

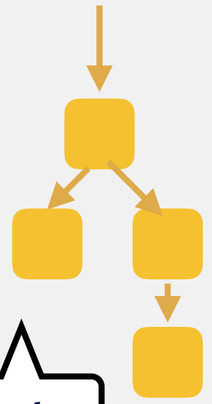
Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems(  (y: string | Array<string>) => void
```

example7.js* 5:1 (1, 10) 100% LF UTF-8 Babel ES6 JavaScript 5 updates

example7.js

Link



Constraints

example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

example6.js

example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

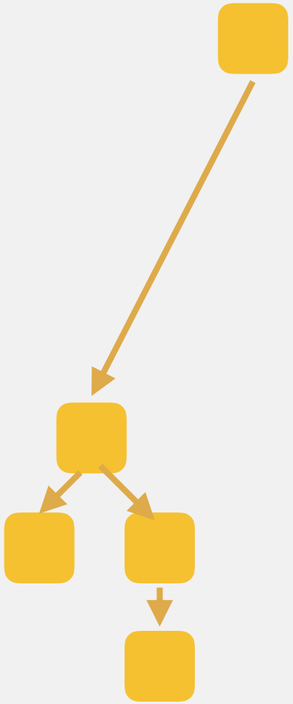
Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems(  (y: string | Array<string>) => void
```

example7.js

example7.js* 5:1 (1, 10) 100% LF UTF-8 Babel ES6 JavaScript 5 updates

Link



Rechecking...

example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems(  (y: string | Array<string>) => void
```

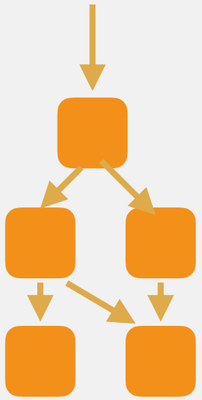
EDIT

example6.js

example7.js

Parse

Compile



example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

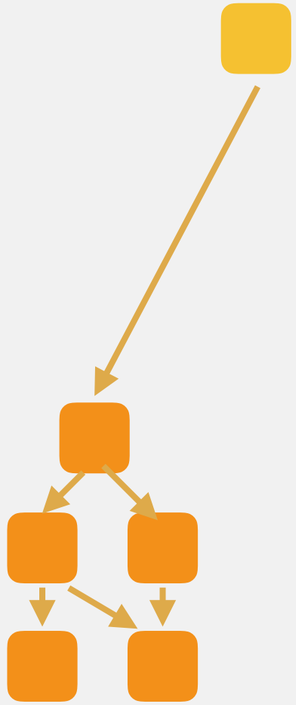
```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems(  (y: string | Array<string>) => void
```

EDIT

example6.js

example7.js

Link



example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

EDIT

example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

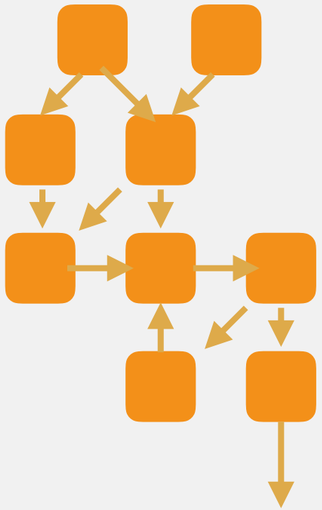
```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems( (y: string | Array<string>) => void
```

2 0 example7.js* 5:1 (1, 10) 100% LF UTF-8 Babel ES6 JavaScript 5 updates

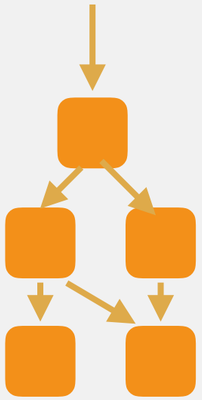
example6.js

Parse

Compile



example7.js



example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

EDIT

example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

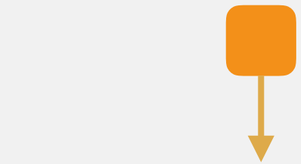
```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems( (y: string | Array<string>) => void
```

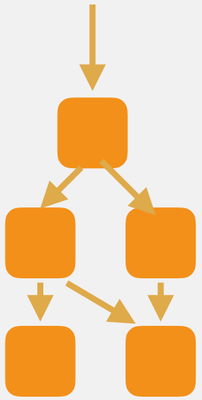
2 0 example7.js* 5:1 (1, 10) 100% LF UTF-8 Babel ES6 JavaScript 5 updates

example6.js

Link

example7.js





example6.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

```
1  // @flow
2
3  function length(x) {
4    return x.length;
5  }
6
7  export function printItems(y: string | Array<string>) {
8    const len = length(y);
9    for (let i = 0; i < len; i++) {
10     console.log("element {i} is {y[i]}");
11   }
12 }
13
```

EDIT

example7.js — /Users/avik/Documents/Flow - ETH Zurich/examples

Home | example1.js | example2.js | example3.js | example4.js | example5.js | example6.js | example7.js

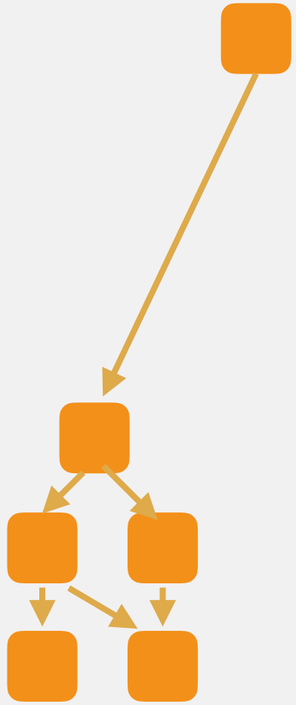
```
1  // @flow
2
3  import { printItems } from './example6';
4
5  printItems(  (y: string | Array<string>) => void
```

2 0 example7.js* 5:1 (1, 10) 100% LF UTF-8 Babel ES6 JavaScript 5 updates

example6.js

example7.js

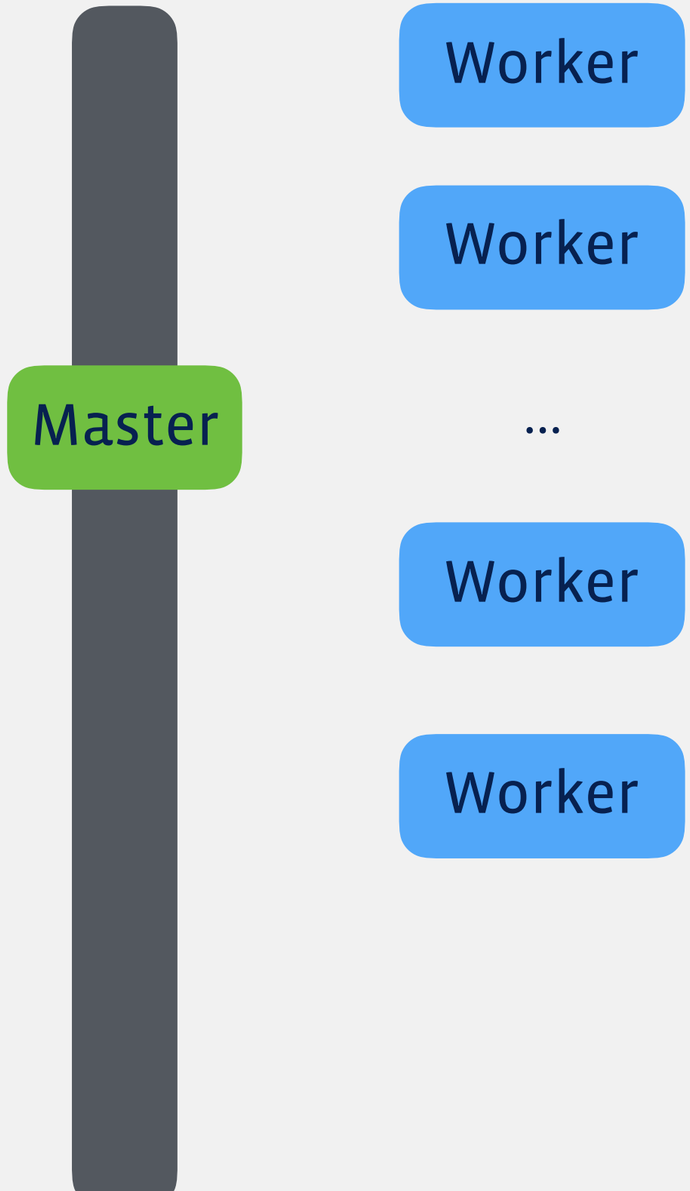
Link



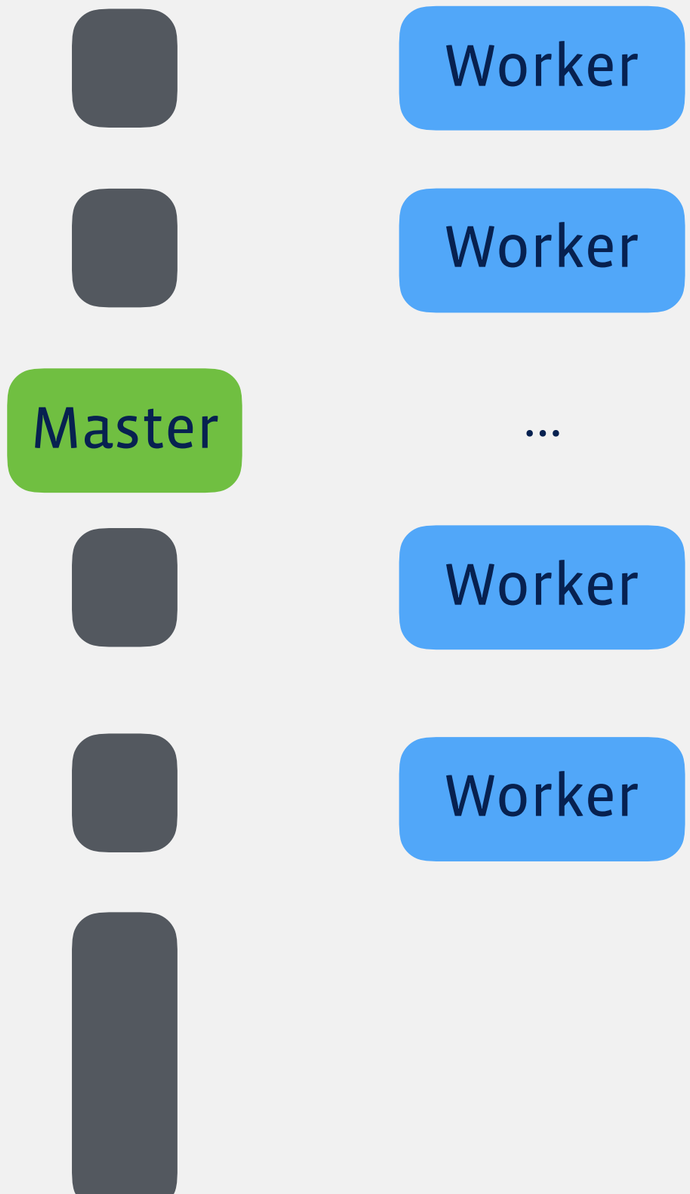
Parallelize everything!

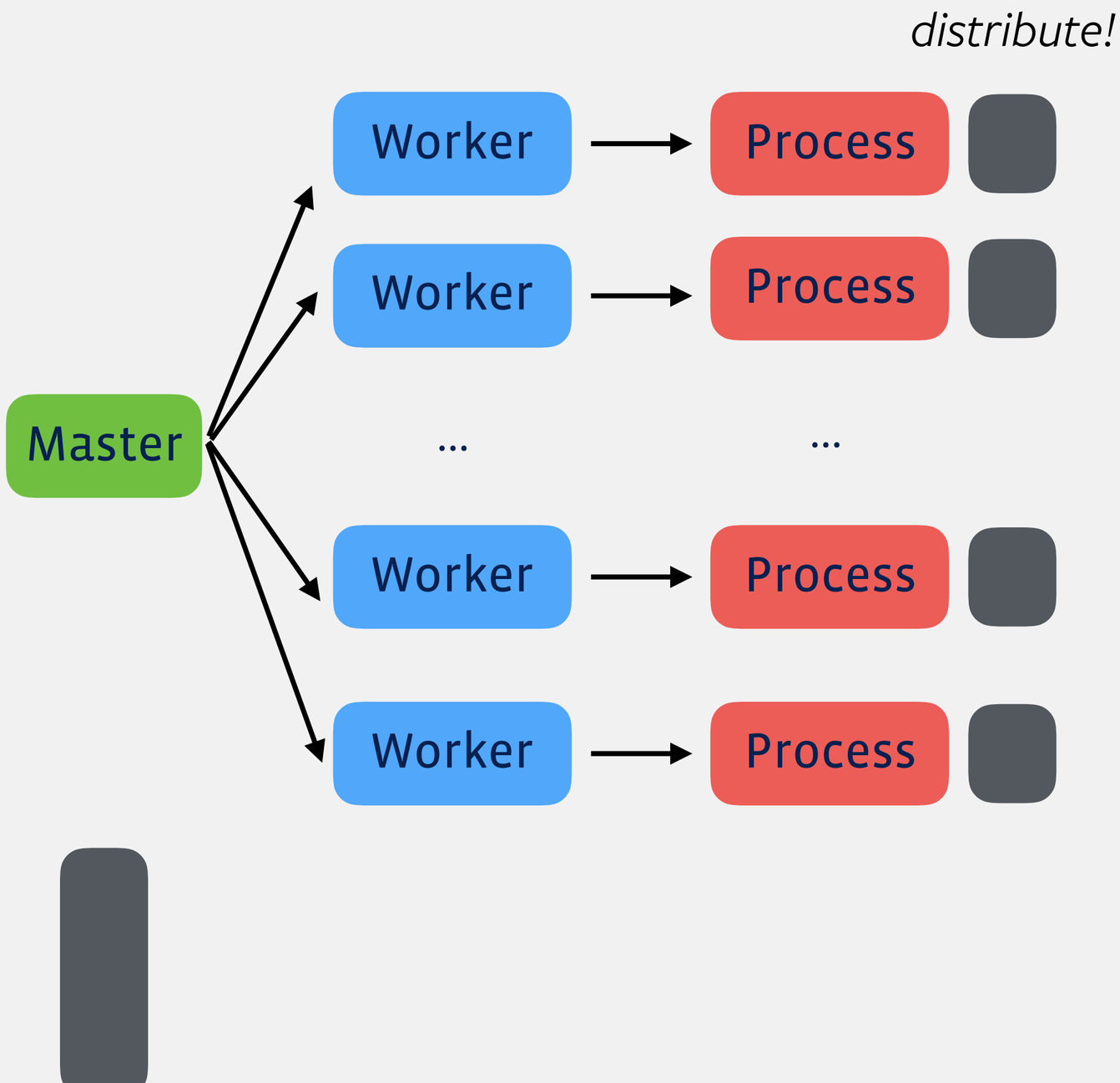
Infrastructure

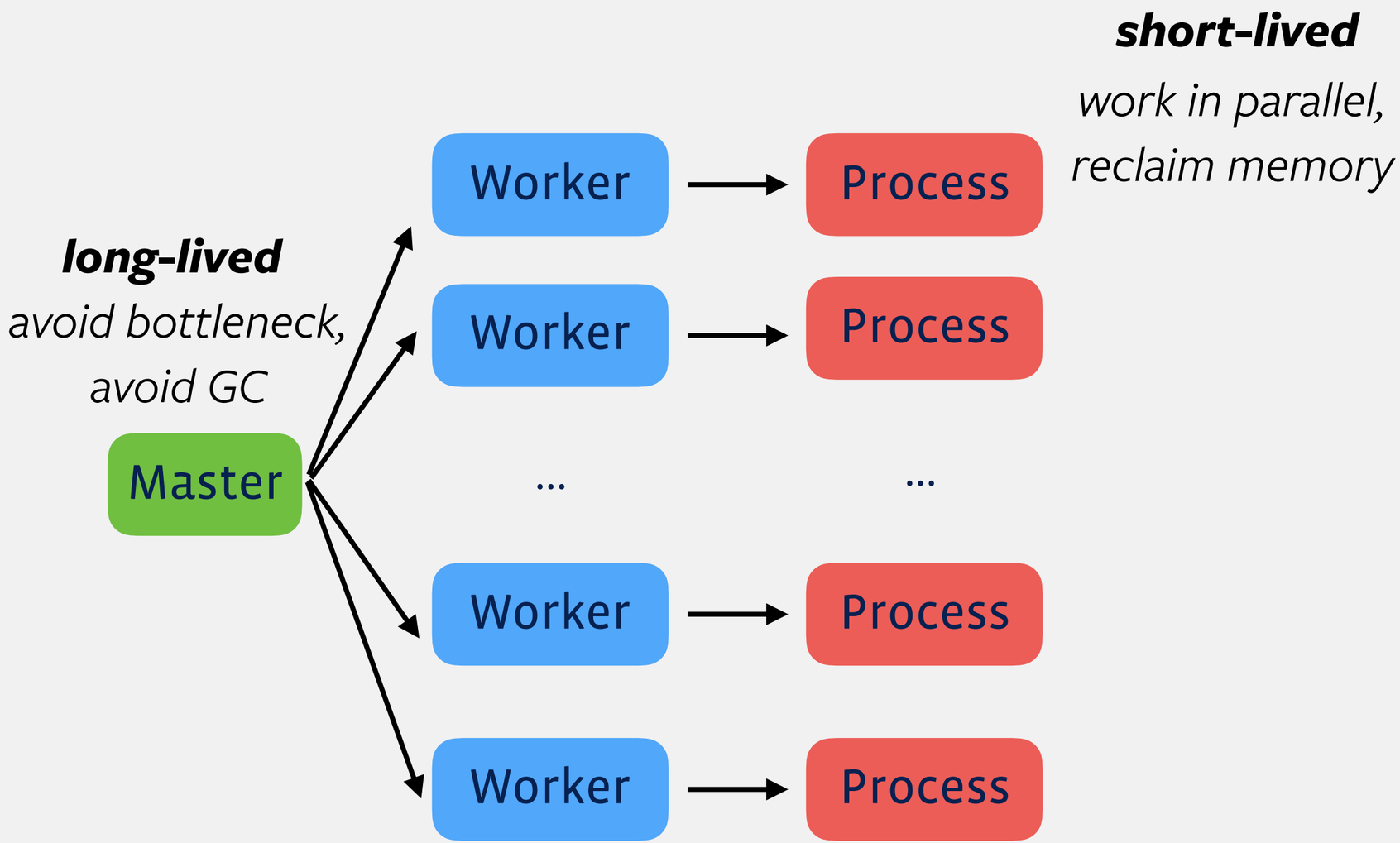
large number of jobs

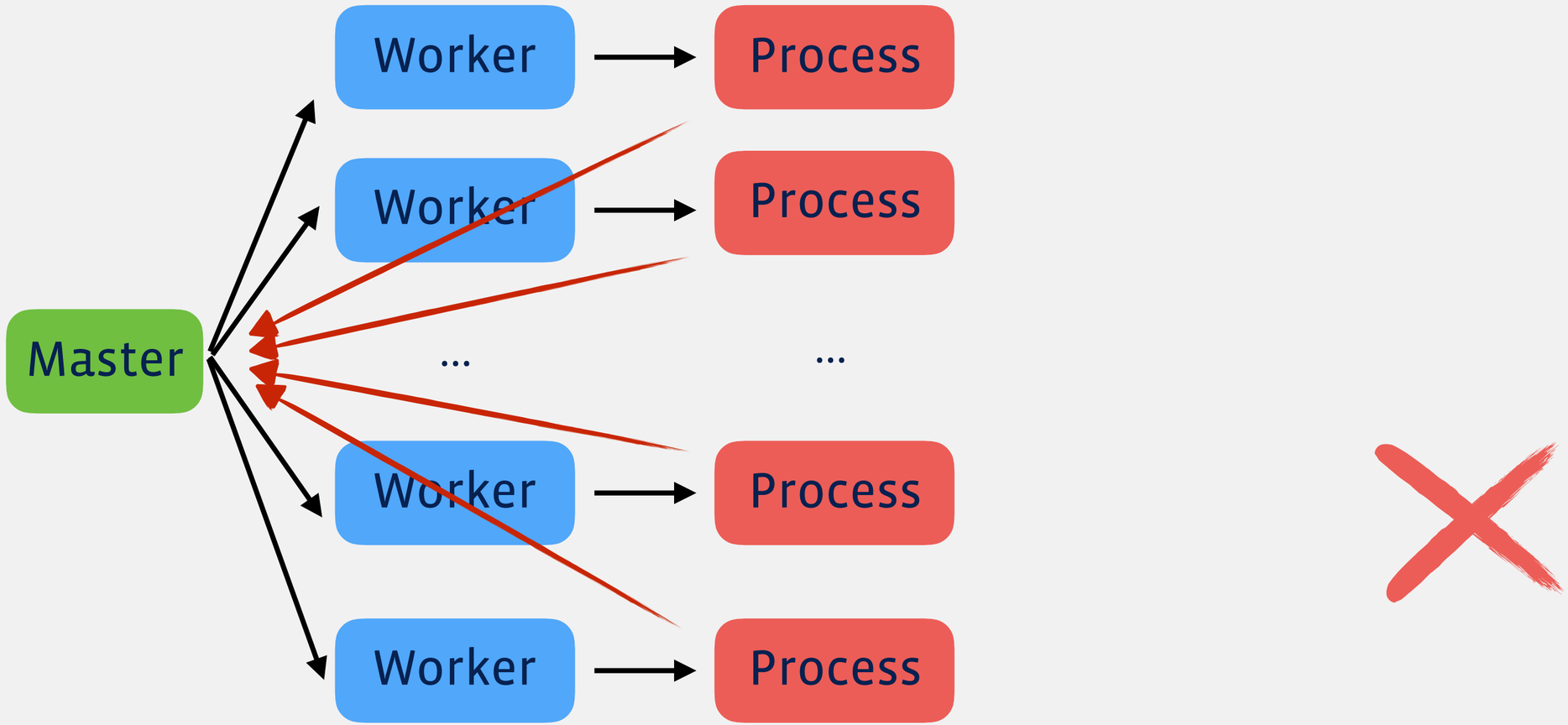


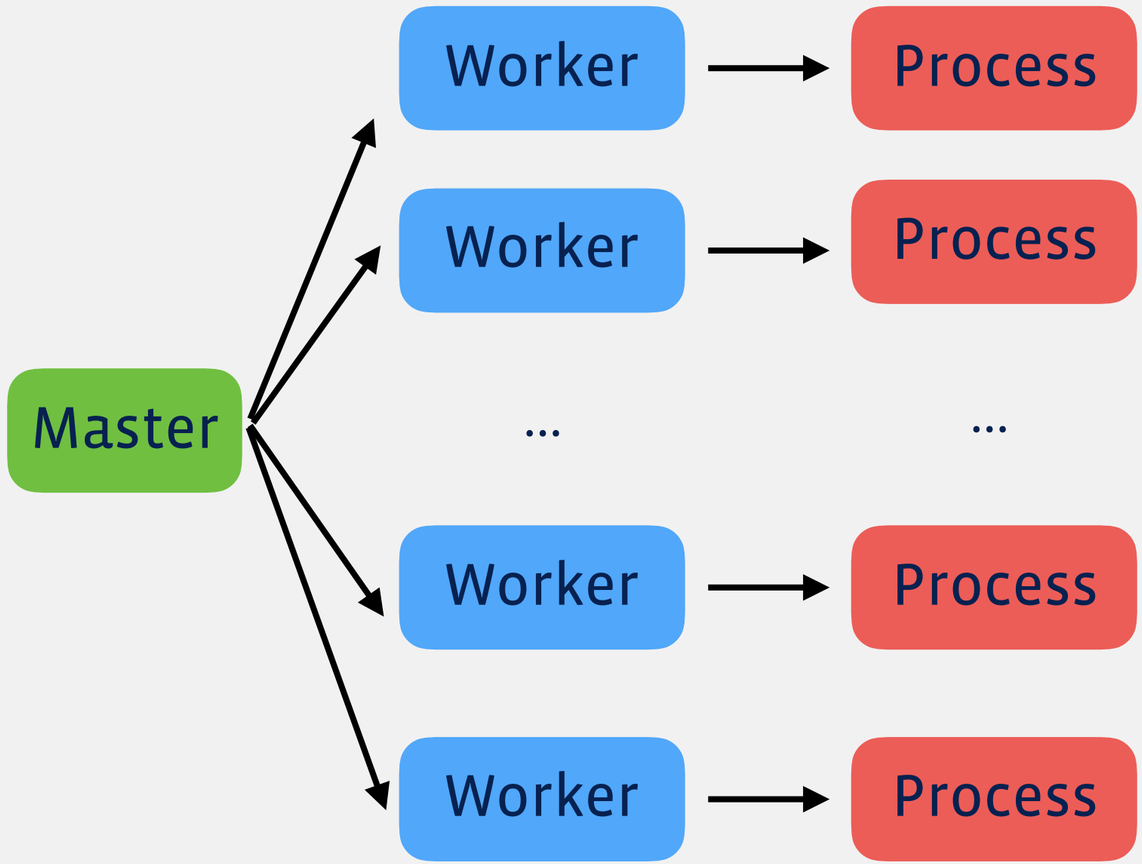
divide into buckets

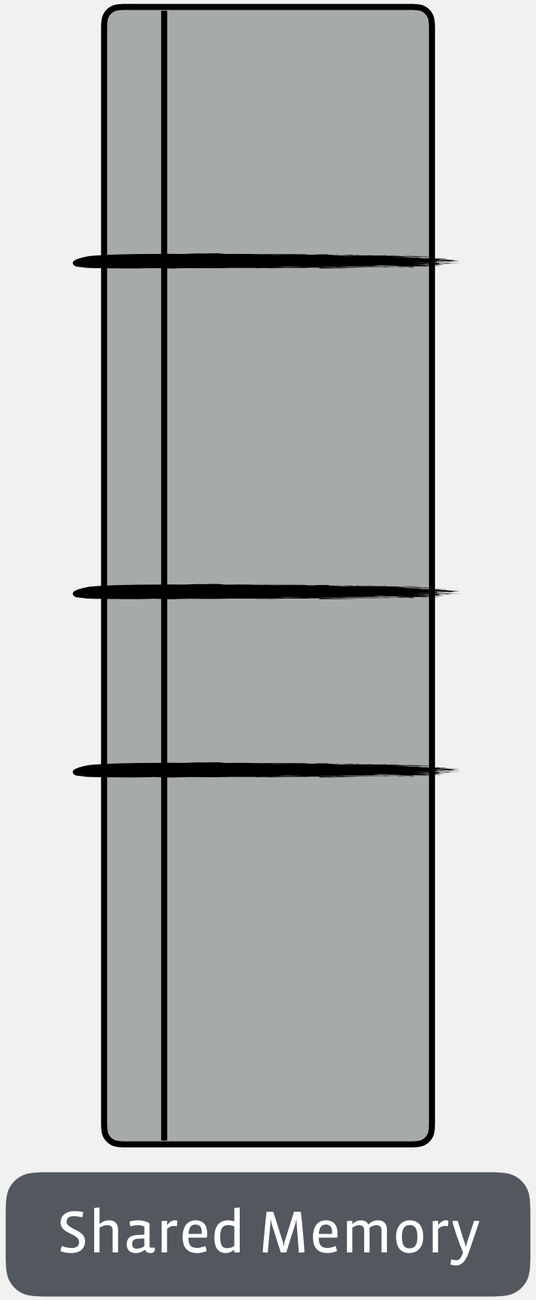


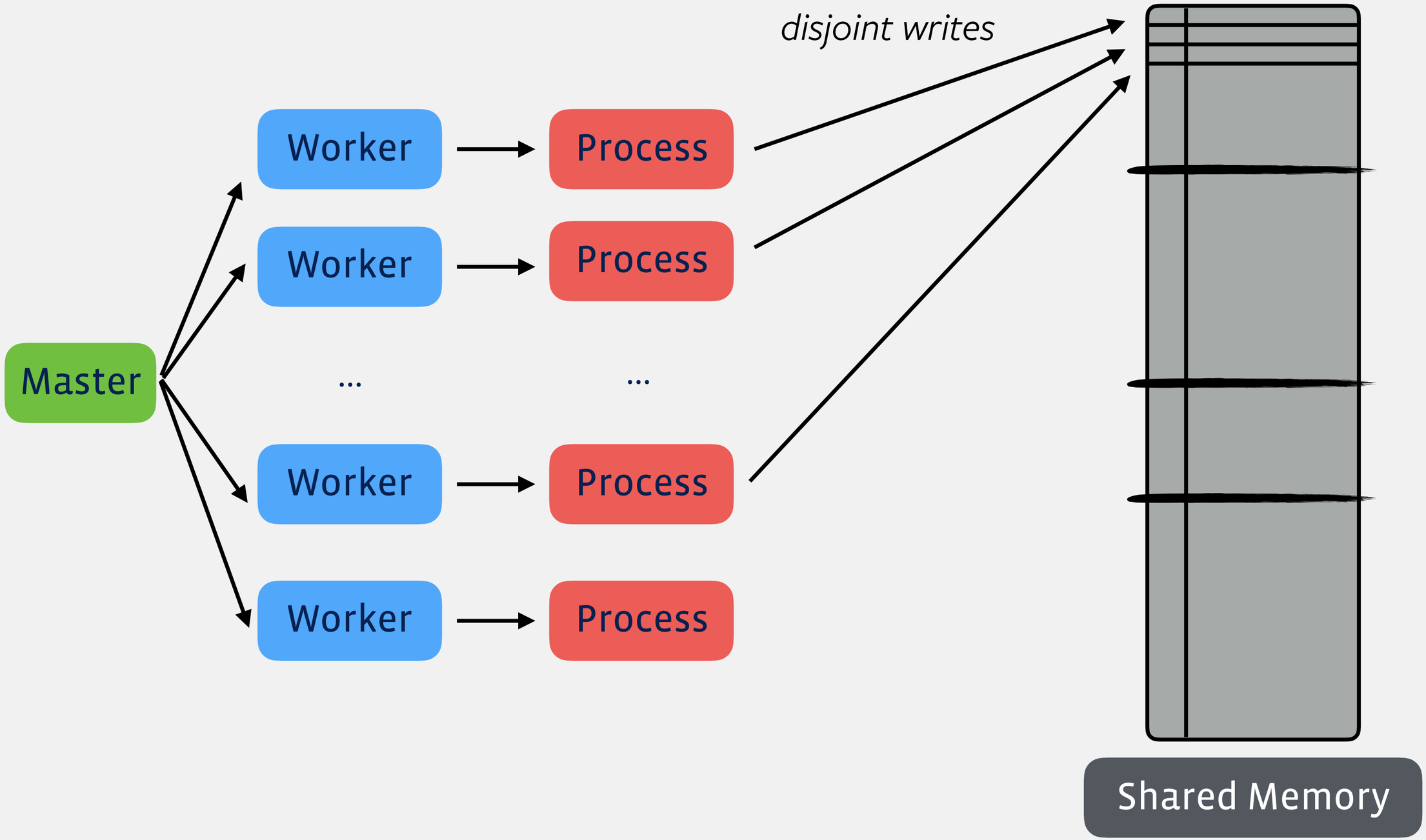

distribute!
Worker
Process
Worker
Process
Master
...
...
Worker
Process
Worker
Process


short-lived
work in parallel,
reclaim memory
long-lived
avoid bottleneck,
avoid GC
Master
Worker
Process
Worker
Process
...
...
Worker
Process
Worker
Process


Master
Worker
Worker
Worker
Worker
Process
Process
Process
Process
...
...


Master
Worker
Process
Worker
Process
...
...
Worker
Process
Worker
Process


Shared Memory


disjoint writes
Master
Worker
Worker
...
Worker
Worker
Process
Process
...
Process
Process
Shared Memory

Worker

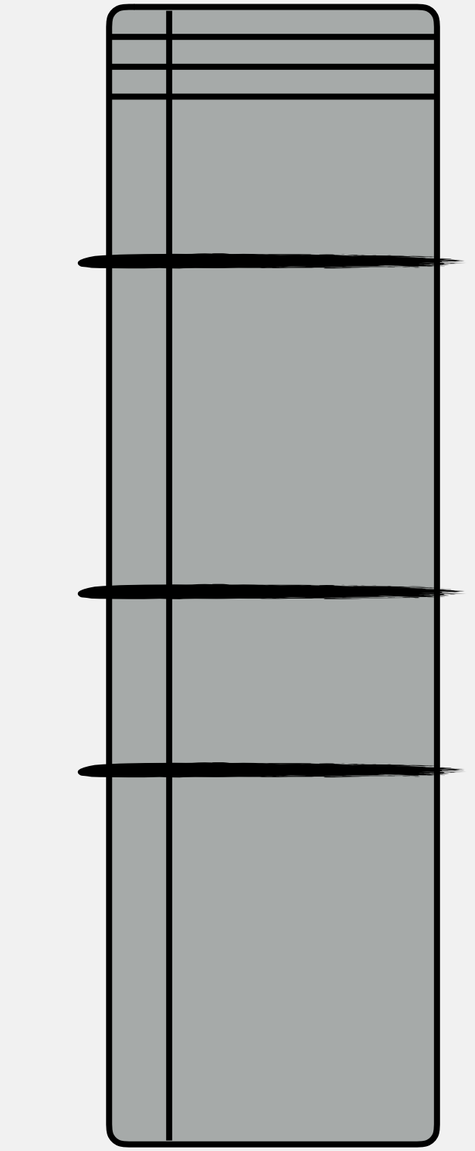
Worker

Master

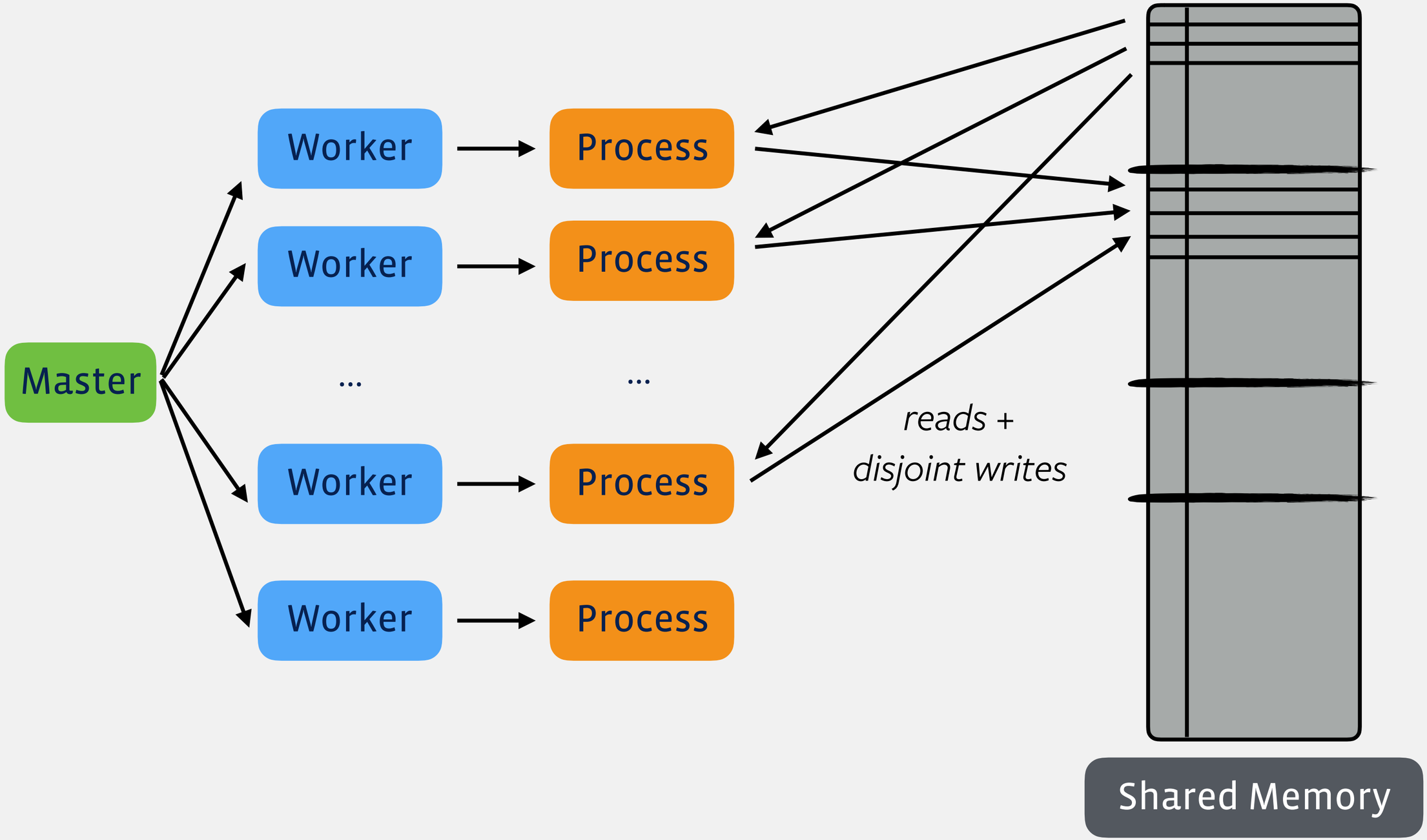
...

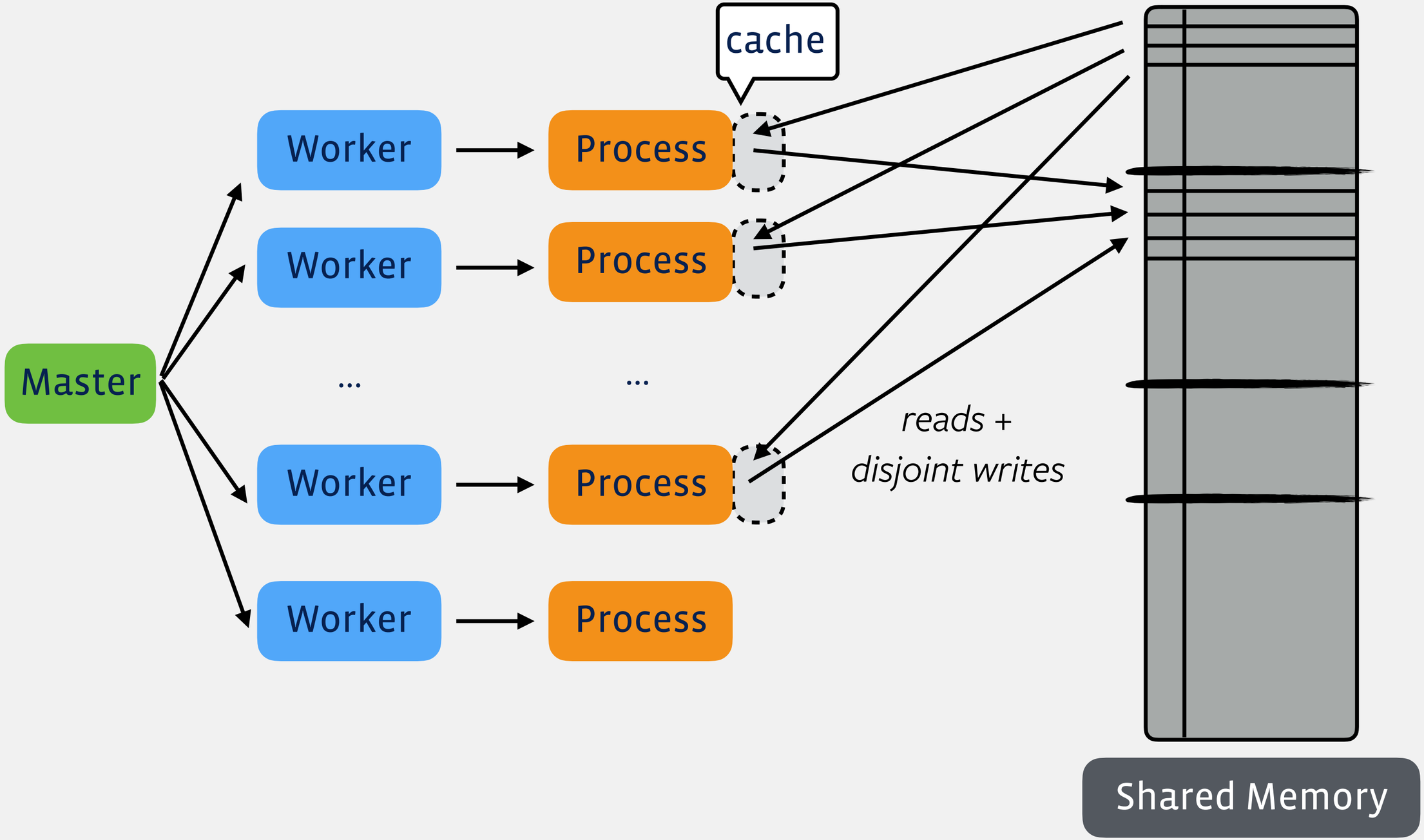
Worker

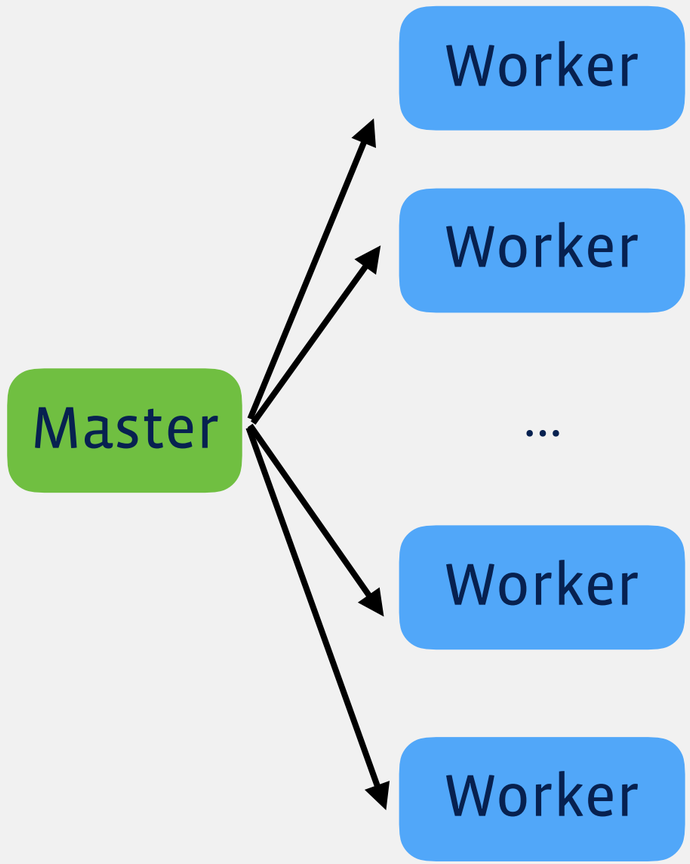
Worker

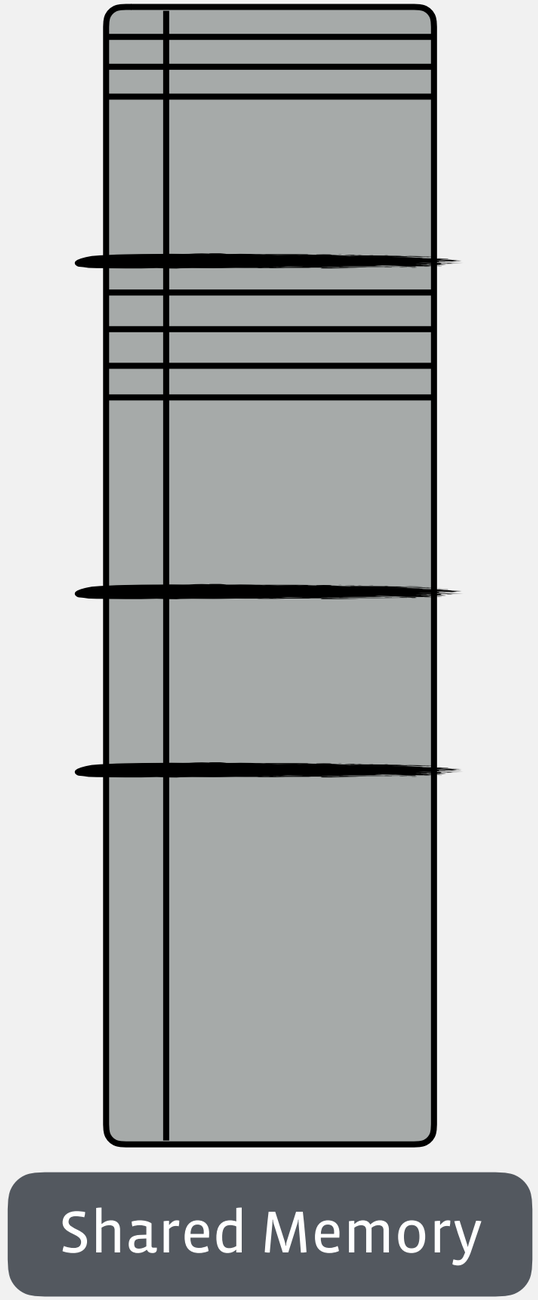


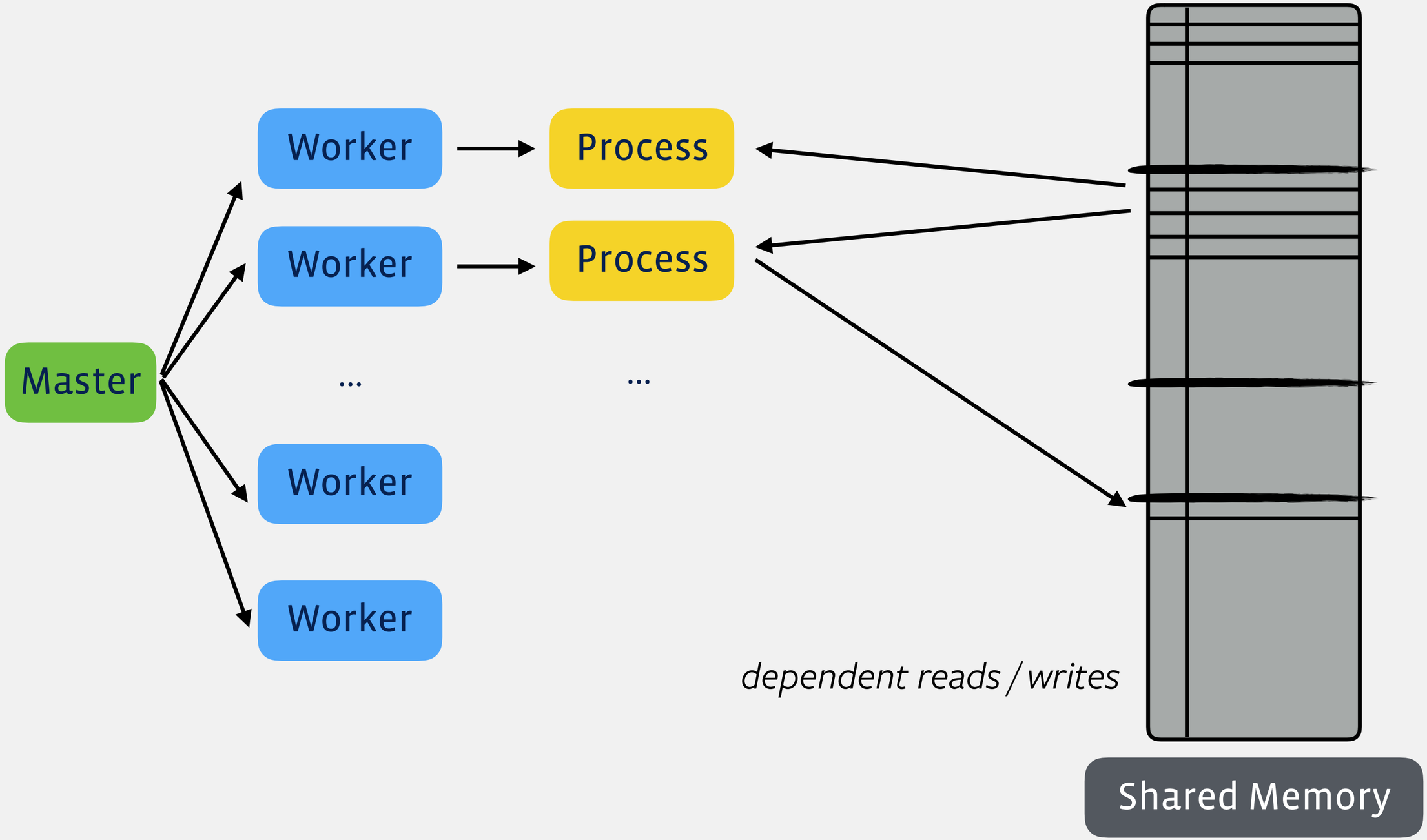
Shared Memory

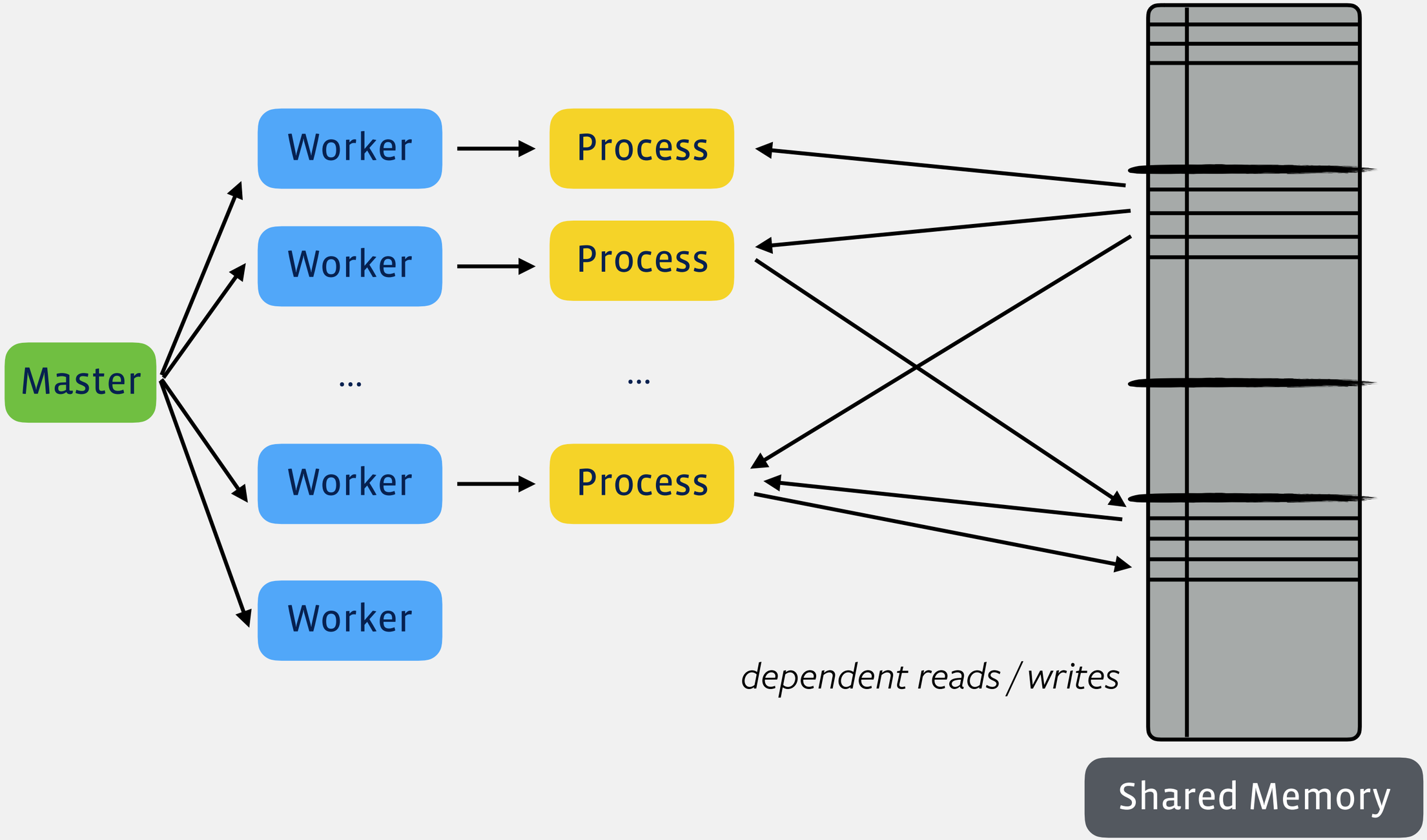

Master
Worker
Worker
...
Worker
Worker
Process
Process
...
Process
Process
reads +
disjoint writes
Shared Memory

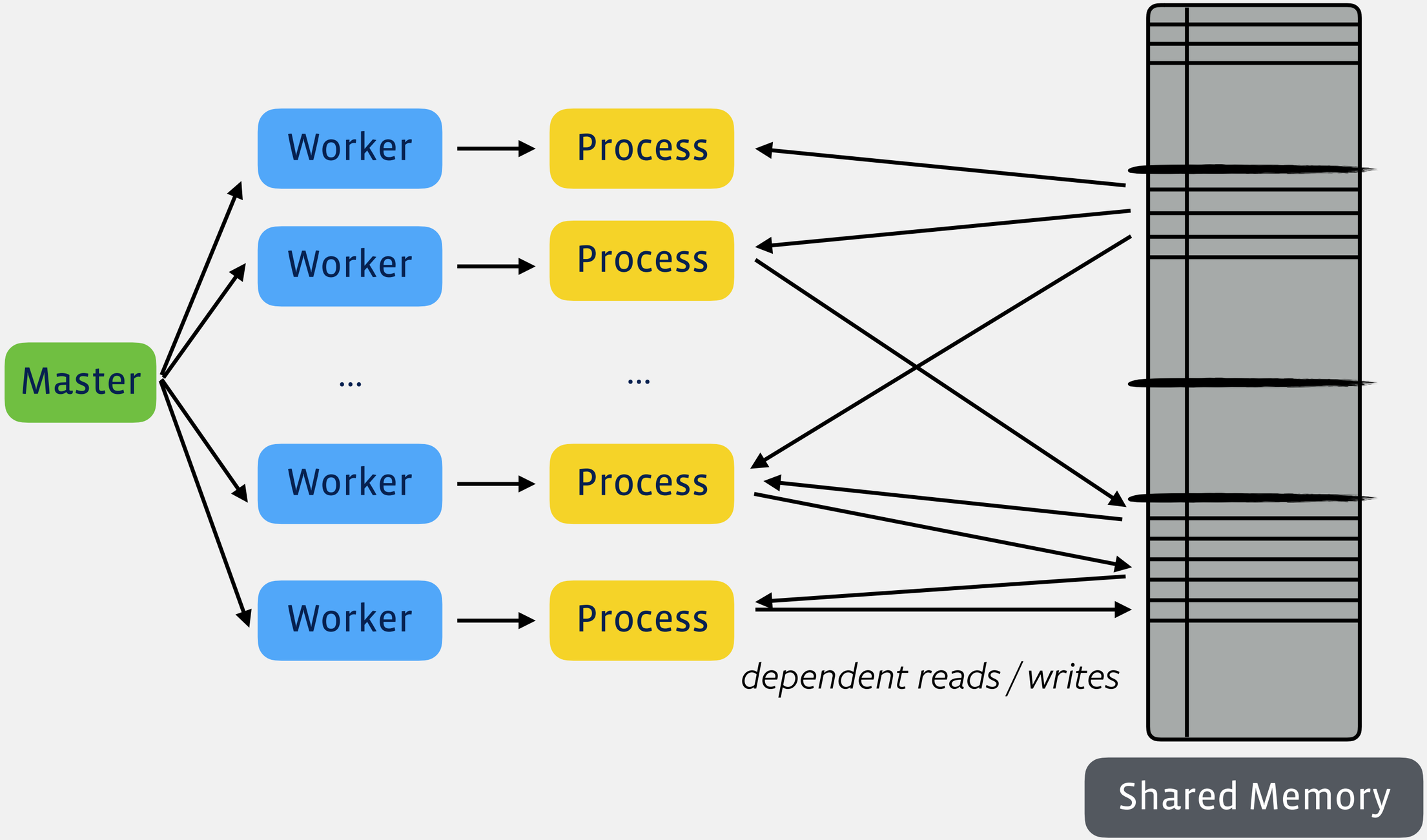

cache
Worker
Process
Worker
Process
Master
...
...
Worker
Process
Worker
Process
reads +
disjoint writes
Shared Memory


Master
Worker
Worker
...
Worker
Worker

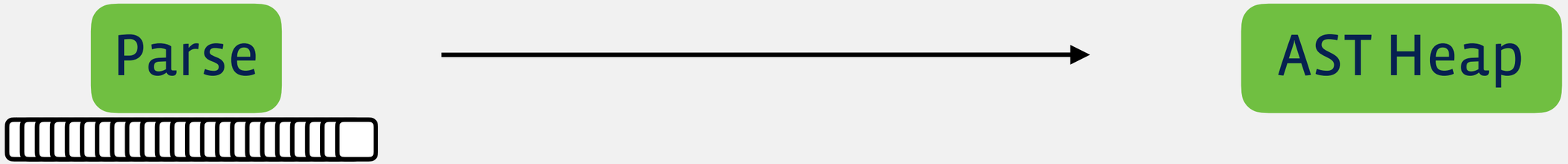

Shared Memory

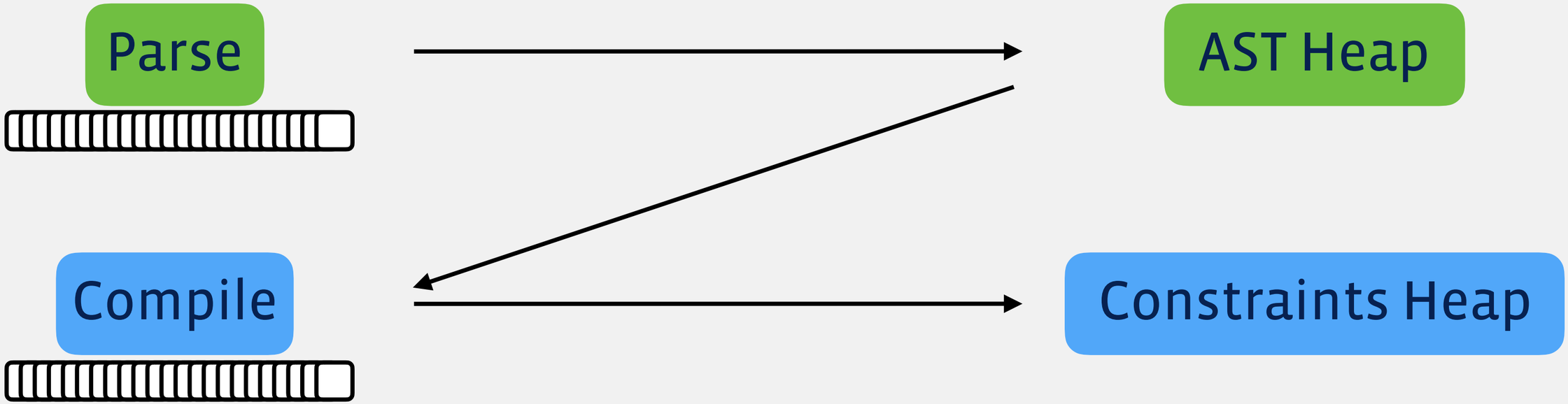

Worker
Process
Worker
Process
Master
...
...
Worker
Worker
dependent reads / writes
Shared Memory

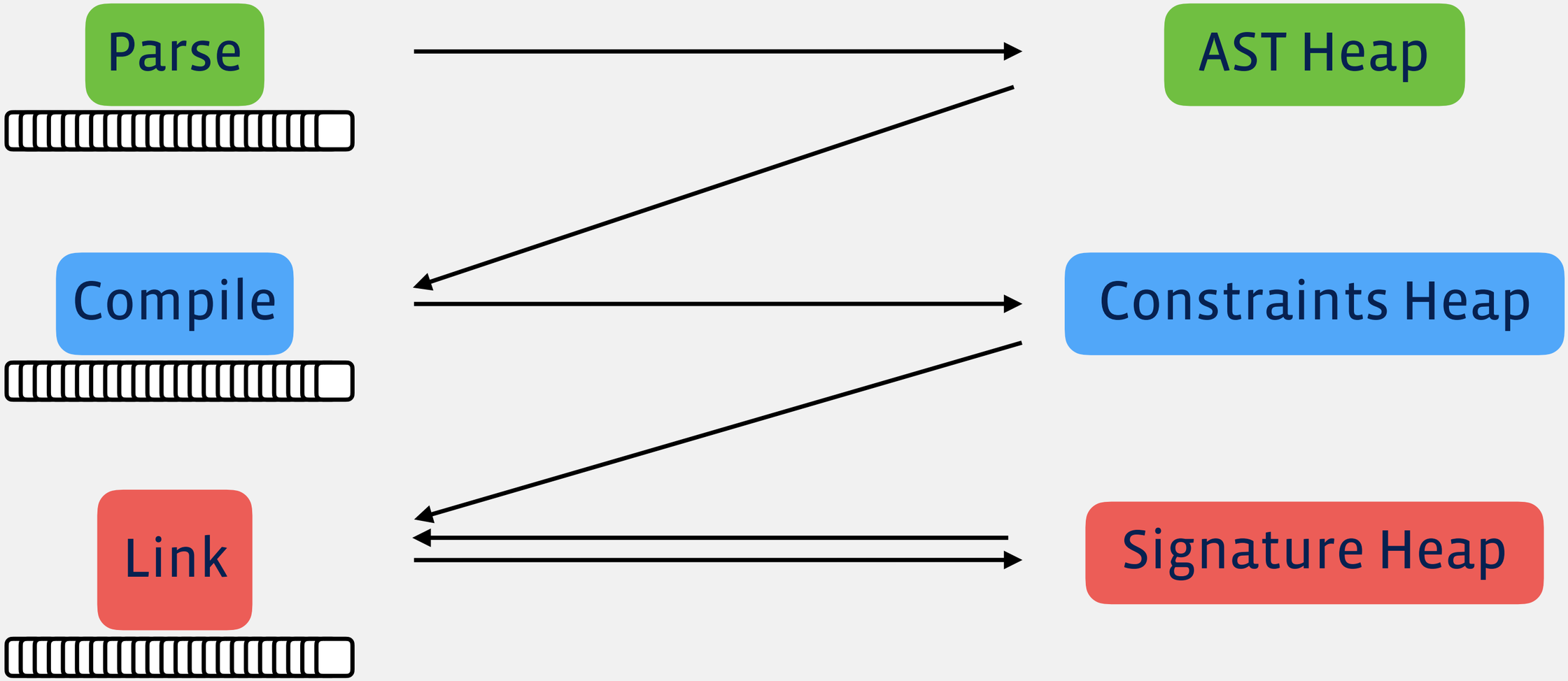

Master
Worker
Worker
...
Worker
Worker
Process
Process
...
Process
dependent reads / writes
Shared Memory

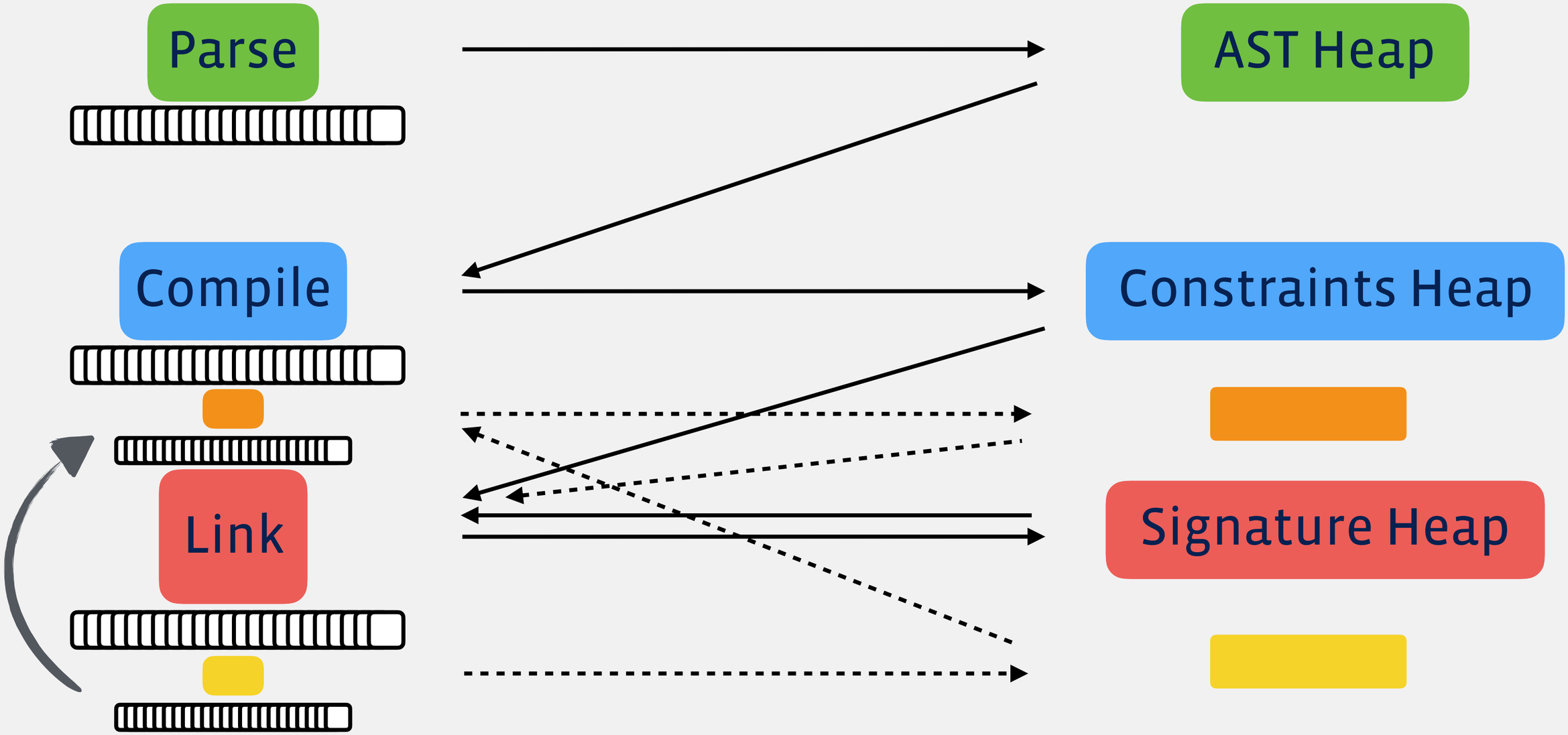

Master
Worker
Worker
...
Worker
Worker
Process
Process
...
Process
Process
dependent reads / writes
Shared Memory

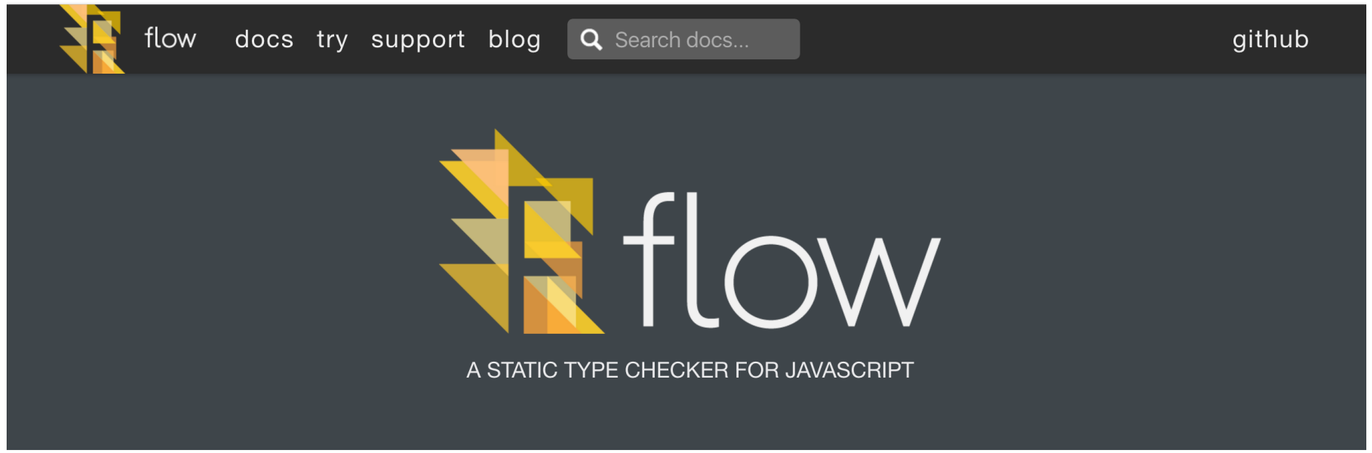
Putting everything together...


Parse
AST Heap


Parse
AST Heap
Compile
Constraints Heap


Parse
Compile
Link
AST Heap
Constraints Heap
Signature Heap


Parse
Compile
Link
AST Heap
Constraints Heap
Signature Heap



Type Inference

Flow uses type inference to find bugs even without type annotations. It precisely tracks the types of variables as they flow through your program.

Idiomatic JS

Flow is designed for JavaScript programmers. It understands common JavaScript idioms and very dynamic code.

Realtime Feedback

Flow incrementally rechecks your changes as you work, preserving the fast feedback cycle of developing plain JavaScript.

Flow can catch common bugs in JavaScript programs before they run, including:

- silent type conversions,
- `null` dereferences,
- and the dreaded `undefined is not a function`.

```
1  // @flow
2  function foo(x) {
3    return x * 10;
4  }
5  foo('Hello, world!');
```

show Flow output

☑ Fast

☑ Precise

Are you in  flow?


flow