Flow

CS-554 – WEB PROGRAMMING

What is Flow?

What is Flow?

Flow is a static type checker for JavaScript; it relies on additional annotations to your JavaScript code to enforce a set of rules.

- https://flow.org/en/
- https://flow.org/try/

Flow is maintained by Facebook, and as such is easily integrated into their app development tools (such as create-react-app)

How is Flow different than TypeScript?

Flow is different from TypeScript, in that it is not used to generate valid JavaScript, but rather is a series of annotations that are stripped out during the build process entirely, leaving your JavaScript in tact.

When maintenance for Flow ends, or when a developer wants to remove Flow from their app, all that must be done is that the Flow source files are checked by Flow and have their annotations removed; the resulting files will have no changes except for the annotations removal, and those files can now be used as source code.

Engineering Decisions Behind Flow

Flow is not a complicated system like TypeScript and attempts to balance **precision** and **speed** of for the developer

- It focuses on using a modular system to keep track of precisely **what** is being affected when you change your code (precision)
- It is able to detect changes quickly by only rechecking rules for things that have changed.

Flow was not designed to be a be-all-end-all tooling to avoid all errors, but rather as a "good-enough, fast-enough, easy-enough" tool

- It's speed is its real key; by being fast and unobtrusive, developers are more likely to use it effectively.

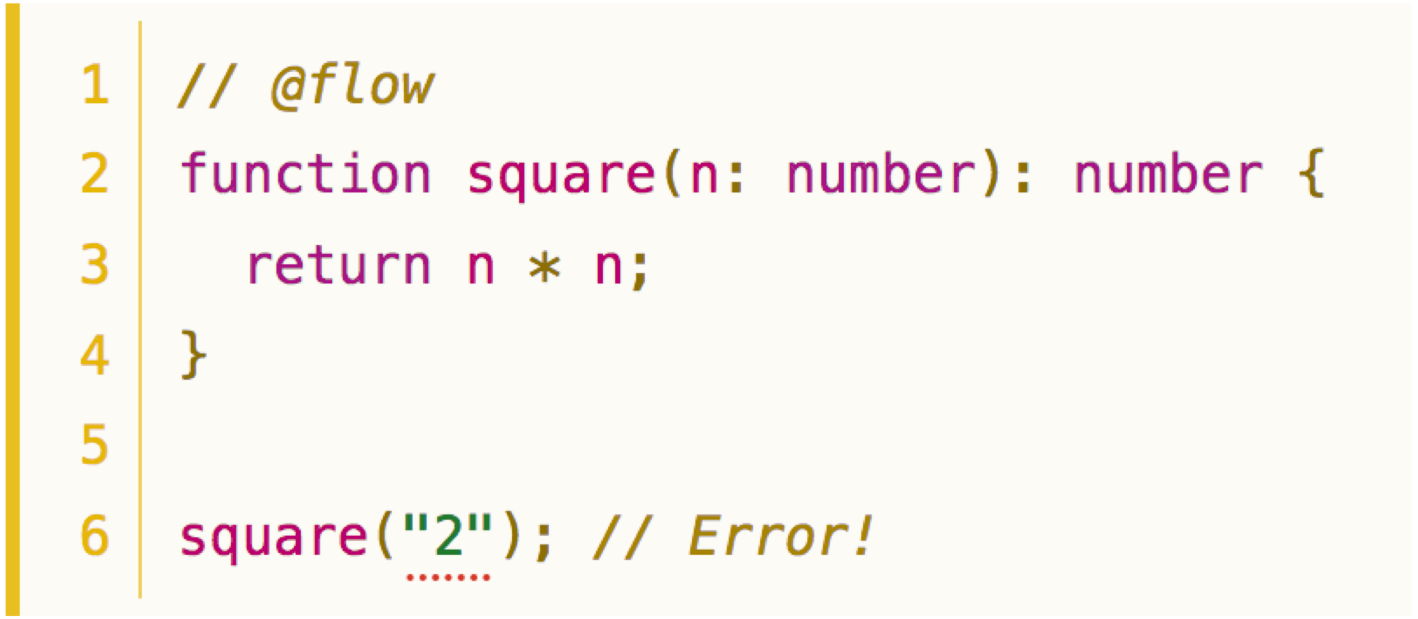
For in depth details about its engine, see the *Type System* area of the documentation

- https://flow.org/en/docs/lang/

Using Flow

In order to have Flow pickup a file for analysis, you must provide a comment in the file that annotates the file for Flow.

For our demonstration purposes, we will be using **flow-remove-types** to pick up files from one directory, run them through Flow, and be copied to a different directory.



```
// @flow
function square(n: number): number {
  return n * n;
}

square("2"); // Error!
```

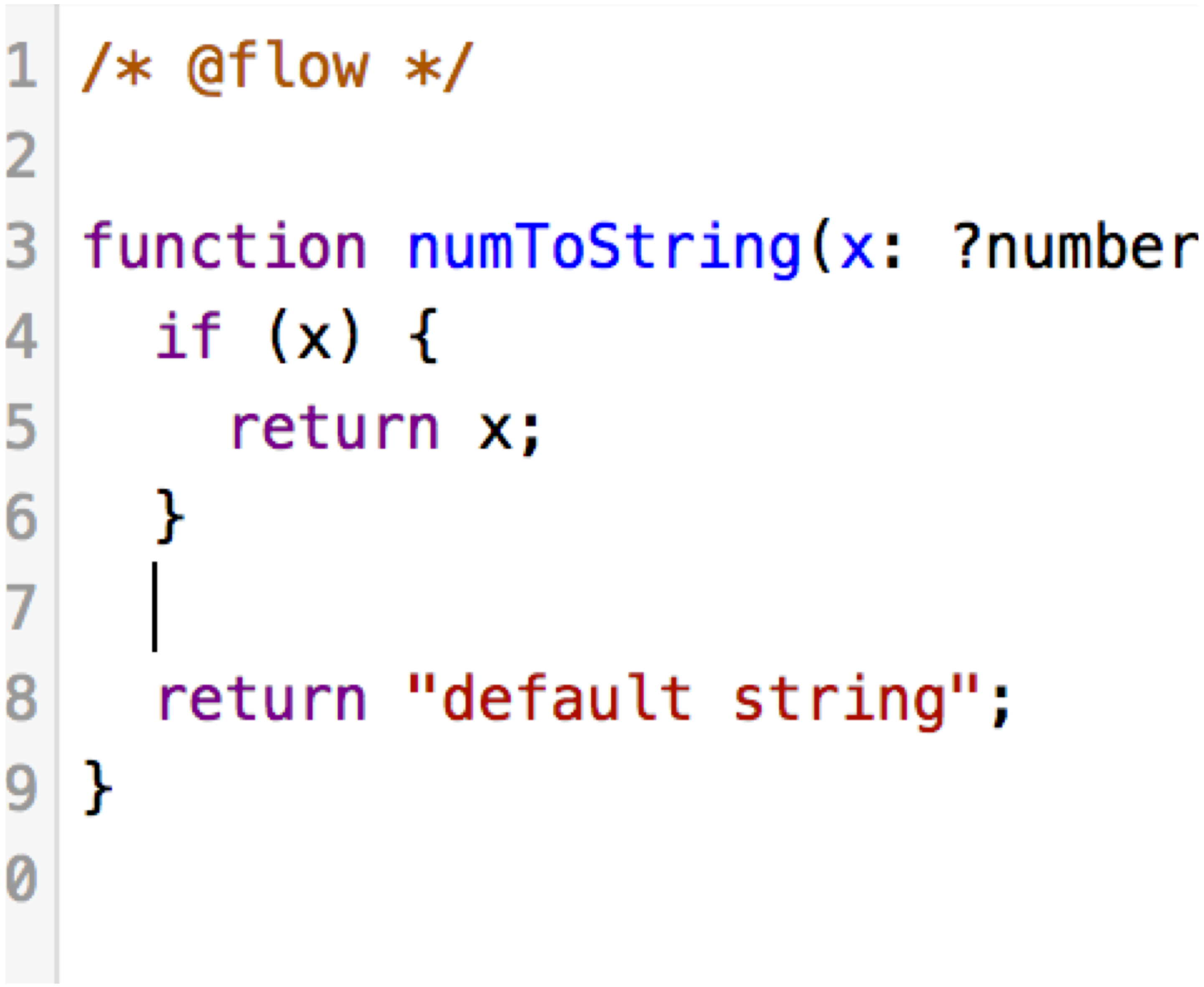
Types in Flow

Flow supports **many** type annotations; much of the syntax is reminiscent of TypeScript, such as with union types and generic types

- https://flow.org/en/docs/types/

Annotating a method

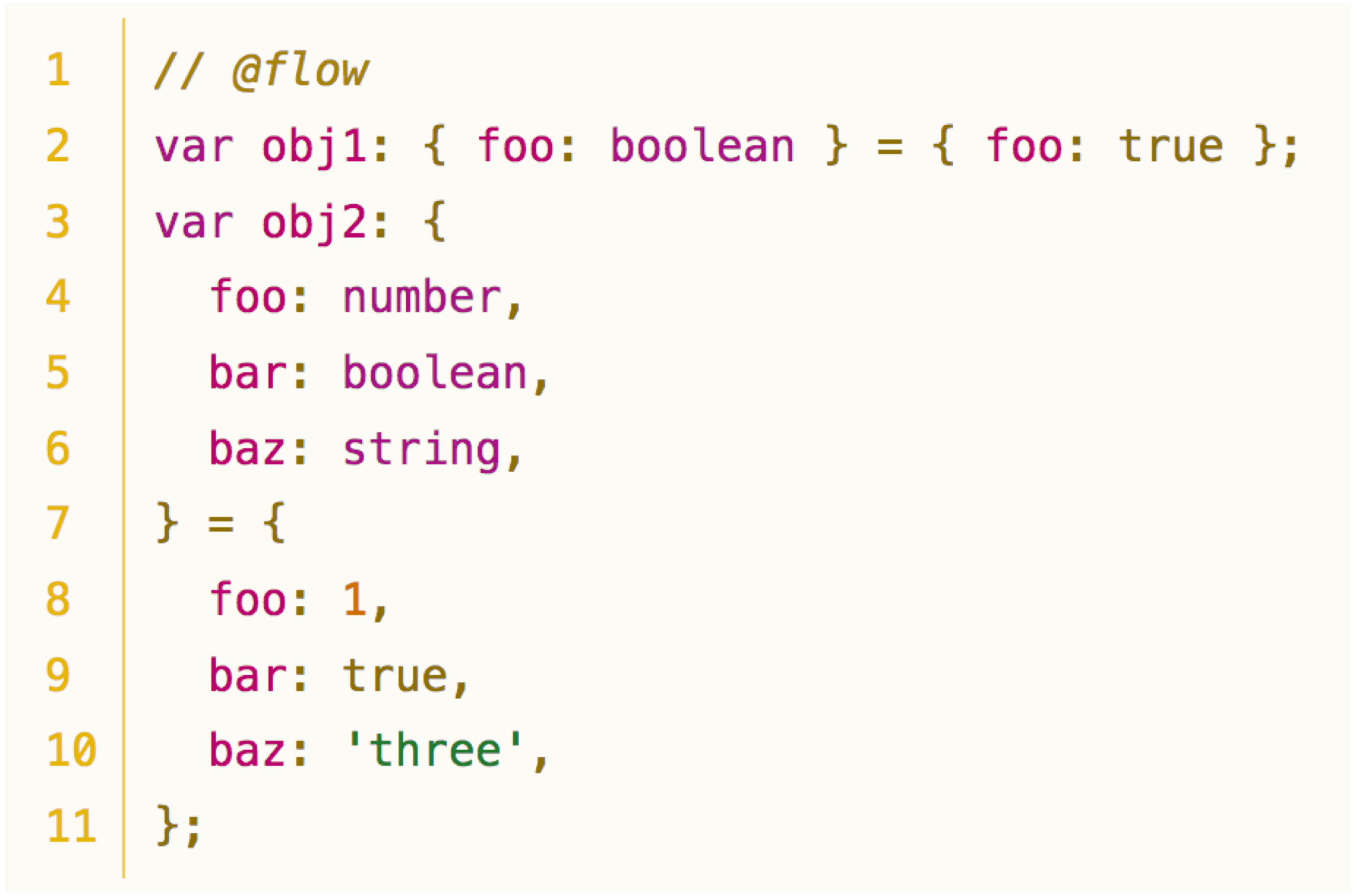
We can annotate methods to declare what types they should be allowed to be called with.



```
/* @flow */

function numToString(x: ?number
  if (x) {
    return x;
  }

  return "default string";
}

```

Annotating a complex object

We can also annotate complex object structures and have the parameter checked for all matching fields. As usual, since JavaScript is really a house of cards awaiting to tragically collapse worse than the Netflix show's most recent season and leaving us extremely unfulfilled when we are picking up life from the ashes, Flow supports duck-typing.

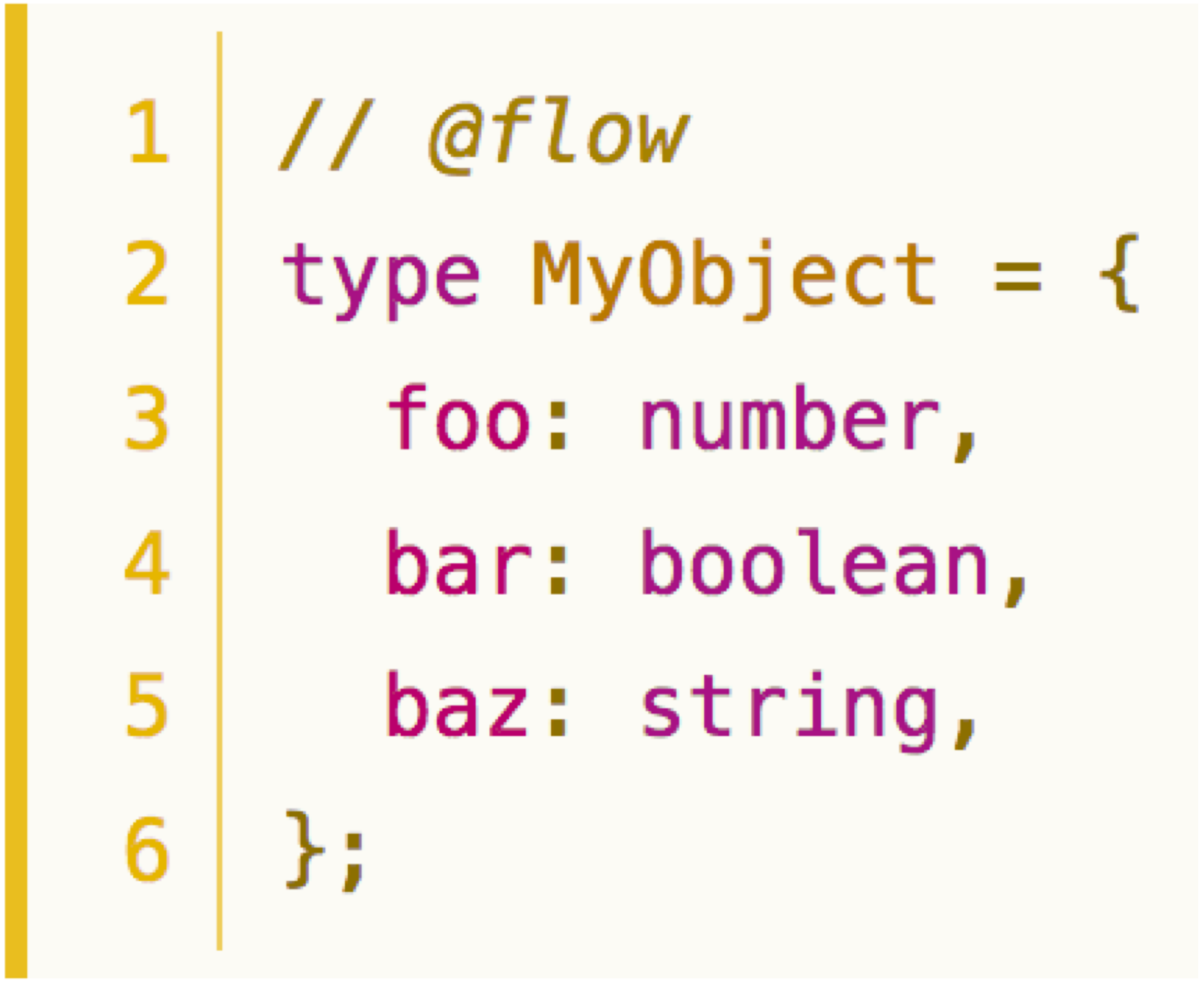


```
// @flow
var obj1: { foo: boolean } = { foo: true };
var obj2: {
  foo: number,
  bar: boolean,
  baz: string,
} = {
  foo: 1,
  bar: true,
  baz: 'three',
};
```

Type Aliasing

Despite our vast egos and generally healthy salaries, developers **are** only human and, if we were to define a complex object each time we wanted to expect it as a parameter we would utterly fail because we're usually under tight deadlines by people who would never cut it as developers.

We can therefore alias a Type and reuse it by Type name, to avoid this issue.



```
// @flow
type MyObject = {
  foo: number,
  bar: boolean,
  baz: string,
};
```

Running Flow

Because it’s 2018 and we need to overwhelm ourselves with options, Flow can be injected into your process in a number of ways:

- Directly through **Babel**
- Using the **flow-remove-types** utility to transpile files
- Through create-react-app, by simply adding a few configuration settings

For details implementing them in each system, see:

- https://flow.org/en/docs/tools/