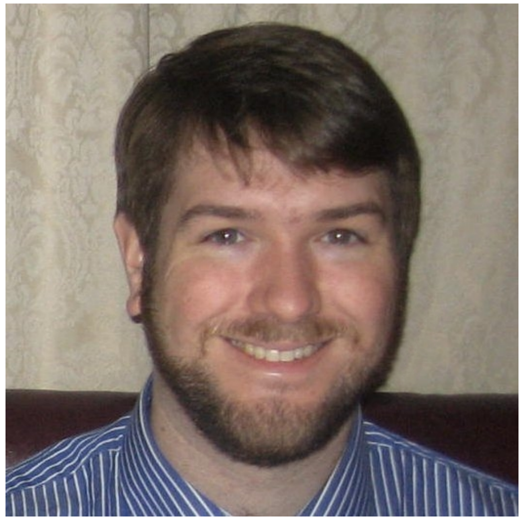
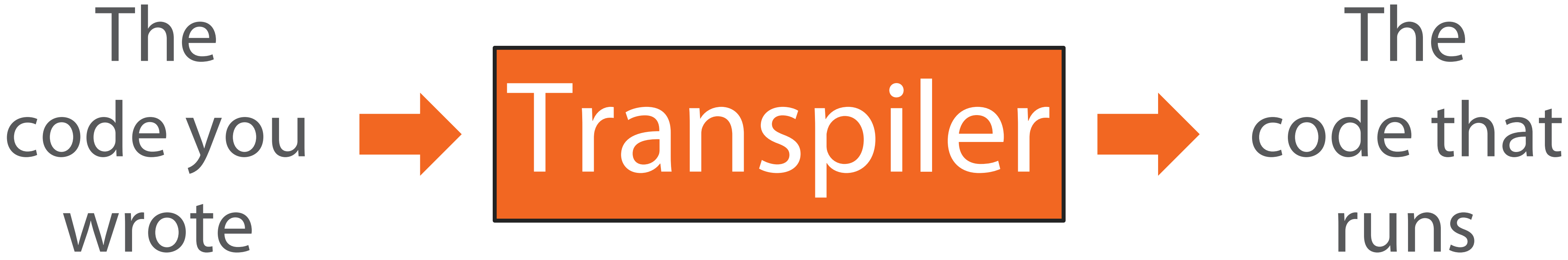
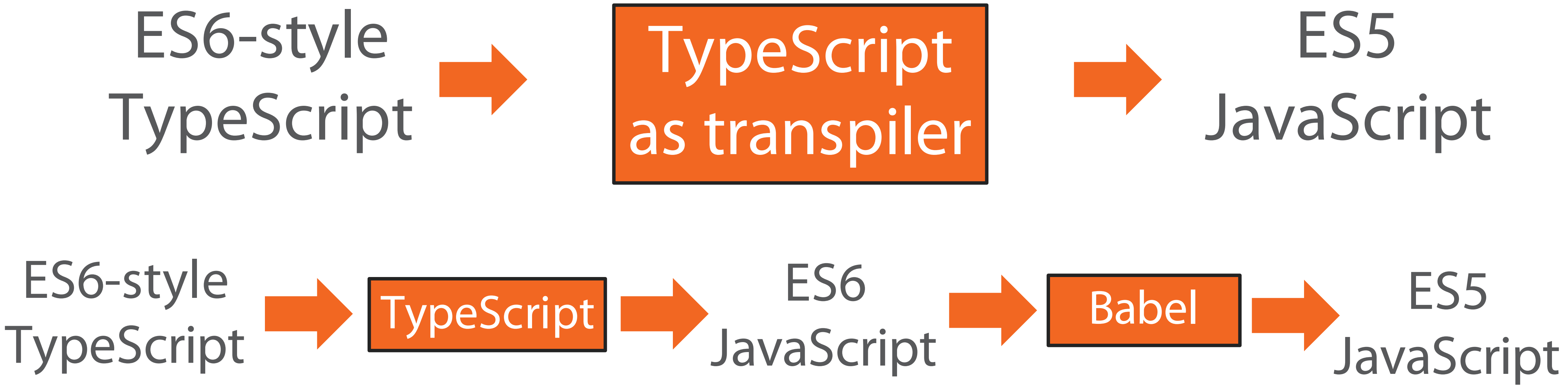
New ES6 Syntax



Steve Ognibene

@NYCdotNet | www.legendaryapps.com


The code you wrote
Transpiler
The code that runs


ES6-style TypeScript
TypeScript as transpiler
ES5 JavaScript
ES6-style TypeScript
TypeScript
ES6 JavaScript
Babel
ES5 JavaScript

`let`

Characteristics of var

Hoisting

Functional scope

Functional Scope

Functional Scope

- `var` inside function:
  - Hoisted to top of the function
  - Usable throughout the function
- `var` outside function:
  - hoisted to “top” of global scope
  - Usable throughout the program

Global Scope Example with var

```
// Script1.js

var person = "Alice";
```

```
// Script2.js

var person = "Bob";
```

```html
<script src="Script1.js"></script>
<script src="Script2.js"></script>
```

Characteristics of var

Hoisting

Functional scope

`var` vs. `let`

`var`:

- Hoists
- Function scope

`let`:

- Not hoisted
- Block scope

for Loop Closure with `let`

```
for (let i = 0; i < count; i += 1) {
    // automatic closure here
}
```

Immediately Invoked Function Expression

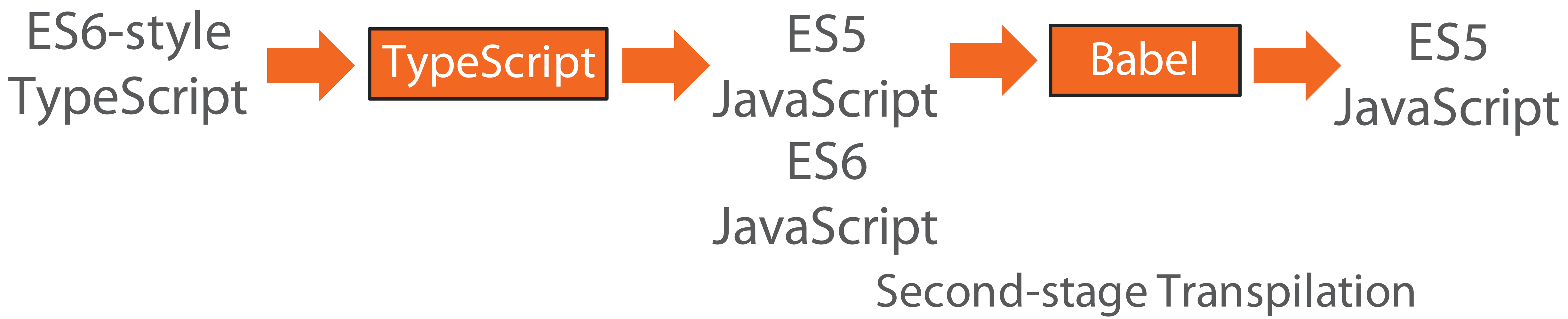
An expression

that creates a function

that is immediately invoked

Using Babel as a Second-stage Transpiler

Usual TypeScript transpilation workflow



ES6 Scope-per-iteration with `for/let`

- TypeScript 1.6 can’t transpile this to ES5
  - Use `var` with an IIFE instead
  - Or use ES6 mode + Babel as second transpiler

New Keyword: `const`

- Same rules as `let`:
  - Block-scoped
  - Not hoisted
- Also:
  - A value must be set on `const` declaration
  - Can’t be changed later

Can an object's properties be held constant using `const`?

`namespace`

New way to create “internal” modules added in TypeScript 1.5

If `const` works with modules,
does it also work with ES6 classes?

`const` and Classes

- Can’t declare class members with `const` in ES6
- Possible if using a TypeScript “class space” or “clodule”.

When to Use `let` and `const`?

- Can clarify purpose and scope of variables
- Can eliminate hand-coding IIFEs in `for` loops
  - (Not in TypeScript 1.6 – use Babel)

When to Use `let` and `const`?

- Use `var`:
  - Globals that are OK to redeclare in same scope.
  - When block scoping is awkward.
- Try to use `const` over `let`.
  - Properties of a `const` *can* be changed.
- TypeScript will provide an error where `let` is required instead of `const` .

Refactoring CoinCounterViewModel

12 `var` → 9 `const`

3 `let`

Find and Replace: `var` to `let`

```javascript
function favoriteFood(name) {
  if (name === "Amy") {
    let fav = "pizza";
  } else {
    let fav = "uncertain";
  }
  return fav;
}
```

Update to `let` or `const` as you go, only if it makes sense.

ES6 Arrow Functions

New in ES6

Shorthand syntax for functions

Simplify `this`

In TypeScript since 2012

Arrow Functions and `this`

Traditional-style Functions

- Value of `this` depends on how you call the function!
  - Could be containing function
  - Could be global namespace
  - Could be something else!!

Arrow Functions

- Value of `this` is always the containing code.
- “Lexical Binding”
- Nested arrow functions share the same `this`.
- See “Practical TypeScript Migration” Module 3, Clip 4: “Lambdas and how ‘this’ works in TypeScript”.

Two More Arrow Function Details

No built-in `arguments` object.

- If you need to iterate over arguments, use an ES6 "rest" parameter instead.

Arrow functions aren't `new`-able.

- You can call a void function with `new` as a "class constructor" in JavaScript.
- An example is in GameClock.ts in the sample code.
- Always use a standard function as the constructor of a functional-style "class".

Destructuring

Break-up an object or array into component variables

ES6 String Templates

```
const moreStuff = 'More Stuff';
const myString = 'Stuff' + moreStuff;

const usesBacktick = `Hello backtick!`;

const myStringToo = `Stuff${moreStuff}`;
```

Tagged String Template Ideas

- HTML escaping
- URL escaping
- Format object into HTTP headers
- Log substitutions to console if a debug variable is set
- Throw an exception if a “bad” substitution value is found

You don’t have to tag a string template.

New ES6 Syntax in TypeScript 1.6

`const`
`let`

Arrow Functions
`this`

Destructuring Objects and Arrays

Using the spread and rest operator

String Templates and tagging

`for` … `of` loops

Coming Up: ES6 Modules