Transpiling

Cory House

@housecor bitnative.com

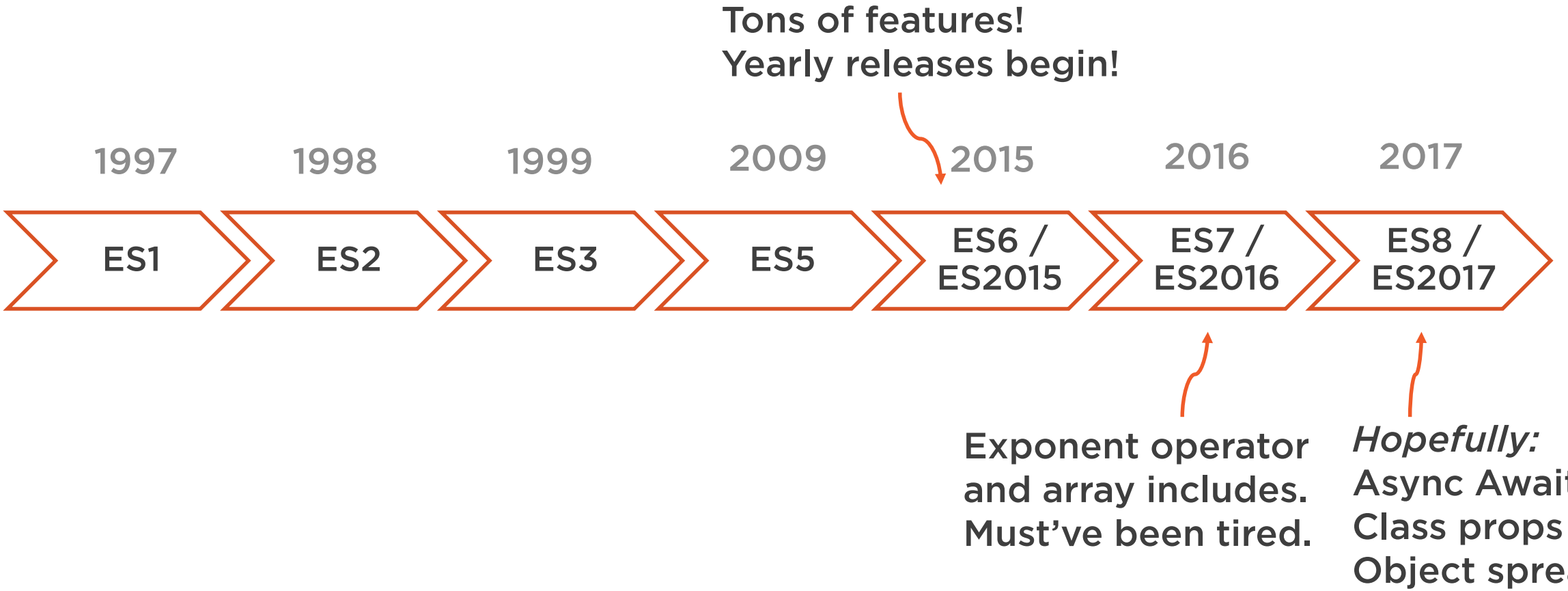
Here's the Plan

Why transpile? - History and future

Transpilers

Set up Babel

ECMAScript Versions



Choosing a Transpiler

List of languages that compile to JS

Edit New Page

Pierre Quentel edited this page 3 days ago · 530 revisions

CoffeeScript Family (& Friends)

- CoffeeScript Unfancy JavaScript
  stars 13k forks 1798 issues 347 open
- CoffeeScript II: The Wrath of Khan Rewrite of the CS compiler
  stars 2k forks 118 issues 106 open

Family (share genes with CoffeeScript)

- Coco A CoffeeScript dialect that aims to be more radical and practical, also acts as a test bed for features that get imported in CoffeeScript.
  stars 494 forks 40 issues 41 open
  - LiveScript is a fork of Coco that is much more compatible with CoffeeScript, more functional, and with more features.
    stars 2k forks 144 issues 160 open
- IcedCoffeeScript A CoffeeScript dialect that adds support for `await` and `defer` keywords which simplify async control flow.
  stars 721 forks 59 issues 76 open
- Parsec CoffeeScript CS based on parser combinators. The project's aim is to add static metaprogramming (i.e. macros + syntax extensibility) to Coffee Script (CS), similar to how Metalua adds such features to Lua. The resulting compiler, once merged with the official compiler, should be usable as a drop-in replacement for it.
  stars 116 forks 13 issues 3 open
- Contracts.coffee A dialect of CoffeeScript that adds built-in support for contracts.
  stars 225 forks 6 issues 28 open

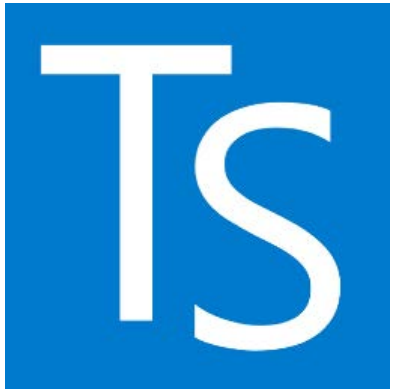
Clone this wiki locally

https://github.com/jashkenas/cof

Clone in Desktop

Popular Transpilers

Babel



TypeScript

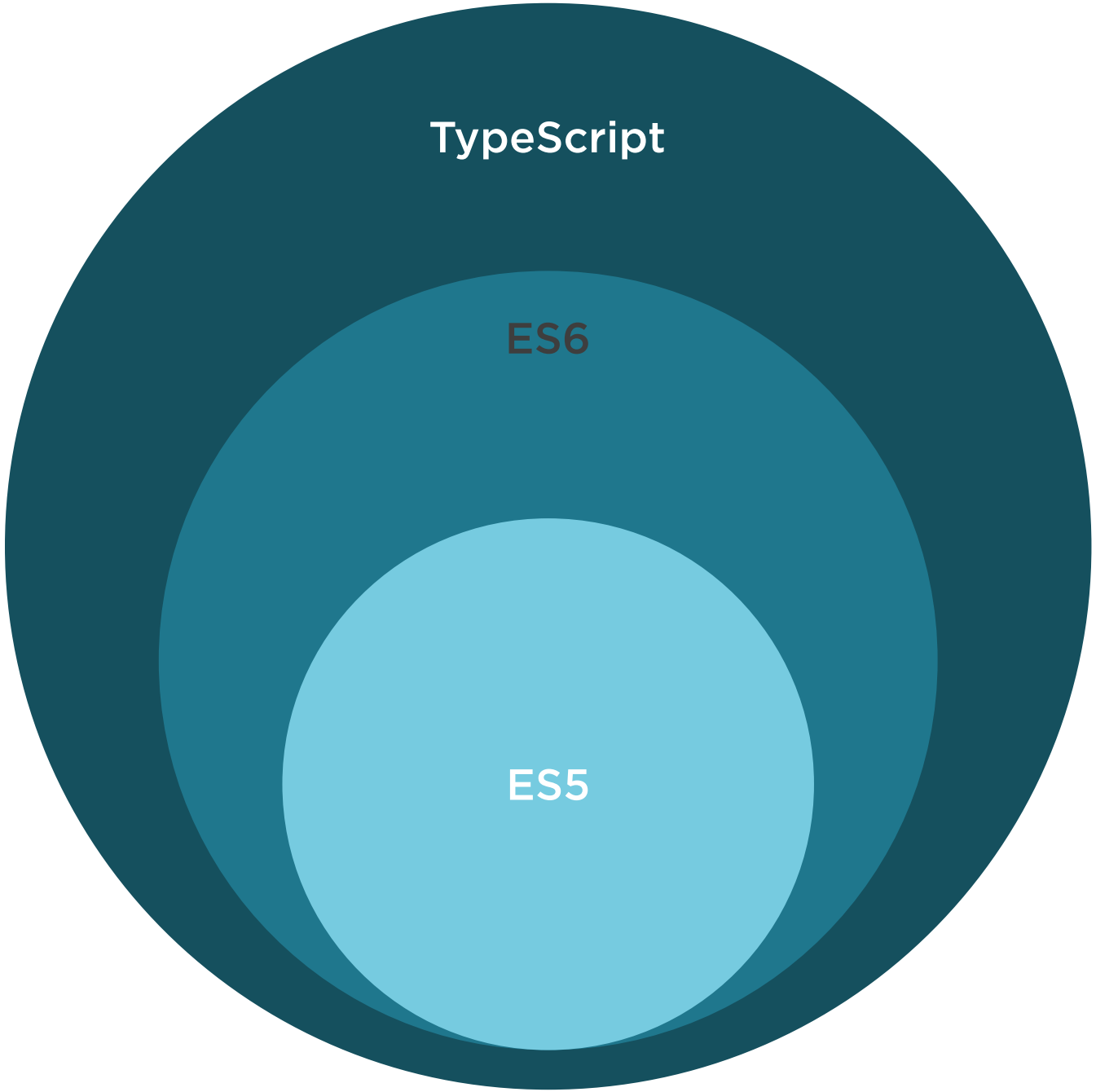
Elm

Why Babel?

Modern, standards-based JS, today.

Why TypeScript?

- Superset of JavaScript
- Enhanced autocompletion
- Safer refactoring
- Clearer intent


TypeScript
ES6
ES5

TypeScript vs Babel

TypeScript

Enhanced Autocomplete

Enhanced readability

Safer refactoring

Additional non-standard features

Babel

Write standardized JS

Leverage full JS Ecosystem

Use experimental features earlier

No type defs, annotations required

ES6 imports are statically analyzable

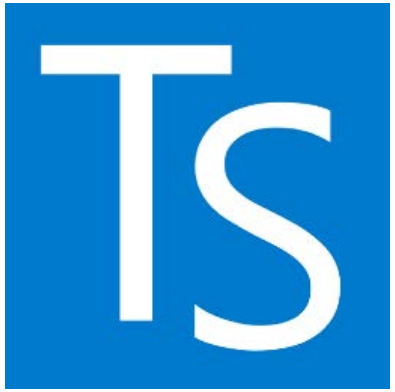
Test, Lint, Babel, Great libs, IDE = safety

Why Elm?

- Compiles down to JS
- Clean Syntax
- Immutable data structures
- Friendly errors
- All errors are compile-time errors
- Interops with JS

Popular Transpilers

Babel



TypeScript

Elm

Babel Configuration Styles

.babelrc

Not npm specific

Easier to read since isolated

package.json

One less file in your project

```
{
  "name": "my-package",
  "version": "1.0.0",
  "babel": {
    // my babel config here
  }
}
```

Stage-X (Experimental Presets)

Any transforms in stage-x presets are changes to the language that haven't been approved to be part of a release of Javascript (such as ES6/ES2015).

> "Changes to the language are developed by way of a process which provides guidelines for evolving an addition from an idea to a fully specified feature"

Subject to change

These proposals are subject to change so ***use with extreme caution***, especially for anything pre stage-3.

The TC39 categorises proposals into 4 stages:

- stage-0 - Strawman: just an idea, possible Babel plugin.
- stage-1 - Proposal: this is worth working on.
- stage-2 - Draft: initial spec.
- stage-3 - Candidate: complete spec and initial browser implementations.
- stage-4 - Finished: will be added to the next yearly release.

Also check out the current tc39 proposals and its process document.

To use these features, install appropriate plugins.

Transpile for Your Environment

Preset	Approach
babel-preset-es2015-node	Version Detection
babel-preset-latest-minimal	Feature Detection

Build Script JS Style

ES5

No waiting for transpile = faster

No transpiler dependency

Transpiled

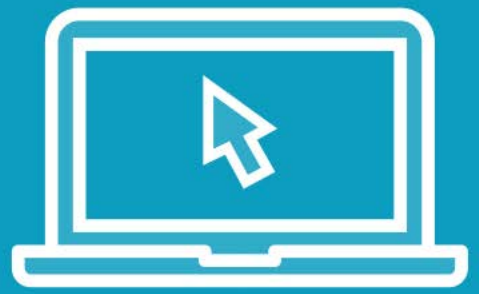
Enjoy the latest features

Consistent coding style

Use the same linting rules everywhere

Can eventually remove transpiler

Demo



Transpiling with Babel

Wrap Up

Transpiling is our present...and future

Transpilers

- Babel, TypeScipt, Elm, dozens more

Configuring Babel

- .babelrc vs package.json
- Experimental features
- Transpiling build scripts

Set up Babel

Next up: Let's set up a bundler!