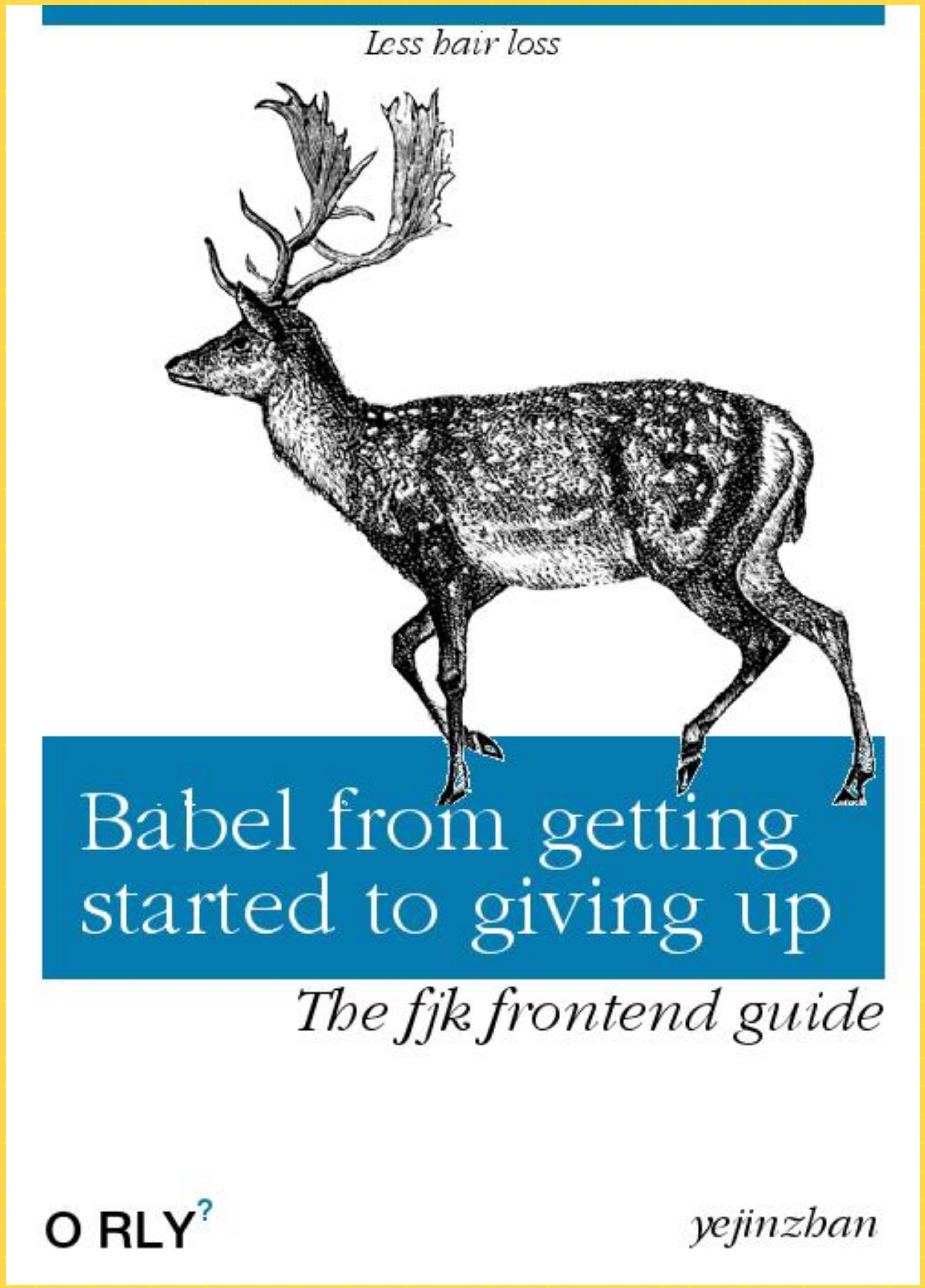
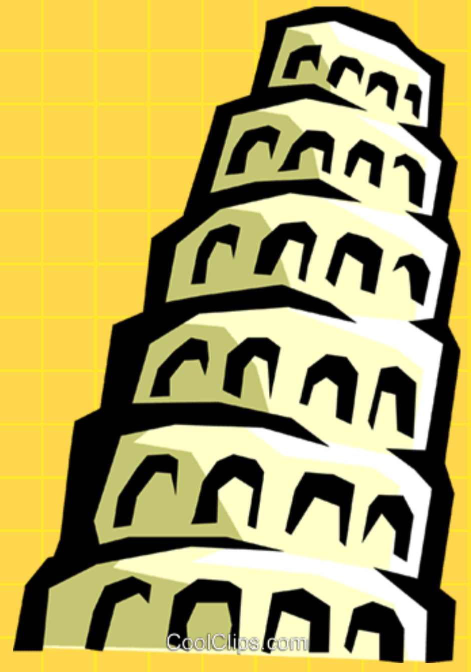


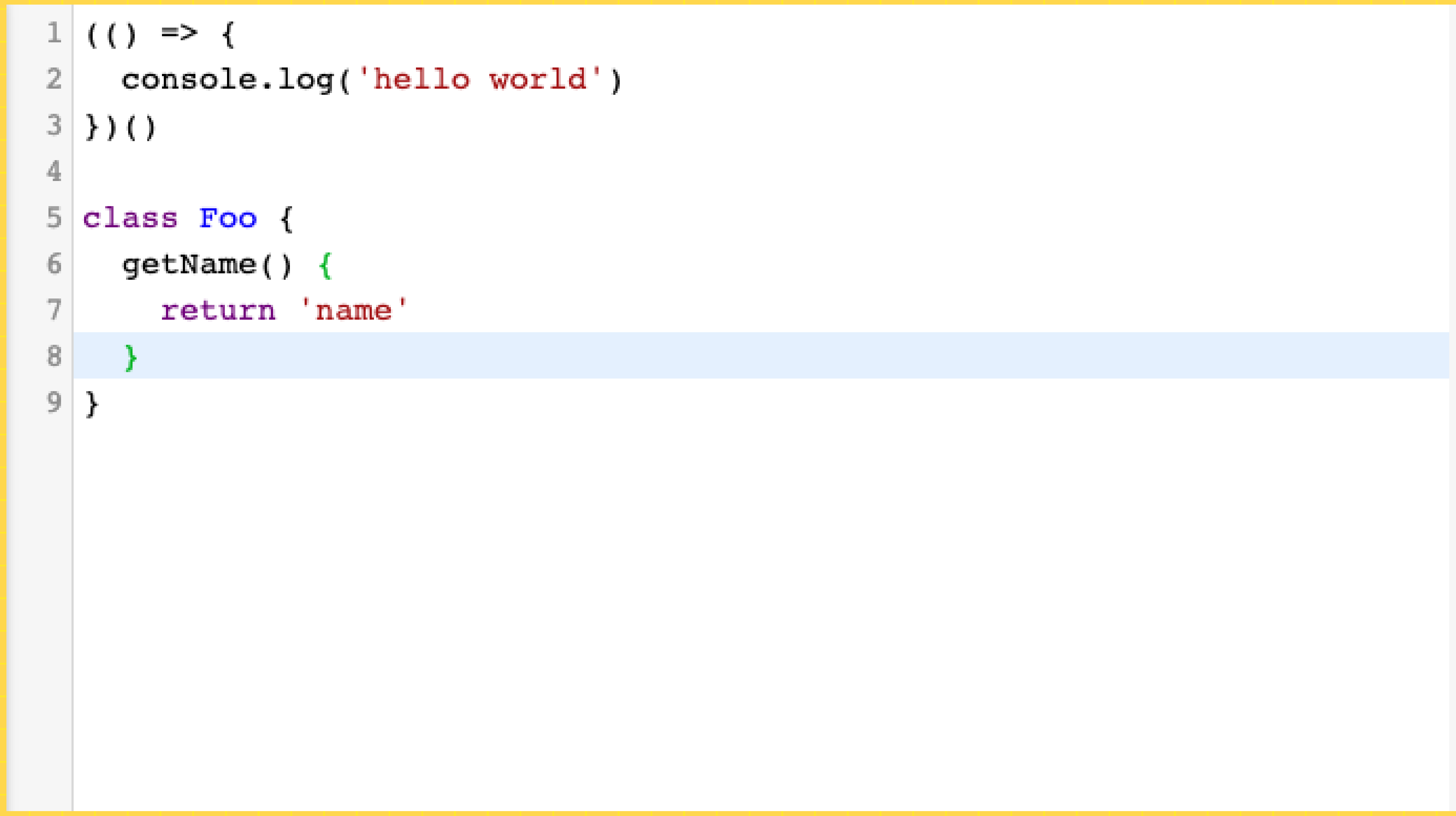
Babel 从入门到放弃



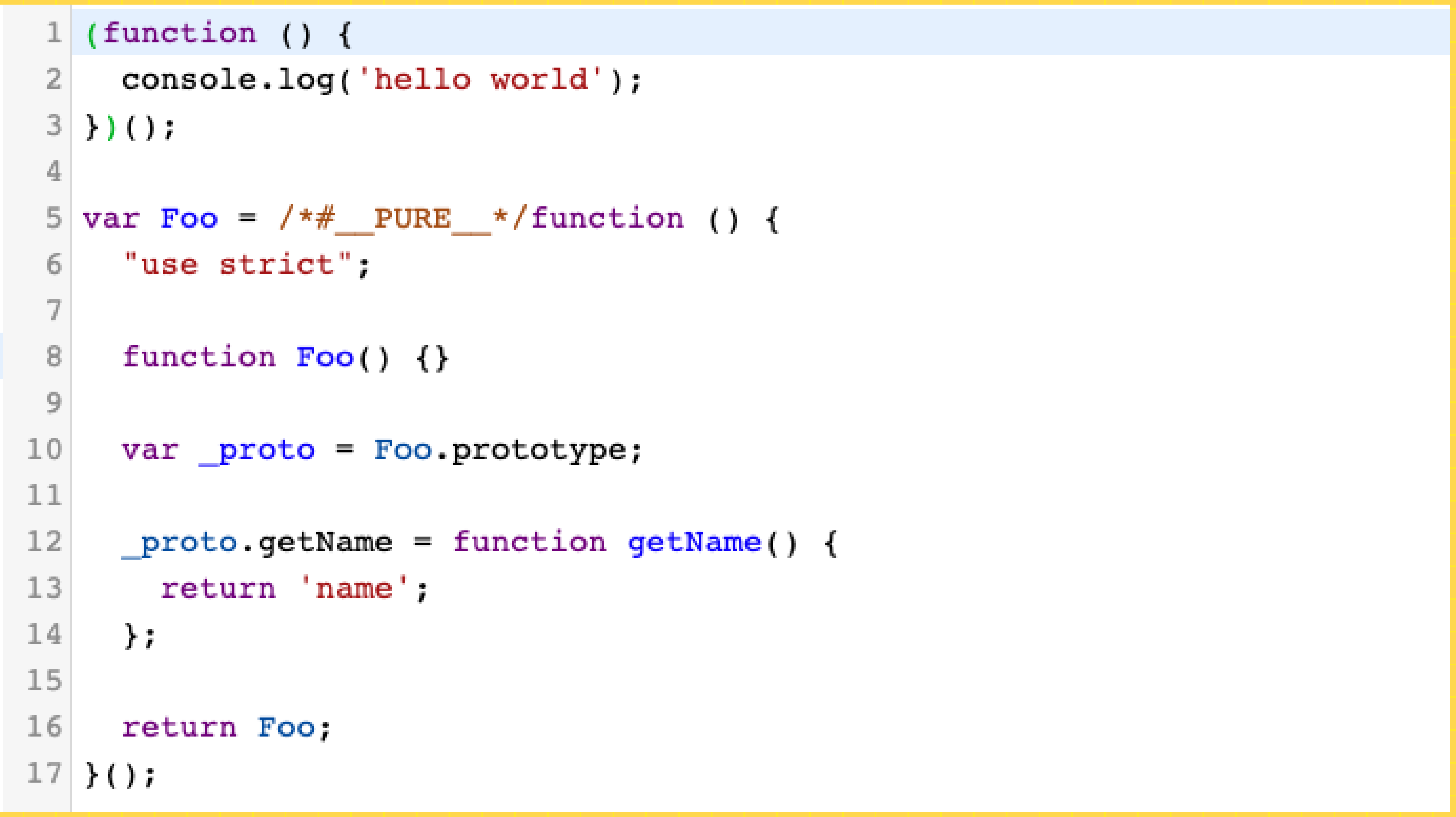
主讲人：叶锦湛

是什么？

- 一个语言转译工具
- ES6(ES2015) => ES5



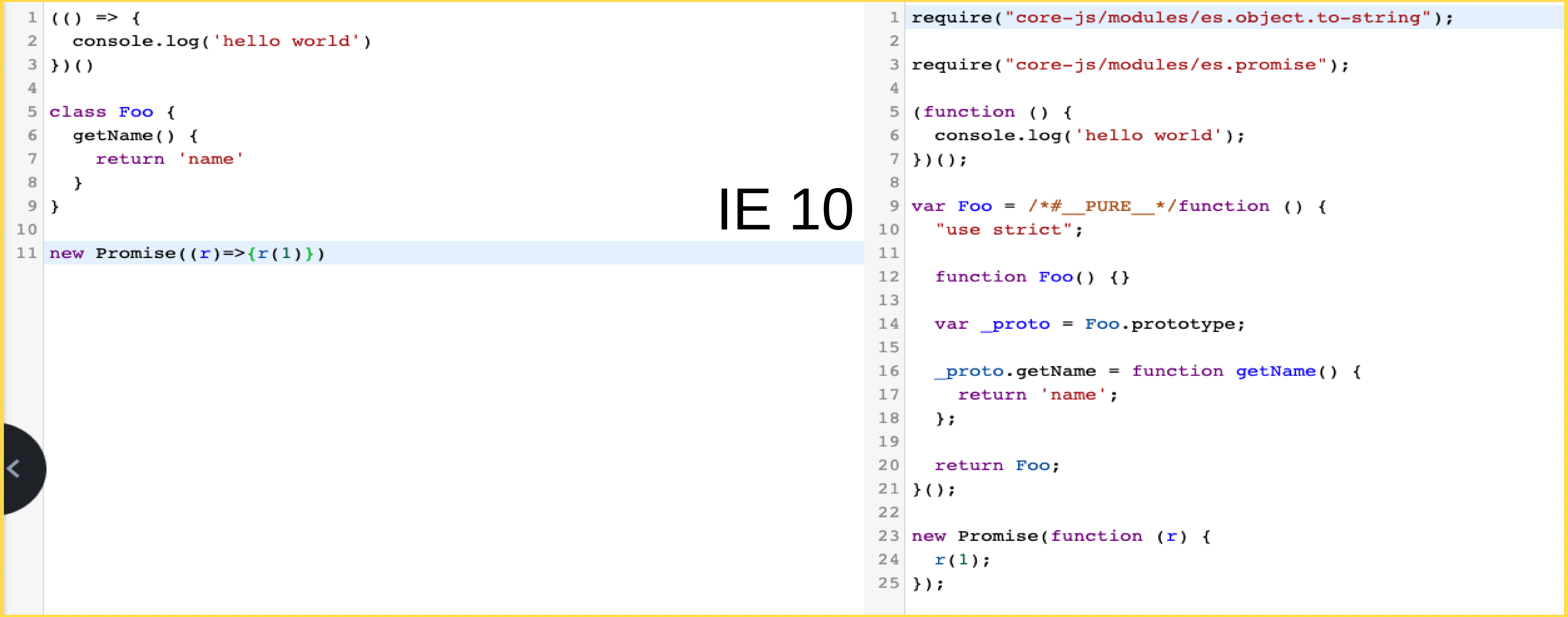
```
(() => {
  console.log('hello world')
})()

class Foo {
  getName() {
    return 'name'
  }
}
```



```
(function () {
  console.log('hello world');
})();

var Foo = /*#__PURE__*/function () {
  "use strict";

  function Foo() {}

  var _proto = Foo.prototype;

  _proto.getName = function getName() {
    return 'name';
  };

  return Foo;
}();
```

为什么？

旧浏览器对 ES6（ES2015） 支持有限，为了让 ES6 的代码能在更多浏览器上运行，获取更多用户，我们需要 Babel



```
(() => {
  console.log('hello world')
})()

class Foo {
  getName() {
    return 'name'
  }
}

new Promise((r)=>{r(1)})
```

IE 10

```
require("core-js/modules/es.object.to-string");

require("core-js/modules/es.promise");

(function () {
  console.log('hello world');
})();

var Foo = /*#__PURE__*/function () {
  "use strict";

  function Foo() {}

  var _proto = Foo.prototype;

  _proto.getName = function getName() {
    return 'name';
  };

  return Foo;
}();

new Promise(function (r) {
  r(1);
});
```

v7.9

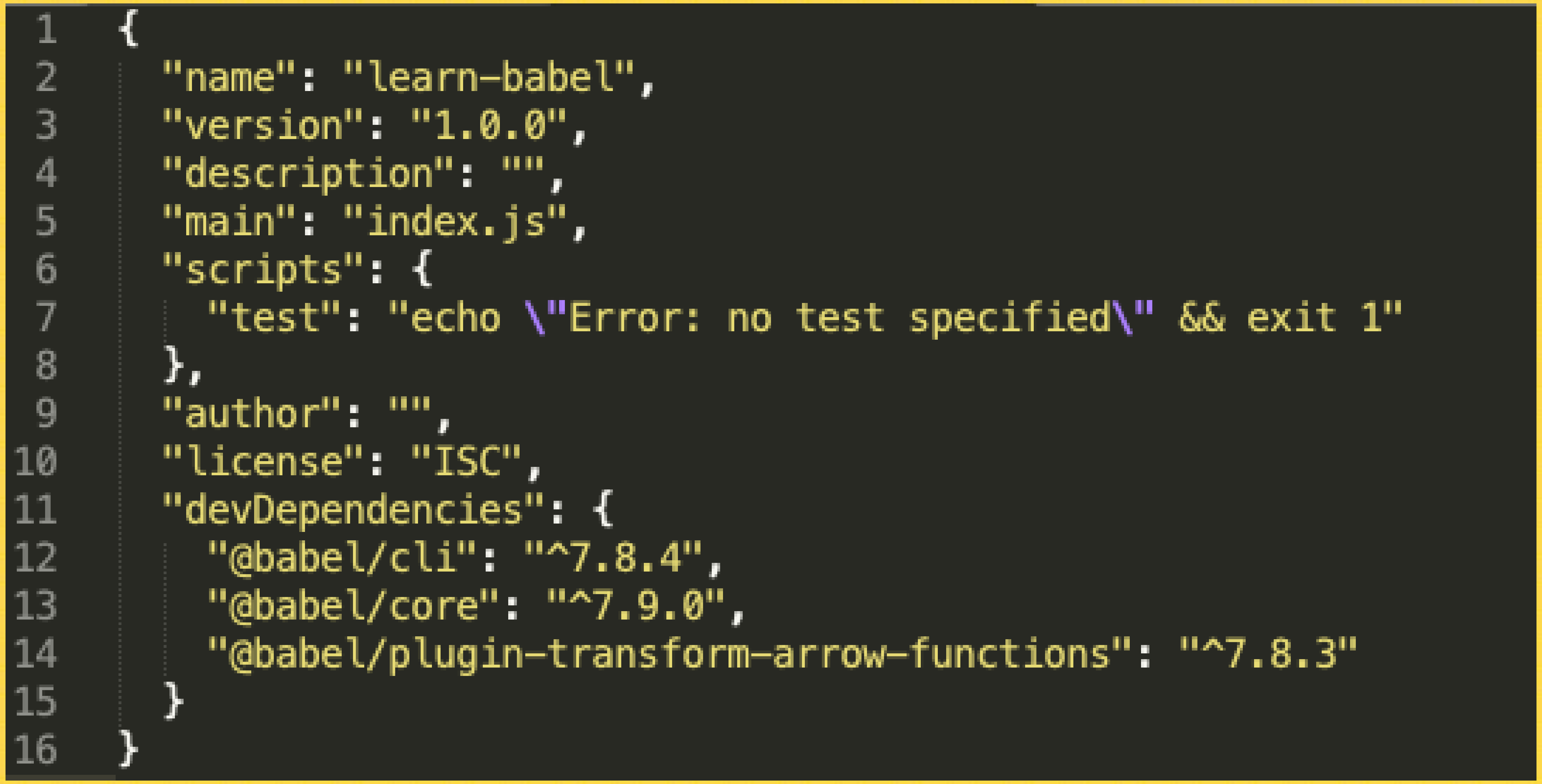
怎么做？

ES6(ES2015)
新语法（ transform 的目标）
块级作用域
let、const
函数
Class
()=>{}
function (...args) {}
.....
迭代器（Iterator）和生成器（Generator）
for-of
async await(2017)
function *createIterator() {}
ES Module
.....
新 API（ polyfill 的目标）
全局 API
Promise
Map/Set
......
实例 API
[].includes
Object.assign
....
polyfill: regenerator-runtime
polyfill: core-js

Babel 组成

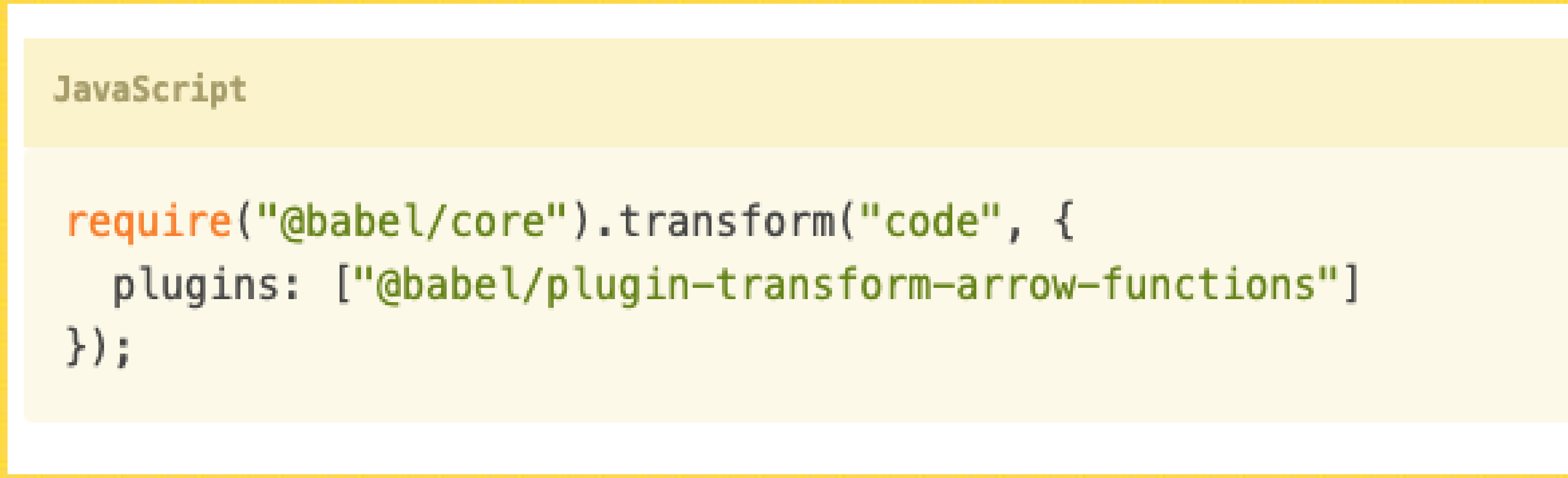
- @babel/cli：命令行工具
- @babel/core：语法转换引擎
- @babel/plugin-x：语法转换插件
- @babel/plugin-transform-runtime：用于生成复用代码、生成非全局污染 polyfill 的插件
- @babel/preset-x：plugin 集合
- @babel/polyfill：新 API 补丁，使用旧浏览器支持的语法特性实现新 API，类似乘法与加法

小试🐮刀



```json
{
  "name": "learn-babel",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@babel/cli": "^7.8.4",
    "@babel/core": "^7.9.0",
    "@babel/plugin-transform-arrow-functions": "^7.8.3"
  }
}
```

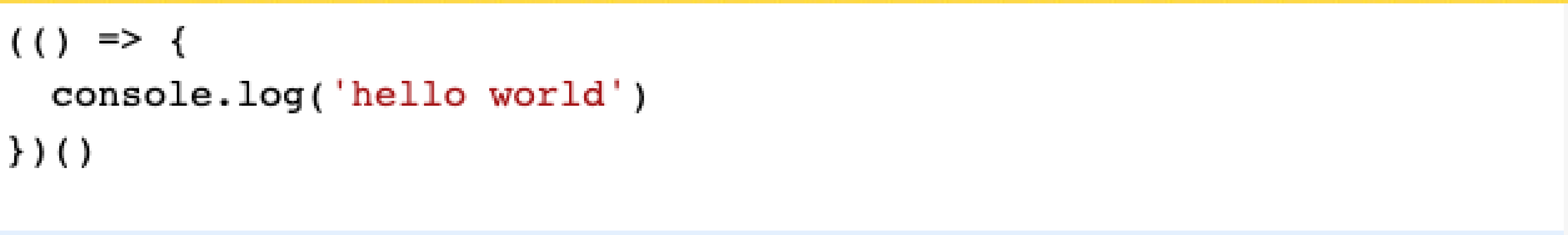
cli 内部调用



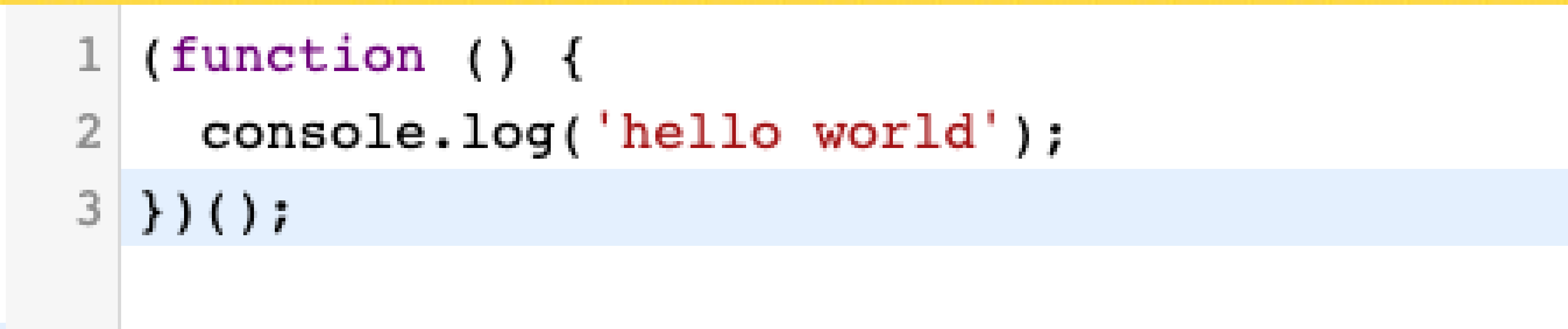
JavaScript

```javascript
require("@babel/core").transform("code", {
  plugins: ["@babel/plugin-transform-arrow-functions"]
});
```

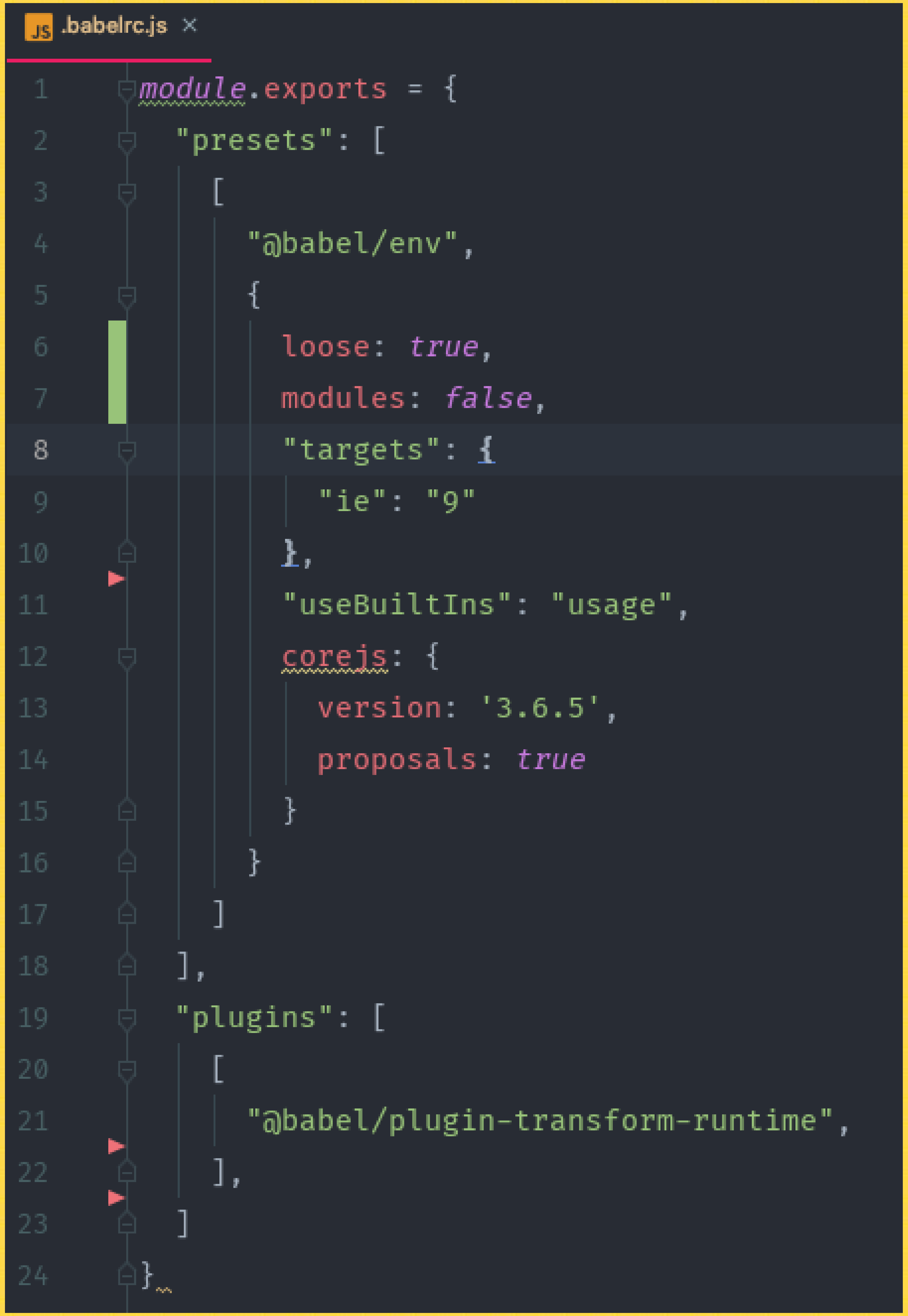
```
npx babel src --out-dir lib --plugins=@babel/plugin-transform-arrow-functions
```



```javascript
(() => {
  console.log('hello world')
})()
```



```javascript
(function () {
  console.log('hello world');
})();
```


.babelrc.js

```
module.exports = {
  "presets": [
    [
      "@babel/env",
      {
        loose: true,
        modules: false,
        "targets": {
          "ie": "9"
        },
        "useBuiltIns": "usage",
        corejs: {
          version: '3.6.5',
          proposals: true
        }
      }
    ]
  ],
  "plugins": [
    [
      "@babel/plugin-transform-runtime",
    ],
  ]
}
```

preset-env

一个能针对特定浏览器进行代码转换的插件集合

loose ：宽松模式，以 new class 为例子，有边界风险，但生成的代码量少、运行快，建议使用 true

modules：将 ES6 Module 转换为其他模块规范，不使用时建议设成 false，默认值 "auto" 让人疑惑

useBuiltIns：polyfill corejs(需要自行安装)/regenerator-runtime(不需要自行安装)，默认为 false，建议使用 "usage"

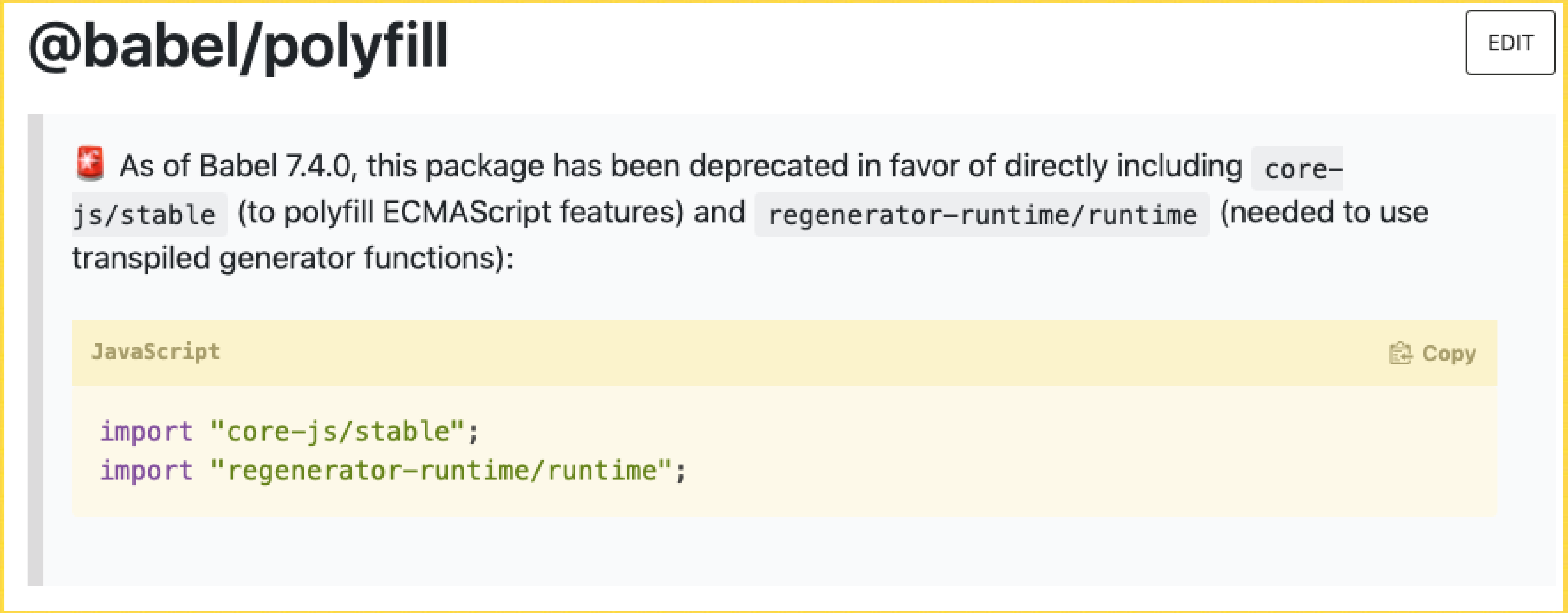
false：禁用 polyfill

"entry": 针对目标浏览器对缺少的API 进行 polyfill

"usage": 只对使用到的API进行 polyfill（类似 tree shaking）

core-js: 指定 corejs 版本，版本越高支持的 API 越多，注意与 package.json 版本对应

@bable/polyfill



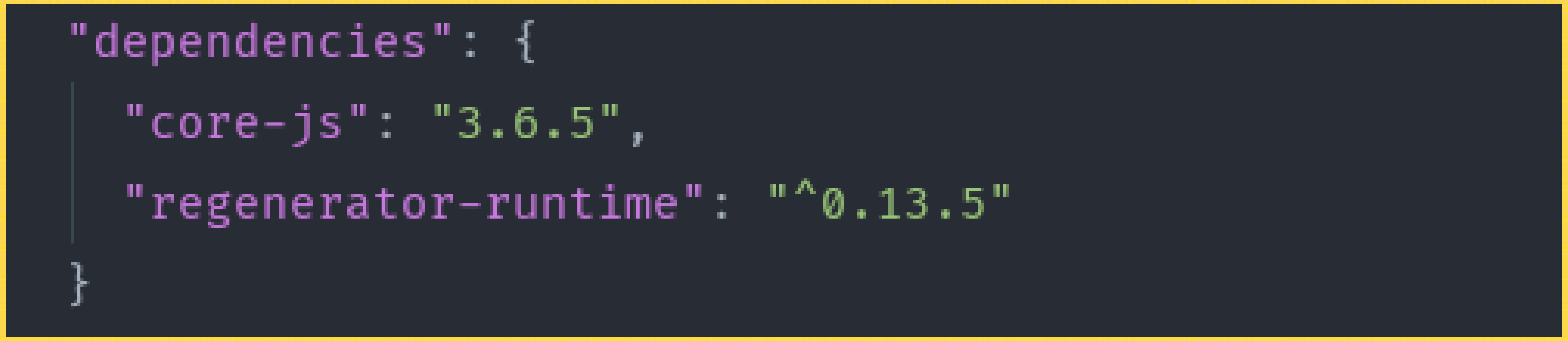
@babel/polyfill

EDIT

🚨 As of Babel 7.4.0, this package has been deprecated in favor of directly including `core-js/stable` (to polyfill ECMAScript features) and `regenerator-runtime/runtime` (needed to use transpiled generator functions):

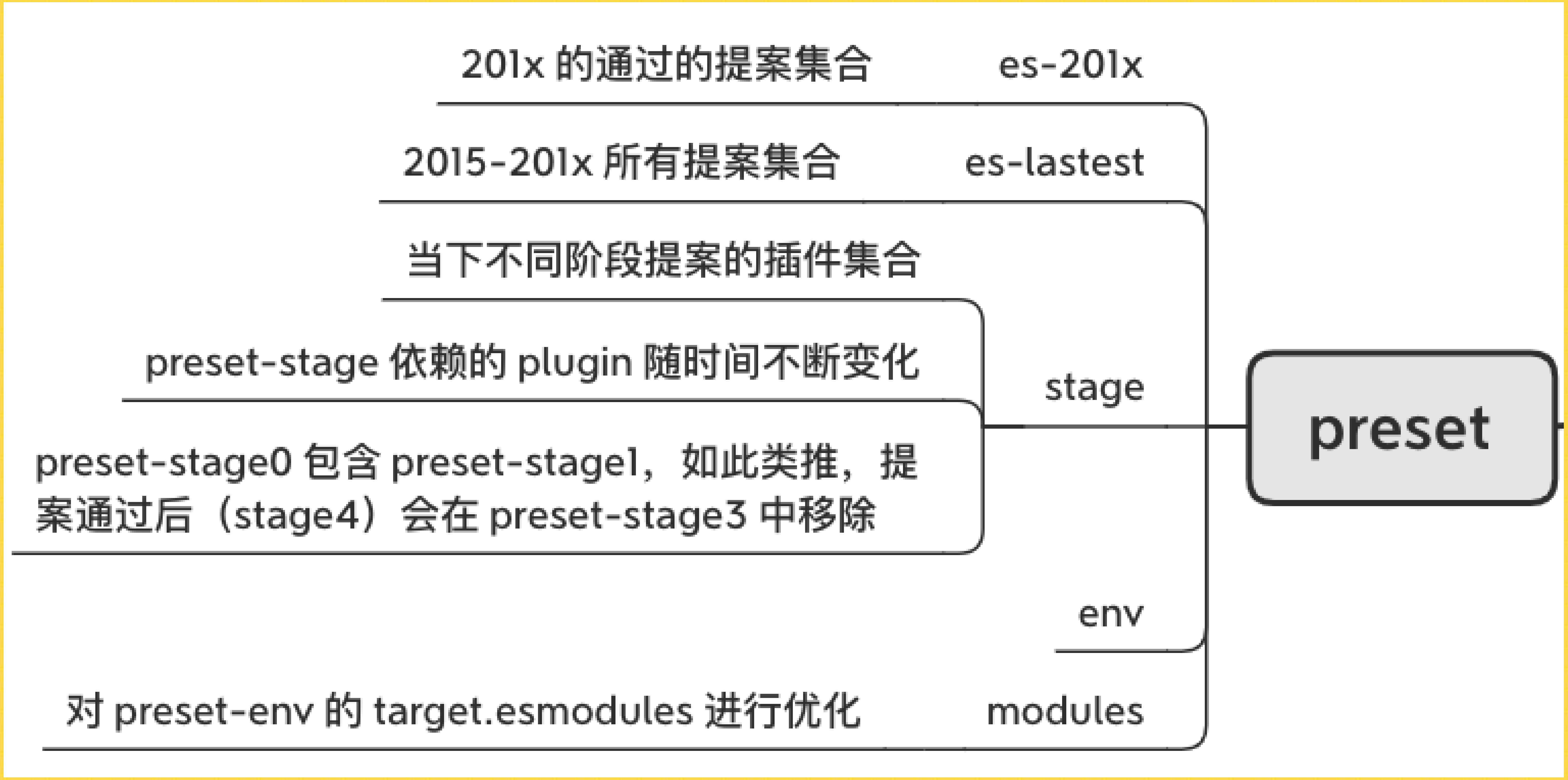
JavaScript Copy

```javascript
import "core-js/stable";
import "regenerator-runtime/runtime";
```



```json
"dependencies": {
  "core-js": "3.6.5",
  "regenerator-runtime": "^0.13.5"
}
```

preset 黑历史


preset
es-201x
201x 的通过的提案集合
es-lastest
2015-201x 所有提案集合
stage
当下不同阶段提案的插件集合
preset-stage 依赖的 plugin 随时间不断变化
preset-stage0 包含 preset-stage1，如此类推，提案通过后（stage4）会在 preset-stage3 中移除
env
modules
对 preset-env 的 target.esmodules 进行优化

preset-201x 包
https://github.com/babel/babel/blob/6.x/packages/babel-preset-latest/package.json
https://github.com/babel/babel/blob/6.x/packages/babel-preset-es2015/package.json

stage 提案说明
stage 包

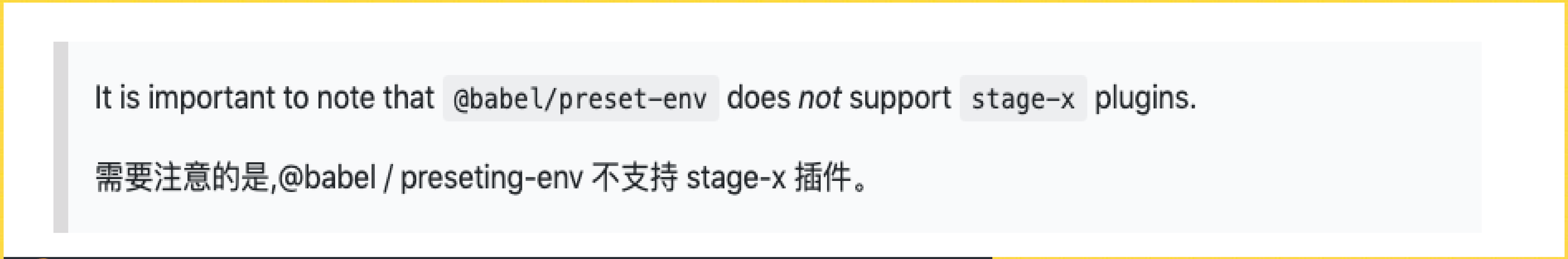
preset-modules

preset-stage2 版本对比

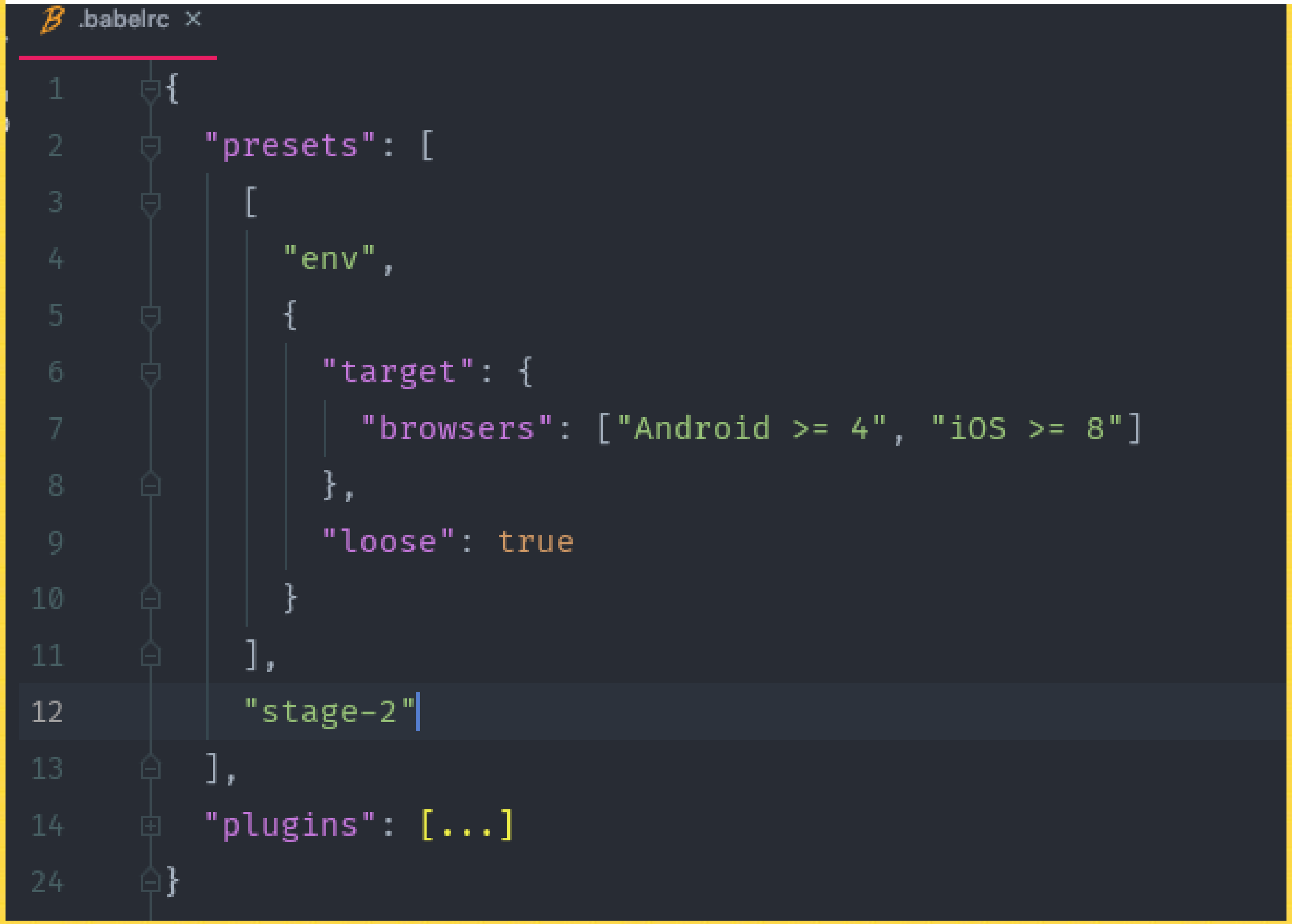
```
{
  "name": "babel-preset-stage-2",
  "version": "6.1.2",
  "description": "Babel preset for stage 2 plugins",
  "author": "Sebastian McKenzie <sebmck@gmail.com>",
  "homepage": "https://babeljs.io/",
  "license": "MIT",
  "repository": "https://github.com/babel/babel/tree/master/packages/babel-p
  "main": "index.js",
  "dependencies": {
    "babel-plugin-syntax-trailing-function-commas": "^6.0.14",
    "babel-plugin-transform-object-rest-spread": "^6.0.14",
    "babel-preset-stage-3": "^6.1.2"
  }
}
```

```
{
  "name": "babel-preset-stage-2",
  "version": "6.24.1",
  "description": "Babel preset for stage 2 plugins",
  "author": "Sebastian McKenzie <sebmck@gmail.com>",
  "homepage": "https://babeljs.io/",
  "license": "MIT",
  "repository": "https://github.com/babel/babel/tree/master/packages/babel-pres
  "main": "lib/index.js",
  "dependencies": {
    "babel-plugin-transform-class-properties": "^6.24.1",
    "babel-plugin-transform-decorators": "^6.24.1",
    "babel-plugin-syntax-dynamic-import": "^6.18.0",
    "babel-preset-stage-3": "^6.24.1"
  }
}
```

stage 的问题


It is important to note that `@babel/preset-env` does *not* support `stage-x` plugins.

需要注意的是,@babel / preseting-env 不支持 stage-x 插件。

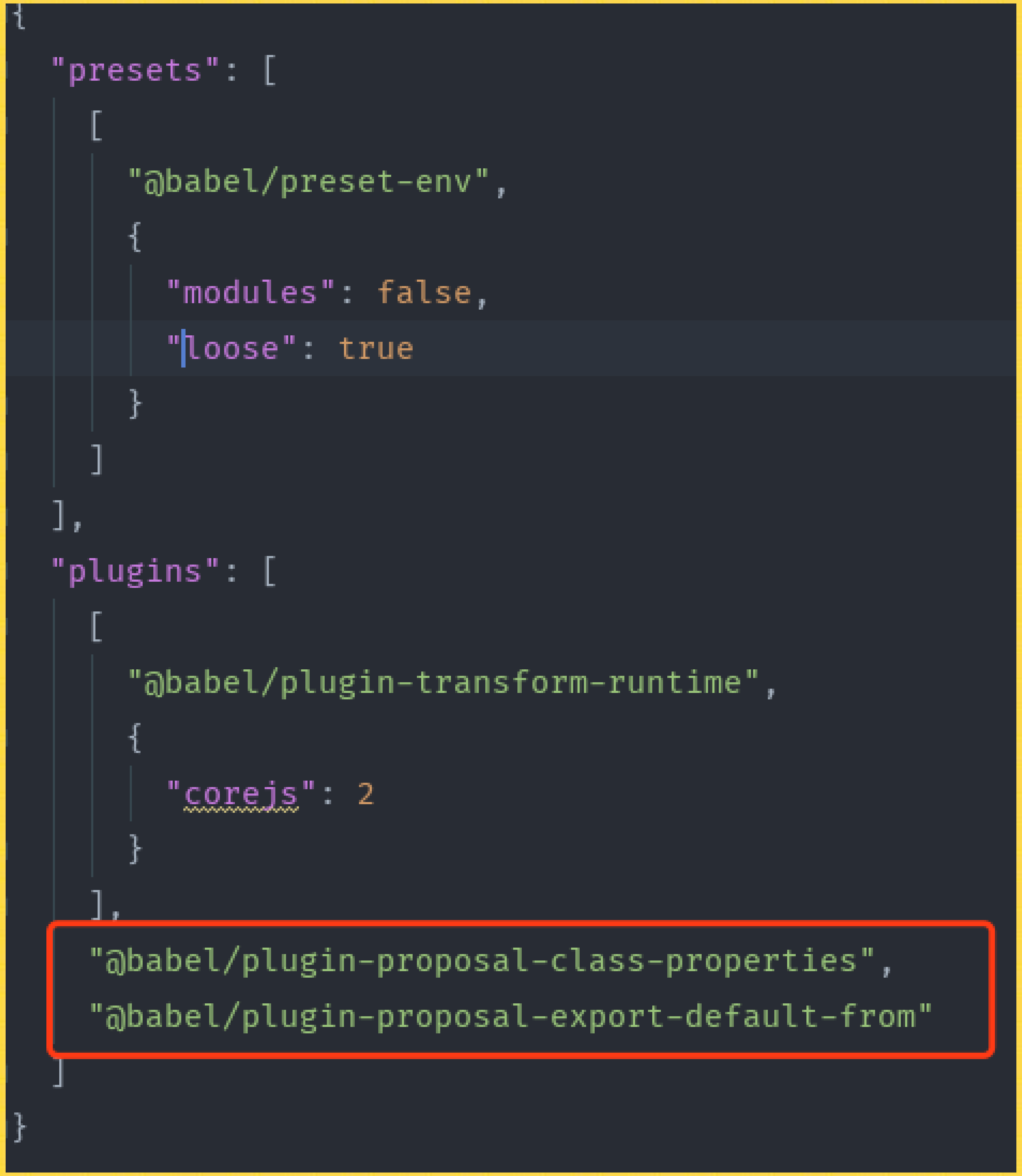

.babelrc

```
{
  "presets": [
    [
      "env",
      {
        "target": {
          "browsers": ["Android >= 4", "iOS >= 8"]
        },
        "loose": true
      }
    ],
    "stage-2"
  ],
  "plugins": [...]
}
```

- stage 需要斟酌使用，提案最后有可能没通过

- 与 preset-env 隐晦的关联，升级版本时如果只升级 stage 不升级 env，废除特性、提案通过的特性会转换失效。

如何使用 stage 语法？



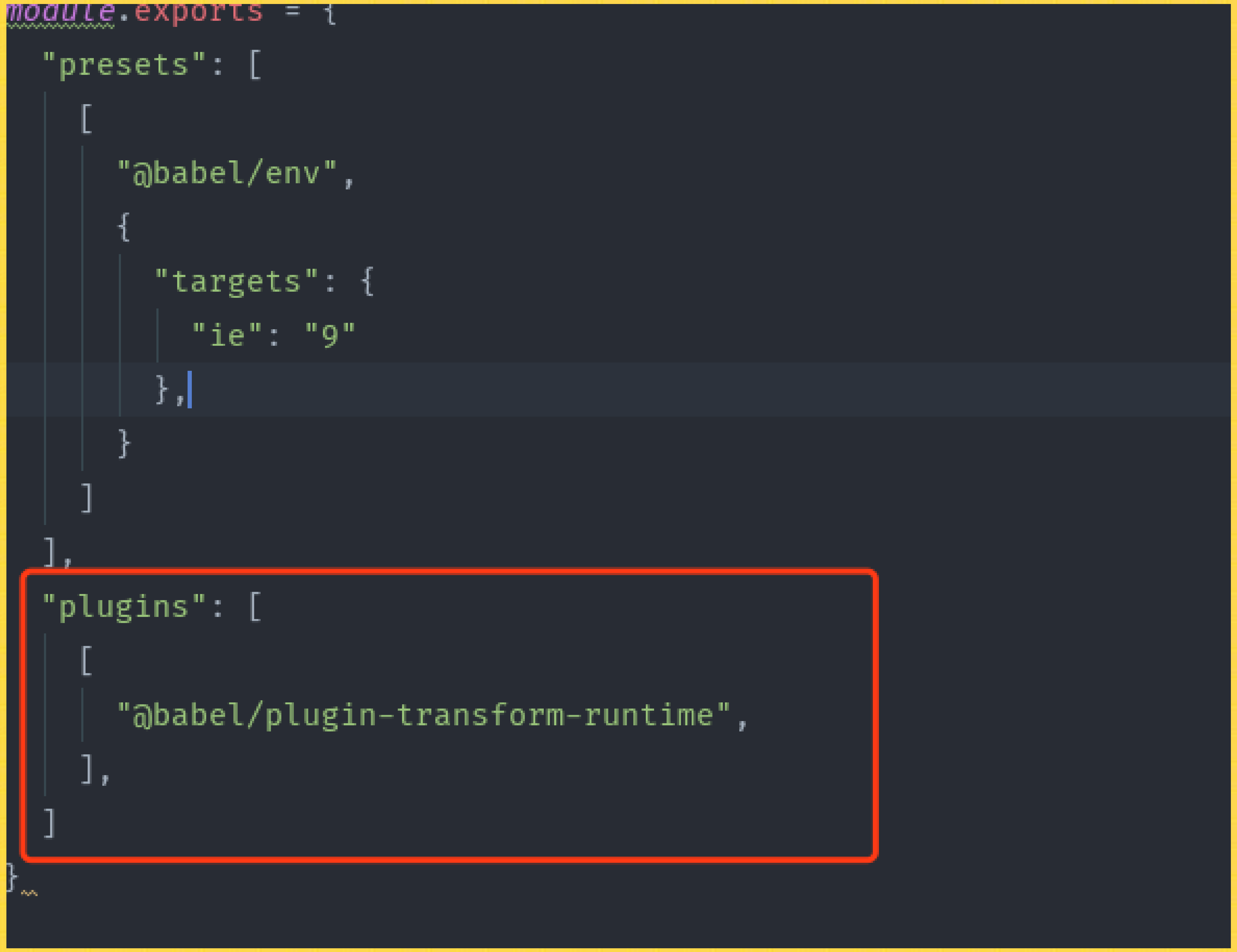
```
{
  "presets": [
    [
      "@babel/preset-env",
      {
        "modules": false,
        "loose": true
      }
    ]
  ],
  "plugins": [
    [
      "@babel/plugin-transform-runtime",
      {
        "corejs": 2
      }
    ],
    "@babel/plugin-proposal-class-properties",
    "@babel/plugin-proposal-export-default-from"
  ]
}
```

使用 proposal plugin

transform-runtime

用于生成复用代码、生成非全局 polyfill 的插件（以 class 为例）

@babel/runtime 需要自动安装



```
module.exports = {
  "presets": [
    [
      "@babel/env",
      {
        "targets": {
          "ie": "9"
        },
      }
    ]
  ],
  "plugins": [
    [
      "@babel/plugin-transform-runtime",
    ],
  ]
}
```

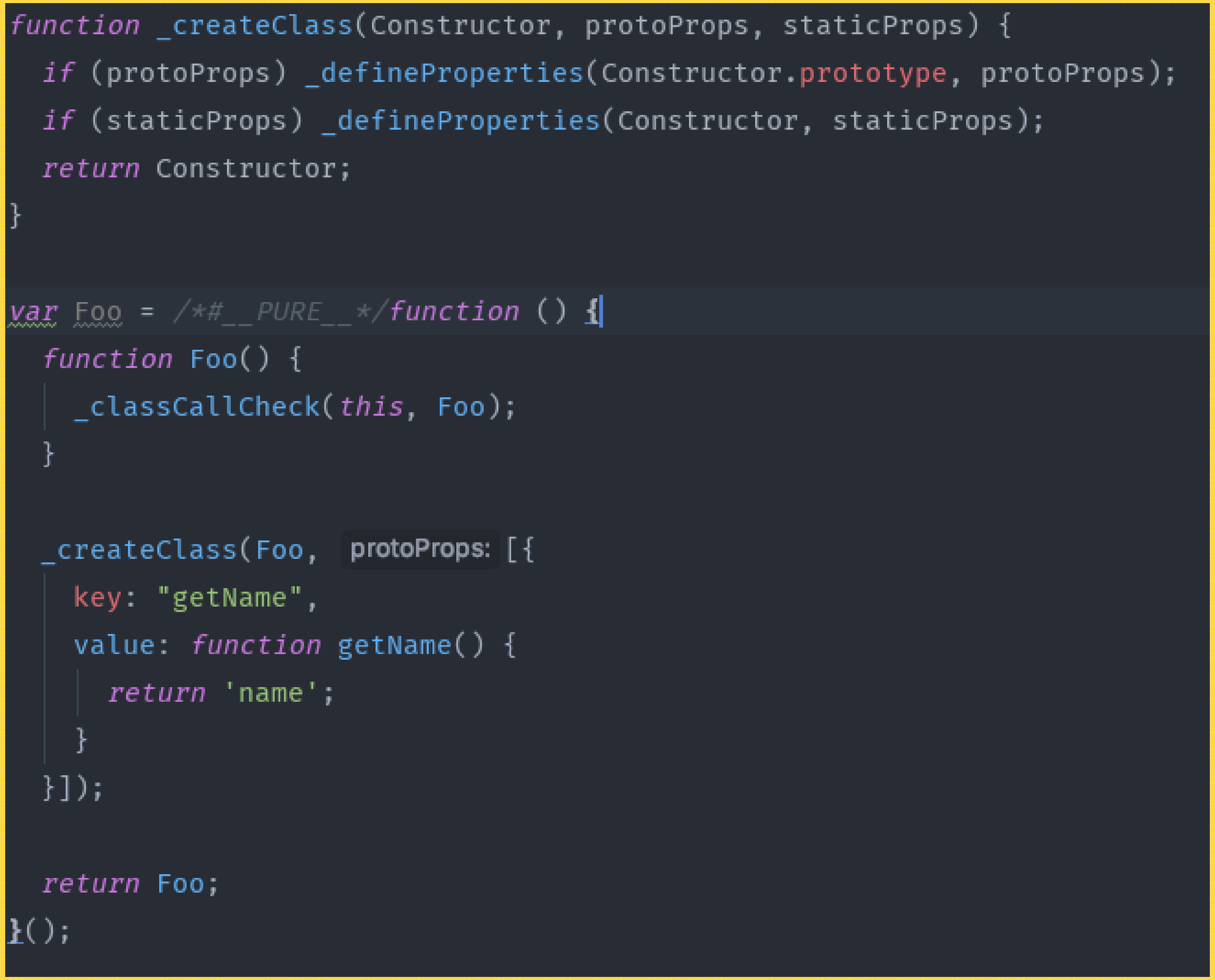
transform-runtime 默认配置

helpers: true, // 将常用 runtime 代码转换成工具类

corejs: false, // 将 corejs 相关代转换成变量

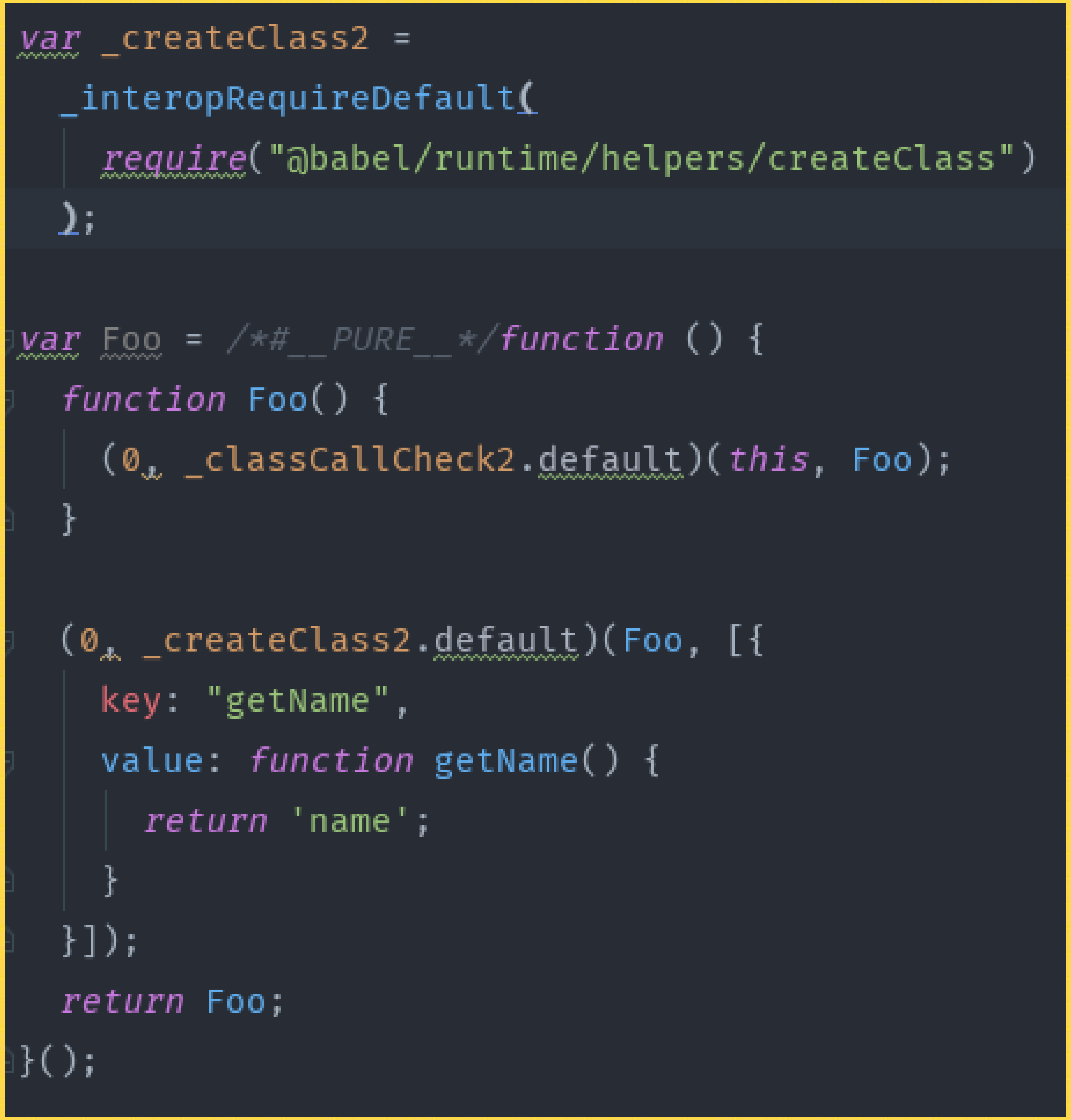
regenerator: true, // 将 generator 相关代转换成变量

helpers: false



```
function _createClass(Constructor, protoProps, staticProps) {
  if (protoProps) _defineProperties(Constructor.prototype, protoProps);
  if (staticProps) _defineProperties(Constructor, staticProps);
  return Constructor;
}

var Foo = /*#__PURE__*/function () {
  function Foo() {
    _classCallCheck(this, Foo);
  }

  _createClass(Foo,  protoProps: [{
    key: "getName",
    value: function getName() {
      return 'name';
    }
  }]);

  return Foo;
}();
```

helpers: true



```
var _createClass2 =
  _interopRequireDefault(
    require("@babel/runtime/helpers/createClass")
  );

var Foo = /*#__PURE__*/function () {
  function Foo() {
    (0, _classCallCheck2.default)(this, Foo);
  }

  (0, _createClass2.default)(Foo, [{
    key: "getName",
    value: function getName() {
      return 'name';
    }
  }]);
  return Foo;
}();
```

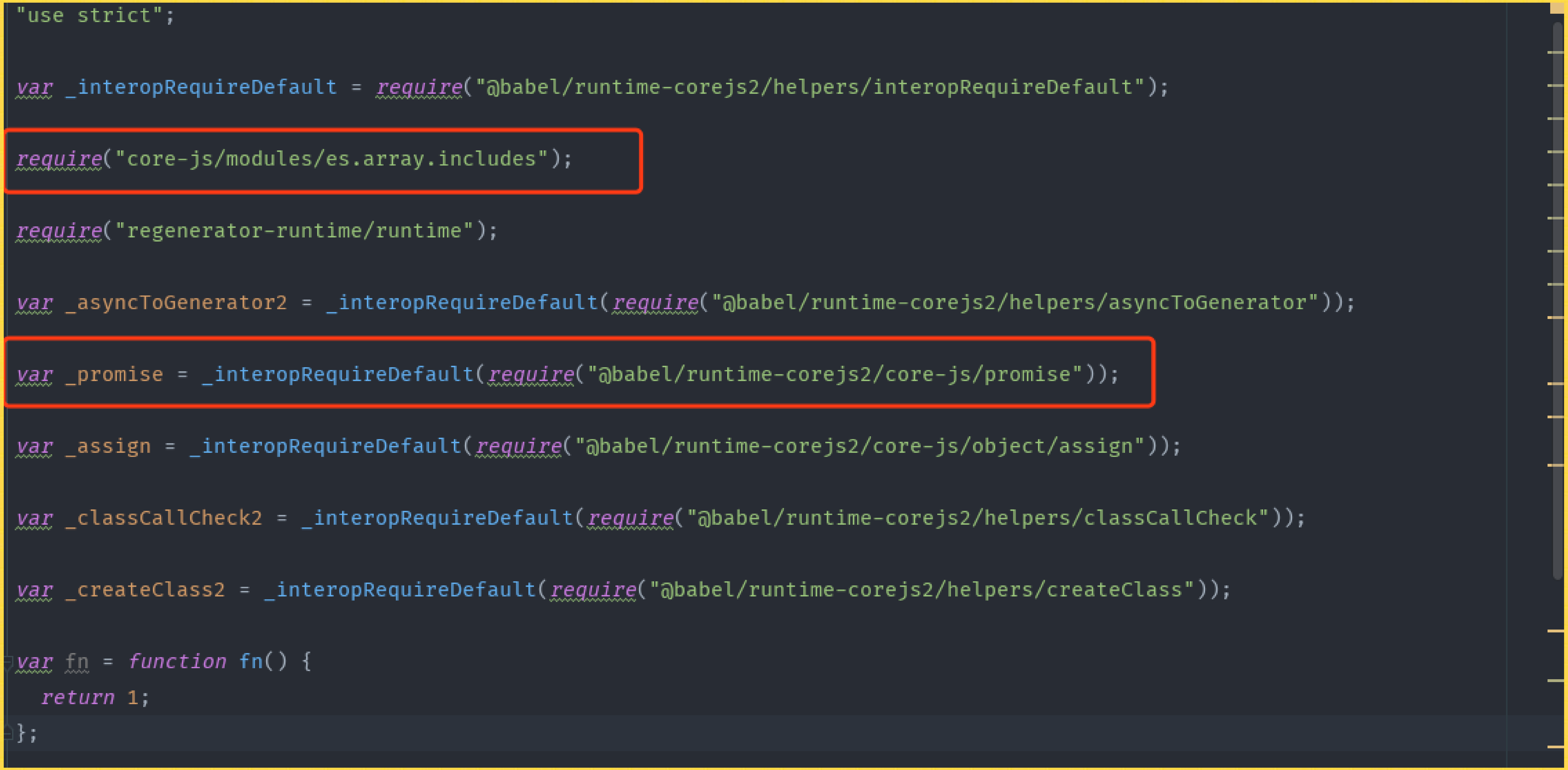
babel-runtime

- @babel/runtime

- @babel/runtime-corejs2
  = runtime + 全局 API

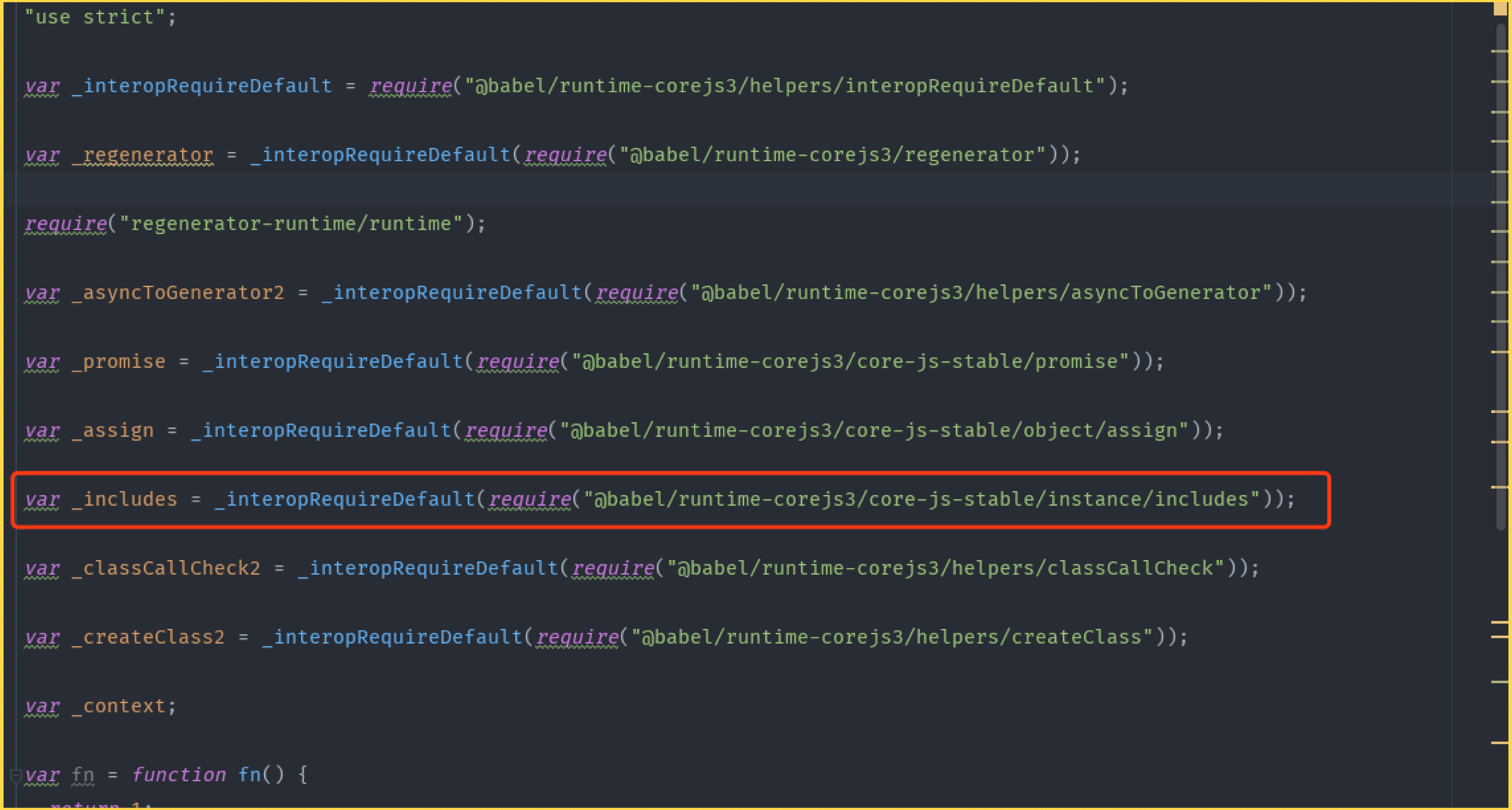
- @babel/runtime-corejs3
  = runtime-corejs2 + 实例 API

runtime-corejs2



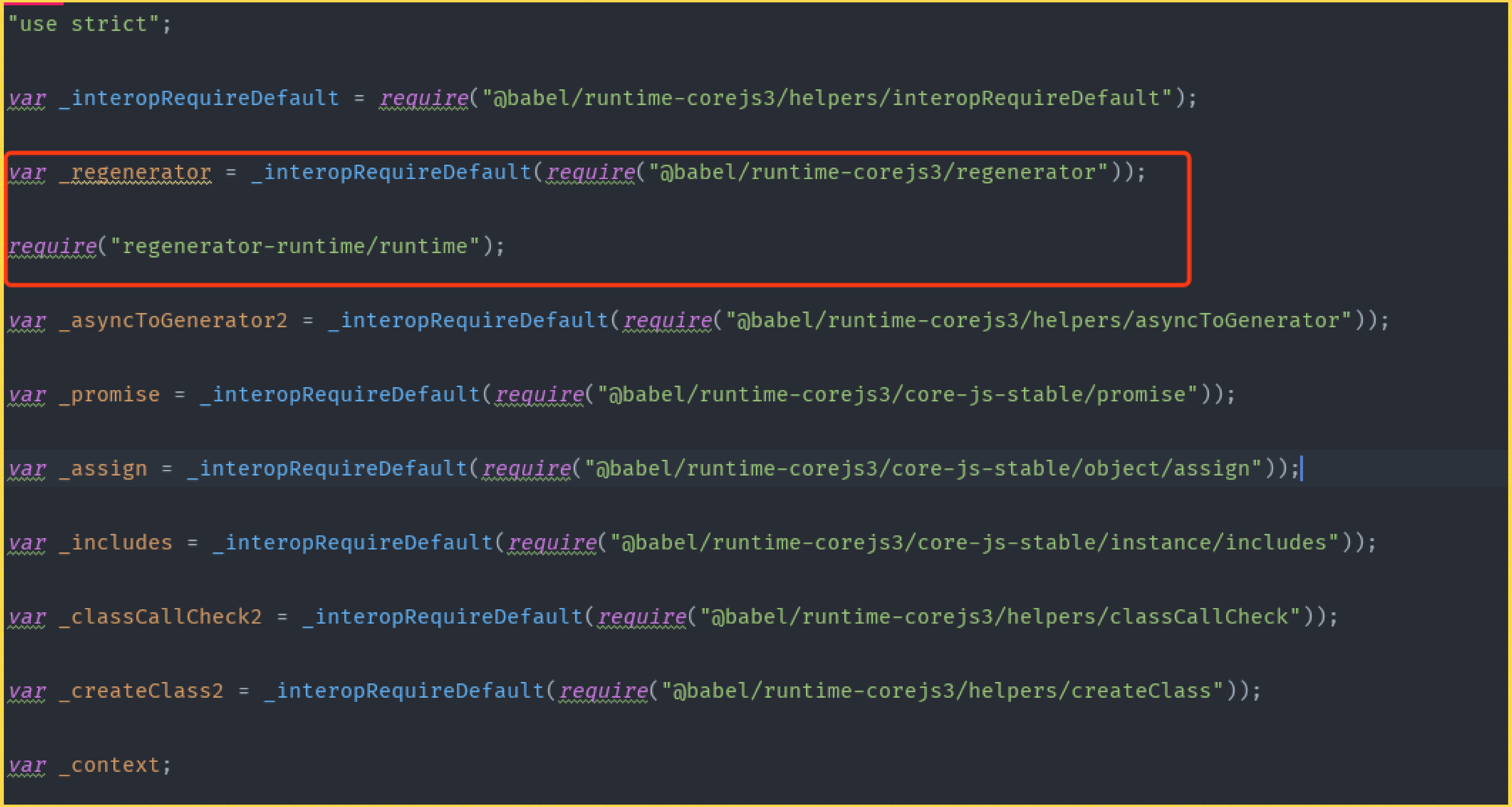
```
"use strict";

var _interopRequireDefault = require("@babel/runtime-corejs2/helpers/interopRequireDefault");

require("core-js/modules/es.array.includes");

require("regenerator-runtime/runtime");

var _asyncToGenerator2 = _interopRequireDefault(require("@babel/runtime-corejs2/helpers/asyncToGenerator"));

var _promise = _interopRequireDefault(require("@babel/runtime-corejs2/core-js/promise"));

var _assign = _interopRequireDefault(require("@babel/runtime-corejs2/core-js/object/assign"));

var _classCallCheck2 = _interopRequireDefault(require("@babel/runtime-corejs2/helpers/classCallCheck"));

var _createClass2 = _interopRequireDefault(require("@babel/runtime-corejs2/helpers/createClass"));

var fn = function fn() {
  return 1;
};
```

runtime-corejs3



```
"use strict";

var _interopRequireDefault = require("@babel/runtime-corejs3/helpers/interopRequireDefault");

var _regenerator = _interopRequireDefault(require("@babel/runtime-corejs3/regenerator"));

require("regenerator-runtime/runtime");

var _asyncToGenerator2 = _interopRequireDefault(require("@babel/runtime-corejs3/helpers/asyncToGenerator"));

var _promise = _interopRequireDefault(require("@babel/runtime-corejs3/core-js-stable/promise"));

var _assign = _interopRequireDefault(require("@babel/runtime-corejs3/core-js-stable/object/assign"));

var _includes = _interopRequireDefault(require("@babel/runtime-corejs3/core-js-stable/instance/includes"));

var _classCallCheck2 = _interopRequireDefault(require("@babel/runtime-corejs3/helpers/classCallCheck"));

var _createClass2 = _interopRequireDefault(require("@babel/runtime-corejs3/helpers/createClass"));

var _context;

var fn = function fn() {
```

regenerator: true



```
"use strict";

var _interopRequireDefault = require("@babel/runtime-corejs3/helpers/interopRequireDefault");

var _regenerator = _interopRequireDefault(require("@babel/runtime-corejs3/regenerator"));

require("regenerator-runtime/runtime");

var _asyncToGenerator2 = _interopRequireDefault(require("@babel/runtime-corejs3/helpers/asyncToGenerator"));

var _promise = _interopRequireDefault(require("@babel/runtime-corejs3/core-js-stable/promise"));

var _assign = _interopRequireDefault(require("@babel/runtime-corejs3/core-js-stable/object/assign"));

var _includes = _interopRequireDefault(require("@babel/runtime-corejs3/core-js-stable/instance/includes"));

var _classCallCheck2 = _interopRequireDefault(require("@babel/runtime-corejs3/helpers/classCallCheck"));

var _createClass2 = _interopRequireDefault(require("@babel/runtime-corejs3/helpers/createClass"));

var _context;
```

BUG https://github.com/babel/babel/issues/10759

两种 polyfill 做法对比

- 全局污染 polyfill ：
  - 配合 useBuiltIns:"usage" 能得到体积较小的包
  - 依赖 window/global 对象
  - 适用于应用开发

- transform polyfill ：
  - 不会有全局污染
  - 一般情况下包体积比全局污染式 polyfill 做法大
  - 不依赖 window/global 对象
  - 适用于第三方库、无 window/global 对象的应用开发

更好的 polyfill 方案

根据 ua 返回不同的包

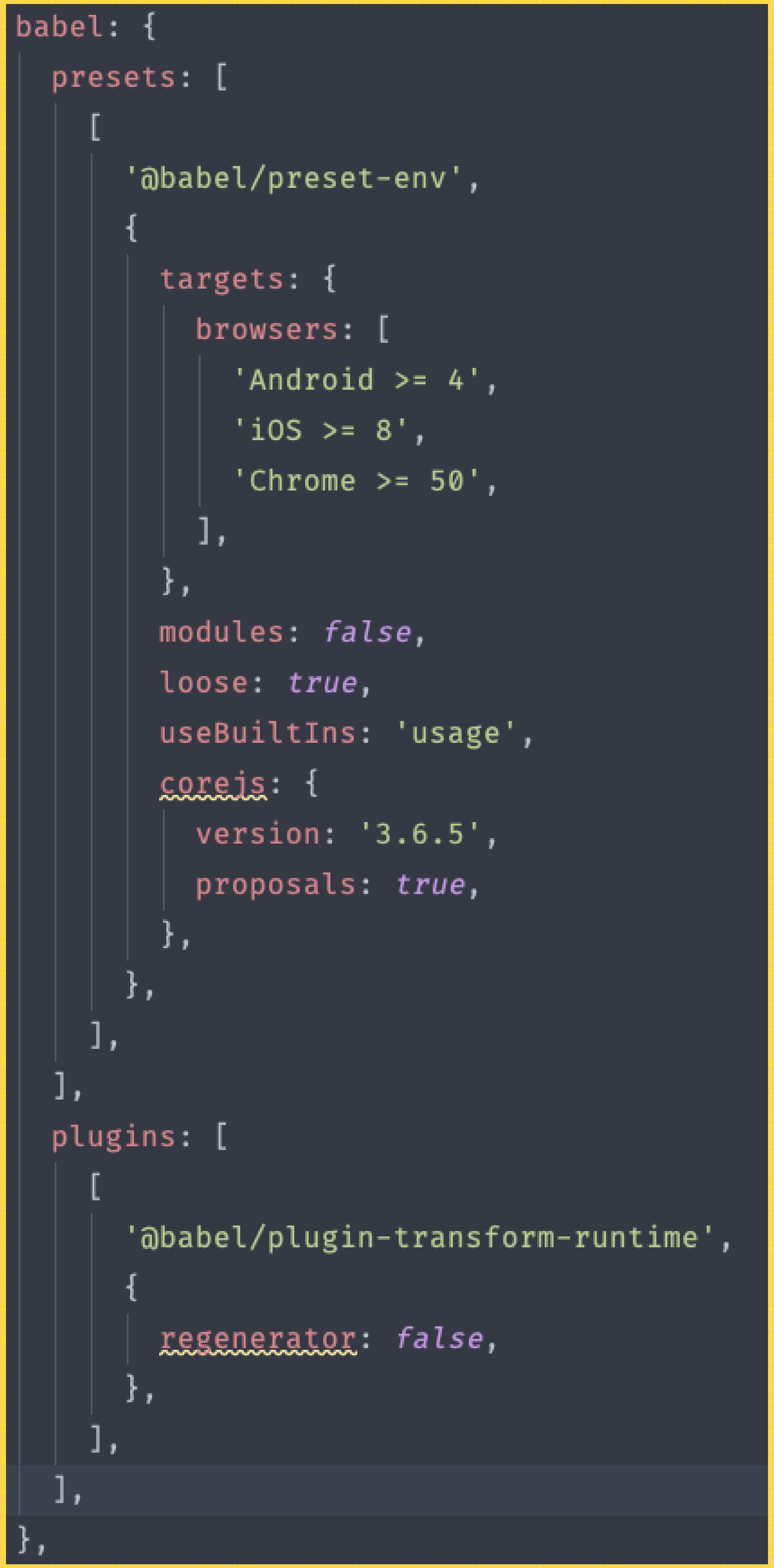
Babel 7 变化概览

- Yearly Preset Deprecations
- Stage Preset Deprecations
- Package Renames
- Switch to -proposal- for TC39 Proposals
- @babel/polyfill Deprecations

Web 项目配置

遇到的坑：

- 不是通过 npm 安装的包注意要使用已经转换成 ES5 的包



```
babel: {
  presets: [
    [
      '@babel/preset-env',
      {
        targets: {
          browsers: [
            'Android >= 4',
            'iOS >= 8',
            'Chrome >= 50',
          ],
        },
        modules: false,
        loose: true,
        useBuiltIns: 'usage',
        corejs: {
          version: '3.6.5',
          proposals: true,
        },
      },
    ],
  ],
  plugins: [
    [
      '@babel/plugin-transform-runtime',
      {
        regenerator: false,
      },
    ],
  ],
},
```

小程序 polyfill 方案1

全局污染 polyfill

小程序环境下 webpack 获取 window（获取到了一个空对象{}）失败导致 polyfill 失败

小程序 polyfill 方案2

transform-runtime

MR 链接

遇到的坑
- es module、commonJs 混用时，需要设置 modules: "commonjs"
- 针对低版本浏览器，只能使用 core-js2 7.3.0 以下版本
- 像 vant 这样的 ES6 包（以及该包所依赖的 ES6 包）需要配置 exclude 进行 babel 转换
- 如果需要转换第三方包，使用 babel.config.json

感谢

- Babel Playground
- Babel 官方文档
- Babel 中文文档
- Babel6: loose mode
- ORLY Cover Generator
- 致我们学前端的小时光—corejs与env、runtime的不解之缘