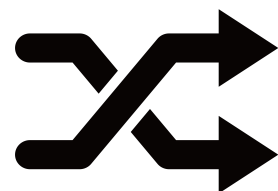


```
JS index.js x
1  const a = 10;
```

ES6



ES5

```
JS index.js x
1  "use strict";
2
3  var a = 10;
```

# BABEL

前端科普系列——

## Babel: 把 ES6 送上天的通天塔

1996.8

1997.7

2000

ECMAScript 6 正式发布  
并且更名为  
“ECMAScript  
2015”

2015.6







# BABEL

## THE BIRTHPLACE OF DIVERSITY



# What`s Babel ?


Babel is a JavaScript compiler.









Use next generation JavaScript, today.

# How to use Babel ?

Shell

```
npm init -y  
npm install --save-dev @babel/core @babel/cli @babel/preset-env
```

✓  babel

- ✓  demo
  - >  node\_modules
  - ✓  src
    -  index.js
    -  package-lock.json
    -  package.json
  - >  img
-  README.md

babel.config.js

```
const presets = [  
  [  
    '@babel/env',  
    {  
      debug: true  
    }  
  ]  
]  
  
const plugins = []  
module.exports = { presets, plugins }
```

## package.json

```
{
  "name": "demo",
  "version": "1.0.0",
  "scripts": {
    "babel": "babel src --out-dir dist"
  },
  "devDependencies": {
    "@babel/cli": "^7.8.4",
    "@babel/core": "^7.9.0",
    "@babel/preset-env": "^7.9.0"
  }
}
```



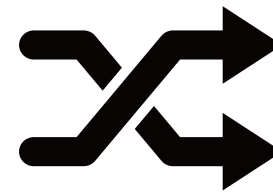
src/index.js

```
const add = (a, b) => a + b
```

```
const arr = [1, 2]
```

```
const hasThreee = arr.includes(3)
```

```
new Promise(resolve=>resolve(10))
```



dist/index.js

```
var add = function add(a, b) {  
  return a + b;  
};
```

```
var arr = [1, 2];
```

```
var hasThreee = arr.includes(3);
```

```
new Promise(function (resolve) {  
  return resolve(10);  
});
```

Why?

# How does Babel work ?

- workflow
- components

Code

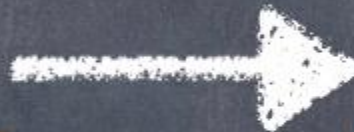


@babel  
/parser

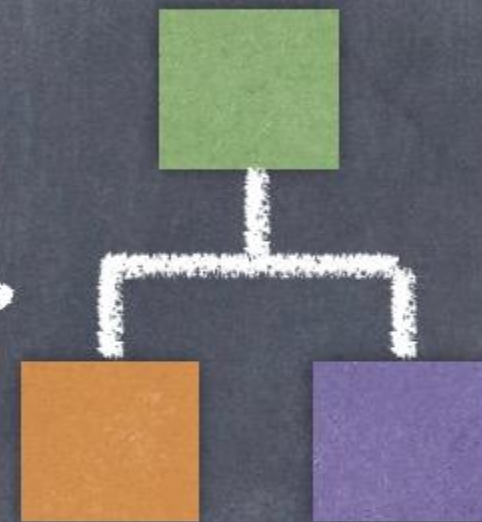


AST

Transform



@babel  
/traverse



AST

Generator

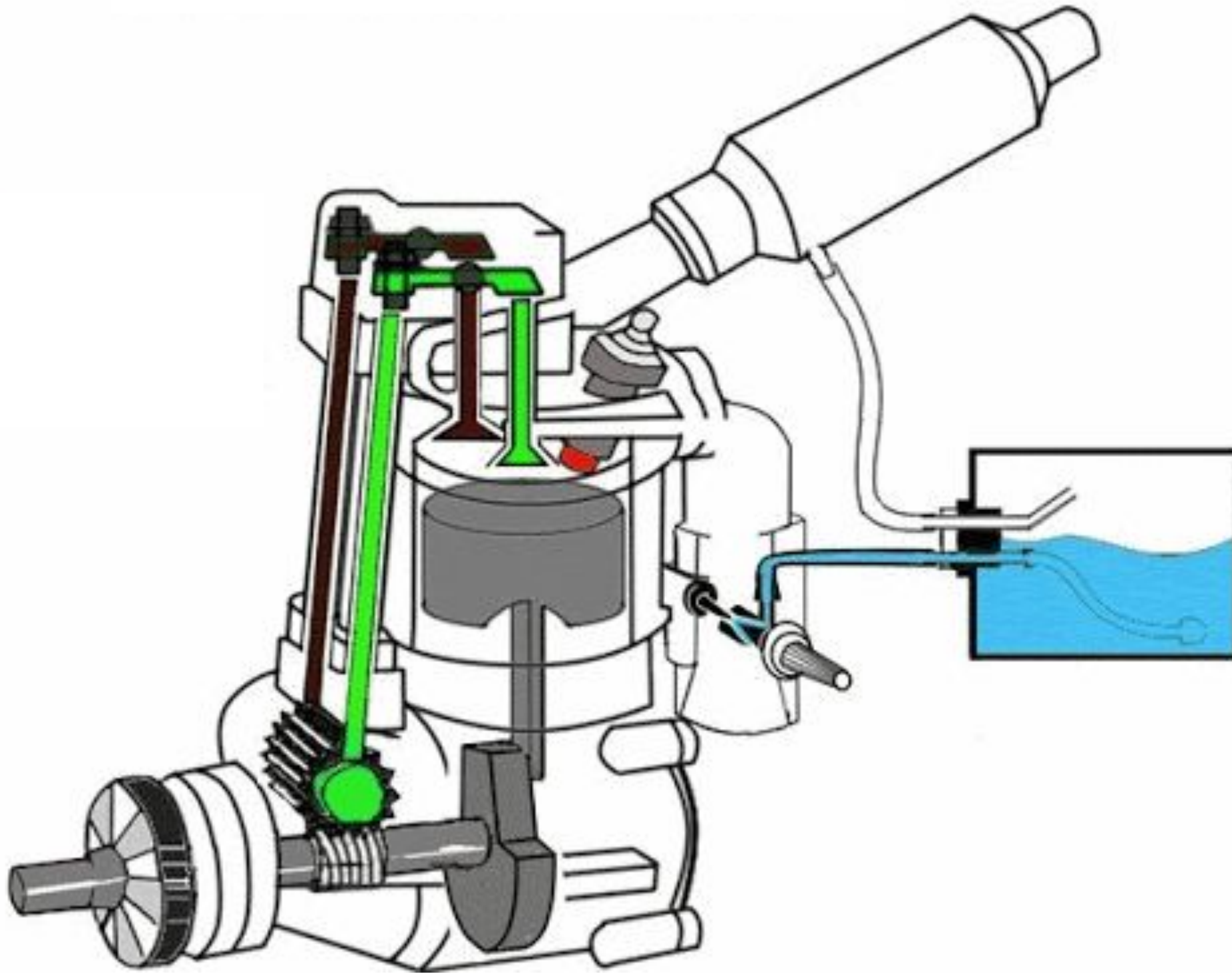


@babel  
/generator

Code







**@babel/core**

the core of transformer for AST

**@babel/cli**

a built-in CLI to compile files from the command line

**@babel/plugin-xxx**

plugin for corresponding syntax in transformation

**@babel/preset-env**

plugins to use, avoid configuration one by one

~~@babel/polyfill~~ --> **core-js**

use to polyfill features, but has been deprecated

**@babel/plugin-transform-runtime**

re-use of Babel's injected helper code to save on codesize, to create a sandboxed environment avoid global pollution

**@babel/preset-env**



```
> babel src --out-dir dist
```

```
@babel/preset-env: `DEBUG` option
```

```
Using targets:  
{}
```

```
Using modules transform: auto
```

```
Using plugins:
```

```
{ SyntaxError: /Users/kongchuiliang/Documents/front-end/babel/demo/src/index.js: Support for the experimental syntax 'classProperties' isn't currently enabled (4:12):
```

```
2 |  
3 | class Person {  
> 4 |   static a = 'a';  
   |         ^  
5 |   static b;  
6 |   name = 'morrain';  
7 |   age = 18
```

```
Add @babel/plugin-proposal-class-properties (https://git.io/vb4SL) to the 'plugins' section of your Babel config to enable transformation.
```

```
at Parser._raise (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:742:17)  
at Parser.raiseWithData (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:735:17)  
at Parser.expectPlugin (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:8762:18)  
at Parser.parseClassProperty (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:12110:12)  
at Parser.pushClassProperty (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:12070:30)  
at Parser.parseClassMemberWithIsStatic (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:12003:14)  
at Parser.parseClassMember (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:11940:10)  
at withTopicForbiddingContext (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:11885:14)  
at Parser.withTopicForbiddingContext (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:10956:14)  
at Parser.parseClassBody (/Users/kongchuiliang/Documents/front-end/babel/demo/node_modules/@babel/parser/lib/index.js:11862:10)
```

```
loc: Position { line: 4, column: 11 },  
pos: 55,  
missingPlugin: [ 'classProperties' ],  
code: 'BABEL_PARSE_ERROR' }
```

```
transform-new-target {}  
transform-regenerator {}  
transform-member-expression-literals {}  
transform-property-literals {}  
transform-reserved-words {}  
transform-modules-commonjs {}  
proposal-dynamic-import {}
```

```
Using polyfills: No polyfills were added, since the `useBuiltIns` option was not set.  
Successfully compiled 1 file with Babel.
```



Shell

```
npm install --save-dev @babel/plugin-proposal-class-properties
```

babel.config.js

```
const presets = [  
  [  
    '@babel/env',  
    {  
      debug: true  
    }  
  ]  
]  
  
const plugins = ['@babel/plugin-proposal-class-properties']  
module.exports = { presets, plugins }
```

## targets

string | Array<string> | { [string]: string }, defaults to {} .

Describes the environments you support/target for your project.

This can either be a browserslist-compatible query (with caveats):

JSON

 Copy

```
{
  "targets": "> 0.25%, not dead"
}
```

Or an object of minimum environment versions to support:

JSON

 Copy

```
{
  "targets": {
    "chrome": "58",
    "ie": "11"
  }
}
```

Only compile syntaxes which are not supported by your target environment

If no targets specified, [@babel/preset-env](#) will transform all ECMAScript 2015+ code by default

Install

How Does it Work?

Browserslist Integration

Options

**targets**

bugfixes

spec

loose

modules

debug

include

exclude

useBuiltIns

corejs

forceAllTransforms

configPath

ignoreBrowserslistConfig

shippedProposals

Caveats

Ineffective browserslist queries

**@babel/polyfill**



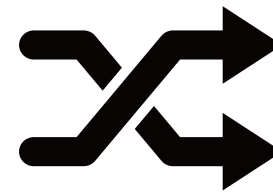
src/index.js

```
const add = (a, b) => a + b
```

```
const arr = [1, 2]
```

```
const hasThreee = arr.includes(3)
```

```
new Promise(resolve=>resolve(10))
```



dist/index.js

```
var add = function add(a, b) {  
  return a + b;  
};
```

```
var arr = [1, 2];
```

```
var hasThreee = arr.includes(3);
```

```
new Promise(function (resolve) {  
  return resolve(10);  
});
```

Why?

# syntax + built-in

syntax

const                      class  
  
...                      =>  
  
import                      let

built-in

Object.assign  
  
Promise  
  
Array.prototype.includes  
  
Array.from  
  
WeakMap



As of Babel 7.4.0, @babel/polyfill has been deprecated in favor of directly including core-js to polyfill ECMAScript features

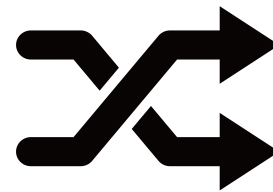
babel.config.js

```
const presets = [
  [
    '@babel/env',
    {
      debug: true,
      useBuiltIns: 'usage',
      corejs: 3
    }
  ]
]
```



src/index.js

```
const arr = [1, 2]
const hasThreee = arr.includes(3)
new Promise(resolve=>resolve(10))
```



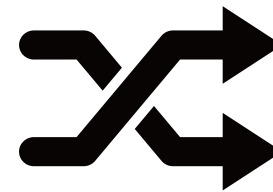
dist/index.js

```
require("core-js/modules/es6.promise");
require("core-js/modules/
                                es7.array.includes");
var arr = [1, 2];
var hasThreee = arr.includes(3);
new Promise(function (resolve) {
    return resolve(10);
});
```

**@babel/plugin-transform-runtime**

src/index.js

```
class Person {  
  static a = 1;  
  static b;  
  name = 'morrain';  
  age = 18  
}
```



dist/index.js

```
function _classCallCheckCheck(ins, Constructor) {...}  
function _defineProperty(obj, key, value) {...}  
var Person = function Person() {  
  _classCallCheckCheck(this, Person);  
  _defineProperty(this, "name", 'morrain');  
  _defineProperty(this, "age", 18);  
};  
_defineProperty(Person, "a", 1);  
_defineProperty(Person, "b", void 0);
```

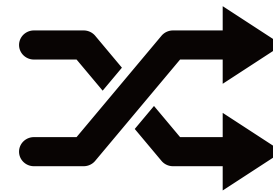


## Shell

```
npm install --save-dev @babel/plugin-transform-runtime
npm install --save @babel/runtime
```

## src/index.js

```
class Person {
  static a = 1;
  static b;
  name = 'morrain';
  age = 18
}
```



## dist/index.js

```
var _classCallCheck =
require("@babel/runtime/helpers/classCallCheck")
var _defineProperty =
require("@babel/runtime/helpers/defineProperty")
var Person = function Person() {
  _classCallCheck(this, Person);
  _defineProperty(this, "name", 'morrain');
  _defineProperty(this, "age", 18);
};
_defineProperty(Person, "a", 1);
_defineProperty(Person, "b", void 0);
```

the built-ins core-js provides such as Promise, Set and Map, will  
pollute the global scope

```
Object.defineProperty(Array.prototype, 'includes', function() {  
    ...  
})
```



Shell

```
npm uninstall @babel/runtime
```

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

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

```
var ac  
  retu  
};
```

but it doesn't consider target environments

```
var arr = [1, 2];  
var hasThreee = (0, _includes["default"])(arr).call(arr, 3);  
new _promise["default"](function (resolve) {  
  return resolve(10);  
});
```

# Rethink polyfilling story

- Currently
- Concepts



Currently Babel has three different ways to inject core-js polyfills in the source code:

- By using `@babel/preset-env`'s `useBuiltIns: "entry"` option, it is possible to inject polyfills for every ECMAScript functionality not natively supported by the target browsers;
- By using `useBuiltIns: "usage"`, Babel will only inject polyfills for unsupported ECMAScript features but only if they are actually used in the input source code;
- By using `@babel/plugin-transform-runtime`, Babel will inject ponyfills (which are "pure" and don't pollute the global scope) for every used ECMAScript feature supported by core-js. This is usually used by library authors.

babel.config.js

```
const targets = ['>1%']
const presets = [
  [
    '@babel/env',
    {
      debug: true
    }
  ],
  '@babel/core'
]
const plugins = ['@babel/plugin-proposal-class-properties']
const polyfills = [
  [
    'corejs3',
    {
      method: 'usage-pure'
    }
  ],
  '@babel/core'
]

module.exports = { targets, presets, plugins, polyfills }
```

Polyfills:

- entry

@babel

- usage

@babel

- usage

@babel

# Q & A

敬请期待~~~~~

前端科普系列 之《webpack:它是如此,但远不止于此》

