Яндекс

Яндекс Карты

Безграничные возможности транспиляции

Андрей Роенко,
Разработчик API Яндекс.Карт

Транспиляция

```
function length({ x, y }) {
    return Math.sqrt(x ** 2 + y ** 2);
}
```

Транспиляция

```
"use strict";

function length(_ref) {
     var x = _ref.x,
         y = _ref.y;

     return Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
}
```

Транспиляция

Компиляция языка в другой язык с похожим набором абстракций или в другую версию того же языка.

- CoffeeScript => JavaScript
- SCSS / LESS / PostCSS => CSS

- ES6+ => ES5 (Babel)
- TypeScript => ES5

- 2to3: Python 2 => Python 3
- 3to2: Python 3 => Python 2

Зачем

- Всем хочется писать на ES6
  - Тем временем ES6 поддерживается (95+%) во всех зеленых браузерах http://kangax.github.io/compat-table/es6/

- Или на JavaScript'е, но с сахарком
- Или вообще не на JavaScript'е

А еще?

- Утилиты: prettier, eslint, tslint, ...
- jscodeshift: js-codemod, react-codemod, ...
- Плагины для Atom/vim/vs code: js-refactor, atom-beautify, ...

Babel

Магия

```
$ babel --presets=env -o output.js source.js
```

```
{ loader: 'babel-loader' }
```

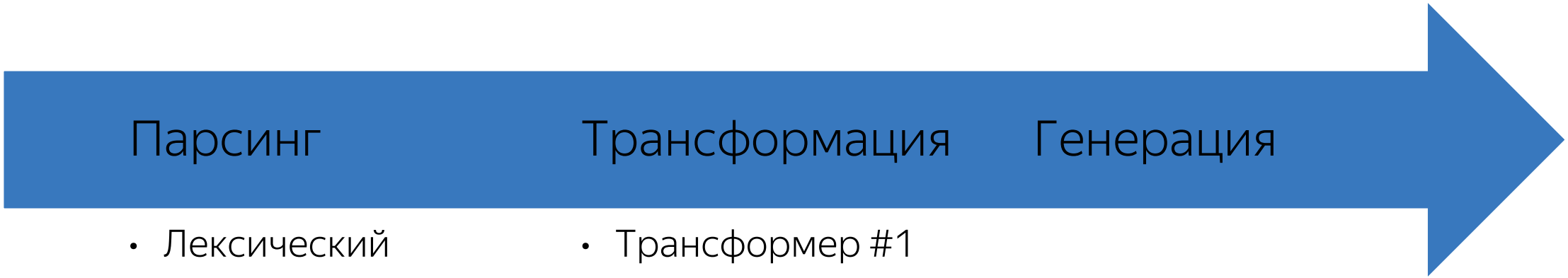
Заглянем под капот

```
const babel = require('@babel/core');

const source = '...';

const result = babel.transform(source, { presets: ['env'] });

console.log(result.code);

/*
"use strict";

function length(_ref) {
    var x = _ref.x,
        y = _ref.y;

    return Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
}
*/
```

Заглянем под капот

```
const babel = require('@babel/core');

const source = '...';

const result = babel.transform(source, { presets: ['env'] });

console.log(result.code);

/*
"use strict";

function length(_ref) {
    var x = _ref.x,
        y = _ref.y;

    return Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
}
*/
```

А под капотом капота



- Лексический анализ
- Семантический анализ

- Трансформер #1
- Трансформер #2
- ...
- Трансформер #N

Лексический анализ

Разбор текста на логические сущности – токены

- Числа
- Идентификаторы
- Строки
- Скобки
- ...

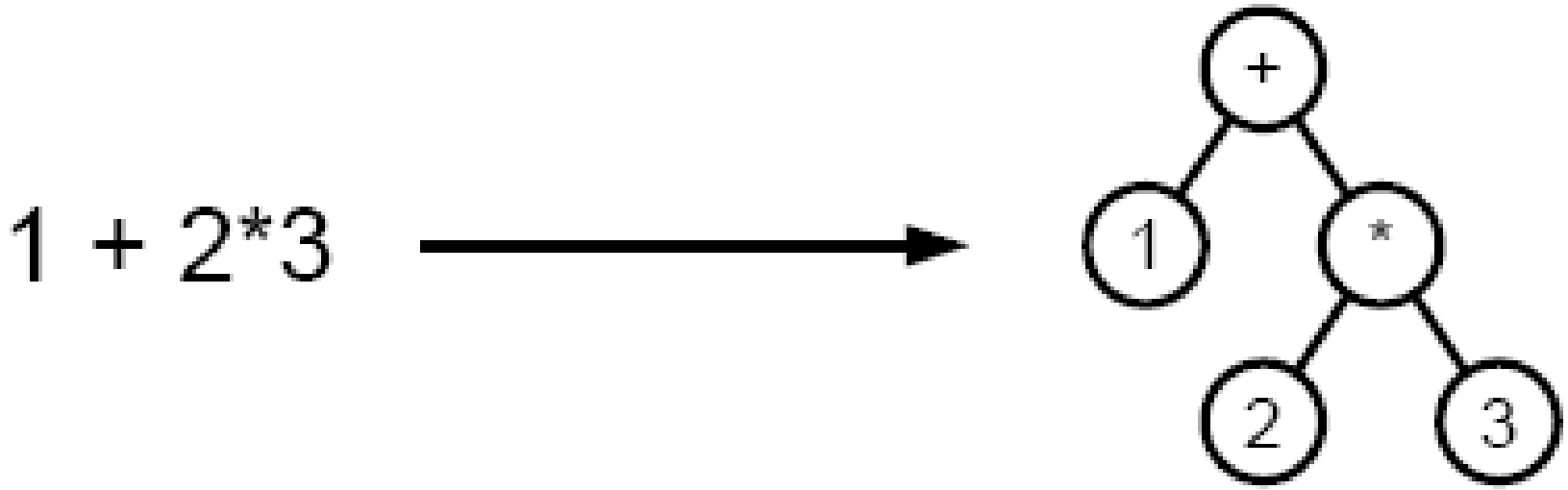
Лексический анализ

```
function length({ x, y }) {
    return Math.sqrt(x ** 2 + y ** 2);
}
```

```
function
length
(
{
x
,
y
...
```

Синтаксический анализ

Создание абстрактного синтаксического дерева (AST) из потока токенов



```
1 function length({ x, y }) {
2   return Math.sqrt(x ** 2 + y ** 2);
3 }
```

Tree JSON 70ms

Autofocus Hide methods Hide empty keys Hide location data Hide type keys

```
- File {
  - program: Program {
      sourceType: "module"
    - body: [
      - FunctionDeclaration {
        + id: Identifier {name}
          generator: false
          expression: false
          async: false
        + params: [1 element]
        - body: BlockStatement {
          - body: [
            - ReturnStatement {
              - argument: CallExpression {
                + callee: MemberExpression {object, property, computed}
                + arguments: [1 element]
                }
              }
            ]
            directives: [ ]
          }
        }
      ]
```

astexplorer.net

Синтаксический анализ

```
function
length
(
{
x
,
y
}
)
{
return
// ...
}
```

```
- FunctionDeclaration {
   - id: Identifier {
        name: "length"
     }
     generator: false
     expression: false
     async: false
   + params: [1 element]
   - body: BlockStatement {
      - body: [
         + ReturnStatement {argument}
        ]
        directives: [ ]
     }
  }
```

Синтаксический анализ

```
Math
.
sqrt
(
x
**
2
+
y
**
2
)
```

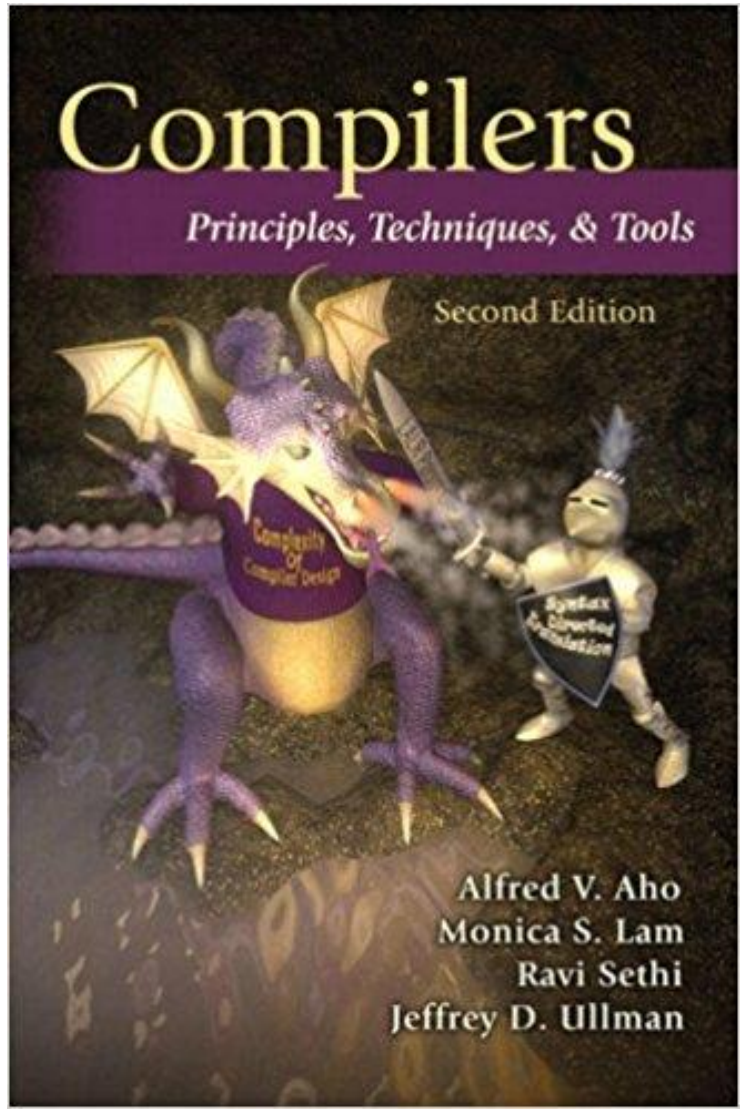
```
- argument: CallExpression {
   - callee: MemberExpression {
      - object: Identifier {
           name: "Math"
        }
      - property: Identifier {
           name: "sqrt"
        }
        computed: false
     }
   - arguments: [
      - BinaryExpression = $node {
         + left: BinaryExpression {left, operator, right}
           operator: "+"
         + right: BinaryExpression {left, operator, right}
        }
     ]
```

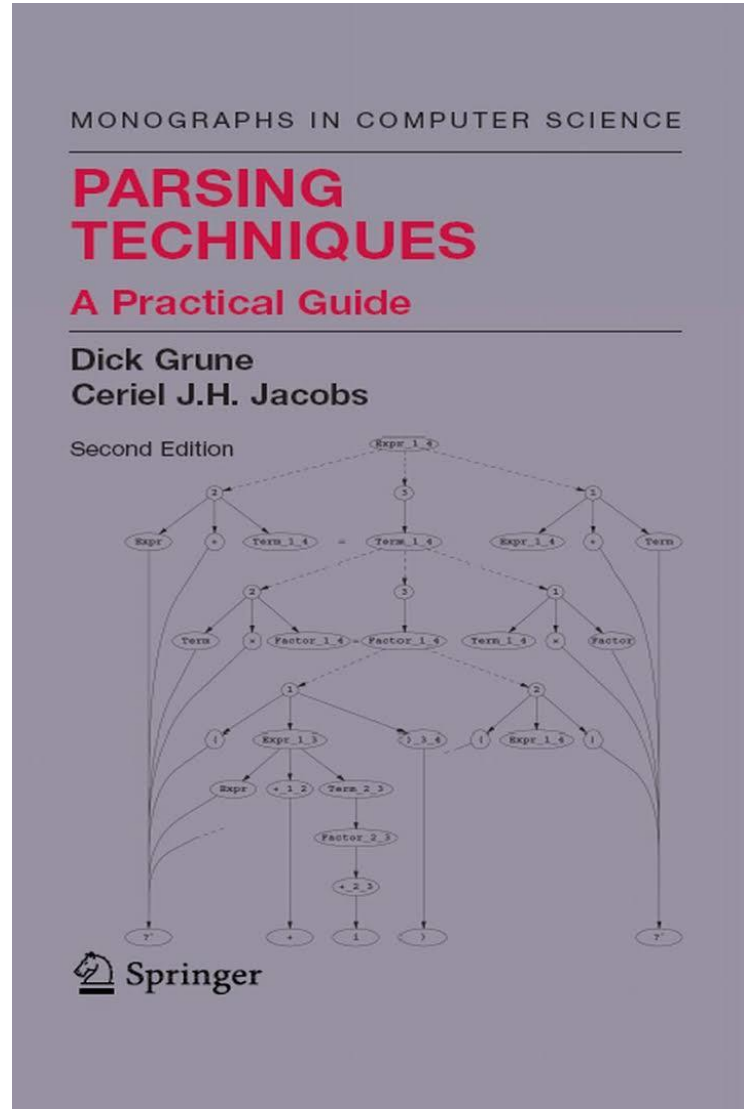
Синтаксический анализ

```
x
**
2
```

```
- left: BinaryExpression {
   - left: Identifier = $node {
        name: "x"
     }
     operator: "**"
   - right: NumericLiteral {
      + extra: {rawValue, raw}
        value: 2
     }
  }
```

en.wikipedia.org/wiki/Parsing


Compilers
Principles, Techniques, & Tools
Second Edition
Alfred V. Aho
Monica S. Lam
Ravi Sethi
Jeffrey D. Ullman


MONOGRAPHS IN COMPUTER SCIENCE
PARSING TECHNIQUES
A Practical Guide
Dick Grune
Ceriel J.H. Jacobs
Second Edition
Springer

Трансформация

Преобразование одного синтаксического дерева в другое

```
x ** 2
```

```
Math.pow(x, 2)
```

```
<A> ** <B>
```

```
Math.pow(<A>, <B>)
```

Трансформация

```
- left: BinaryExpression {
   - left: Identifier = $node {
        name: "x"
     }
     operator: "**"
   - right: NumericLiteral {
      + extra: {rawValue, raw}
        value: 2
     }
  }
```

```
- left: CallExpression {
   - callee: MemberExpression {
      - object: Identifier {
           name: "Math"
        }
      - property: Identifier {
           name: "pow"
        }
        computed: false
     }
   - arguments: [
      - Identifier {
           name: "x"
        }
      - NumericLiteral {
         + extra: {rawValue, raw}
           value: 2
```

Генерация

```
- FunctionDeclaration {
   - id: Identifier {
        name: "length"
     }
     generator: false
     expression: false
     async: false
   + params: [1 element]
   - body: BlockStatement {
      - body: [
         + ReturnStatement {argument}
        ]
        directives: [ ]
     }
  }
```

```
"use strict";

function length(_ref) {
    var x = _ref.x,
        y = _ref.y;

    return Math.sqrt(Math.pow(x, 2) +
                     Math.pow(y, 2));
}
```

Где все это в Babel

Парсинг

```
const babelParser = require('@babel/parser');

const code = 'function length({ x, y }) { ... }';

const ast = babelParser.parse(code);


const exprAst = babelParser.parseExpression('1+2*fn()');
```

Парсинг

```
const babelParser = require('@babel/parser');

const code = 'function length({ x, y }) { ... }';

const ast = babelParser.parse(code);

const exprAst = babelParser.parseExpression('1+2*fn()');
```

Парсинг

```
const babelParser = require('@babel/parser');

const code = 'function length({ x, y }) { ... }';

const ast = babelParser.parse(code);

const exprAst = babelParser.parseExpression('1+2*fn()');
```

Ограничения

- Нельзя менять синтаксис

babel-plugin-syntax-decorators

```
parserOpts.plugins.push(
  legacy
    ? "decorators-legacy"
    : ["decorators", { decoratorsBeforeExport }],
);
```

Ограничения

- Зато есть куча прикольных синтаксических конструкций:

Метки

```
foobar: {
    const x = 10;
    break foobar;
    console.log(x);
}
```

Строки aka директивы

```
function foo() {
    "use strict";
    "my own magic";
}
```

Работа с AST

```
- FunctionDeclaration {
    type: "FunctionDeclaration"
  + id: Identifier {type, name}
    generator: false
    async: false
    params: [ ]
  + body: BlockStatement {type, body, directives}
  }
```

AST-хелперы

```
const t = require('@babel/types');

const newAst = t.callExpression(fnAst, [ast1, ast2]);

if (t.isCallExpression(ast)) {
    // ...
}

t.assertCallExpression(ast);
```

AST-хелперы

```
const t = require('@babel/types');

const newAst = t.callExpression(fnAst, [ast1, ast2]);

if (t.isCallExpression(ast)) {
    // ...
}

t.assertCallExpression(ast);
```

AST-хелперы

```
const t = require('@babel/types');

const newAst = t.callExpression(fnAst, [ast1, ast2]);

if (t.isCallExpression(ast)) {
    // ...
}

t.assertCallExpression(ast);
```

AST-хелперы

```
const t = require('@babel/types');

const newAst = t.callExpression(fnAst, [ast1, ast2]);

if (t.isCallExpression(ast)) {
    // ...
}

t.assertCallExpression(ast);
```

Трансформация

```
- left: BinaryExpression {
   - left: Identifier = $node {
        name: "x"
     }
     operator: "**"
   - right: NumericLiteral {
      + extra: {rawValue, raw}
        value: 2
     }
  }
```

```
- left: CallExpression {
   - callee: MemberExpression {
      - object: Identifier {
           name: "Math"
        }
      - property: Identifier {
           name: "pow"
        }
        computed: false
     }
   - arguments: [
      - Identifier {
           name: "x"
        }
      - NumericLiteral {
         + extra: {rawValue, raw}
           value: 2
```

Трансформация

```
const traverse = require('@babel/traverse');

const newAst = traverse(ast, {
    CallExpression(path) {
        path.parent
        path.node
        path.get('callee')
    }
    FunctionDeclaration(path) {
        if (path.node.name) {
            path.skip()
        }
    }
});
```

Трансформация

```
const traverse = require('@babel/traverse');

const newAst = traverse(ast, {
    CallExpression(path) {
        path.parent
        path.node
        path.get('callee')
    }
    FunctionDeclaration(path) {
        if (path.node.name) {
            path.skip()
        }
    }
});
```

Трансформация

```
const traverse = require('@babel/traverse');

const newAst = traverse(ast, {
    CallExpression(path) {
        path.parent
        path.node
        path.get('callee')
    }
    FunctionDeclaration(path) {
        if (path.node.name) {
            path.skip()
        }
    }
});
```

Трансформация

```
const traverse = require('@babel/traverse');

const newAst = traverse(ast, {
    CallExpression(path) {
        path.parent
        path.node
        path.get('callee')
    }
    FunctionDeclaration(path) {
        if (path.node.name) {
            path.skip()
        }
    }
});
```

Трансформация

```
const traverse = require('@babel/traverse');

const newAst = traverse(ast, {
    CallExpression(path) {
        path.parent
        path.node
        path.get('callee')
    }
    FunctionDeclaration(path) {
        if (path.node.name) {
            path.skip()
        }
    }
});
```

Скоуп

```
const x = 10;

function a(x = 20) {
    for (let x = 0; x < 30; x++) {
        /* ... */
    }
    function b() {
        console.log(x);
    }
    b()
}
a()
```

Скоуп

```
const x = 10;

function a(x = 20) {
    for (let x = 0; x < 30; x++) {
        /* ... */
    }
    function b() {
        console.log(x);
    }
    b()
}
a()
```

Скоуп

```
const x = 10;

function a(x = 20) {
    for (let x = 0; x < 30; x++) {
        /* ... */
    }
    function b() {
        console.log(x);
    }
    b()
}
a()
```

Скоуп

```
const x = 10;

function a(x = 20) {
    for (let x = 0; x < 30; x++) {
        /* ... */
    }
    function b() {
        console.log(x);
    }
    b()
}
a()
```

Скоуп

```
const x = 10;

function a(x = 20) {
    for (let x = 0; x < 30; x++) {
        /* ... */
    }
    function b() {
        console.log(x);
    }
    b()
}
a()
```

Скоуп

```
const x = 10;

function a(x = 20) {
    for (var x = 0; x < 30; x++) {
        /* ... */
    }
    function b() {
        console.log(x);
    }
    b()
}
a()
```

Скоуп

```
const newAst = traverse(ast, {
    Identifier(path) {
        path.scope.hasBinding('x')

        path.scope.hasOwnBinding('x')
        path.scope.parent

        path.scope.generateUidIdentifier('x')
        path.scope.rename('x', 'y')
    }
});
```

Скоуп

```
const newAst = traverse(ast, {
    Identifier(path) {
        path.scope.hasBinding('x')

        path.scope.hasOwnBinding('x')
        path.scope.parent

        path.scope.generateUidIdentifier('x')
        path.scope.rename('x', 'y')
    }
});
```

Скоуп

```
const newAst = traverse(ast, {
    Identifier(path) {
        path.scope.hasBinding('x')

        path.scope.hasOwnBinding('x')
        path.scope.parent

        path.scope.generateUidIdentifier('x')
        path.scope.rename('x', 'y')
    }
});
```

Скоуп

```
const newAst = traverse(ast, {
    Identifier(path) {
        path.scope.hasBinding('x')

        path.scope.hasOwnBinding('x')
        path.scope.parent

        path.scope.generateUidIdentifier('x')
        path.scope.rename('x', 'y')
    }
});
```

Скоуп

```
const newAst = traverse(ast, {
    Identifier(path) {
        path.scope.hasBinding('x')

        path.scope.hasOwnBinding('x')
        path.scope.parent

        path.scope.generateUidIdentifier('x')
        path.scope.rename('x', 'y')
    }
});
```

Изменение AST

```
const newAst = traverse(ast, {
    FunctionDeclaration(path) {
        path.remove()

        path.replaceWith(ast0)
        path.replaceWithMultiple([ast1, ast2])

        path.node.body
        path.node.body.push(ast1)
        path.node.body.unshift(ast2)
        path.node.body.splice(10, 1, ast3)
    }
});
```

Изменение AST

```
const newAst = traverse(ast, {
    FunctionDeclaration(path) {
        path.remove()

        path.replaceWith(ast0)
        path.replaceWithMultiple([ast1, ast2])

        path.node.body
        path.node.body.push(ast1)
        path.node.body.unshift(ast2)
        path.node.body.splice(10, 1, ast3)
    }
});
```

Изменение AST

```
const newAst = traverse(ast, {
    FunctionDeclaration(path) {
        path.remove()

        path.replaceWith(ast0)
        path.replaceWithMultiple([ast1, ast2])

        path.node.body
        path.node.body.push(ast1)
        path.node.body.unshift(ast2)
        path.node.body.splice(10, 1, ast3)
    }
});
```

Осторожно

Все нетипизированное, легко написать нерабочий код

- Тесты спасают

Можно сгенерировать невалидный AST

- Модуль cst – Concrete Syntax Tree

Генерация

```
const generate = require('@babel/generator');

const ast = babelParser.parse(fs.readFileSync('./index.js', 'utf8'));

console.log(generate(ast, {
    comments: true,
    minified: false
}, code).code);
```

Генерация

```
const generate = require('@babel/generator');

const ast = babelParser.parse(fs.readFileSync('./index.js', 'utf8'));

console.log(generate(ast, {
    comments: true,
    minified: false
}, code).code);
```

Генерация

```
const generate = require('@babel/generator');

const ast = babelParser.parse(fs.readFileSync('./index.js', 'utf8'));

console.log(generate(ast, {
    comments: true,
    minified: false
}, code).code);
```

Генерация

- Или вы можете обойти и сгенерировать все руками руками
- 6000 строк генерации в prettier
  prettier/src/language-js/printer-estree.js

Плагин для Babel

Плагин

```
function dropDebugger(babel) {
    const { types: t } = babel;

    return {
        name: 'drop-debugger',
        visitor: {
            DebuggerStatement(path) {
                path.remove();
            }
        }
    };
}

babel.transform(code, { plugins: [dropDebugger] })

babel.transform(code, { plugins: ['./dropDebugger'] })
                              require('./dropDebugger')
```

Плагин

```
function dropDebugger(babel) {
    const { types: t } = babel;

    return {
        name: 'drop-debugger',
        visitor: {
            DebuggerStatement(path) {
                path.remove();
            }
        }
    };
}

babel.transform(code, { plugins: [dropDebugger] })

babel.transform(code, { plugins: ['./dropDebugger'] })
                              require('./dropDebugger')
```

Плагин

```
function dropDebugger(babel) {
    const { types: t } = babel;

    return {
        name: 'drop-debugger',
        visitor: {
            DebuggerStatement(path) {
                path.remove();
            }
        }
    };
}

babel.transform(code, { plugins: [dropDebugger] })

babel.transform(code, { plugins: ['./dropDebugger'] })
                          require('./dropDebugger')
```

Что почитать

github.com/babel/babel

Babel Plugin Handbook

Исходники плагинов

Интересные плагины

Удаляет все assert’ы

- github.com/unassert-js/babel-plugin-unassert

Контрактное программирование

- github.com/codemix/babel-plugin-contracts

Макросы

- github.com/codemix/babel-plugin-macros

Весь github.com/babel/awesome-babel

Кейс API Яндекс.Карт

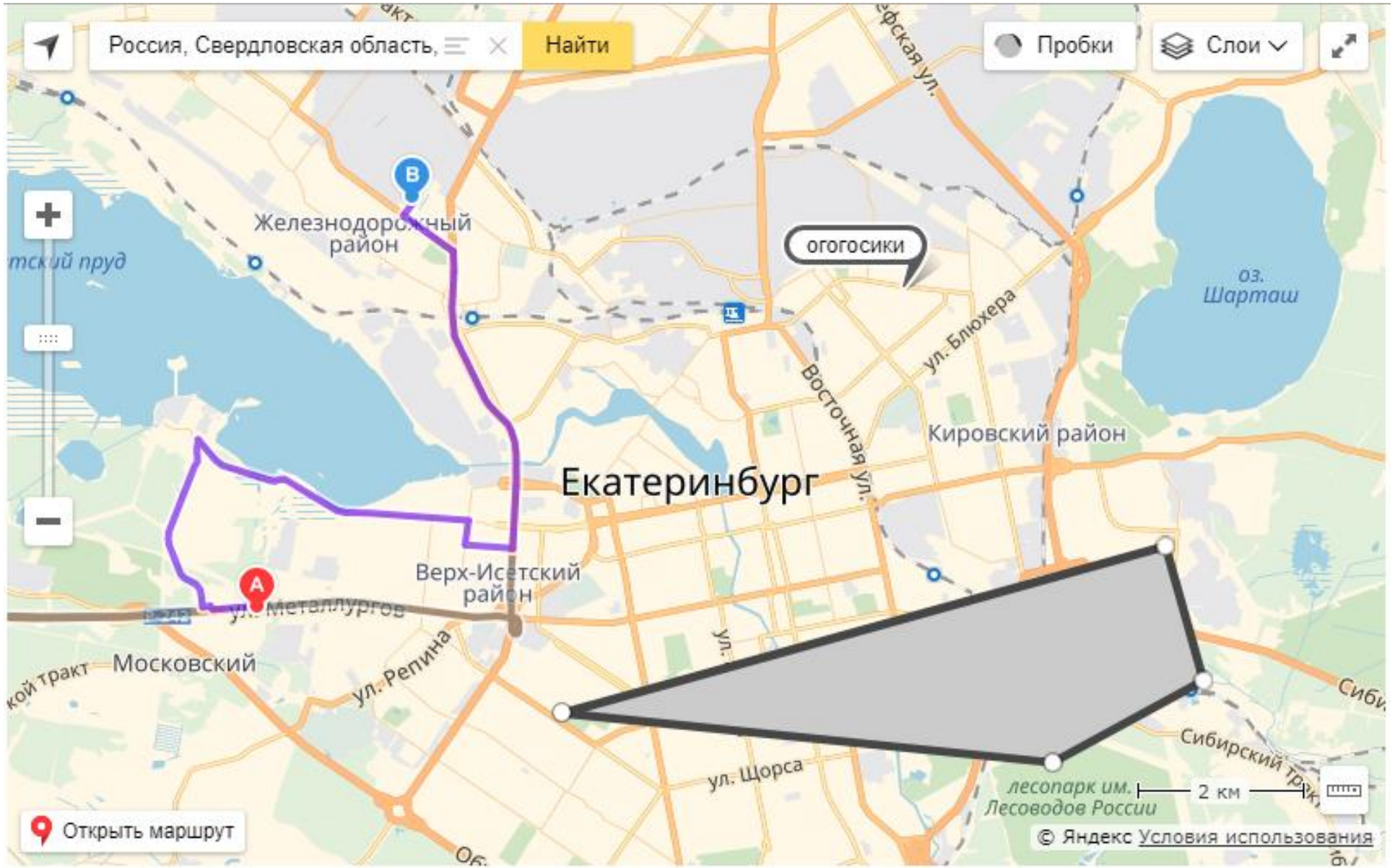
API Яндекс.Карт

```
<script src="https://api-maps.yandex.ru/2.1/?lang=ru_RU">
</script>
<script>
    ymaps.ready().then(function () {
        var map = new ymaps.Map(
            'map-div-id',
            { /* state */ },
            { /* options */ }
        );
    });
</script>
```

API Яндекс.Карт

```
<script src="https://api-maps.yandex.ru/2.1/?lang=ru_RU">
</script>
<script>
    ymaps.ready().then(function () {
        var map = new ymaps.Map(
            'map-div-id',
            { /* state */ },
            { /* options */ }
        );
    });
</script>
```

API Яндекс.Карт



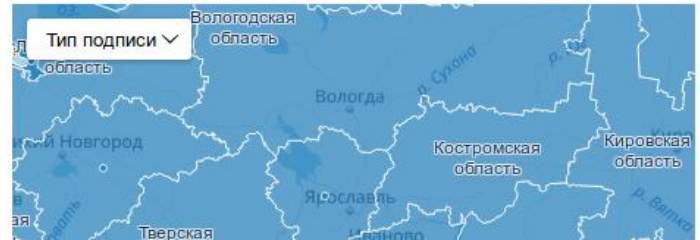
Модули JS API Карт

Тепловые карты

Как на физической карте интенсивность цвета указывает на возвышенности, так на тепловой — на высокую концентрацию объектов или предложений, соответствующих запросу. Например, по запросу [снять квартиру] вы получите карту, на которой спальные районы будут гораздо более «тёплыми», чем центр города.

Подписывание полигонов на карте

Модуль Polylabeler позволяет автоматически расставить подписи внутри произвольных многоугольников на карте.
В зависимости от вида подписи и формы полигона модуль рассчитывает для неё подходящую позицию, которую затем можно изменить.



Модуль подписей полигонов

github.com/yandex/mapsapi-polylabeler

У нас своя модульная система

```
ymaps.modules.define('module.name', [
    'global.dependency.name.first',
    'global.dependency.name.second'
], function(provide, first, second) {
    // ...
    var myExports = 42;
    provide(myExports);
});
```

У нас своя модульная система

```
ymaps.modules.define('module.name', [
    'global.dependency.name.first',
    'global.dependency.name.second'
], function(provide, first, second) {
    // ...
    var myExports = 42;
    provide(myExports);
});
```

У нас своя модульная система

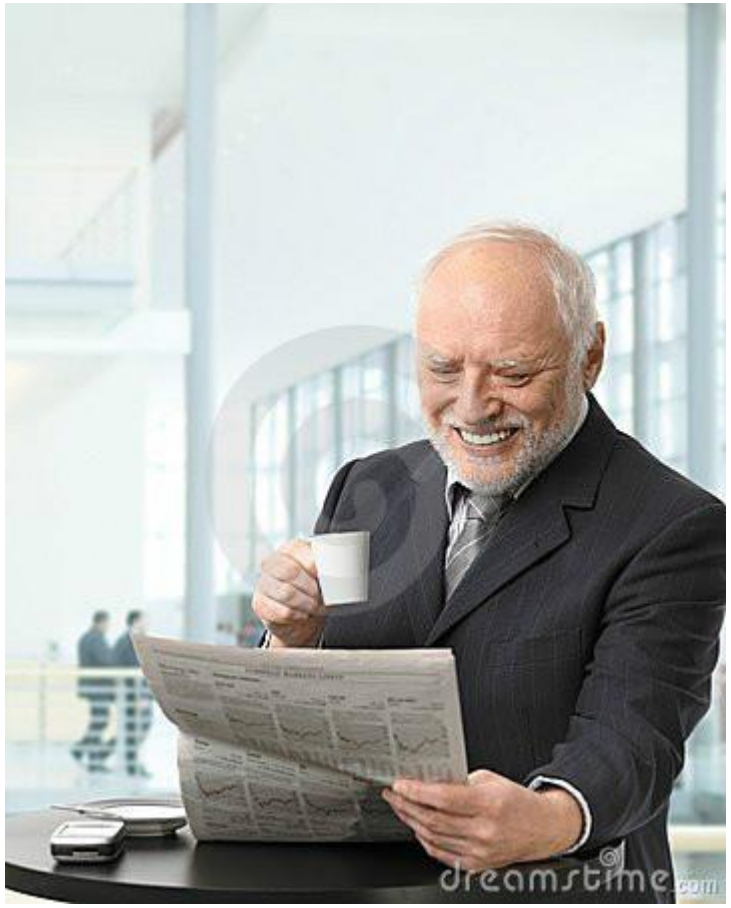
```
ymaps.modules.define('module.name', [
    'global.dependency.name.first',
    'global.dependency.name.second'
], function(provide, first, second) {
    // ...
    var myExports = 42;
    provide(myExports);
});
```

У нас своя модульная система

```
ymaps.modules.define('module.name', [
    'global.dependency.name.first',
    'global.dependency.name.second'
], function(provide, first, second) {
    // ...
    var myExports = 42;
    provide(myExports);
});
```

У нас своя модульная система

```
ymaps.modules.define('module.name', [
    'global.dependency.name.first',
    'global.dependency.name.second'
], function(provide, first, second) {
    // ...
    var myExports = 42;
    provide(myExports);
});
```



За окном мир победившего ES2015

```
import first from './dependency/name/first';
import second from './dependency/name/second';

export default myExports;
```


BABEL

ES2015 в ymaps.modules

```
for (const statement of program.body) {
    if (statement.isImportDeclaration()) {
        saveForLater(statement);
        statement.remove();
    }

    if (statement.isExportDeclaration()) {
        statemenet.replaceWith(
            t.callExpression(provideIdentifier,statement.node);
        )
    }
}

const template = babelTemplate(
    'ymaps.modules.define(NAME, [DEPS], function(PARAMS) { BODY; });'
);

program.body = template({ BODY: program.body, ... });
```

ES2015 в ymaps.modules

```
for (const statement of program.body) {
    if (statement.isImportDeclaration()) {
        saveForLater(statement);
        statement.remove();
    }

    if (statement.isExportDeclaration()) {
        statemenet.replaceWith(
            t.callExpression(provideIdentifier,statement.node);
        )
    }
}

const template = babelTemplate(
    'ymaps.modules.define(NAME, [DEPS], function(PARAMS) { BODY; });'
);

program.body = template({ BODY: program.body, ... });
```

ES2015 в ymaps.modules

```
for (const statement of program.body) {
    if (statement.isImportDeclaration()) {
        saveForLater(statement);
        statement.remove();
    }

    if (statement.isExportDeclaration()) {
        statemenet.replaceWith(
            t.callExpression(provideIdentifier,statement.node);
        )
    }
}

const template = babelTemplate(
    'ymaps.modules.define(NAME, [DEPS], function(PARAMS) { BODY; });'
);

program.body = template({ BODY: program.body, ... });
```

ES2015 в ymaps.modules

```
for (const statement of program.body) {
    if (statement.isImportDeclaration()) {
        saveForLater(statement);
        statement.remove();
    }

    if (statement.isExportDeclaration()) {
        statemenet.replaceWith(
            t.callExpression(provideIdentifier,statement.node);
        )
    }
}

const template = babelTemplate(
    'ymaps.modules.define(NAME, [DEPS], function(PARAMS) { BODY; });'
);

program.body = template({ BODY: program.body, ... });
```

ES2015 в ymaps.modules

```
for (const statement of program.body) {
    if (statement.isImportDeclaration()) {
        saveForLater(statement);
        statement.remove();
    }

    if (statement.isExportDeclaration()) {
        statemenet.replaceWith(
            t.callExpression(provideIdentifier,statement.node);
        )
    }
}

const template = babelTemplate(
    'ymaps.modules.define(NAME, [DEPS], function(PARAMS) { BODY; });'
);

program.body = template({ BODY: program.body, ... });
```

Пишем

```
import PCollection from './polylabel/PolylabelCollection';
import PObjectManager from './polylabel/PolylabelObjectManager';
import ObjectManager from 'api/ObjectManager';

export default function (map, data) {
    initStyles();
    return data instanceof ObjectManager ?
        new PObjectManager(map, data) :
        new PCollection(map, data);
}
```

Выполняем

```
ymaps.modules.define('polylabel.create', [
    'polylabel.polylabel.PolylabelCollection',
    'polylabel.polylabel.PolylabelObjectManager',
    'ObjectManager'
], function (_provide, PCollection, PObjectManager, ObjectManager) {
    _provide(function (map, data) {
        initStyles();
        return data instanceof ObjectManager ?
            new PObjectManager(map, data) :
            new PCollection(map, data);
    });
});
```

Профит

- Пишем на ES2015, на выходе ES3
- Бесплатная поддержка IDE и линтеров
- Проще мигрировать на другую модульную систему и бандлер

ES2015 в ymaps.modules

github.com/flapenguin/babel-plugin-transform-es2015-modules-ym

npmjs.com/package/babel-plugin-transform-es2015-modules-ym

TypeScript

Похожая магия

```
tsc source.ts
```

```
{ loader: 'awesome-typescript-loader' }
```

Под капотом чуть сложнее

```
const config = { /* ... */ };
const host = ts.createCompilerHost(config);
const program = ts.createProgram(['source.js'], config, host);

const result = program.emit(
    undefined, // explicit AST
    writeFile,
    cancellationToken,
    emitOnlyDts,
    {
        before: [transformer1],
        after: [transformer2]
    }
);
```

Под капотом чуть сложнее

```
const config = { /* ... */ };
const host = ts.createCompilerHost(config);
const program = ts.createProgram(['source.js'], config, host);

const result = program.emit(
    undefined, // explicit AST
    writeFile,
    cancellationToken,
    emitOnlyDts,
    {
        before: [transformer1],
        after: [transformer2]
    }
);
```

Под капотом чуть сложнее

```
const config = { /* ... */ };
const host = ts.createCompilerHost(config);
const program = ts.createProgram(['source.js'], config, host);

const result = program.emit(
    undefined, // explicit AST
    writeFile,
    cancellationToken,
    emitOnlyDts,
    {
        before: [transformer1],
        after: [transformer2]
    }
);
```

Трансформация

```
const visitor = node => {
    if (ts.isIdentifer(node)) {
        // ...
        ts.updateIdentifier(node)
    } else {
        ts.visitEachChild(node, visitor, ctx);
    }
};
```

Трансформация

```
const visitor = node => {
    if (ts.isIdentifer(node)) {
        // ...
        ts.updateIdentifier(node)
    } else {
        ts.visitEachChild(node, visitor, ctx);
    }
};
```

Создание переменных

- `ts.createUniqueName`
- `ts.createTempVariable`

Плагины

```
function myTransformer(myArgs) {
    return function(transformationContext) {
        return function(sourceFile) {
            return ts.visitNode(sourceFile, visitor);
        }
    };
}
```

Плагины

```
function myTransformer(myArgs) {
    return function(transformationContext) {
        return function(sourceFile) {
            return ts.visitNode(sourceFile, visitor);
        }
    };
}
```

Плагины

```
function myTransformer(myArgs) {
    return function(transformationContext) {
        return function(sourceFile) {
            return ts.visitNode(sourceFile, visitor);
        }
    };
}
```

Плагины

```
function myTransformer(myArgs) {
    return function(transformationContext) {
        return function(sourceFile) {
            return ts.visitNode(sourceFile, visitor);
        }
    };
}
```

Информация о типах

```
const typeChecker = program.getTypeChecker();
const type = typeChecker.getTypeAtLocation(ast)


type.getProperties()
type.getCallSignatures()
type.getBaseTypes()
type.getConstructSignatures()

const symbol = type.getSymbol()

symbol.name // 'FooBar'
symbol.valueDeclaration // FooBar declaration ./FooBar.ts:42
```

Информация о типах

```
const typeChecker = program.getTypeChecker();
const type = typeChecker.getTypeAtLocation(ast)


type.getProperties()
type.getCallSignatures()
type.getBaseTypes()
type.getConstructSignatures()

const symbol = type.getSymbol()

symbol.name // 'FooBar'
symbol.valueDeclaration // FooBar declaration ./FooBar.ts:42
```

Информация о типах

```
const typeChecker = program.getTypeChecker();
const type = typeChecker.getTypeAtLocation(ast)
```

```
type.getProperties()
type.getCallSignatures()
type.getBaseTypes()
type.getConstructSignatures()
```

```
const symbol = type.getSymbol()

symbol.name // 'FooBar'
symbol.valueDeclaration // FooBar declaration ./FooBar.ts:42
```

Информация о типах

```
const typeChecker = program.getTypeChecker();
const type = typeChecker.getTypeAtLocation(ast)


type.getProperties()
type.getCallSignatures()
type.getBaseTypes()
type.getConstructSignatures()

const symbol = type.getSymbol()

symbol.name // 'FooBar'
symbol.valueDeclaration // FooBar declaration ./FooBar.ts:42
```

Осторожно

- TypeScript не будет проверять типы для кода, который вы нагенерили

Нет стандартного API для плагинов

github.com/Microsoft/TypeScript/issues/14419

```
tsc ??? source.ts
```

Нет стандартного API для плагинов

- Чтобы воткнуться в пайплайн, придется писать свой tsc
  - github.com/cevek/ttypescript

- Во многих бандлерах уже есть поддержка в плагинах
  - ts-loader
  - parcel-plugin-typescript
  - rollup-plugin-typescript2

Что почитать

github.com/Microsoft/TypeScript

TypeScript compiler APIs revisited

How to Write a TypeScript Transform (Plugin)

TypeScript Deep Dive

Автоматические понифилы

Понифилы?

```
// Polyfill
Array.prototype.every = function(cb) {
    // ...
};

[].every(...);

// Ponyfill
function arrayEvery(arr, cb) {
    // ...
};

arrayEvery([], ...)
```

Понифилы?

```
// Polyfill
Array.prototype.every = function(cb) {
    // ...
};

[].every(...);

// Ponyfill
function arrayEvery(arr, cb) {
    // ...
};

arrayEvery([], ...)
```

Понифилы?

```
// Polyfill
Array.prototype.every = function(cb) {
    // ...
};

[].every(...);

// Ponyfill
function arrayEvery(arr, cb) {
    // ...
};

arrayEvery([], ...)
```

Понифилы?

```
// Polyfill
Array.prototype.every = function(cb) {
    // ...
};

[].every(...);

// Ponyfill
function arrayEvery(arr, cb) {
    // ...
};

arrayEvery([], ...)
```

Понифилы?

```
// Polyfill
Array.prototype.every = function(cb) {
    // ...
};

[].every(...);

// Ponyfill
function arrayEvery(arr, cb) {
    // ...
};

arrayEvery([], ...)
```


TypeScript

Поиск мест, где нужен понифил

```
if (
    ts.isCallExpression(node) &&
    ts.isPropertyAccessExpression(node.expression)
) {
    // object.method(...)
    const object = node.expression.expression;
    const methodName = node.expression.name;

    const objectType = typeChecker.getTypeAtLocation(object);
    const objectTypeName = objectType?.getSymbol().name;

    if (objectTypeName === 'Array' && hasPonyfill(methodName)) {
        ts.updateCall(node, ...);
    }
}
```

Поиск мест, где нужен понифил

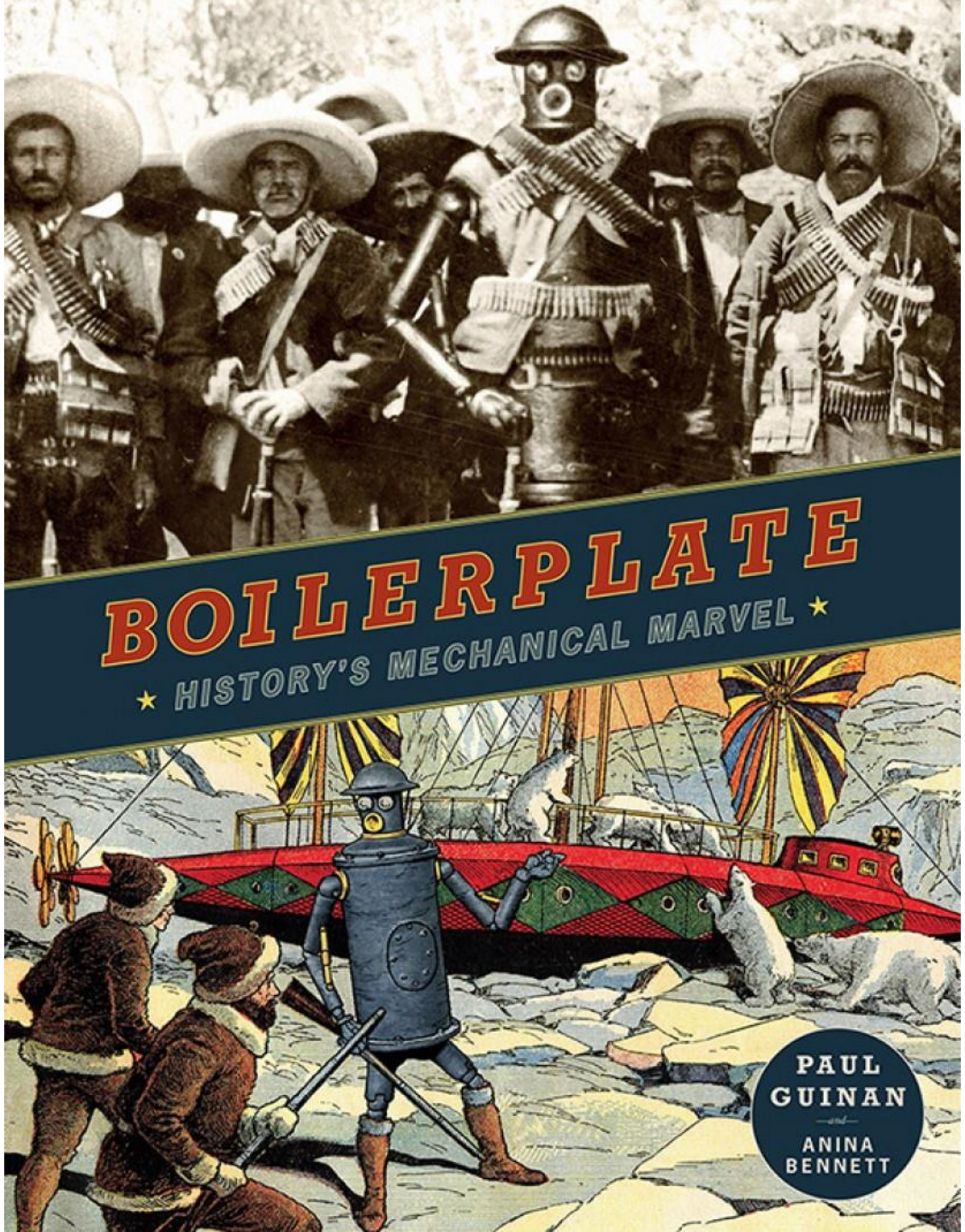
```
if (
    ts.isCallExpression(node) &&
    ts.isPropertyAccessExpression(node.expression)
) {
    // object.method(...)
    const object = node.expression.expression;
    const methodName = node.expression.name;

    const objectType = typeChecker.getTypeAtLocation(object);
    const objectTypeName = objectType?.getSymbol().name;

    if (objectTypeName === 'Array' && hasPonyfill(methodName)) {
        ts.updateCall(node, ...);
    }
}
```

Поиск мест, где нужен понифил

```
if (
    ts.isCallExpression(node) &&
    ts.isPropertyAccessExpression(node.expression)
) {
    // object.method(...)
    const object = node.expression.expression;
    const methodName = node.expression.name;

    const objectType = typeChecker.getTypeAtLocation(object);
    const objectTypeName = objectType?.getSymbol().name;

    if (objectTypeName === 'Array' && hasPonyfill(methodName)) {
        ts.updateCall(node, ...);
    }
}
```

Поиск мест, где нужен понифил

```
if (
    ts.isCallExpression(node) &&
    ts.isPropertyAccessExpression(node.expression)
) {
    // object.method(...)
    const object = node.expression.expression;
    const methodName = node.expression.name;

    const objectType = typeChecker.getTypeAtLocation(object);
    const objectTypeName = objectType?.getSymbol().name;

    if (objectTypeName === 'Array' && hasPonyfill(methodName)) {
        ts.updateCall(node, ...);
    }
}
```


BOILERPLATE
★ HISTORY'S MECHANICAL MARVEL ★
PAUL GUINAN
and
ANINA BENNETT

Использование с awesome-typescript-loader

```
loader: 'awesome-typescript-loader',
options: {
  configFileName: __dirname + '/tsconfig.json’,
  getCustomTransformers: (program) => ({
    before: [
      autoPonyfillTransformer({
        typeChecker: program.getTypeChecker(),
        ponyfillMethods: {
          [`Array@lib.es6.d.ts'`]: {
            file: __dirname + '/arrayPonyfills.ts’,
            methods: {
              map: true,
              filter: 'myFilter’
```

Использование с awesome-typescript-loader

```
loader: 'awesome-typescript-loader',
options: {
  configFileName: __dirname + '/tsconfig.json’,
  getCustomTransformers: (program) => ({
    before: [
      autoPonyfillTransformer({
        typeChecker: program.getTypeChecker(),
        ponyfillMethods: {
          [`Array@lib.es6.d.ts'`]: {
            file: __dirname + '/arrayPonyfills.ts’,
            methods: {
              map: true,
              filter: 'myFilter’
```

Использование с awesome-typescript-loader

```
loader: 'awesome-typescript-loader',
options: {
  configFileName: __dirname + '/tsconfig.json’,
  getCustomTransformers: (program) => ({
    before: [
      autoPonyfillTransformer({
        typeChecker: program.getTypeChecker(),
        ponyfillMethods: {
          [`Array@lib.es6.d.ts'`]: {
            file: __dirname + '/arrayPonyfills.ts’,
            methods: {
              map: true,
              filter: 'myFilter’
```

Было

```
[1, 2, 3, 4].find(cb);

returnsArray().filter(cb);
```

Стало

```
import * as utilArray from './arrayPonyfills';

utilArray.find([1, 2, 3, 4], cb);

utilArray.myFilter(returnsArray(), cb);
```

Автоматические понифилы

github.com/flapenguin/ts-transform-auto-ponyfill

npmjs.com/package/ts-transform-auto-ponyfill

Слайды: flapenguin.me/talks/transpilation-ft

Вопросы?

github.com/flapenguin

twitter.com/fla_penguin

flapenguin.me

Flow

- На OCaml'е с системой типов OCaml'а
- Ad-hoc решение Facebook для React'а
  - Нельзя объявить функцию с именем invariant
    github.com/facebook/flow/issues/112
  - ~125 PR от 48 людей. 15 открыли >1 PR, 9 открыли >2 PR
- Зато поддержка в Babel'е из коробки

```
- VariableDeclarator {
   - id: Identifier {
        name: "x"
      - typeAnnotation: TypeAnnotation {
           typeAnnotation: StringTypeAnnotation {
           }
```