Инструментируй это

Роман Дворнов
Avito

Минск 2015

Я...

- Работаю в Avito
- Делаю SPA
- Автор basis.js

За любую движуху,
кроме голодовки ;)

DEMO

поговорим про боль и немного интриги

Инструментирование кода

«под инструментированием понимают возможность отслеживания или установления количественных параметров …, а также возможность диагностировать ошибки и записывать информацию для отслеживания причин их возникновения…»

en.wikipedia.org/wiki/Instrumentation_(computer_programming)

Области применения

- Трассировка
- Отладка
- Регистрация событий
- Подмена кода
- Счетчики производительности
- ...

Пример №1

Code coverage

Как узнать что исполнялось?

Оригинальный код

```
function foo(a) {
  if (a > 5) {
    return 1;
  } else {
    return 2;
  }
}
```

Как узнать что исполнялось?

Оригинальный код

```
function foo(a) {
  if (a > 5) {
    return 1;
  } else {
    return 2;
  }
}
```

Трансформированный код

```
function foo(a) {
  __visit(1);
  if (a > 5) {
    __visit(2);
    return 1;
  } else {
    __visit(3);
    return 2;
  }
}
```

API (runtime)

Упрощенная версия

```
var __count = [0, 0, 0, ...]; // столько нулей, сколько
                              // частей в коде, которые
                              // нас интересуют

window.__visit = function(idx) {
  __count[idx - 1]++;
};
```

После выполнения тестов в __count будет необходимая информация

Пример результатов

функция исполнялась 3 раз

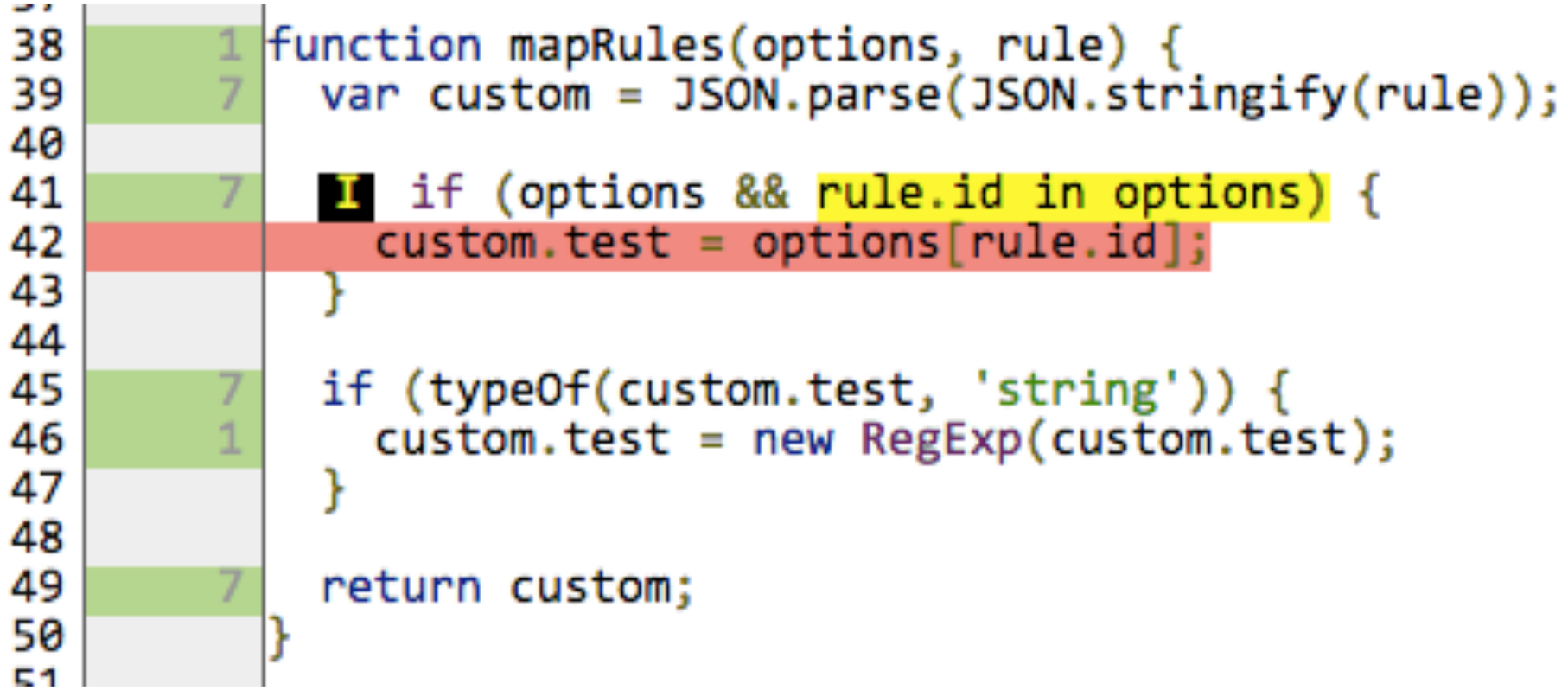
```
__count = [3, 3, 0]
```

ветка **if** исполнялась 3 раза

ветка **else** не исполнялась

Покрытие кода
~67%

Проецируем данные на исходники



Code coverage: все вместе

- трансформация оригинального кода
- запуск тестов
- агрегация данных или визуализация
- PROFIT!!!

Code coverage: все вместе

- трансформация оригинального кода
- запуск тестов
- агрегация данных или визуализация
- PROFIT!!!

это и есть **инструментирование**
(делается автоматически)

Не стоит путать с другими типами трансформаций, например, транспиляцией

Транспиляция – преобразование код в равнозначный код, например, ES6 → ES5

en.wikipedia.org/wiki/Source-to-source_compiler

Инструментирование –
дополнение кода другим кодом,
который делает что-то еще

Инструментирование = код, добавляющий код в код

Инструментирование = код, добавляющий код в код

Пример №2

Loupe

Визуализация того, как работают некоторые части в JavaScript

(call stack, event loop, callback queue)

latentflip.com/loupe

(там есть видео доклада, рекомендую)

DEMO

Как это работает

- Инструментируем код
- отправляем код в Web Worker, где он исполняется
- по мере исполнения, в основной процесс отправляются сообщения о том, что происходит – это визуализируется
- немного «магии» ;)

Пример №3

React Hot Loader

Модификация React-компонентов

в реальном времени

gaearon.github.io/react-hot-loader/

DEMO

Как это работает

- Достаточно сложно объяснить в двух словах
- Ключевое:
  - исходный код инструментируется
  - патчатся классы компонент
  - подменяются методы классов при изменении исходного кода и делается полный re-render

Пример №4

Ищем концы

или с чего все начиналось

Начальная идея

Имея ссылку на объект уметь

определять местоположение

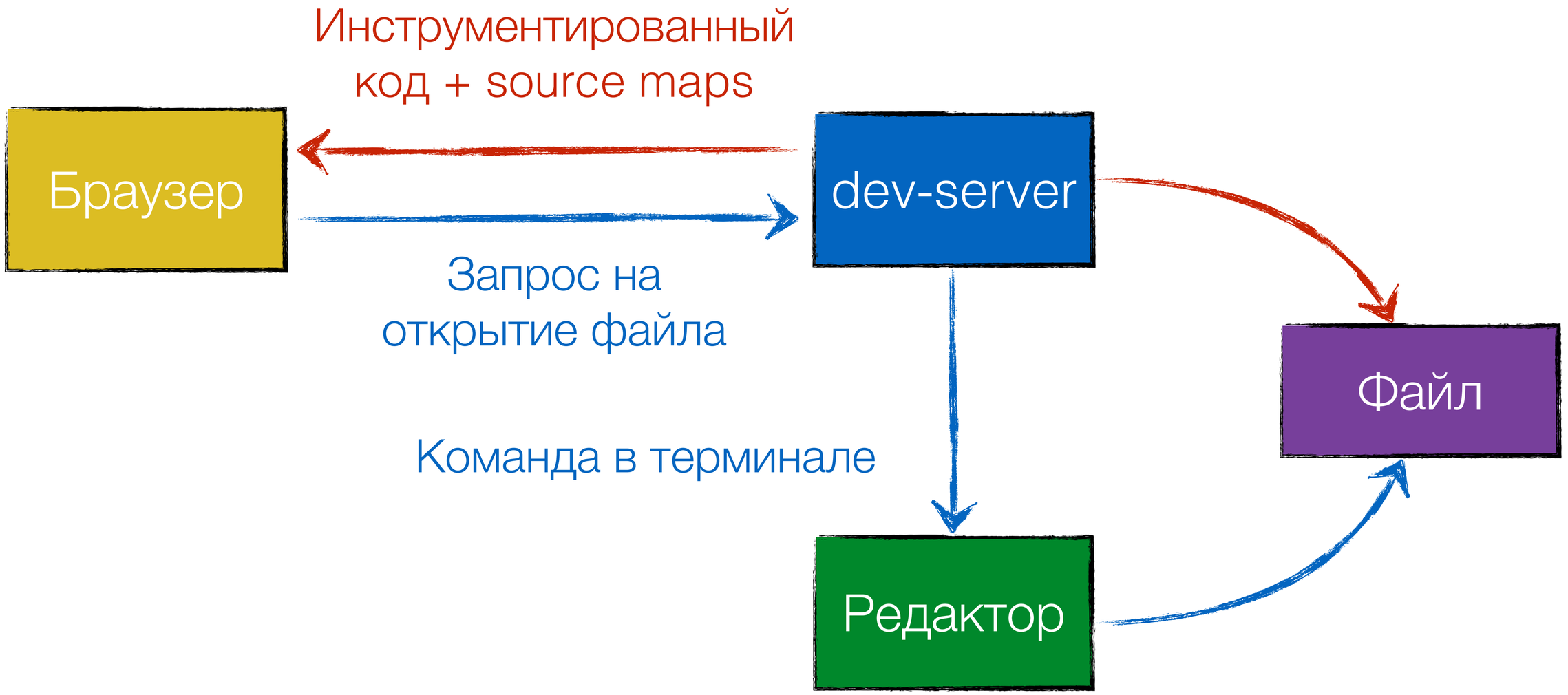
фрагмента кода его декларации

DEMO

Главное – идея

- Не завязано на фреймворк/библиотеку
- Библиотеки могут расширять возможности и предоставлять дополнительную информацию

За пределами браузера

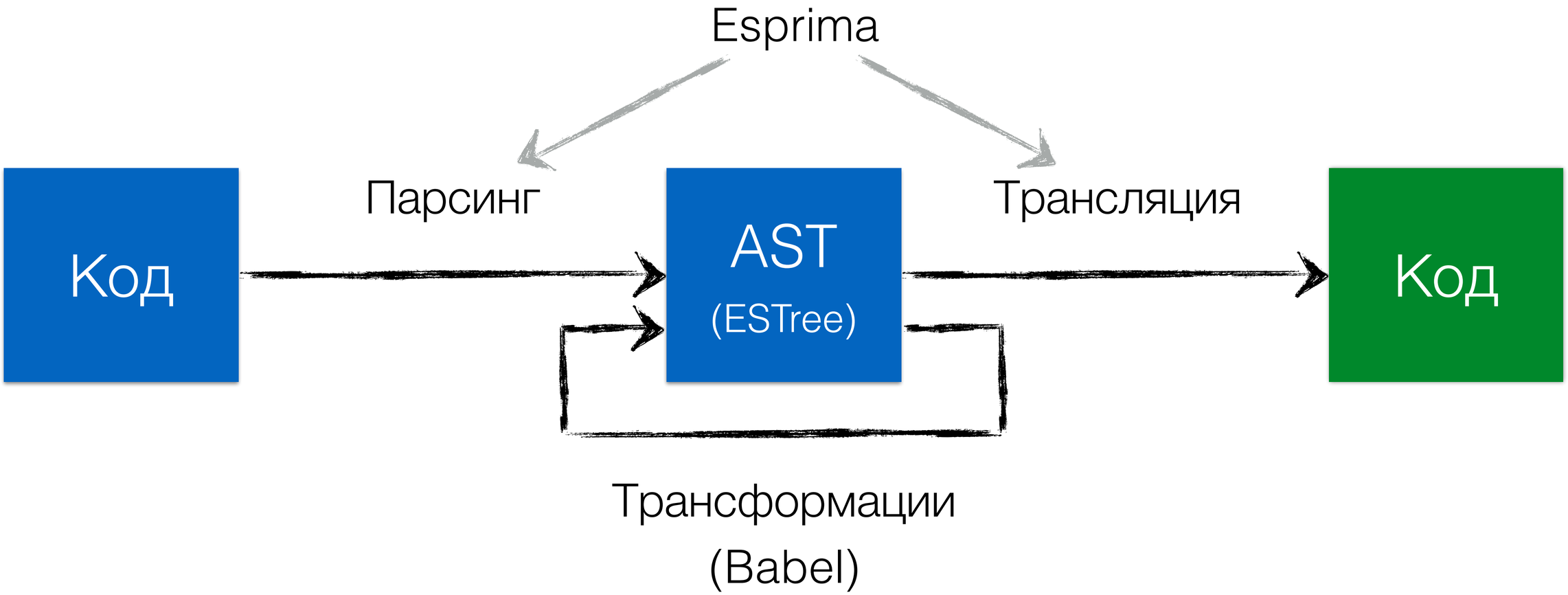


Части решения

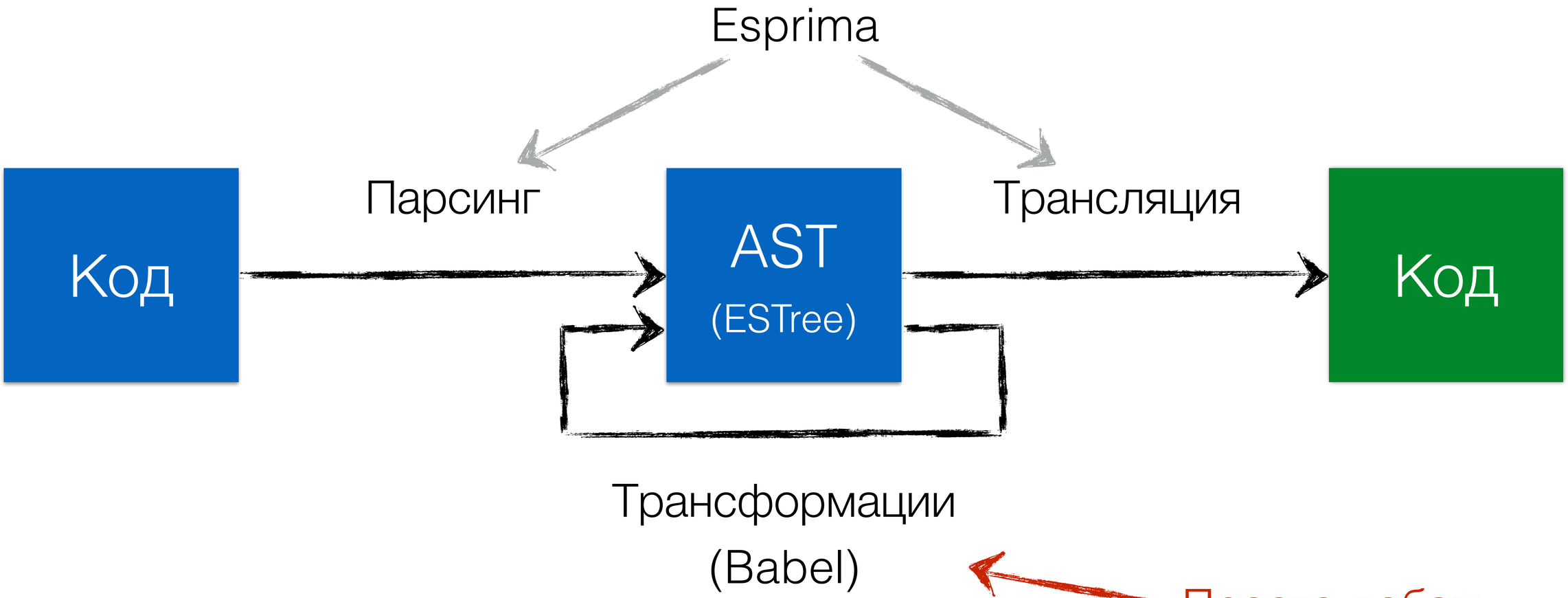
- Инструментирование кода
- Runtime API
- Инспекция DOM дерева
- dev-сервер

Инструментирование

Делаем трансформацию кода правильно



Делаем трансформацию кода правильно



Инструментирование делается плагином для Babel

github.com/restrry/babel-plugin-source-wrapper

Что делает плагин

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

var obj = { foo: 1 };
```

```
var foo = $devinfo(function(a, b) {
    return a + b;
}, { … });

var obj = $devinfo({ foo: 1 }, { … });
```

Делается автоматически

и только на этапе разработки

Runtime API

Основное API

Упрощенный код

```
window.$devinfo = (function() {
  var map = new WeakMap();
  var api = function(ref, data) {
    if (!map.has(ref))
      map.set(ref, data);
    return ref;
  };
  api.get = function(ref) {
    return ref ? map.get(ref) : undefined;
  }
  return api;
})();
```

Основное API

Упрощенный код

```
window.$devinfo = (function() {
  var map = new WeakMap();
  var api = function(ref, data) {
    if (!map.has(ref))
      map.set(ref, data);
    return ref;
  };
  api.get = function(ref) {
    return ref ? map.get(ref) : undefined;
  }
  return api;
})();
```

Используем WeakMap для хранения информации:

- объекты как ключи
- не трансформирует объекты
- не создает утечек памяти

Основное API

Упрощенный код

```
window.$devinfo = (function() {
  var map = new WeakMap();
  var api = function(ref, data) {
    if (!map.has(ref))
      map.set(ref, data);
    return ref;
  };
  api.get = function(ref) {
    return ref ? map.get(ref) : undefined;
  }
  return api;
})();
```

Основная функция регистрации, "добавляет" информацию только если ее еще нет у объекта

Основное API

Упрощенный код

```
window.$devinfo = (function() {
  var map = new WeakMap();
  var api = function(ref, data) {
    if (!map.has(ref))
      map.set(ref, data);
    return ref;
  };
  api.get = function(ref) {
    return ref ? map.get(ref) : undefined;
  }
  return api;
})();
```

Получение информации,
используется инструментами

Использование

```
var obj = {};

$devinfo(obj, { data: 123 });

console.log($devinfo.get(obj));
// > { data: 123 }
```

Возможность хранить информацию, можно использовать для других задач

DEMO

github.com/lahmatiy/t8

Инспектирование DOM

Для инструмента нужно

- знать границу компонента, т.е. определять элемент-контейнер компонента (DOM узел)
- определять владельца компонента (view) по элементу-контейнеру (по DOM узлу)

Решение в лоб

- если у DOM узла есть свойство __view – значит это контейнер компонента
- в свойстве __view хранится ссылка на view

Но лучше использовать
WeakMap (node → view)

Крохотный патч для Backbone

```
var _setElement = Backbone.View.prototype._setElement;
Backbone.View.prototype._setElement = function() {
    _setElement.apply(this, arguments);
    this.el.__view = this;
};
```

Или с WeakMap

```
var map = new WeakMap();
var _setElement = Backbone.View.prototype._setElement;
Backbone.View.prototype._setElement = function() {
    _setElement.apply(this, arguments);
    map.add(this.el, this);
};
window.getBackboneViewByNode = function(el) {
   return map.get(el)
};
```

Готовые решения

basis.js

Работает из коробки

(нужен basis.js 1.5)

Другие фреймворки

component-inspector

github.com/lahmatiy/component-inspector

```
npm install component-inspector --save-dev
```

React

Подключение в html **перед** React

```
<script src="node_modules/component-inspector/dist/react.js">
</script>
<script src="react.js"></script>
```

Backbone

Подключение в html **после** Backbone

```
<script src="backbone.js"></script>
<script src="node_modules/component-inspector/dist/backbone.js">
</script>
```

Свое решение?

Запросто!

Инициализация

```
<script src="node_modules/component-inspector/dist/base.js"></script>
<script>
  initComponentInspector({
    // просто добавь API
  });
</script>
```

Описываем API

```
initComponentInspector({
    isComponentRootNode: function(node) {
      return Boolean(node && node.__view);
    },
    getInstanceByNode: function(node) {
      if (node) {
        return node.__view;
      }
    },
    …
 });
```

Можно определить многое

- isComponentRootNode(node)
- getComponentNameByNode(node)
- getInstanceByNode(node)
- getInstanceRootNode(instance)
- getInstanceClass(instance)
- getInstanceLocation(instance)
- ...

github.com/lahmatiy/component-inspector#api-free-build

dev-сервер

dev-сервер должен

- отдавать инструментированный код
- исполнять команды, например, открытие файла в редакторе

Вариант настройки: webpack

Инструментирование

webpack

+ babel + babel-plugin-source-wrapper

Открытие файла в редакторе

express или webpack-dev-server

+ express-open-in-editor

Пример конфигурации для Webpack + React

tinyurl.com/pwj6bln

Вариант настройки: basisjs-tools

```
> npm install basisjs-tools -g
> npm install basisjs-tools-instrumenter
// создать конфиг basis.config
> basis server
```

Сервер будет отдавать инструментированный код и встраивать необходимое в html

```
{
  "plugins": [
    "basisjs-tools-instrumenter"
  ]
}
```

Побочные продукты

- open-in-editor

  npm пакет для программного открытия файла в редакторе

  github.com/lahmatiy/open-in-editor

- express-open-in-editor

  расширение для Express для открытия файла по урлу

  github.com/lahmatiy/express-open-in-editor

- extract-code-fragment

  получение раскрашенного фрагмента кода из файла

  скоро

Проблемки

Проблема №1

«Захламляется» код

Решение: Source maps

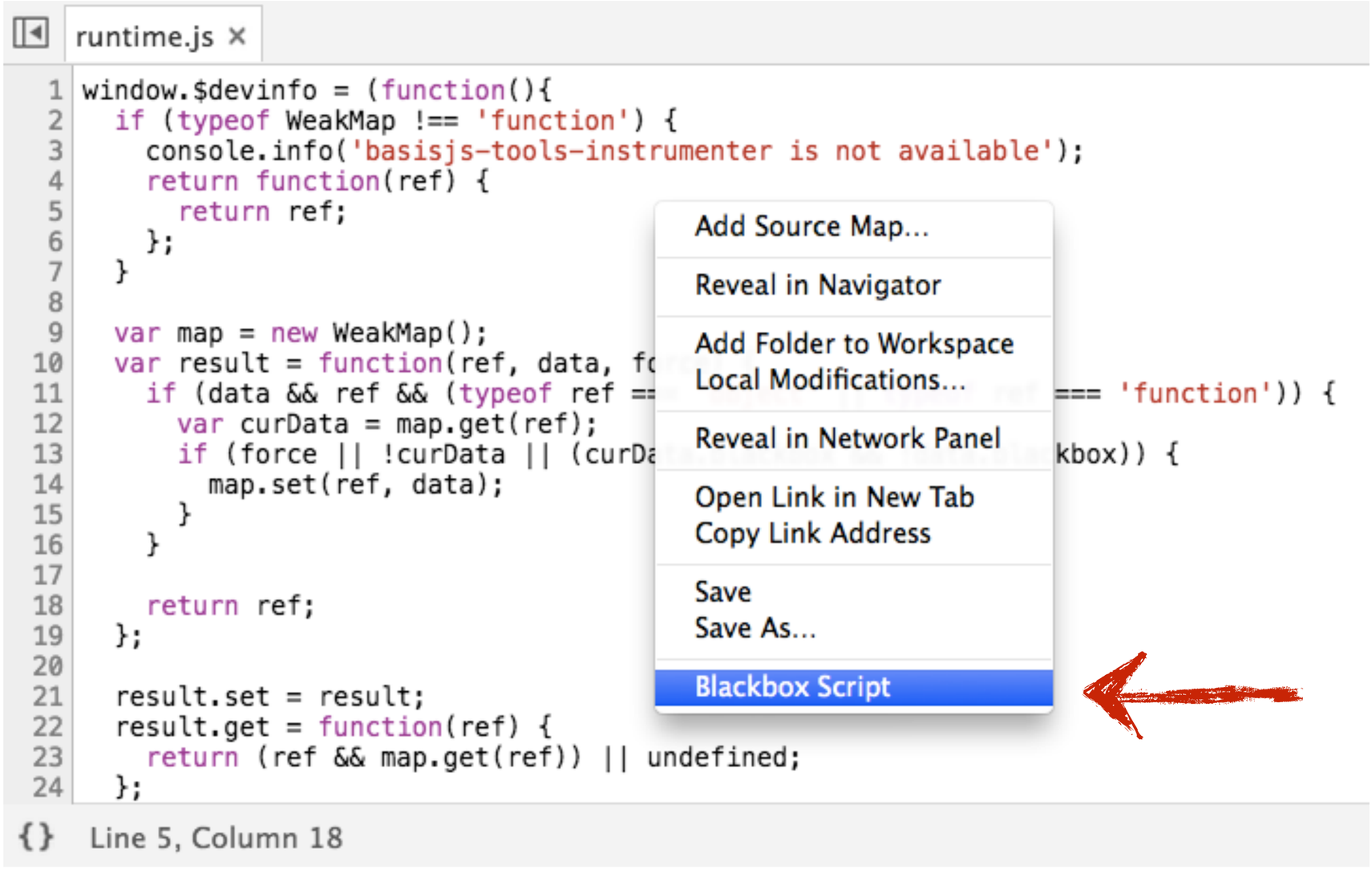
```
var a = $loc_h8tz9yd7({
  foo: 1,
  bar: $loc_h8tz9yd7(function (a, b) {
    return a + b;
  }, {
    loc: "/src/app.js:3:8:5:4"
  })
}, {
  loc: "/src/app.js:1:9:6:2",
  map: {
    foo: "/src/app.js:2:8:2:9",
    bar: "/src/app.js:3:8:5:4"
  }
});
```

```
var a = {
  foo: 1,
  bar: function(a, b) {
    return a + b;
  }
};
```

Проблема №2

Runtime API мешает отладке

Решение: Blackbox Script



Проблема №2

Показывается не то место

app.js

```
var obj = $devinfo(createObj(), {
  loc: "app.js:…"
});
```

Нам нужно это место

lib.js

```
function createObj() {
  return $devinfo({
    example: true
  }, {
    loc: "lib.js:…"
  });
}
```

Но покажется это

Решение: blackbox в инструментере

```
require('babel-plugin-source-wrapper')({
  blackbox: [
    "/path/to/lib.js"
  ]
})
```

По умолчанию:

```
blackbox: [
  "/bower_components/**",
  "/node_modules/**"
]
```

Решение: blackbox в инструментере

app.js

```
var obj = $devinfo(createObj(), {
  loc: "app.js:…"
});
```

Теперь будет показываться это место

lib.js

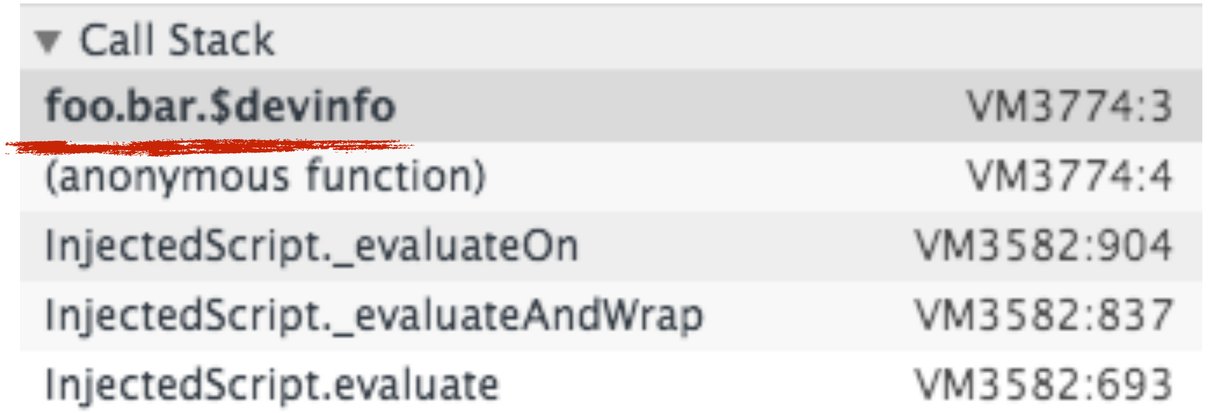
```
function createObj() {
  return $devinfo({
    example: true
  }, {
    loc: "lib.js:…",
    blackbox: true
  });
}
```

У такой информации меньший приоритет

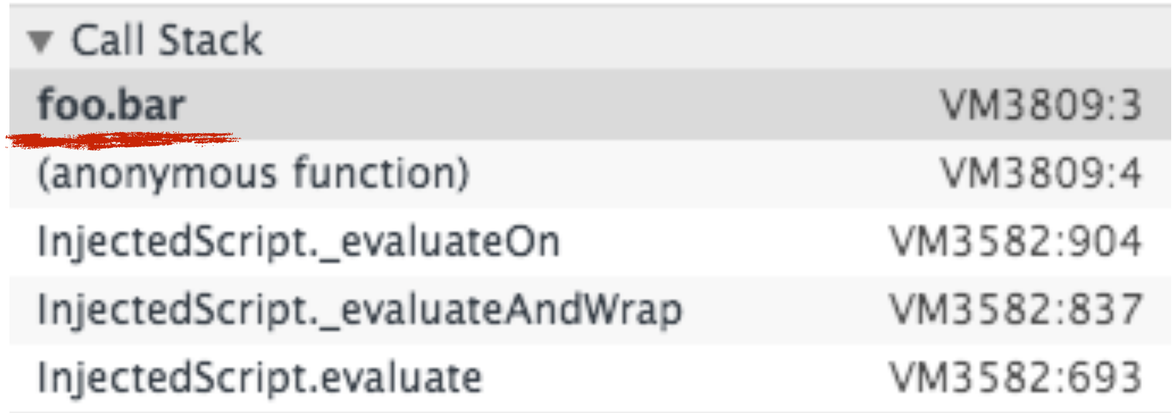
Проблема №4

«Зашумляется» стек вызова

Видим


Call Stack
foo.bar.$devinfo VM3774:3
(anonymous function) VM3774:4
InjectedScript._evaluateOn VM3582:904
InjectedScript._evaluateAndWrap VM3582:837
InjectedScript.evaluate VM3582:693

Хотим


Call Stack
foo.bar VM3809:3
(anonymous function) VM3809:4
InjectedScript._evaluateOn VM3582:904
InjectedScript._evaluateAndWrap VM3582:837
InjectedScript.evaluate VM3582:693

Решение: магия :)

```
$devinfo(function() {
  …
}, { … })
```

```
($devinfo)(function() {
  …
}, { … })
```

Заключение

Инструментирование – метод, расширяющий возможности разработки

Местами это сложно,

но безумно интересно!

Попробуйте!

Интересно, что получится у вас ;)

Вопросы?

Роман Дворнов

@rdvornov

rdvornov@gmail.com

github.com/lahmatiy

Component Inspector

tinyurl.com/wsd-ci