

Legacy. Иллюзия контроля

Заурбек Будтуев

Руководитель группы фронтенд-разработки



Search Results

[Advanced Search Tips](#)[Ask Question](#)

Results for legacy

[Search](#)

83,104 results

[Relevance](#)[Newest](#)[More ▾](#)[Search Jobs](#)[👤 HR Tools](#)[📁 Post Jobs](#)[Sign In ▾](#)

[Technology](#) > [Software Development & Architecture](#) > [Software Development](#)

3,152 Legacy code jobs [?](#)

[Relevance](#)[Date](#)[Job Type ▾](#)[Minimum Salary ▾](#)[Date Added ▾](#)

Legacy по версии Wikipedia

Legacy code

Legacy code is [source code](#) that relates to a no-longer supported^{[\[citation needed\]](#)} or manufactured [operating system](#) or other [computer](#) technology. The term can also mean code inserted into modern software for the purpose of maintaining an older or previously supported feature – for example supporting a serial interface even though many modern systems do not have a [serial port](#). It may also be in the form of supporting older file formats^{[\[citation needed\]](#)} that may have been encoding in non-[ASCII characters](#), such as [EBCDIC](#).^{[\[citation needed\]](#)}

Современная интерпретация



Michael
Stonebraker,
Michael L.
Brodie

Any information system that significantly resists modification and evolution.



Dylan Beattie

Code that's too scary to update and too profitable to delete.

Современная интерпретация



Eli Lopian

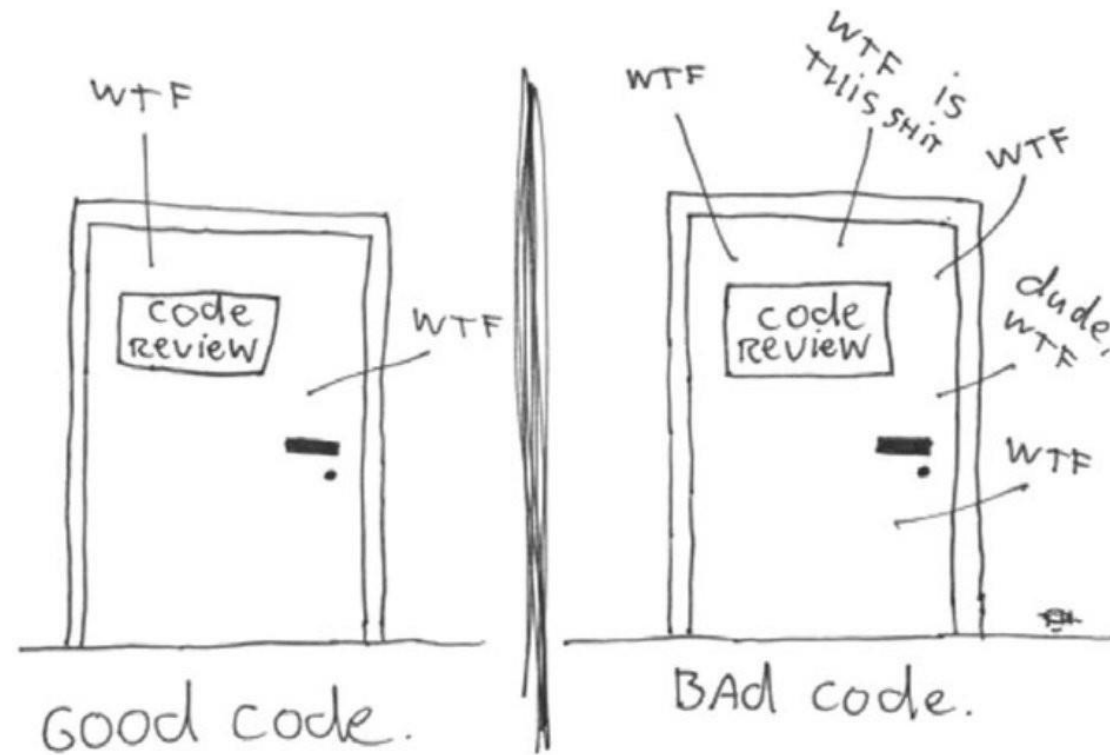
I think of legacy code as code that developers are afraid to change.



Michael
Feathers

Legacy code is code without tests.

The ONLY VALID MEASUREMENT
OF CODE QUALITY: WTFs/MINUTE



Признаки кода

1. Дорог в поддержке и развитии.
2. Критически важный для бизнеса.
3. Не покрыт тестами.

Новый проект

1. Аналитика.
2. Документация API.
3. Тесты.
4. Соблюдение стиля.
5. Работа с техническим долгом.



Иллюзия контроля как причина



Иллюзия контроля как причина

1. Благоприятный исход.

✔ frontend » tests #9478
built by Jenkins @ <https://ci.funbox.ru/>

✔ frontend » #11667
built by Jenkins @ <https://ci.funbox.ru/>

Тесты

1. Нестабильность.

❗ frontend » tests #9481
built by Jenkins @ <https://ci.funbox.ru/>

✅ frontend » #11670
built by Jenkins @ <https://ci.funbox.ru/>

[Flaky Tests at Google and How We Mitigate Them](#)

[Making System User Interactive Tests Repeatable: When and What Should we Control?](#)



```
it('При создании происходит переход на страницу списка',
  async () => {
    await utils.addStub('POST', '/api/create', {});
    // await utils.addStub('GET', '/api/packages', {});

    await browser.click('[data-test-create-button]');

    await browser.waitForUrl('packages');
  }
);
```



```
afterEach(async function() {  
  if (browser.browserErrors.length > 0 && this.currentTest.state !== 'failed') {  
    test.callback(new Error(browser.browserErrors[0].msg));  
  }  
});
```

[@funbox/phantom-lord](https://twitter.com/funbox/phantom-lord)



```
afterEach(async function() {  
  if (browser.browserErrors.length > 0 && this.currentTest.state !== 'failed') {  
    test.callback(new Error(browser.browserErrors[0].msg));  
  }  
});
```



```
Error: TypeError: Cannot read property 'timestamp' of undefined  
at logger (webpack:///./node_modules/@frontend-js-group-1/fetcher/dist/index.js?:101:26)  
at logError (webpack:///./node_modules/@frontend-js-group-1/fetcher/dist/index.js?:139:18)  
at eval (webpack:///./node_modules/@frontend-js-group-1/fetcher/dist/index.js?:203:14)
```

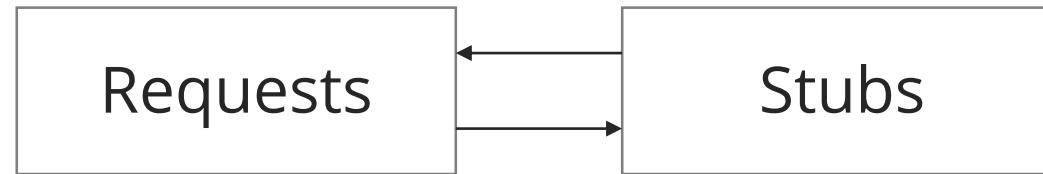


```
afterEach(async function() {  
  if (browser.browserErrors.length > 0 && this.currentTest.state !== 'failed') {  
    test.callback(new Error(browser.browserErrors[0].msg));  
  }  
});
```



```
async closePage() {  
  if (this.browserErrors.length > 0) {  
    console.log('Browser error occurred');  
  
    throw new Error(this.browserErrors[0].msg);  
  }  
}
```

[@funbox/phantom-lord](https://twitter.com/funbox/phantom-lord)



```
async afterEach() {  
  const [stubs, stubRequests] = await browser.evaluate(() => (  
    [window.stubs, window.stubRequests].map(i => Object.values(i).map(({ url }) => url))  
  ));  
  
  const requestsWithoutStubs = stubRequests  
    .filter(stubRequest => !stubs.some(stub => (/^re:/.test(stub)  
      ? new RegExp(stub.replace(/^re:/, '')).test(stubRequest)  
      : stub === stubRequest)));  
  
  if (requestsWithoutStubs.length) console.log(requestsWithoutStubs.join(', '));  
}
```




```
it('При создании происходит переход на страницу списка',
  async ( ) => {
    await utils.addStub('POST', '/api/create', {});
    await utils.addStub('GET', '/api/packages', {});

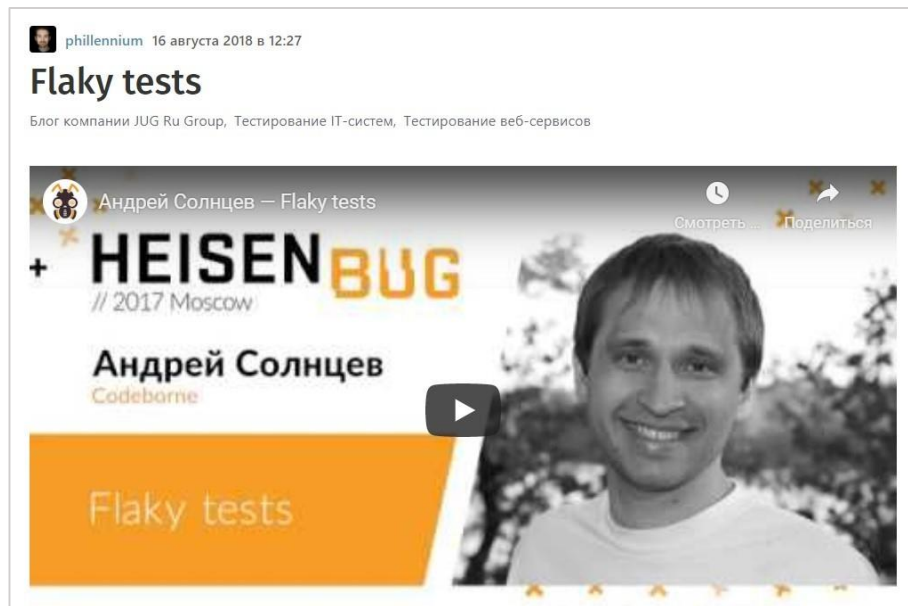
    await browser.click('[data-test-create-button]');

    await browser.waitForUrl('packages');
    await browser.waitForSelector('[data-test-packages]');
  }
);
```

Тесты

1. Нестабильность.

Flaky Tests



Тесты































1. Нестабильность.
2. Дублирование.



```
it('Должна быть кнопка сохранения',  
  async () => {  
    await browser.waitForSelectorText('[data-test-submit]', 'Сохранить и отправить');  
  }  
);
```



```
it('Должна быть кнопка сохранения',  
  async () => {  
    await browser.waitForSelectorText('[data-test-submit]', 'Сохранить и отправить');  
  }  
);  
  
it('Должен отобразиться диалог по нажатию на кнопку сохранения',  
  async () => {  
    await browser.clickSelectorText('[data-test-submit]', 'Сохранить и отправить');  
    await browser.waitForSelectorText('[data-test-submit-dialog]', 'Сохранить');  
  }  
);
```

 #6424 (pending—Already running 1 builds across all nodes)	 #6414 (pending—Already running 1 builds across all nodes)	 #6414 (pending—Already running 1 builds across all nodes)
 #6423 (pending—Already running 1 builds across all nodes)	 #6413 (pending—Already running 1 builds across all nodes)	 #6413 (pending—Already running 1 builds across all nodes)
 #6422 (pending—Already running 1 builds across all nodes)	 #6412 (pending—Already running 1 builds across all nodes)	 #6412 (pending—Already running 1 builds across all nodes)
 #6421 (pending—Already running 1 builds across all nodes)	 #6411 (pending—Already running 1 builds across all nodes)	 #6411 (pending—Already running 1 builds across all nodes)
 #6420 (pending—Already running 1 builds across all nodes)	 #6410 (pending—Already running 1 builds across all nodes)	 #6410 (pending—Already running 1 builds across all nodes)
 #6419 (pending—Already running 1 builds across all nodes)	 #6409 (pending—Already running 1 builds across all nodes)	 #6409 (pending—Already running 1 builds across all nodes)
 #6418 (pending—Already running 1 builds across all nodes)	 #6408 (pending—Already running 1 builds across all nodes)	 #6408 (pending—Already running 1 builds across all nodes)
 #6417 (pending—Already running 1 builds across all nodes)	 #6407 (pending—Already running 1 builds across all nodes)	 #6407 (pending—Already running 1 builds across all nodes)
 #6416 (pending—Already running 1 builds across all nodes)	 #6406 (pending—Already running 1 builds across all nodes)	 #6406 (pending—Already running 1 builds across all nodes)
 #6415 (pending—Already running 1 builds across all nodes)	 #6405 (pending—Already running 1 builds across all nodes)	 #6405 (pending—Already running 1 builds across all nodes)

Миллениалы переизобрели BDD

Требования → Тесты



Миллениалы переизобрели BDD

Требования ↔ Тесты

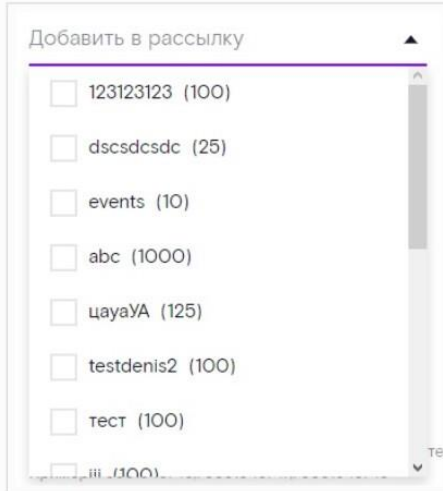


- **Добавить в рассылку** - выпадающий список групп получателей (созданных ранее в разделе "Клиенты"), которым будут рассылаться сообщения;
- **Исключить из рассылки** - выпадающий список групп получателей (созданных ранее в разделе "Клиенты"), которые будут исключены из рассылки;

❗ В истории **S RTT-3388** - Доработать функционал добавления клиентов и работы с группами **CLOSED**



1. **добавляется кол-во абонентов** в группе, для селектов "Добавить в рассылку", "Исключить из рассылки".

Примерно, но не обязательно, так:



В методе GET /client_groups эти данные передаются.

2. Группы в списках отсортированы по алфавиту

Добавить в рассылку	Указываются группы адресной книги – будущие получатели рассылки		<p>Для выбора доступны группы:</p> <ul style="list-style-type: none"> • С типом Белый; • В которых есть клиенты. <p>Указать можно несколько значений.</p>	<ul style="list-style-type: none"> • Всегда; • Указываются названия выбранных групп
Исключить из рассылки	Указываются группы адресной книги – клиенты, которым не должны отправляться сообщения рассылки		<p>Для выбора доступны группы:</p> <ul style="list-style-type: none"> • С типом Черный; • В которых есть клиенты. <p>Указать можно несколько значений.</p>	<ul style="list-style-type: none"> • Всегда; • Указываются названия выбранных групп

Тесты

1. Нестабильность.
2. Дублирование.
3. Анализ покрытия.

Puppeteer to Istanbul

build **passing** coverage **100%** release **standard version**

Convert coverage from the format outputted by [puppeteer](#) to a format consumable by [Istanbul](#).

Usage

To Output Coverage in Istanbul Format with Puppeteer

1. install *puppeteer*, `npm i -D puppeteer` .
2. install *puppeteer-to-istanbul*, `npm i -D puppeteer-to-istanbul` .
3. run.

[Puppeteer to Istanbul](#)



```
async function writePtiCoverage() {  
  const jsCoverage = await browser.page.coverage.stopJSCoverage();  
  
  await fs.emptyDir('.nyc_output');  
  
  pti.write(jsCoverage);  
}  
  
...  
  
it.only('Главная страница', async () => {  
  await browser.open(utils.url('home'));  
  await browser.page.coverage.startJSCoverage();  
  
  await browser.waitForSelector('[data-test-home]');  
  await writePtiCoverage();  
});
```

```

1698
1699 function _getPrototypeOf(o) { _getPrototypeOf = Object.setPrototypeOf ? Object.getPrototypeOf : function _getPrototypeOf(o) { return o.__proto__ || Object.getPrototypeOf(o); }; return _getPrototypeOf(o); }
1700
1701 function _defineProperty(obj, key, value) { if (key in obj) { Object.defineProperty(obj, key, { value: value, enumerable: true, configurable: true, writable: true }); } else { obj[key] = value; } return obj; }
1702
1703 var __signature__ = typeof reactHotLoaderGlobal !== 'undefined' ? reactHotLoaderGlobal.default.signature : function (a) {
1704   return a;
1705 };
1706
1707 var Settings = /*#__PURE__*/function (_PureComponent) {
1708   _inherits(Settings, _PureComponent);
1709
1710   var _super = _createSuper(Settings);
1711
1712   function Settings(props) {
1713     var _this;
1714
1715     _classCallCheck(this, Settings);
1716
1717     cov_nefqvzylj().f[0]++;
1718     cov_nefqvzylj().s[0]++;
1719     _this = _super.call(this, props);
1720
1721     _defineProperty(_assertThisInitialized(_this), "submit", (cov_nefqvzylj().s[31]++, function (event) {
1722       cov_nefqvzylj().f[8]++;
1723       cov_nefqvzylj().s[32]++;
1724       event.preventDefault();
1725       cov_nefqvzylj().s[33]++;
1726
1727       _this.validateForm();
1728
1729       cov_nefqvzylj().s[34]++;
1730
1731       if (!_this.isFormCanSubmit()) {
1732         cov_nefqvzylj().b[12][0]++;
1733         cov_nefqvzylj().s[35]++;
1734         return;
1735       } else {
1736         cov_nefqvzylj().b[12][1]++;
1737       }
1738     }

```

github.com/istanbuljs/puppeteer-to-istanbul/issues/18



```
module: {  
  rules: [  
    {  
      use: {  
        loader: 'babel-loader',  
        options: {  
          plugins: ['istanbul'],  
        },  
      },  
    },  
  ],  
},
```

[babel-plugin-istanbul](#)



```
async function writeCoverage() {
  const coverage = await browser.page.evaluate(() => window.__coverage__);

  await Promise.all(
    Object.values(coverage).map(cov => fs.writeFileSync(
      `.nyc_output/${cov.hash}-${moment().unix()}.json`,
      { [cov.path]: cov }
    )),
  );
}

afterEach(() => {
  if (process.env.COVERAGE) await utils.writeCoverage();
});
```

istanbul.js.org/docs/advanced/coverage-object-report


```

80         ? ERROR_MESSAGE.INVALID_PASSWORD
81         : '';
82     },
83 },
84 newPasswordConfirm: {
85     validator: value => (
86     6x     value !== this.getFieldValue('newPassword')
87         ? ERROR_MESSAGE.DIFFERENCE_PASSWORD
88         : ''
89     ),
90 },
91 email: {
92     validator: value => {
93     6x     switch (true) {
94     3x     case !value:
95         return ERROR_MESSAGE.EMPTY_EMAIL;
96
97         case !FORMAT_REGEX.EMAIL.test(value):
98         return ERROR_MESSAGE.INVALID_EMAIL;
99
100     default:
101     3x     return '';
102     }
103 },
104 },
105 });

```

Иллюзия контроля как причина

1. Благоприятный исход.



Иллюзия контроля как причина

1. Благоприятный исход.
2. Личная вовлеченность и привычность.



Неиспользуемый код (DCE)





```
// export.js  
export function square(x) {  
  return x * x;  
}  
  
export function cube(x) {  
  return x * x * x;  
}  
  
// import.js  
import { cube } from './export.js';
```



```
mode: 'development',  
optimization: {  
  usedExports: true,  
}
```

Tree Shaking





```
'use strict';

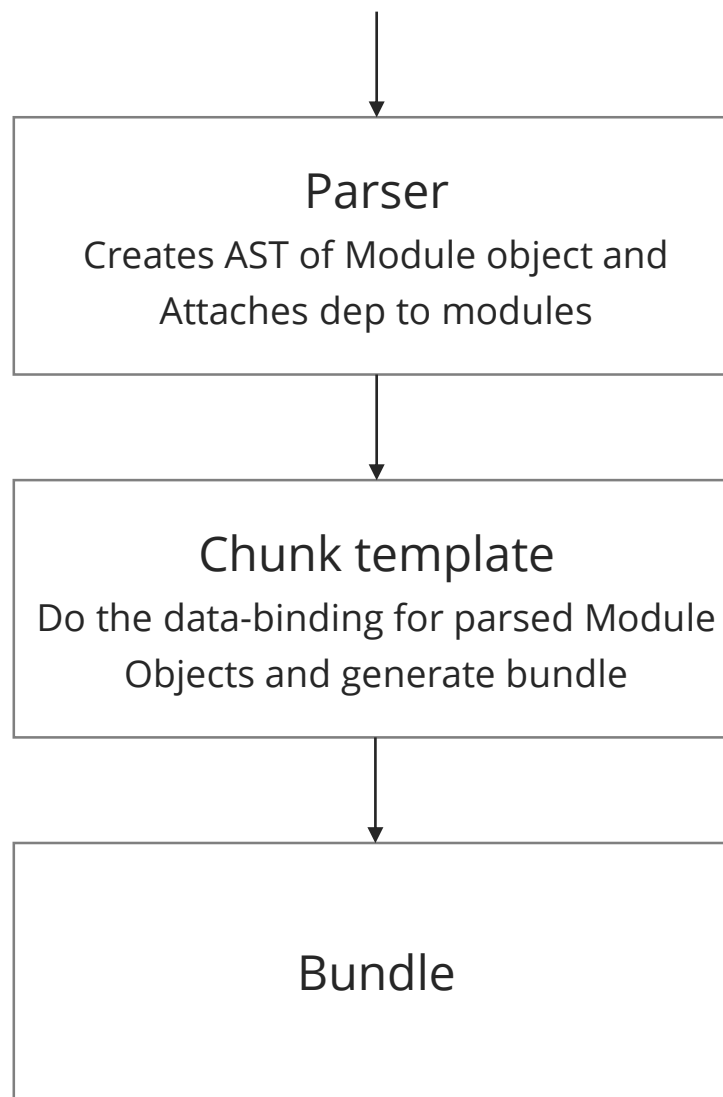
/* unused harmony export square */
/* harmony export (immutable) */
__webpack_exports__['a'] = cube;

function square(x) {
  return x * x;
}

function cube(x) {
  return x * x * x;
}
```

[Tree-shaking? Let's implement it!](#)

[Webpack — Behind the Scenes](#)





```
rules: {  
  "import/no-unused-modules": [1, { "missingExports": true }],  
}
```



```
// export.js  
export default List;  
export { List, List__Item };  
  
// import.js  
import { List, List__Item } from 'export.js';
```




```
switch (Direction) {  
  case 'Up':  
    // ...  
  case 'Down':  
    // ...  
  case 'Right': // Недостижимо  
    // ...  
  default:  
}
```



TS

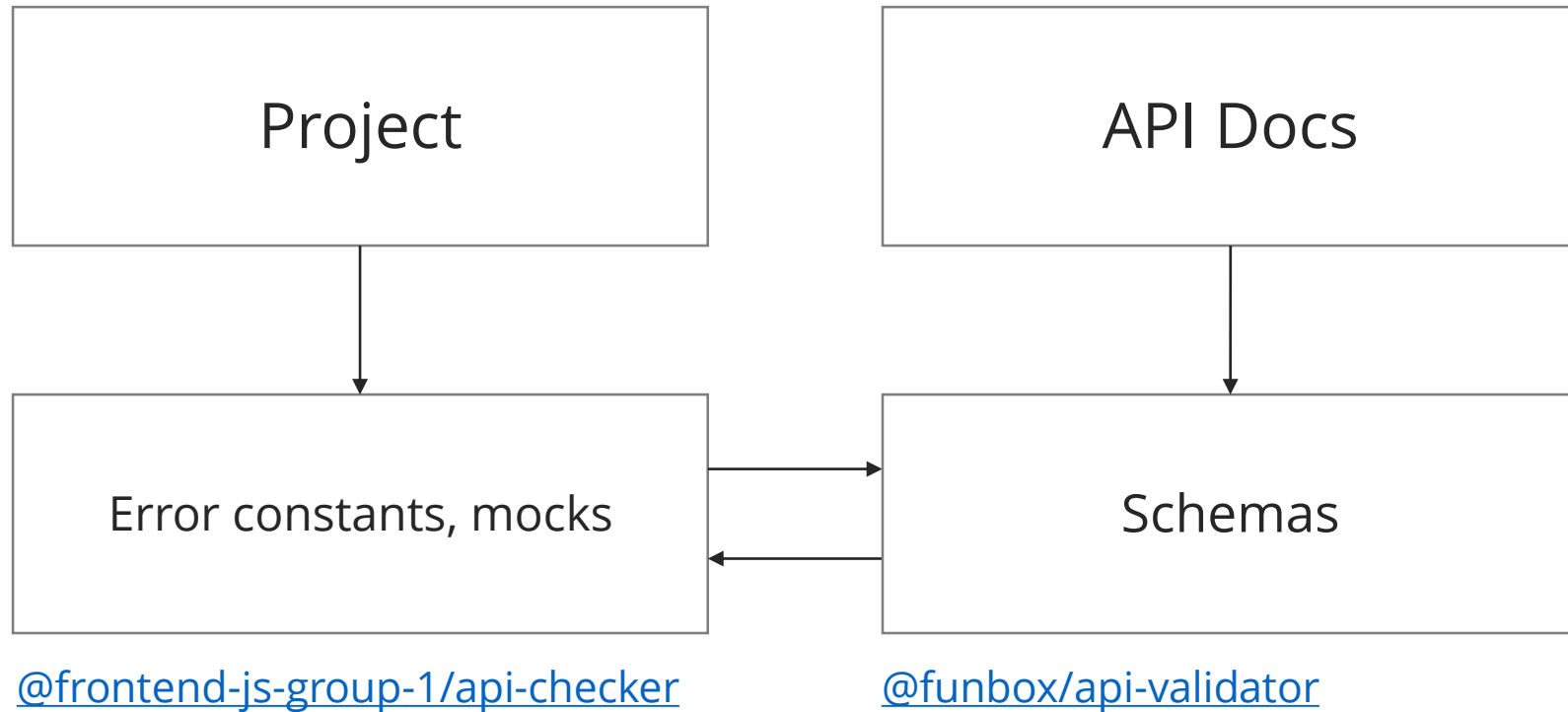
```
const enum Directions { Up, Down }

function someFunc(op: Directions) {
  switch (op) {
    case Directions.Up:
      // ...
    case Directions.Down:
      // ...
    case Directions.Right: // Error
      // ...
    default:
  }
}
```

```

80         ? ERROR_MESSAGE.INVALID_PASSWORD
81         : '';
82     },
83 },
84 newPasswordConfirm: {
85     validator: value => (
86     6x     value !== this.getFieldValue('newPassword')
87         ? ERROR_MESSAGE.DIFFERENCE_PASSWORD
88         : ''
89     ),
90 },
91 email: {
92     validator: value => {
93     6x     switch (true) {
94     3x     case !value:
95         return ERROR_MESSAGE.EMPTY_EMAIL;
96
97         case !FORMAT_REGEX.EMAIL.test(value):
98         return ERROR_MESSAGE.INVALID_EMAIL;
99
100     default:
101     3x     return '';
102     }
103 },
104 },
105 });

```



И ещё

[depcheck — dependencies](#)

[unused-webpack-plugin — files](#)

[\[Open\] Detect unused class component method](#)



```
plugins: [  
  "no-unused-react-component-methods"  
],  
rules: {  
  "react/no-unused-class-component-methods": 'error',  
}
```

Неиспользуемый код (DCE)

1. ESLint (модули, методы, переменные и т. д.).
2. TypeScript (недостижимые ветки).
3. Отчеты о покрытии (недостижимые сценарии).
4. webpack (файлы, модули).
5. API документация (методы, ошибки).
6. 🕒



Иллюзия контроля как причина

1. Благоприятный исход.
2. Личная вовлеченность и привычность.



Иллюзия контроля как причина

1. Благоприятный исход.
2. Личная вовлеченность и привычность.
3. Направленность на успех.



Миграция

Стек

1. Angular.
2. CoffeeScript.
3. Bootstrap.
4. Slim.

Миграция

Стек

1. Angular.
2. CoffeeScript.
3. Bootstrap.
4. Slim.
5. jQuery.



Миграция

StranglerFig?

[StranglerFigApplication, Martin Fowler](#)



Миграция

StranglerFig? Переписать с нуля

1. Нет тестов.
2. Нет документации API.
3. Нет дизайна.
4. Есть резерв.
5. Период без нового функционала.

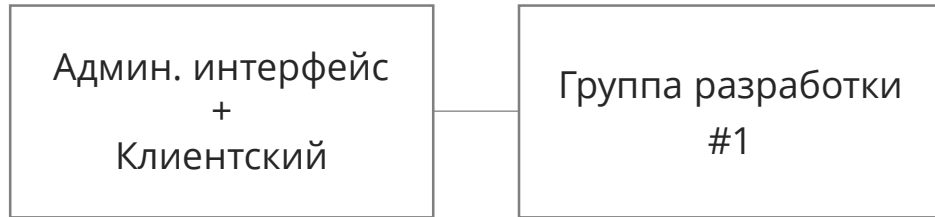
[StranglerFigApplication, Martin Fowler](#)



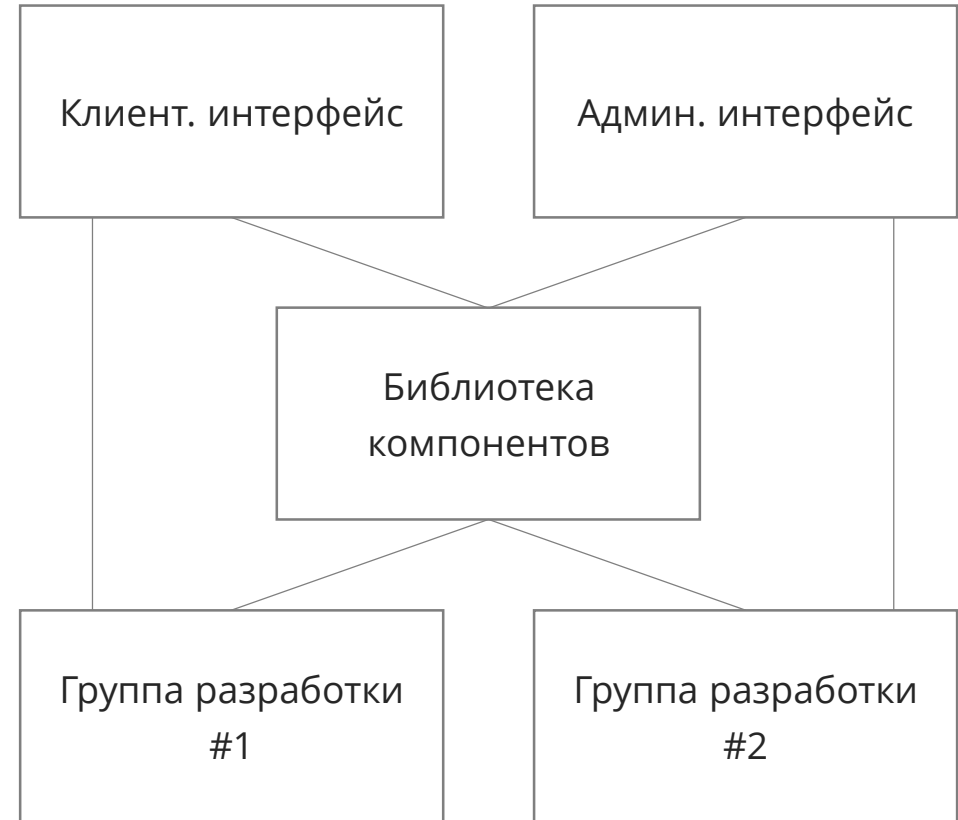
1. Отсутствие дублирования.
2. Высокая стоимость интеграций.



1. Отсутствие дублирования.
2. Высокая стоимость интеграций.



1. Проблема дублирования преувеличена.
2. Независимое окружение и процессы.



Миграция

Ожидание



Миграция

Реальность



Материалы

1. [Flaky Tests at Google and How We Mitigate Them](#)
2. [Making System User Interactive Tests Repeatable: When and What Should we Control?](#)
3. [Flaky Tests, Андрей Солнцев](#)
4. [StranglerFigApplication, Martin Fowler](#)
5. [Ctrl-Alt-Del: Learning to Love Legacy Code, Dylan Beattie](#)
6. [Tree-shaking? Let's implement it!](#)
7. [webpack — Behind the Scenes](#)
8. Книга «Рефакторинг кода на JavaScript», М. Фаулер

Иллюстрации

1. artify.co
2. WTFs/m
3. Max and Maurice
4. technoir.nl

Спасибо за внимание

