УДК 004.416.6

# МИГРАЦИЯ ПРОЕКТА НА ЯЗЫК ПРОГРАММИРОВАНИЯ TYPESCRIPT ПРИ РАЗРАБОТКЕ НА ФРЕЙМВОРКЕ NODE.JS

**Вахрамов С.В., Мусин А.М., Ризванов Д.А., Хамитов М.А.**

*ФГБОУ ВО «Уфимский государственный авиационный технический университет», Уфа, e-mail: s@vakhramoff.ru*

Большинство современных Node.js-проектов написаны на языке программирования JavaScript. За последние два года TypeScript (надмножество над JavaScript) набрал очень большую популярность. В статье рассматривается процесс миграции Node.js проектов с языка программирования JavaScript на TypeScript. Проанализированы тренды популярности языка, приведены графики, показывающие её рост, пошагово описан алгоритм действий для решения поставленной задачи. Данный способ учитывает особенности языка TypeScript и специализированных библиотек, которые работают с фреймворком Node.js. Кроме того, описан процесс типизации библиотек, включая коды команд, позволяющие выполнить миграцию с помощью утилиты терминала в macOS или командной строки в Windows. В дополнение описаны особенности данного перехода, а также его необходимость при использовании достаточно больших проектов, имеющих свойство увеличиваться в объёме в пересчёте на количество строк кода и количество исходных файлов. Результаты миграции подробно описывают преимущества и недостатки самого языка TypeScript. В заключение приводится статистика, полученная в результате применения данного подхода на одном из собственных проектов. Удалость снизить количество ошибок разработки на 15%, облегчить отладку кода на 10%, а знакомство с проектом новых разработчиков стало до 50% быстрее.

**Ключевые слова: TypeScript, JavaScript, Node.js, npm, API, back-end, lebab, EcmaScript, миграция проекта, популярность TypeScript**

# MIGRATION OF A PROJECT WRITTEN USING NODE.JS FRAMEWORK TO TYPESCRIPT PROGRAMMING LANGUAGE

**Vakhramov S.V., Musin A.M., Rizvanov D.A., Khamitov M.A.**

*Ufa State Aviation Technical University, Ufa, e-mail: s@vakhramoff.ru*

Many Node.js projects are written in TypeScript (a superset of JavaScript). This article describes the migration process of Node.js projects from JavaScript to TypeScript programming language. The popularity of TypeScript has been analized, and step-by-step algorithm of migration process was described to solve the problem. This approach also considers TypeScript and special libraries' features, which work with Node.js framework. Besides that, the typification process for libraries was described, including special terminal (in macOS) or console (in Windows) commands which lead to perform necessary tasks. In addition to all of that, the migration specificity and necessity was described in case of developing huge projects, which grow in a long-term period of time counting in lines of codes and source files amount. The results of the migration show advantages and disadvantages of the TypeScript language itself. In conclusion, the statistics of this approach was shown with regard to one of our personal projects. By using this approach mistakes in project were reduced by 15%.

**Keywords: TypeScript, JavaScript, Node.js, npm, API, back-end, lebab, EcmaScript, project migration, TypeScript popularity**

Успешное продвижение языка TypeScript в рейтинге языков программирования обусловлено его основанностью на шаблонах: это способствует реализации принципа повторного использования и позволяет смешиваться с существующей кодовой базой в JavaScript, способствуя получению более безопасного типизированного кода.

По данным Google Trends с рисунка можно увидеть растущую популярность TypeScript. За последние 5 лет рост популярности запроса составил 80 баллов: с 12 до 92 баллов, приблизившись к максимальному пику в 100 баллов – когда запрос показывает наивысший уровень популярности.

Весомая причина перехода на TypeScript – уменьшение количества затрачиваемого времени на тестирование своего кода: объём кода уменьшается на 10–20% при миграции с JavaScript на TypeScript.

Интерес среди пользователей сети Интернет равномерно увеличивается из года в год на протяжении более пяти лет, об этом свидетельствует рейтинг популярности языков программирования, по версии аналитической компании RedMonk, за период первого квартала 2020 года, изображенного на рис. 2.

Согласно рейтингу, TypeScript набирает популярность, занимая 9-ю строчку, что видно из рейтинга, представленного на рис. 3.

Согласно статистике с сайта github.com, TypeScript находится на 5-м месте среди языков, набирающих популярность.

Сообщество активно развивает TypeScript: на сегодняшний день существует около 7000 пакетов, поддерживающих описание типов для языка программирования TypeScript для уже существующих JavaSript-пакетов [4]. Количество пакетов,

изначально написанных на самом языке TypeScript, исчисляется десятками тысяч.

Проекты на Node.js используются для организации работы с серверной частью, преимущественно с данными пользователей. Эти данные часто включают в себя персональные данные людей.

Правильный проект на Node.js должен быть грамотно структурирован: отдельно выделены модели и контроллеры. Использование TypeScript позволяет с помощью интерфейсов дополнительно описать представления, что в будущем не даст допустить ошибку в программном коде проекта, например путём передачи строки вместо числа, что является самой частой проблемой при разработке на JavaScript, которую особенно трудно обнаружить при отладке.

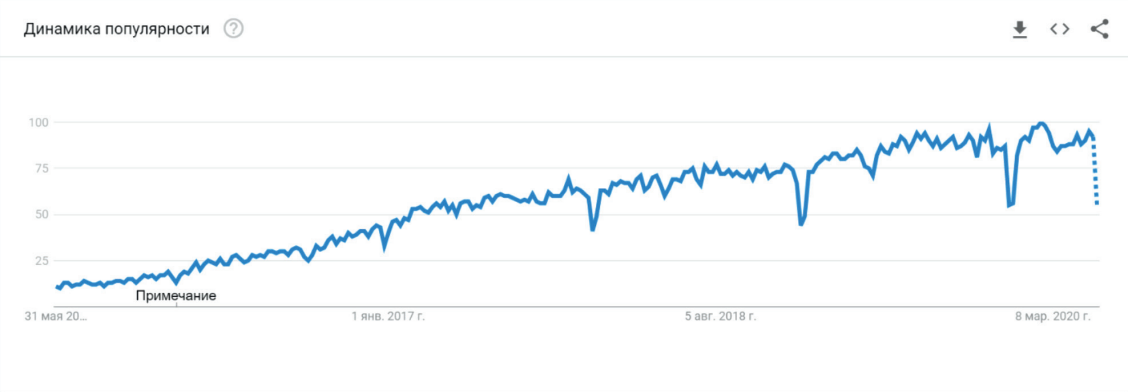


*Рис. 1. Динамика популярности Google Trend по запросу TypeScript за 2015–2020 гг. [1]*

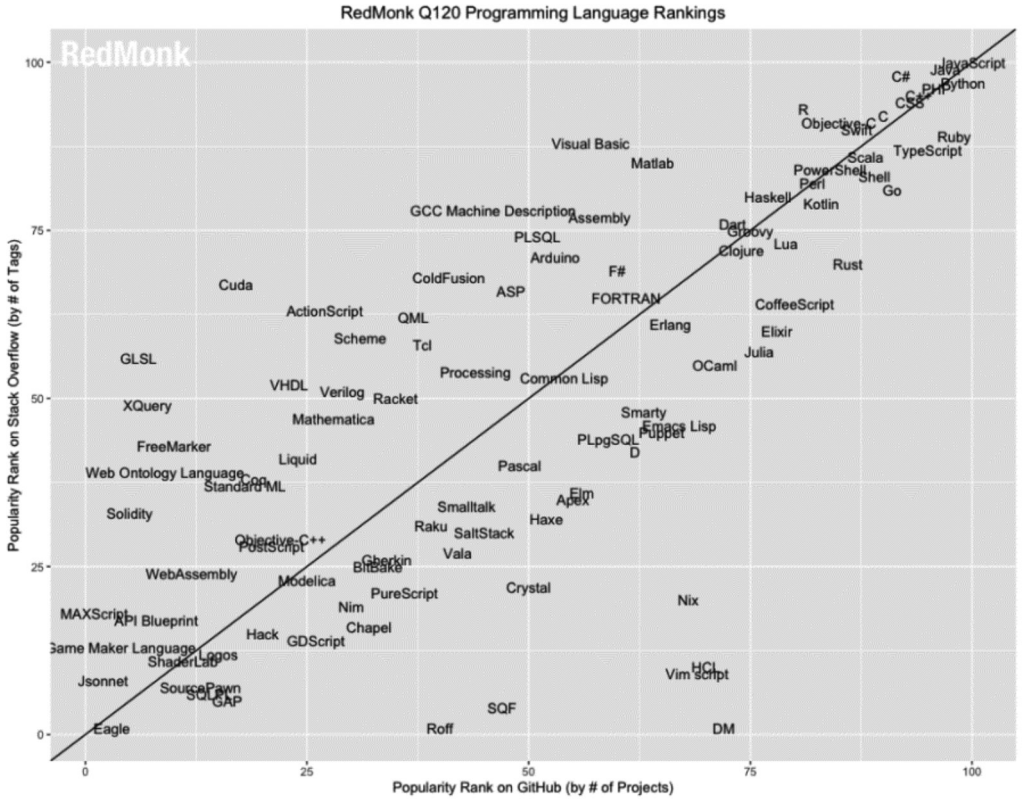


*Рис. 2. Рейтинг популярности языков программирования, по версии аналитической компании RedMonk, за первый квартал 2020 года [2]*

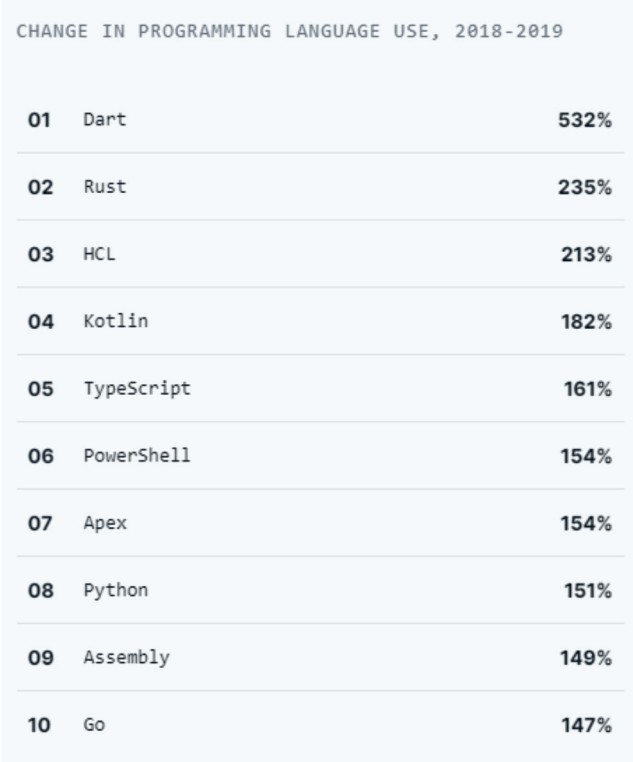


*Рис. 3. Рост популярности языков программирования за 2018–2019 гг. [3]*

Дополнительным преимуществом использования TypeScript является мощнейший механизм описания типов: мы не просто описываем базовые типы, с помощью общих (Generic) типов имеем возможность описать абстрактную конструкцию, которую в дальнейшем сможем использовать вкупе с различными базовыми типами.

Например:

```
interface GenericIdentityFn {
  <T>(arg: T): T;
}
function identity<T>(arg: T): T {
  return arg;
}
let myIdentity: GenericIdentityFn = identity;
```

JavaScript является языком динамическим строго типизированным, в то время как TypeScript статическим слабо типизированным. Рассмотрим далее преимущество каждого из подходов [5].

*Преимущества статической типизации (static typing):*

– проверки типов происходят один раз на этапе компиляции, не нужно выяснять, нет ли попытки деления числа на строку (и затем выдать ошибку или осуществить преобразование);

– скорость выполнения: статически типизированные языки в большинстве случаев быстрее динамически типизированных ввиду предыдущего пункта;

– позволяет обнаруживать потенциальные ошибки уже на этапе компиляции;

– ускоряет процесс разработки (при выдаче подсказок заранее отсеиваются варианты, не подходящие по типу).

*Преимущества динамической типизации (dynamic typing):*

– легко создаются универсальные коллекции, содержащие в себе данные разных типов;

– удобно писать обобщённые алгоритмы (сортировка не только числового массива, а массива строк или произвольных структур данных);

– языки с динамической типизацией легки для освоения. Они очень хороши для того, чтобы начать программировать.

*Преимущества сильной типизации (strong typing):*

– надежность – вместо неожиданного поведения в рантайме программист получает ошибку на этапе компиляции;

– скорость – в коде нет скрытых преобразований типов, при написании кода программист может точно предположить, будет ли участок кода затрачивать много времени на выполнения;

– понимание работы программы – чётко понятно, какая структура с какой сравнивается, как преобразуются данные: не возникает ситуации «волшебного» приведения типов;

– определенность – при выполнении преобразований вручную ясно, какие данные и каким образом преобразуются из начальной формы в заданную.

*Преимущества слабой типизации (weak typing):*

– *возможно* использование смешанных выражений (например, из вещественных и целых чисел);

– абстрагирование от типизации и сосредоточение на задаче;

– краткость записи.

Цель исследования: исследование процесса миграции проекта, написанного на фреймворке Node.js, с языка программирования JavaScript на язык TypeScript с целью ускорения разработки проекта. Выявление преимуществ и недостатков использования языка TypeScript.

## Материалы и методы исследования

Миграция проекта требует использование терминала (командной строки). При выполнении команд, описанных далее, будут установлены необходимые зависимости для перехода на язык TypeScript.

1. Установить npm-пакет TypeScript:

```
npm install --save-dev typescript@latest
```

2. Установить TS-Node:

```
npm install --save-dev ts-node@latest
```

3. Сконфигурировать компилятор TypeScript минимально рекомендуемыми настройками:

```
{
  "compilerOptions": {
    "module": "commonjs",
    "target": "esnext",
    "sourceMap": true,
    "moduleResolution": "node",
    "outDir": "dist"
  },
  "lib": ["es2020"],
  "compileOnSave": false,
  "exclude": [
    "node_modules"
  ]
}
```

4. Переписать код на ES6-синтаксис.

5. Автоматизировать процесс можно с помощью утилиты lebab [6].

6. Переименовать файлы с расширений .js на расширения .ts.

7. Установить типизированные пакеты для инструментов, используемых с Node.js.

В нашем случае необходимыми для установки были следующие пакеты:

```
npm install --save-dev @types/argon2
npm install --save-dev @types/dotenv
npm install --save-dev @types/express
npm install --save-dev @types/express-jwt
npm install --save-dev @types/jsonwebtoken
npm install --save-dev @types/mongoose
npm install --save-dev @types/node
```

8. Настроить nodemon для перекомпиляции проекта при изменении ts-файлов путём изменения nodemon.json

```
{
  "watch": ["**/*.ts"],
  "ignore": ["**/*.spec.ts"],
  "exec": "ts-node ./index.ts"
}
```

9. Написать скрипт запуска проекта, добавив необходимую конфигурацию в package.json

```
"scripts": {
  "start": "tsc && ts-node dist/app.js"
}
```

10. Дальнейший запуск и разработка с автоматической перекомпиляцией проекта могут вестись по следующей команде в терминале:

```
npm run start
```

## Результаты исследования и их обсуждение

TypeScript активно развивается и обладает рядом преимуществ над JavaScript. В следующем разделе будут описаны положительные и отрицательные стороны миграции на TypeScript.

Преимущества:

– Строгая типизация.

– При использовании Node.js можно пользоваться последними возможностями стандартов ES7 и ES8 [7; 8], в то время как на JavaScript максимальный доступный стандарт – ES6.

– Типизированные сигнатуры методов в IDE позволяют видеть полное описание метода и принимаемых параметров.

– При разработке фронтенда с помощью TypeScript можно понимать, результаты каких типов ждать [9].

– Упрощённая автогенерация документации.

– Тестирование и отладка кода проходят легче [9].

– Понятия из мира ООП:

● Interface – позволяет описывать контракты взаимодействующих сущностей, а также методы и свойства, доступные для кода внутри и вне приложения [10].

● Enum – определяет именованные константы.

● Generics – механизм декларации классов и методов для работы со множественными типами данных.

– Async/await

– Await-функции высшего уровня:

**Код**

```
let f = async () => {
   return new Promise((resolve) => {
      setTimeout(() => resolve(`I'm a cool async response!`), 1000);
   });
};
console.info(await f());
```

Выведет в консоль через 1 секунду сообщение:

I'm a cool async response!

**Недостатки**:

– Типизировано большинство библиотек, но не все. Могут возникнуть ситуации, когда придётся использовать нетипизированный JavaScript код [11].

– Много времени уходит на описание интерфейсов.

– Некоторые думают, что привнесение TypeScript в проект: увеличит время на изучение проекта новыми разработчиками, усложнит поддержку, увеличит время разработки, оставит проект в хипстерском состоянии (когда через 1 год язык станет неподдерживаемым и ненужным), предотвратит вход в проект JavaScript-разработчиков, сделает невозможным использование кодовой базы не на TypeScript и усложнит изменение функциональности приложения в отдалённом будущем.

## Выводы

Мигрируя на TypeScript, удалось снизить количество ошибок на 15% при разработке и сократить время написания кода на примерно 10%: облегчилась отладка, а благодаря инструментам автоматической генерации документации отпала необходимость в ручном комментировании кода. Знакомство с проектом для новых разработчиков при этом стало на 30% быстрее. Для опытных разработчиков этот показатель достиг 50%.

## Список литературы

1. Google Trends: Typescript. [Electronic resource]. URL: https://trends.google.com/trends/explore?date=today%205-y&q=%2Fm%2F0n50hxv (date of access: 12.07.2020).

2. The RedMonk Programming Language Rankings: January 2020. [Electronic resource]. URL: https://redmonk.com/sogrady/2020/02/28/language-rankings-1-20 (date of access: 12.07.2020).

3. ECMAScript 2016+ compatibility table. [Electronic resource]. URL: http://kangax.github.io/compat-table/es2016plus (date of access: 12.07.2020).

4. Search: @types – npm. [Electronic resource]. URL: https://www.npmjs.com/search?q=%40types&ranking=popularity (date of access: 12.07.2020).

5. Ликбез по типизации в языках программирования. [Электронный ресурс]. URL: https://habr.com/ru/post/161205 (дата обращения: 12.07.2020).

6. The State of the Octoverse celebrates a year of building across teams, time zones, and millions of merged pull requests. [Electronic resource]. URL: https://octoverse.github.com (date of access: 12.07.2020).

7. Turn your ES5 code into readable ES6. Lebab does the opposite of what Babel does. [Electronic resource]. URL: https://github.com/lebab/lebab (date of access: 12.07.2020).

8. What is TypeScript and why would I use it in place of JavaScript? [Electronic resource]. URL: https://stackoverflow.com/questions/12694530/what-is-typescript-and-why-would-i-use-it-in-place-of-javascript/35048303#35048303 (date of access: 12.07.2020).

9. TypeScript Roadmap: January – June 2020. [Electronic resource]. URL: https://github.com/microsoft/TypeScript/issues/36948 (date of access: 12.07.2020).

10. Why TypeScript is the best way to write Front-end in 2019–2020. [Electronic resource]. URL: https://medium.com/@jtomaszewski/why-typescript-is-the-best-way-to-write-front-end-in-2019-feb855f9b164 (date of access: 12.07.2020).

11. The Benefits of Migrating from JavaScript to TypeScript [Electronic resource]. URL: https://www.appdynamics.com/blog/engineering/the-benefits-of-migrating-from-javascript-to-typescript (date of access: 12.07.2020).