**Typescript**

**Книги**

TypeScript Deep Dive стр. 9 – 22

Pro TypeScript: Application-Scale JavaScript Development cтр.47

**Сайты**

TypeScript

<https://metanit.com/web/typescript/1.1.php>

<http://typescript-lang.ru/docs/>

<https://www.typescriptlang.org/docs/home.html>

Статическая и динамическая типизация

<https://habr.com/ru/post/308484/>

JavaScript vs TypeScript: что в каких случаях лучше использовать

<https://proglib.io/p/javascript-vs-typescript/>

Мысли вслух о TypeScript

<https://habr.com/ru/post/272055/>

TypeScript – Изучаем с нуля

<https://webformyself.com/typescript-izuchaem-s-nulya/>

* **Необходимость использования TypeScript.**

JavaScript — очень динамичный язык программирования, который многие противопоставляют С-подобным языкам. Его не типизированная природа и уникальное прототипное наследование послужили переходом большого количества разработчиков на JavaScript.

Динамически типизированные языки не требуют указывать тип, но и не определяют его сами. Типы переменных неизвестны до того момента, когда у них есть конкретные значения при запуске. Большинство динамических языков выдадут ошибку, если типы используются некорректно (JavaScript — известное исключение; он пытается вернуть значение для любого выражения, даже когда оно не имеет смысла). При использовании динамически типизированных языков даже простая ошибка вида "a" + 1 может возникнуть в боевом окружении.

TypeScript больше оправдывает себя в крупных проектах. Это связано с тем, что разработка на нем занимает больше времени чем на JavaScript из-за того, что приходится помимо методов и классов описывать и их декларации. Но тем не менее, пока в JavaScript нет статической типизации, TypeScript является отличной альтернативой.

* **Как решает проблемы.**

TypeScript - это строго типизированный и компилируемый язык, чем, возможно, будет ближе к программистам Java, C# и других строго типизированных языков. Хотя на выходе компилятор создает все тот же JavaScript, который затем исполняется браузером, так как ориентируется прежде всего на стандарт ECMAScript 3, хотя TS также поддерживает и стандарты ECMAScript 5 и ECMAScript 2015 / 2017. Хотя в процессе разработки мы можем сами задать целевой стандарт ECMAScript.. В TS можно использовать все те конструкции, которые применяются в JS - те же операторы, условные, циклические конструкции. Более того код на TS компилируется в Javascript. В конечном счете, TS - это всего лишь инструмент, который призван облегчить разработку приложений. Однако строгая типизация уменьшает количество потенциальных ошибок, которые могли бы возникнуть при разработке на JavaScript.

Потенциал TypeScript-а позволяет быстрее и проще писать большие сложные комплексные программы, соответственно их легче поддерживать, развивать, масштабировать и тестировать, чем на стандартном JavaScript.

* **Детальное описание технологии.**

TypeScript возник из-за предполагаемых недостатков JavaScript в крупномасштабных приложениях как в Microsoft, так и у прочих пользователей JavaScript. Проблемы с разработкой сложных программ на JavaScript привели к необходимости облегчения разработки компонентов языка.

Разработчики TypeScript искали решение, которое не будет нарушать совместимость со стандартом и его кроссплатформенной поддержкой. Зная, что только стандарт ECMAScript предлагает поддержку в будущем для программирования на базе классов (Class-based programming), TypeScript был основан на этом предположении. Это привело к созданию компилятора JavaScript с набором синтаксических языковых расширений, увеличенным на основе предложения, которое трансформирует расширения в JavaScript. В этом смысле TypeScript является представлением того, что ожидать от ECMAScript 6. Уникальный аспект не в предложении, а в добавлении в TypeScript статической типизации, что позволяет статически анализировать язык, облегчая оснастки и IDE Для написания кода на языке TypeScript можно использовать любой самый простейший текстовый редактор. Многие текстовые редакторы и среды разработки, например, Visual Code Studio, Atom, Sublime, Visual Studio, Netbeans, WebStorm и другие, имеют поддержку TypeScript на уровне плагинов, что позволяет воспользоваться рядом преимуществом, например, подцветкой кода или всплывающей подсказкой по типам и конструкциям языка.

TypeScript реализует многие концепции ООП, такие как наследование, полиморфизм, инкапсуляция и модификаторы доступа. В нем есть классы, интерфейсы и абстрактные классы.

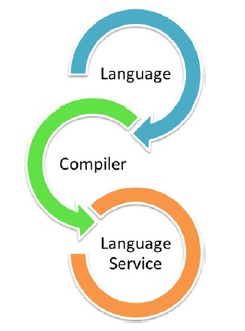
Потенциал языка позволяет быстрее и проще писать сложные комплексные решения, которые легче развивать и тестировать в дальнейшем, чем на стандартном JavaScript.

Поскольку данный язык является OpenSource, то все его инструменты доступны для всех желающих. Адрес репозитория - https://github.com/Microsoft/TypeScript. Кроме того, он является кроссплатформенным, а это значит, для работы с TypeScript мы можем использовать как Windows, так и Linux и MacOS.

*Typescript имеет не так много недостатков:*

* В процессе разработки имеем дело с файлами \*.ts, \*.d.ts, \*.map, \*.js. Слишком много дополнительных файлов, что бывает неудобно, если ваш проект небольшой.
* Не все браузеры поддерживают отладку TypeScript в консоли без лишних настроек.
* Множество нетривиальных классов. Чтобы писать код, опираясь на классы, приходится держать в голове какое свойство где находится. Например, вместо одного класса Event существуют еще такие как MouseEvent, TouchEvent, KeyboardEvent и другие...
* Неявная статическая типизация. Всегда можно описать тип как any, что по факту отключит приведение к конкретному типу этой переменной.
* Это транспайлер, что подразумевает, что мы должны всегда иметь под рукой tsc
* d.ts декларации поддерживаются сообществом DefinitelyTyped и часто не соответствуют текущей версии библиотеки. Либо не учитывают сложных вариантов (generic-функции, возвращаемые значения нескольких типов)

*В основе TypeScript лежат три компонента:*



*Основные три компонента TypeScript(Язык, Компилятор, Языковая служба)*

* Язык – Он включает в себя синтаксис, ключевые слова и сигнатуры типа.
* Компилятор TypeScript − Компилятор TypeScript (tsc) конвертирует инструкции, написанные в TypeScript для его JavaScript эквивалентов.
* Языковая служба TypeScript – «Языковая служба» предоставляет дополнительный уровень вокруг конвейера основного компилятора, которым являются редакторы. Языковая служба поддерживает стандартный набор стандартных операций по редактированию, например, автодополнение инструкций, автопоподстановки для сигнатур, форматирования и т.д.
* **Обоснование выбора TypeScript.**

Основное достоинство TypeScript – это отличный инструментарий. Здесь обеспечивается сложное авто завершение, навигация и рефакторинг. В больших проектах без таких инструментов практически не обойтись. Своими силами вечно не решаешься окончательно изменить код, вся база кода пребывает в сыром виде, и любой крупный рефакторинг становится рискованным и затратным.

TypeScript во многом превосходит все аналоги, к примеру, такие языки программирования как CoffeeScript и Dart, в том плане, что TypeScript – это расширенный JavaScript. Для сравнения, Dart и CoffeeScript являются новыми языками, и для них требуется среда выполнения, ориентированная на конкретный язык.

Angular построен на TypeScript. Рекомендуется писать приложения Angular на TypeScript. Необходимо писать код на современном JS (ES6).