Сергеєв Єгор Ігорович
студент,
Національний технічний університет України «Київський політехнічний інститут»
Прасолов Андрій Павлович
студент,
Національний технічний університет України «Київський політехнічний інститут»

Сергеев Егор Игоревич
студент,
Национальный технический университет Украины «Киевский политехнический институт»
Прасолов Андрей Павлович
студент,
Национальный технический университет Украины «Киевский политехнический институт»

Serheiev Y.
student
National Technical University of Ukraine «Kyiv Polytechnic Institute»
Prasolov A.
student
National Technical University of Ukraine «Kyiv Polytechnic Institute»

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ TYPESCRIPT ТА JAVASCRIPT В РОЗРОБЦІ СУЧАСНИХ ВЕБ-ДОДАТКІВ

СРАВНЕНИЕ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ TYPESCRIPT И JAVASCRIPT В РАЗРАБОТКЕ СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ

COMPARING TYPESCRIPT AND JAVASCRIPT PROGRAMMING LANGUAGES IN MODERN WEB APPLICATIONS DEVELOPMENT

Анотація. *Порівняно мови програмування TypeScript та JavaScript, показано, що використання TypeScript полегшує розробку та тестування веб-додатків.*
Ключові слова: *JavaScript, TypeScript, веб-додаток, програмування.*

Аннотация. *Сравнены языки программирования TypeScript и JavaScript, показано, что использование TypeScript облегчает разработку и тестирование веб-приложений.*
Ключевые слова: *JavaScript, TypeScript, веб-приложение, программирование.*

Summary. *Were compared TypeScript and JavaScript programming languages, were shown, that TypeScript usage simplifies web-application development and testing.*
Keywords: *JavaScript, TypeScript, web-application, programming.*

Вступ

Як усім відомо, кожен із веб розробників, хоч бек-енд чи фронт-енд повинен досконало знати HTML та JavaScript. Без цих мов, включаючи й CSS неможливо створити повноцінний, сучасний веб-ресурс. Звичайно, сучасні веб-додатки створюються з використанням серверних технологій, таких як NodeJS чи ASP.NET та ін., разом фронт-енд та бек-енд частини утворюють гармонійний веб-додаток, що не поступається десктопним, або, в гарних руках, працює набагато краще. За останні роки, неможливо не помітити зріст популярності інтерпретованої мови про-

грамування JavaScript на світовому ринку. Ця мова, будучи створеною для вирішення питання маніпулювання елементами веб-сторінки, зараз використовується майже усюди, від десктопних додатків, як наприклад, під Windows 8,10 до смартфонів, що працюють на таких операційних системах, як Android чи iOS. Дана мова має таку популярність, оскільки вона досить легка в засвоєнні, хоч і має велику кількість «підводних каменів», вона інтерпретована, і набагато легша в плані написання коду, ніж C++ чи C, оскільки не потребує коду програміста для очищення вже використаних ресурсів процесора, оскільки очищенням займається збирач сміття (garbage collector). Попри усі ці переваги, дана мова має й свої недоліки. Дану мову не було спроектовано для вирішення більших задач, ніж ті, що зв'язані з маніпулюванням веб-сторінкою, проте розробники з кожним роком покращують її, додаючи нові можливості, а також виправляючи помилки, підтримуючи сумісність з попередніми версіями.

В даній роботі, розглянуто мову, котра називається TypeScript, що було створено в Microsoft під керівництвом Андерса Хейлсберга, який створив такі мови як C#, Delphi і т.д.. TypeScript вирішує основні проблеми JavaScript, як, наприклад, використання статичних типів, що значно спрощує дебагінг коду та його тестування. Спеціалісти з Google використовують TypeScript як основну мову свого нового фреймворка Angular2, що на момент написання статті знаходиться в «release candidate» версії.

Принцип роботи TypeScript

TypeScript має свій компілятор, що компілює код в чистий JavaScript, програміст також може писати й JavaScript код в одному файлі з TypeScript кодом, компілятор просто пропустить дані строки, компілюючи інші. Кожен файл TypeScript має розширення «.ts», компілятор TypeScript проходить по кожному такому файлу та компілює його в аналогічний файл з JavaScript кодом, в проекті достатньо ссилатися на JavaScript файли. Однак, TypeScript надає й можливість настроювання компілятора, використовуючи спеціальний файл «tsconfig.json». Даний файл є корінним в проекті для TypeScript. В ньому можна додати файли або папки з файлами для компіляції, ну і, аналогічно, виключити файли папки з процесу компіляції, змінити кодування для компільованого файла і т.д. (всі опції доступні на офіційному сайті мови TypeScript). Компілятор під час роботи при неправильному співставленні типів не буде зупиняти компіляцію, як, наприклад, в C++, Java, а лише видасть попередження про незпівставлення типів, оскільки це дозволено мовою JavaScript, де число може з легкістю бути приведено до строки та інше, використовуючи спеціальні методи для типів значення та об'єктів.

Порівняння TypeScript та JavaScript

В даному розділі наведено основні можливості мови TypeScript, що неможливо використати в JavaScript:

1. Використання статичних типів

На відміну від JavaScript, в TypeScript розробник повинен писати тип змінної разом з її об'явленням. TypeScript підтримує основні типи, що доступні і в JavaScript — number, string, boolean, null, object, undefined, symbol, promise та інші. Однак, в TypeScript розробник може створити ще й enum(перечислення), void, any(тип параметра не визначено під час компіляції), tuple (кортеж — щось на зразок масиву, що може мати значення різних типів, однак дані типи потрібно описати, тобто кортеж — не те саме, що й масив в JavaScript).

Приклад створення змінної в TypeScript:

```
let myVar: string = «Hello»;
```

2. TypeScript дозволяє створювати інтерфейси

Розробникам мов Java, C# та ін.. відомо, що таке інтерфейси — це контракт, що може приймати функція та реалізує певний клас, задля того, аби код не залежав від конкретної реалізації, інтерфейси використовуються в такому популярному патерні як «dependency injection». Проте, в TypeScript інтерфейси використовуються в основному задля передачі комплексних типів — об'єктів в певну функцію, при чому функція може повертати об'єкт, що реалізує даний інтерфейс(під реалізацією мається на увазі те, що в об'єкта є ті параметри, що описані в інтерфейсі, тобто їх може бути і більше, проте перевіряються лише необхідні), тобто інтерфейси використовуються задля перевірки типів, що передаються в функцію і використовуються компілятором. Також, основною особливістю інтерфейсів є те, що інтерфейс може мати необов'язкові параметри, що можуть бути проігноровані.

Приклад інтерфейсу в TypeScript:

```
interface Named {
    name: string;
    (a: string, b: string): string;
    surname?: string;
}
```

В даному інтерфейсі є поле name з типом строка, функція, що приймає два параметри типу строка, а також поле surname, що є необов'язковим.

3. TypeScript дозволяє створювати класи

Так, з виходом EcmaScript 6 в JavaScript також можливе використання класів, проте в TypeScript класи мають більше значення та мають більше функцій. По-перше, класи в TypeScript можуть реалізовувати інтерфейси, тобто, їх спокійно можна передавати в функції, як і об'єкти, що реалізують даний інтерфейс. По-друге, в TypeScript класи мають

модифікатори доступу — public, private, protected, чого не має JavaScript. По-третє, в TypeScript можна створити абстрактний клас. Отже, TypeScript на даний момент є більш пристосованою для створення ООП коду. Звичайно, в TypeScript, як і в JavaScript є наслідування, статичні методи, конструктори та властивості (getters, setters).

Найпростіший приклад класу:

```
class Person {
        private _name: string;
        constructor(name: string) {
            this._name = name;
    }
    public sayName(): string {
        return this._name;
    }
};
```

В даному класі є лише конструктор, приватне поле типу строка та публічний метод.

4. Оскільки TypeScript має статичну типізацію то в ньому доступне приведення типів, завдяки спеціальним конструкціям. В версії 1.5 розробникам рекомендується використовувати спеціальний оператор «as» [1], що і використовується для приведення одного типу до іншого.

5. В TypeScript можна створювати generic-типи та функції, — аналог шаблонів в C++. Звичайно, в простому JavaScript даний функціонал не потрібен, оскільки у функцію можна передати параметр будь-якого типу, проте супроводжувати такий код дуже важко. В TypeScript можна замість таких типів використовувати й спеціальний тип any [2], але він має зовсім інше значення. Як і в шаблонах C++, generic типи використовуються для створення класів, методів, що можуть приймати параметри різних типів. Даний функціонал є дуже важливим в будь-якій мові з статичною типізацією, приклад generic типу:

```
function genFunc<T>(arg: T): T {
return arg;
}
let output = genFunc<number>(5);
```

В даному прикладі було створено просту generic фунцію, що приймає параметр та повертає його назад. В змінну output буде записано число 5 і тип цієї змінної — також число. Однак, ми могли використовувати будь-який інший тип — інтерфейс Named, string чи клас Person.

6. Для функцій в TypeScript зарезервовано спеціальний тип, тобто функція в TypeScript — це не об'єкт, як в звичайному JavaScript, при чому ми не зможемо присвоїти параметру одного типу функції іншому, якщо в них не є однакова кількість параметрів та тип, а також тип значення, що повертається, проте є й виключення, що детальніше описані на офіційному сайті мови TypeScript.

7. TypeScript підтримує асинхронні методи, використовуючи зарезервовані слова async та await[1]. Для цього метод повинен повертати спеціальний тип — promise, що доступний в JavaScript[1]. Завдяки асинхронності, програмісту тепер не потрібно писати код, з багатьма зв'язаними методами then, success, error, тепер достатньо обрамити метод спеціальним словом async, а код, що викликає даний метод повинен лише використати спеціальне слово await, результатом буде повернені дані з promise. Дана можливість мови не є простою і не одразу зрозуміла і виходить за рамки цієї статті, тому, радимо прочитати про асинхронність і чим вона відрізняється від багатопоточності на спеціальних ресурсах, а про її реалізацію в мові — на сайті мови TypeScript.

8. Оскільки TypeScript має статичну типізацію, а більшість фреймворків написані на чистому JavaScript, то потрібно якимось чином переводити чистий JavaScript код в код TypeScript, або просто описувати основні типи, що використовуються тією чи іншою функцією в коді фреймворка. Звичайно, в Microsoft передбачили це, тому в TypeScript доступні спеціальні так звані «declaration» файли, тобто файли з об'явленням усіх типів та функцій, класів, що є в JavaScript файлах фреймворка. Розробник може сам писати такі файли, для необхідної бібліотеки, або використати вже написані, їх можна знайти на GitHub чи скачати, використовуючи Bower, Nuget та ін. Дані файли мають розширення «d.ts». Щоб зсилатися на код в іншому файлі TypeScript використовується синтаксис із трьома флешами[3], наприклад:

```
///<reference path=»»/>
```

Для параметра path потрібно задати лише шлях розміщення файла у файловій системі. Завдяки цьому компілятор зможе перевірити, що функція, в яку розробник передає параметри або використовує описана, або описані типи, що вона приймає, тобто таким чином забезпечується правильність написання коду при компіляції.

В даному розділі було описано лише ті особливості, що підтримує TypeScript, яких немає в JavaScript. Те, ж, що підтримується обома мовами не було описано, оскільки не має значення для порівняння цих мов. Звичайно, на момент написання статті, не усе, що доступно в JavaScript, доступно в TypeScript, детальніше можна прочитати на сайті мови. Однак, розробники TypeScript досить активно розширюють можливості своєї мови і додають навіть і деякі реалізації з наступних специфікацій JavaScript. Також в цьому розділі не було описано таку функціональність, як JSX, деякі особливості приведення типів, оператори, не розгор-

нуто поняття модулів та просторів імен, не розкрито значення функцій та їх можливостей, усі оновлення мови з версій 1.7 та 1.8, проте інформації, що описано в даній статті достатньо для розуміння того, для чого було створено дану мову і розуміння її недоліків та переваг в порівнянні з JavaScript.

Висновок

Було розглянуто основні особливості мови TypeScript. Наведено принцип роботи компілятора, а також наведено спосіб конфігурування параметрів компілятора. Порівняно TypeScript з JavaScript, показані основні можливості, що надає мова для програміста, такі як підтримка асинхронного програмування, статична типізація, створення класів, інтерфейсів, вдосконалена типізація, тобто розробниками мови надано більше типів і т.д.

В результаті, можна з впевненістю сказати, що, оскільки мова JavaScript є однією з найпопулярніших мов в наш час, що використовується майже усюди, то використання TypeScript значно спростить розробку додатків, підвищуючи продуктивність програміста, зменшить кількість часу на розробку та дебагінг коду, покращить тестування коду та його структуризацію. А оскільки дана мова розвивається завдяки співпраці Microsoft та Google, а також спільноти, то розробники будуть і надалі радувати нас гарними нововведеннями.

Література

1. Документація мови TypeScript — Режим доступу: https://www.typescriptlang.org/docs/tutorial.html

2. TypeScript. The Definitive Guide — Режим доступу: https://basarat.gitbooks.io/typescript/content/docs/getting-started.html

3. Dan Maharry. TypeScript Revealed / Maharry D. — Apress, 2013. — 81 c.