# От JavaScript к TypeScript

JavaScript сегодня на столько популярен, что Вы можете найти проекты, которые содержат сотни тысяч строк кода. Вы можете использовать JavaScript, чтобы создавать web сайты, Windows приложения и серверные компоненты. Но, в случае действительно сложных проектов у разработчиков появляется проблема с поддержкой кода, ведь в сравнении с C#, C++ и Java, языку JavaScript не хватает статических типов, классов, интерфейсов и других атрибутов современных ООП языков. В результате, классические средства разработки лишены многих полезных возможностей, связанных с IntelliSense и рефакторингом, а любые ошибки становится найти не очень просто.

Именно поэтому многие разработчики предпочитают настройки над JavaScript, такие как TypeScript, CoffeScript или системы на подобии Dart.

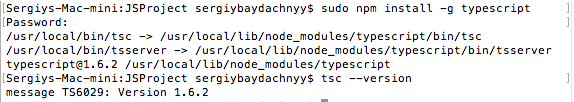
В этом разделе мы посмотрим, как использовать TypeScript, так как этот язык достаточно популярен сегодня и имеет «родную» поддержку в Code, а также в Visual Studio, WebStorm и других редакторах.

Как я уже говорил, TypeScript представляет собой надстройку над JavaScript, добавляя туда классы, интерфейсы, обобщенные типы и позволяет комбинировать TypeScript и JavaScript код. Благодаря компилятору TypeScript Вы можете достаточно просто преобразовать TypeScript код в JavaScript, соответственно Вы можете использовать TypeScript везде, где может быть использован JavaScript.

Чтобы приступить к работе с TypeScript Вы можете посетить <http://www.typescriptlang.org> сайт и найти там множество примеров и сам компилятор.

Чтобы установить TypeScript компилятор на Ваш Mac, проще всего использовать node.js менеджер пакетов:

**sudo npm install –g typescript**



Как только компилятор установлен, Вы можете вызвать его из командной строки, используя tsc утилиту:

**tsc --version**

Если все прошло хорошо, то Вы можете открыть Visual Studio Code и приступить к работе с TypeScript.

Обратите внимание, что для инсталляции TypeScript мы использовали node.js менеджер пакетов. Если node.js у Вас не установлен, обратитесь на сайт nodejs.org и выполните установку.

Если рассматривать редактор Visual Studio Code применительно к TypeScript, то тут Вы можете найти все те же возможности, что и для JavaScript. Но для конфигурации TypeScript проектов используется tsconfig.json, который, впрочем, имеет схожий формат с jsconfig.json. Базовая конфигурация может выглядеть следующим образом:

{

"compilerOptions": {

"target": "ES5",

"module": "amd",

"sourceMap": true

}

}

Этого достаточно, чтобы приступить к написанию кода и использовать все возможности редактора. Но, чтобы выполнить компиляцию созданного TypeScript кода в JavaScript нам понадобиться еще настроить задачу. О задачах мы говорили ранее. В нашем случае нужно создать файл tasks.json, где разместить следующий код:

{

"version": "0.1.0",

"command": "tsc",

"isShellCommand": true,

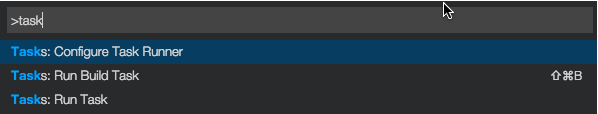
"showOutput": "silent",

"args": ["index.ts"],

"problemMatcher": "$tsc"

}

В последней версии VS Code tasks.json должен быть размещен в каталоге .vscode, поэтому, чтобы избежать ошибок, лучше всего создать этот файл используя Command Palette и выполнить там команду Configure Task Runner:



Чтобы протестировать нашу задачу, попробуем создать немного кода. Для этого можно открыть руководство на сайте http://www.typescriptlang.org и скопировать код оттуда:

class Student {

fullname : string;

constructor(public firstname, public middleinitial, public lastname) {

this.fullname = firstname + " " + middleinitial + " " + lastname;

}

}

interface Person {

firstname: string;

lastname: string;

}

function greeter(person : Person) {

return "Hello, " + person.firstname + " " + person.lastname;

}

var user = new Student("Jane", "M.", "User");

document.body.innerHTML = greeter(user);

Как только код готов, нажмите ⇧⌘B или используйте Command Palette для запуска задачи. Если все прошло хорошо, то Вы сможете найти увидеть новый JavaScript файл с результирующим кодом. Проверить его работу можно создав простой HTML файл и открыв его в браузере:

<!DOCTYPE html>

<html>

<head><title>TypeScript Greeter</title></head>

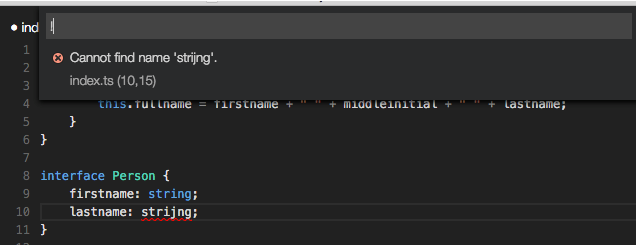
<body>

<script src="index.js"></script>

</body>

</html>

Поговорим немного об IntelliSense системе. Попробуйте намеренно совершить несколько ошибок и посмотреть на реакцию редактора:

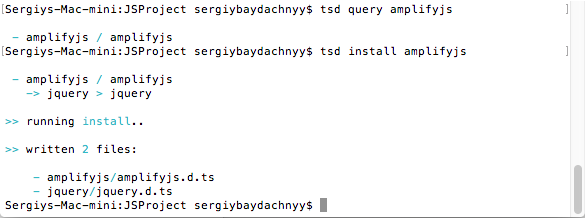


Как мы видим, IntelliSense система замечательно работает для базовых типов, наших конструкций и стандартных синтаксических элементов. Но современное программирование на JavaScript предполагает использование большого количества сторонних библиотек. Естественно, мы можем использовать все эти библиотеки и в TypeScript, но как быть с IntelliSense системой и заставить ее работать с максимальной отдачей? На помощь приходят TypeScript definition files (файлы определения типов), которые позволяют описать типы, которые входят в сторонний Фреймворк или библиотеку. Описание служит для того, чтобы предоставить максимум информации редактору и использовать все возможности системы даже при работе с файлами на чистом JavaScript. Остается последний вопрос, где взять эти файлы определения типов? К счастью, для более чем 90% доступных библиотек эти файлы уже созданы и поддерживаются сообществом. Чтобы воспользоваться ими Вам нужно установить TypeScript Definition Manager, используя следующую команду:

sudo npm install -g tsd

Как только менеджер установлен, вы можете использовать tsd утилиту в контексте рабочей директории для поиска и загрузки пакетов. Наиболее популярными командами являются tsd install и tsd query:

tsd install amplifyjs



Как только .d.ts файлы с определения попадают в каталог проекта, Visual Studio Code автоматически вычитывает из них информацию и подключает к системе.