# 1 Общая информация

**В этой главе**

- ✓ Какие возможности предоставляет TypeScript разработчикам.
- ✓ Когда стоит использовать TypeScript в проекте.
- ✓ Ограничения TypeScript.
- ✓ Как устроена эта книга.
- ✓ Сообщения об ошибках в книге.
- ✓ Связь с автором.

TypeScript — это надстройка над JavaScript, ориентированная на создание безопасного и предсказуемого кода, который способен запускаться в любой среде выполнения JavaScript. Главная особенность TypeScript — статическая типизация — делает работу с JavaScript более предсказуемой для программистов, знакомых с такими языками, как C# и Java. В книге я объясняю, что делает TypeScript и какие возможности он предоставляет.

## 1.1. СТОИТ ЛИ ИСПОЛЬЗОВАТЬ TYPESCRIPT

TypeScript — это не решение всех проблем, и важно понимать, когда уместно использовать TypeScript, а когда он просто будет мешать. В последующих разделах я описываю главные возможности TypeScript и ситуации, в которых они могут быть полезны.

### 1.1.1. Чем полезен TypeScript для повышения производительности разработчиков

Основные возможности TypeScript направлены на повышение продуктивности разработчиков, в частности, за счет использования статических типов, которые облегчают работу с системой типов JavaScript. Другие возможности повышения производительности, такие как ключевые слова управления доступом и лаконичный синтаксис конструкторов классов, помогают предотвратить распространенные ошибки при кодировании.

Средства повышения производительности TypeScript применяются к коду JavaScript. Пакет TypeScript включает в себя компилятор, обрабатывающий файлы TypeScript и генерирующий чистый JavaScript, который можно запускать в среде выполнения JavaScript, например в Node.js или в браузере (рис. 1.1).


**Рис. 1.1.** Преобразование TypeScript в JavaScript-код

Благодаря сочетанию возможностей JavaScript и TypeScript язык JavaScript сохраняет гибкость и динамичность, ограничивая при этом использование типов данных, что делает их более привычными и предсказуемыми для большинства разработчиков. Это также означает, что в проектах на TypeScript все еще могут применяться сторонние пакеты JavaScript, включая поддержку использования TypeScript в готовых фреймворках для разработки приложений, описанных в части III.

Возможности TypeScript можно применять выборочно, то есть допускается использовать только те, которые подходят для конкретного проекта. Если вы новичок, то, скорее всего, захотите использовать все и сразу. Однако с опытом вы научитесь работать с TypeScript более осознанно, применяя его возможности только к тем частям кода, которые особенно сложны или которые, по вашему мнению, могут вызвать проблемы.

Некоторые возможности TypeScript полностью реализуются компилятором и не оставляют следов в исполняемом JavaScript-коде. Другие возможности реализованы за счет стандартного JavaScript и выполнения дополнительных проверок при компиляции. Это означает, что для достижения наилучших результатов часто необходимо вникать в работу той или иной TypeScript-функции, из-за чего они могут казаться сложными и нелогичными.

В более широком смысле TypeScript дополняет JavaScript, но не заменяет его, а разработка проекта на TypeScript в основном представляет собой процесс написания кода на JavaScript. Некоторые программисты считают, что, перейдя на

TypeScript, они смогут писать веб-приложения, не вникая, как работает JavaScript. Они видят, что TypeScript выпускает компания Microsoft, и полагают, что это C# или Java, но только для веб-разработки, однако подобное предположение приводит к путанице и разочарованию.

Чтобы эффективно работать с TypeScript, необходимо не только хорошо владеть JavaScript, но и понимать причины того или иного его поведения. В главах 3 и 4 описаны особенности JavaScript, которые необходимо знать, чтобы извлечь максимальную пользу из TypeScript и обеспечить себе прочный фундамент для понимания того, почему TypeScript является таким мощным инструментом.

Если вы готовы разбираться в системе типов JavaScript, то работа с TypeScript вам понравится. Но если вы не готовы потратить время на освоение JavaScript, то и не следует использовать TypeScript. Добавление TypeScript в проект, когда у вас нет знаний JavaScript, усложняет разработку, поскольку в таком случае придется работать с двумя наборами языковых функций, ни одна из которых не будет вести себя так, как вы ожидаете.

### 1.1.2. Особенности версий JavaScript

JavaScript развивался довольно хаотично, однако в последнее время наблюдается целенаправленная работа по стандартизации и совершенствованию этого языка, что привело к появлению новых возможностей, упрощающих его использование. Но и здесь не обошлось без проблем: до сих пор многие среды выполнения JavaScript не поддерживают эти современные возможности, особенно это касается старых браузеров. Из-за этого разработчикам на JavaScript приходится ограничиваться небольшим набором доступных функций языка, которые поддерживаются повсеместно. Сам по себе JavaScript непрост в освоении, а невозможность использовать функции, облегчающие разработку, делает его еще сложнее.

Компилятор TypeScript способен преобразовывать JavaScript-код, написанный с применением новейших функций, в код, соответствующий более старым версиям языка JavaScript. Это позволяет программистам использовать современные возможности JavaScript при работе с TypeScript, в то время как предыдущие версии JavaScript могут выполнять код, написанный в рамках конкретного проекта.

Компилятор TypeScript хорошо справляется с большинством языковых особенностей, однако некоторые из них не всегда возможно эффективно транслировать под устаревшие версии. Если вы работаете с самыми ранними вариациями JavaScript, то обнаружите, что не каждую его современную возможность получится использовать в процессе разработки, поскольку у компилятора TypeScript попросту нет инструментов для их поддержки в устаревшем JavaScript.

При этом необходимость генерировать устаревший JavaScript-код важна не во всех проектах, поскольку компилятор TypeScript — лишь одна из частей общей цепочки инструментов. Он отвечает за внедрение возможностей TypeScript,

но результатом является современный JavaScript-код, который затем обрабатывается другими инструментами. Такой подход широко используется при разработке веб-приложений, и его примеры вы увидите в части III.

## 1.2. ЧТО НУЖНО ЗНАТЬ

Если вы пришли к выводу, что TypeScript подходит для вашего проекта, то важно иметь как минимум базовые знания в области разработки на JavaScript. В главах 3 и 4 я даю начальные сведения о возможностях JavaScript, которые будут полезны для понимания TypeScript, однако учтите, что данное издание не является полноценным учебником по JavaScript. В части III я демонстрирую, как можно использовать TypeScript с популярными фреймворками для разработки веб-приложений, но для этих примеров требуется знание HTML и CSS.

## 1.3. КАК НАСТРОИТЬ СРЕДУ РАЗРАБОТКИ

О том, как настроить среду разработки, написано в главе 2, где вы создадите свое первое приложение на TypeScript. В последующих главах могут потребоваться дополнительные пакеты. Все необходимые инструкции по их установке и использованию также будут даны.

## 1.4. КАКОВА СТРУКТУРА ЭТОЙ КНИГИ

Книга состоит из трех частей, каждая из которых охватывает ряд смежных тем.

Часть I «Начало работы с TypeScript». Здесь содержится информация, необходимая для начала разработки на TypeScript. Она включает в себя краткое описание того, как создавать приложения на TypeScript, а также вводную главу о важных функциях JavaScript. В главах 5 и 6 представлены средства разработки на языке TypeScript.

Часть II «Возможности TypeScript». Во второй части рассматриваются возможности языка TypeScript, которые помогают повысить продуктивность разработчиков, в том числе статические типы. TypeScript предоставляет множество различных возможностей для работы с типами, которые я подробно описываю и демонстрирую на примерах.

Часть III «Создание веб-приложений на основе TypeScript». TypeScript не используется сам по себе, поэтому часть III посвящена тому, как использовать TypeScript для создания веб-приложений с помощью наиболее популярных фреймворков. Здесь рассказывается о возможностях TypeScript, полезных для каждого фреймворка, и демонстрируются способы решения задач, обычно необходимых при разработке веб-приложений. Для того чтобы продемонстрировать способности фреймворков, я также покажу, как создать автономное веб-приложение, не зависящее от фреймворков.

## 1.5. МНОГО ЛИ ПРИМЕРОВ?

Примеров очень много. Лучший способ изучения TypeScript — через примеры, и в эту книгу я включил их как можно больше. Чтобы максимально увеличить количество примеров, я принял простое правило, позволяющее избегать повторного перечисления одного и того же кода или содержимого. Когда я создаю файл, я показываю его полное содержимое, как это сделано в листинге 1.1. В заголовке листинга я указываю имя файла и его папки, а внесенные мной изменения выделяю жирным шрифтом.

**Листинг 1.1.** Выражение неизвестного значения в файле index.ts из папки src

```
function calculateTax(amount: number, format: boolean): string | number {
    const calcAmount = amount * 1.2;
    return format ? '$${calcAmount.toFixed(2)}' : calcAmount;
}

let taxValue = calculateTax(100, false);

switch (typeof taxValue) {
    case "number":
        console.log('Number Value: ${taxValue.toFixed(2)}');
        break;
    case "string":
        console.log('String Value: ${taxValue.charAt(0)}');
        break;
    default:
        let value: never = taxValue;
        console.log('Unexpected type for value: ${value}');
    }

let newResult: unknown = calculateTax(200, false);
let myNumber: number = newResult as number;
console.log('Number value: ${myNumber.toFixed(2)}');
```

Это листинг из главы 7, в котором показано содержимое файла `index.ts`, расположенного в папке src. Не обращайте внимания на содержание листинга и назначение файла, просто имейте в виду, что в этом типе листинга приводится полное содержимое файла, а изменения, которые вам необходимо внести самостоятельно, чтобы следовать примеру, выделены жирным шрифтом. Некоторые файлы с кодом могут быть довольно объемными, а описываемая мною функциональность требует лишь небольших изменений. Вместо включения всего файла я использую многоточие, показывая только нужную для объяснения часть файла (листинг 1.2).

**Листинг 1.2.** Настройка инструментов в файле package.json из папки reactapp

```
...
"scripts": {
  "json": "json-server data.js -p 4600",
  "serve": "react-scripts start",
  "start": "npm-run-all -p serve json",
  "build": "react-scripts build",
```

```
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
...
```

Это листинг из части III, который показывает набор изменений, примененных к одной части большого файла. Когда вы увидите такой неполный листинг, знайте, что изменились только строки, выделенные жирным шрифтом, а остальная часть файла не меняется.

Иногда от вас потребуется внести изменения в разных частях файла, а это затрудняет его представление в виде частичного листинга. В такой ситуации я опускаю часть содержимого файла, как показано в листинге 1.3.

**Листинг 1.3.** Применение декоратора в файле abstractDataSource.ts из папки src

```
import { Product, Order } from "./entities";
import { minimumValue } from "../decorators";

export type ProductProp = keyof Product;

export abstract class AbstractDataSource {
    private _products: Product[];
    private _categories: Set<string>;
    public order: Order;
    public loading: Promise<void>;

    constructor() {
        this._products = [];
        this._categories = new Set<string>();
        this.order = new Order();
        this.loading = this.getData();
    }

    @minimumValue("price", 30)
    async getProducts(sortProp: ProductProp = "id",
            category? : string): Promise<Product[]> {
        await this.loading;
        return this.selectProducts(this._products, sortProp, category);
    }

    // ...другие методы для краткости опущены...

}
```

В этом листинге изменения по-прежнему выделены жирным шрифтом, а те части файла, которые в листинге опущены, в примере не затрагиваются.

## 1.6. ГДЕ МОЖНО ПОЛУЧИТЬ КОД ПРИМЕРА

Примеры проектов для всех глав книги вы можете бесплатно скачать с сайта https://github.com/manningbooks/essential-typescript-5. Здесь содержится все необходимое для выполнения примеров, и вам не придется набирать код вручную.

## 1.7. ЧТО ДЕЛАТЬ, ЕСЛИ У ВАС ВОЗНИКЛИ ПРОБЛЕМЫ С ВЫПОЛНЕНИЕМ ПРИМЕРОВ

Во-первых, следует вернуться к началу главы и сделать все заново. Большинство проблем возникает из-за пропуска какого-либо шага или неполного применения изменений, показанных в листинге. Обратите внимание на выделения в листингах, которые указывают на необходимые изменения.

Затем следует проверить список ошибок и исправлений, расположенный в репозитории книги на GitHub. Технические книги сложны, и, несмотря на все усилия автора и редакторов, ошибки неизбежны. Проверьте список исправлений, чтобы получить перечень известных ошибок и инструкции по их устранению.

Если у вас все равно возникают проблемы, то скачайте проект для главы, которую читаете, из репозитория (https://github.com/manningbooks/essential-typescript-5) и сравните его со своим проектом. Я писал код для репозитория GitHub, прорабатывая каждую главу, поэтому в вашем проекте должны быть те же файлы с тем же содержимым.

Если вам все же не удается заставить примеры работать, то вы можете обратиться ко мне за помощью, написав на почту adam@adam-freeman.com. Пожалуйста, укажите в письме, какую книгу вы читаете и в какой главе или примере возникла проблема. Всегда полезно указать номер страницы или листинга. Учитывайте также, что я получаю много писем и могу ответить не сразу.

### 1.7.1. Что делать, если вы обнаружили ошибку в книге

Вы можете сообщать мне об ошибках по электронной почте adam@adam-freeman.com, но перед тем, как это сделать, пожалуйста, проверьте список ошибок и исправлений к данной книге, который найдете в репозитории GitHub по адресу https://github.com/manningbooks/essential-typescript-5, — возможно, об этой ошибке уже известно.

Ошибки, которые могут привести читателей в замешательство, особенно проблемы с кодом примеров, я добавляю в файл с исправлениями в репозитории GitHub с благодарностью первому читателю, сообщившему о них. Я также публикую список менее серьезных проблем, под которыми обычно подразумеваются опечатки в тексте, сопровождающем примеры, и которые вряд ли вызовут путаницу.

## 1.8. КАК СВЯЗАТЬСЯ С АВТОРОМ

Вы можете связаться со мной по адресу adam@adam-freeman.com. Уже несколько лет я указываю этот адрес электронной почты в своих книгах. Изначально я сомневался в правильности такого решения, но в итоге рад, что сделал это. Я получаю письма от читателей со всего мира. Они работают или учатся в различных областях, и большинство писем очень позитивные, вежливые и приятные.

Я пытаюсь отвечать на письма оперативно, но их приходит так много, что я просто не успеваю, особенно когда погружен в работу над книгой. Я стараюсь

помочь читателям, застрявшим на примерах из книги, но прошу вас: прежде чем обратиться ко мне, выполните действия, описанные выше.

Мне приятно получать письма от читателей, но есть ряд вопросов, на которые я сразу отвечаю «нет». Боюсь, что не смогу написать код для вашего стартапа, помочь вам с заданием в университете, принять участие в споре о дизайне вашей команды разработчиков или научить вас программированию.

## 1.9. ЕСЛИ ВАМ ПОНРАВИЛАСЬ ЭТА КНИГА

Пожалуйста, напишите мне по адресу adam@adam-freeman.com и сообщите об этом. Мне всегда приятно читать отзывы довольных читателей, и я ценю время, которое уходит на отправку этих писем. Писать книги — нелегкий труд, и ваши сообщения стимулируют и поддерживают меня.

## 1.10. ЕСЛИ ВАМ НЕ ПОНРАВИЛАСЬ ЭТА КНИГА

Вы по-прежнему можете написать мне по адресу adam@adam-freeman.com, и я постараюсь вам помочь. Имейте в виду, что я смогу помочь только в том случае, если вы опишете проблему и скажете, что именно вы хотели бы, чтобы я с ней сделал. Поймите, иногда единственный выход — признать, что я просто не ваш автор. Я обязательно учту все замечания, но после 25 лет работы я пришел к выводу, что не всем нравится мое творчество.

## РЕЗЮМЕ

В этой главе я объяснил, в каких случаях следует использовать TypeScript для своих проектов. Я также описал содержание и структуру книги, указал, где взять исходный код, и рассказал о том, как связаться со мной, если у вас возникнут проблемы с примерами.

- TypeScript — надстройка над JavaScript, которая для эффективного использования предполагает знание последнего.
- TypeScript не является подмножеством C#, несмотря на схожий стиль кода.
- Основной особенностью TypeScript является добавление статических типов в JavaScript.
- Компилятор TypeScript может адаптироваться к определенным версиям JavaScript, что позволяет использовать современные возможности языка в приложениях, работающих на старых платформах.

В следующей главе я расскажу о системе типов JavaScript, лежащей в основе возможностей TypeScript.