

TYPESCRIPT



План лекции

Основные моменты:



1. TypeScript
2. Setting up TS
3. Scripts
4. TSC и TSX
5. Union Type
6. Interface
7. Enum
8. Functions
9. Tuple






TypeScript




TypeScript (TS) — это надстройка над JavaScript, которая добавляет статическую типизацию. Она позволяет разработчикам указывать типы переменных, функций и объектов, что помогает:

- **находить ошибки на этапе компиляции (до запуска кода),**
 - **делать код более читаемым и поддерживаемым,**
 - **улучшать автодополнение в редакторах.**
- 


TypeScript компилируется в обычный JavaScript, который уже и работает в любом браузере.

Создан Microsoft. Широко используется в крупных проектах и фреймворках как в бэкенде, так и во фронтенде (например, Angular, React, Vue).





Как настроить ts вручную и создать базовый проект
(для теоретических целей)

1. **Создайте папку** под новый проект: например, **alice-app**
 2. **Откройте ее** в отдельном окне в VSCode (верхнее меню → open folder → выбираете alice-app)
 3. После открытия - папка должна показываться слева сверху **крупными буквами**, это значит, что папка стала текущей.
-
- 

3. Откройте терминал в этой папке и введите команду:

npm init -y

Это команда объявляет текущую папку проектом и создает package.json



3. Установи typescript как зависимость разработки

npm install typescript --save-dev

Проверить установился ли он можно командой:

npx tsc --version



4. Запусти команду **npx tsc --init**
Она создаст файл tsconfig.json

5. Добавь в него следующий код:

```
{  
  "include": ["src"],  
  "compilerOptions": {  
    "target": "es2016",  
    "module": "commonjs",  
    "skipLibCheck": true,  
    "outDir": "./build"  
  },  
  "exclude": ["node_modules"]  
}
```



6. Создай папку `src` и файл `index.ts`:

Можешь сделать это при помощи команд:

`mkdir src`

`touch src/index.ts`



7. Добавь код в index.ts

```
function greet(name: string): string {  
  return `Hello, ${name}!`;  
}
```

```
console.log(greet("TypeScript"));
```



8. Выполни компиляцию кода

`npx tsc`

Компиляция TypeScript (TS) — это процесс преобразования TypeScript-кода в JavaScript, который может выполняться в браузере или среде вроде Node.js.

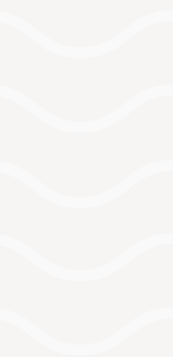
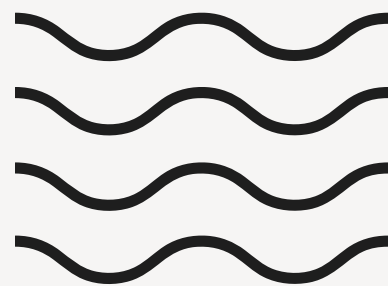
TypeScript не понимается напрямую движками JS, поэтому перед запуском его нужно **скомпилировать (transpile)** в стандартный JavaScript.



Обрати внимание на то, что у тебя появилась папка **build** с сгенерированными js файлами.

Их можно запустить

Поздравляю, мы научились преобразовывать ts в js!

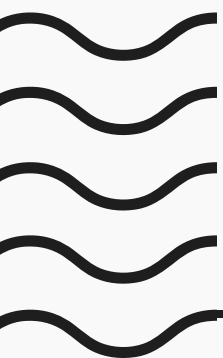


Скрипты



Для упрощения работы можно заменить в **package.json** скрипты:

```
"scripts": {  
  "build": "tsc",  
  "start": "node build/index.js",  
  "dev": "tsc --watch"  
}
```



Скрипты



Вызывать их можно вводя в терминале соответствующие команды:

npm start запустит получившийся код

npm run build создает билд папку с готовым js кодом

npm run dev будет преобразовывать код в js при любом изменении





Скрипты

Чтобы остановить выполнение программы введите
Ctrl + C
(контрол с)

P.S. Сам плюс вводить не нужно :)





Если ты хочешь запускать отдельные файлы ts через расширение CodeRunner напрямую, выполни единократно команду:

`npm i -g tsx`

Она установит на твой компьютер глобальную зависимость

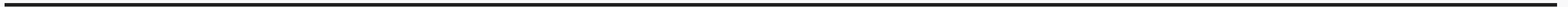


`npm i -g tsx`

делает следующее:



- `npm` — Node.js пакетный менеджер.
- `i` (или `install`) — команда установки.
- `-g` — означает глобальную установку, т.е. пакет будет доступен во всей системе, а не только в конкретном проекте.
- `tsx` — это название пакета, который ты устанавливаешь.



Что такое tsx?

tsx — это инструмент для запуска TypeScript (и TSX) файлов напрямую, без предварительной компиляции.

- Позволяет запускать .ts и .tsx файлы так же просто, как .js

Зачем это нужно?

Обычно TypeScript нужно сначала скомпилировать с помощью tsc, а потом запускать. tsx делает это на лету, экономя время и упрощая разработку.





Типизация

- имплицитная (неявная)
- ЭКСПЛИЦИТНАЯ





Примитивы (string, number, boolean)

```
const aliceName: string = "Alice";  
const aliceAge: number = 10;  
const isCurious: boolean = true;
```





Массивы

```
const teaGuests: string[] = ["Mad Hatter", "March Hare",  
"Dormouse"];  
const rabbitHolesVisited: number[] = [1, 3, 5];
```





Кортежи (tuple)

```
const doorSize: [string, number] = ["small", 15]; // [размер, см]
```



Enum

```
enum WonderlandLocation {  
  Garden = "garden",  
  TeaParty = "tea party",  
  Courtroom = "courtroom",  
  RabbitHole = "rabbithole"  
}
```



```
console.log(WonderlandLocation.Garden); // garden
```

```
const currentLocation: WonderlandLocation =  
  WonderlandLocation.TeaParty;
```



Объекты и интерфейсы (interface)

```
interface Character {  
  name: string;  
  species: "human" | "animal" | "card";  
  isFriendly: boolean;  
}
```

ТИП



```
const cheshireCat: Character = {  
  name: "Cheshire Cat",  
  species: "animal",  
  isFriendly: false,  
};
```

значение



Union

```
type Size = "tiny" | "huge";  
// Размер может быть "tiny" или "huge"
```

```
let cakeSize: Size = "tiny";
```



Union

```
type CardSuit = "hearts" | "spades" | "clubs" | "diamonds";
```

```
interface CardSoldier {  
  suit: CardSuit;  
  number: number;  
}
```



Functions

```
function drinkPotion(size: "big" | "small"): void {  
  console.log(`Alice drinks a ${size} potion`);  
}
```

```
function grow(height: number): string {  
  return `Alice grows to ${height}cm`;  
}
```



Arrow Functions

```
const describeDrink = (label: string): string => {  
  return `The bottle says: "${label}"`;   
};
```



Generics

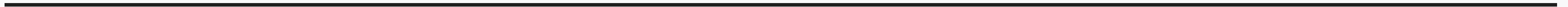
```
function print<T>(arg: T): void {  
  console.log(arg);  
}
```

```
print<string>("asd");
```



Официальная документация TS

<https://www.typescriptlang.org/>



Если вы нашли опечатку или ошибку в презентации сообщите
Alisher Khamidov

