**Продвинутые возможности TypeScript: Декораторы и модули**

TypeScript предоставляет мощные возможности для работы с типами и разработкой приложений, включая поддержку **декораторов** и **модулей**. Эти два концепта позволяют значительно расширить функциональность и улучшить структуру приложений, особенно в больших проектах. Декораторы позволяют изменять поведение классов и их членов, а модули обеспечивают удобную организацию кода и его повторное использование. В этой лекции мы рассмотрим, что такое декораторы и модули в TypeScript, а также как их использовать для создания более гибких и удобных приложений.

**Декораторы в TypeScript**

Декораторы в TypeScript — это специальные функции, которые позволяют добавлять метаданные или изменять поведение классов, их свойств, методов или параметров. Декораторы являются частью метапрограммирования и предоставляют мощный инструмент для разработки, который активно используется, например, в фреймворках, таких как Angular.

Декоратор — это функция, которая будет вызвана с дополнительными параметрами в зависимости от того, на что он применяется. Например, декораторы могут изменять поведение методов, свойств класса или даже самих классов. В TypeScript декораторы используются с помощью аннотации @, и чтобы использовать их, необходимо включить соответствующую опцию в настройках компилятора.

**Включение декораторов в TypeScript**

Для того чтобы работать с декораторами, необходимо в файле tsconfig.json включить опцию experimentalDecorators. Эта опция разрешает использование экспериментальных функций языка TypeScript, таких как декораторы.



После того как опция включена, можно использовать декораторы в своем коде.

**Типы декораторов**

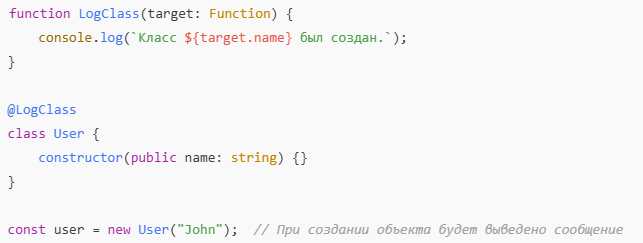
TypeScript поддерживает несколько типов декораторов:

1. **Декоратор класса** — применяется к самому классу.
2. **Декоратор метода** — применяется к методам класса.
3. **Декоратор свойства** — применяется к свойствам класса.
4. **Декоратор параметра** — применяется к параметрам метода.

Каждый тип декоратора принимает разные параметры и имеет различное поведение.

**Декоратор класса**

Декоратор класса используется для изменения поведения самого класса. Например, можно добавлять дополнительные методы или изменять уже существующие.



В этом примере декоратор LogClass выводит сообщение в консоль при создании класса User. Декоратор класса принимает конструктор класса в качестве параметра и может изменять его или добавлять дополнительное поведение.

**Декоратор метода**

Декоратор метода позволяет изменять поведение метода класса. Он получает два параметра: целевой объект и имя метода. Это может быть полезно для логирования, измерения времени выполнения метода или изменения логики работы метода.

function LogMethod(

  target: any,

  key: string,

  descriptor: PropertyDescriptor,

) {

  const originalMethod = descriptor.value;

  descriptor.value = function (...args: any[]) {

    console.log(

      `Метод ${key} был вызван с аргументами: ${args}`,

    );

    return originalMethod.apply(this, args);

  };

  return descriptor;

}

class User {

  constructor(public name: string) {}

  @LogMethod

  greet(greeting: string) {

    return `${greeting}, ${this.name}!`;

  }

}

const user = new User('John');

console.log(user.greet('Hello'));

**Модули в TypeScript**

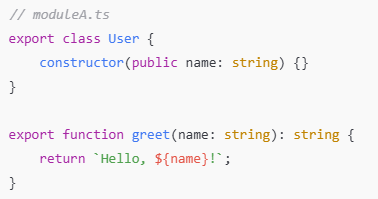
Модули в TypeScript — это способ организации кода, который позволяет разделить приложение на небольшие, легко управляемые и повторно используемые части. Модули в TypeScript соответствуют концепции модулей в ECMAScript 6 (ES6), что позволяет разрабатывать более структурированные и поддерживаемые приложения.

Модуль — это любой файл, который использует ключевые слова import и export. Это позволяет импортировать функции, классы или переменные из других модулей и экспортировать их для использования в других частях приложения.

**Экспорт и импорт**

В TypeScript для работы с модулями используется два ключевых слова: export и import.

1. **Экспорт** позволяет делать доступными для других файлов (модулей) функции, переменные, классы и интерфейсы.
2. **Импорт** позволяет использовать элементы, экспортированные из других модулей.





В следующем примере мы используем как декоратор, так и модули для создания простого приложения, где класс экспортируется из одного модуля, а декоратор используется для логирования вызовов метода.

// loggerDecorator.ts

export function LogMethod(

  target: any,

  key: string,

  descriptor: PropertyDescriptor,

) {

  const originalMethod = descriptor.value;

  descriptor.value = function (...args: any[]) {

    console.log(

      `Метод ${key} был вызван с аргументами: ${args}`,

    );

    return originalMethod.apply(this, args);

  };

  return descriptor;

}

// user.ts

import { LogMethod } from './loggerDecorator';

export class User {

  constructor(public name: string) {}

  @LogMethod

  greet(greeting: string) {

    return `${greeting}, ${this.name}!`;

  }

}

// app.ts

import { User } from './user';

const user = new User('John');

console.log(user.greet('Hello'));