Лекция 8
Typescript

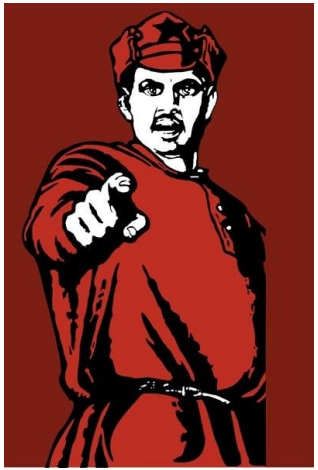
Мартин Комитски

План на сегодня

- Введение
- Настройка окружения, Hello World
- Основные принципы и возможности TS
- Практика

Минутка бюрократии

- Внимание
- Отметки о посещении занятий
- Обратная связь о лекциях



Введение

Typescript. Введение

Typescript is:

- A JavaScript that scales.
- A typed superset of JavaScript that compiles to plain JavaScript.

https://www.typescriptlang.org

Typescript. Зачем он нужен? Что дает?

- Дает возможность привести в порядок код больших приложений
- Привносит новые концепции классических ЯП
- Дополнительные возможности в IDE
- Дополнительные возможности из ECMAScript

Typescript. Требования

- node
- npm
- tsc
- Продвинутая IDE или текстовый редактор с расширенной поддержкой TS, например, VS Code

Типы данных

Typescript. Типы данных

```
1.  // Boolean
2.  let isDone: boolean = false;
3.
4.  // Number
5.  let decimal: number = 6;
6.  let hex: number = 0xf00d;
7.  let binary: number = 0b1010;
8.  let octal: number = 0o744;
9.
10. // String
11. let color: string = "blue";
12. color = 'red';
13.
14. // Array
15. let list: number[] = [1, 2, 3];
16. let list: Array<number> = [1, 2, 3];
17.
18. // Tuple
19. let x: [string, number];
20. x = ["hello", 10]; // OK. How получить error?
21.
```

Typescript. Типы данных

```
1.  // Enum
2.  enum Color {Red, Green, Blue}
3.  let c: Color = Color.Green;
4.
5.  // Any
6.  let notSure: any = 4;
7.  notSure = "maybe a string instead";
8.  notSure = false; // okay, definitely a boolean
9.
10. // Unknown aka type-safe any
11. let notSure: unknown = 5;
12. let sure: number = notSure; // Error
13.
14. // Void
15. function warnUser(): void {
16.     console.log("This is my warning message");
17. }
18.
19. // Null, Undefined
20. // Not much else we can assign to these variables!
21. let u: undefined = undefined;
22. let n: null = null;
23.
```

Typescript. Типы данных

```
1.  // Never
2.  // Function returning never must have unreachable end point
3.  function error(message: string): never {
4.      throw new Error(message);
5.  }
6.
7.  // Inferred return type is never
8.  function fail() {
9.      return error("Something failed");
10. }
11.
12. // Function returning never must have unreachable end point
13. function infiniteLoop(): never {
14.     while (true) {
15.     }
16. }
17.
```

Typescript. Типы данных

```
1.  // Object
2.  declare function create(o: object | null): void;
3.
4.  create({ prop: 0 }); // OK
5.  create(null); // OK
6.
7.  create(42); // Error
8.  create("string"); // Error
9.  create(false); // Error
10. create(undefined); // Error
11.
12.
```

Typescript. Типы данных. Type assertions

```
1.  // Type assertions
2.  let someValue: any = "this is a string";
3.
4.  let strLength: number = (<string>someValue).length;
5.
6.  let someValue: any = "this is a string";
7.
8.  let strLength: number = (someValue as string).length;
9.
```

Typescript. Типы данных. Пересечение и объединение типов

```
1.  type TRandomIntersection = number & string;
2.  type TRandomUnion = number | string;
```

Type Inference

Typescript. Типы данных. Type Inference

```
1.  // Type Inference
2.  let x = 3; // typeof x === 'number'
3.
4.
```

Функции

Typescript. Функции

```
1.  // Typing the function
2.  function add(x: number, y: number): number {
3.      return x + y;
4.  }
5.
6.  let myAdd = function(x: number, y: number): number { return x +
    y; };
7.
8.  // Writing the function type
9.  let myAdd: (x: number, y: number) => number =
10.     function(x: number, y: number): number { return x + y; };
11.
12. let myAdd: (baseValue: number, increment: number) => number =
13.     function(x: number, y: number): number { return x + y; };
14.
```

Typescript. Функции

```
1.  // Inferring the types
2.  // myAdd has the full function type
3.  let myAdd = function(x: number, y: number): number { return  x +
    y; };
4.
5.  // The parameters 'x' and 'y' have the type number
6.  let myAdd: (baseValue: number, increment: number) => number =
7.      function(x, y) { return x + y; };
8.
```

Typescript. Функции

```
1.  // Optional Parameters
2.  function buildName(firstName: string, lastName: string) {
3.      return firstName + " " + lastName;
4.  }
5.
6.  let result1 = buildName("Bob");                  // error, too
    few parameters
7.  let result2 = buildName("Bob", "Adams", "Sr."); // error, too
    many parameters
8.  let result3 = buildName("Bob", "Adams");         // ah, just
    right
9.
```

Typescript. Функции

```
1.  // Optional Parameters
2.  function buildName(firstName: string, lastName?: string) {
3.      if (lastName)
4.          return firstName + " " + lastName;
5.      else
6.          return firstName;
7.  }
8.
9.  let result1 = buildName("Bob");                  // works
    correctly now
10. let result2 = buildName("Bob", "Adams", "Sr."); // error, too
    many parameters
11. let result3 = buildName("Bob", "Adams");         // ah, just
    right
12.
```

Typescript. Функции

```
1.  // Default Parameters
2.  function buildName(firstName = "Will", lastName: string) {
3.      return firstName + " " + lastName;
4.  }
5.
6.  let result1 = buildName("Bob");                  // error, too
    few parameters
7.  let result2 = buildName("Bob", "Adams", "Sr."); // error, too
    many parameters
8.  let result3 = buildName("Bob", "Adams");         // okay and
    returns "Bob Adams"
9.  let result4 = buildName(undefined, "Adams");     // okay and
    returns "Will Adams"
10.
11.
```

Typescript. Функции

```
1.  // Overload
2.  function sum(x: any,y: any) {
3.      if (typeof x === 'number') {
4.          return x + y;
5.      } else {
6.          return `${x} ${y}`;
7.      }
8.  }
9.
10. function sum(x: number,y: number)
11. function sum(x: string,y: number) {
12.     if (typeof x === 'number') {
13.         return x + y;
14.     } else {
15.         return `${x} ${y}`;
16.     }
17. }
18.
19.
```

Интерфейсы

Typescript. Интерфейсы

```
1.  // Our first interface
2.  function printLabel(labeledObj: { label: string }) {
3.      console.log(labeledObj.label);
4.  }
5.
6.  let myObj = {size: 10, label: "Size 10 Object"};
7.  printLabel(myObj);
8.
9.  // Our second first interface
10. interface LabeledValue {
11.     label: string;
12. }
13.
14. function printLabel(labeledObj: LabeledValue) {
15.     console.log(labeledObj.label);
16. }
17.
18. let myObj = {size: 10, label: "Size 10 Object"};
19. printLabel(myObj);
20.
```

Классы

Typescript. Классы

```
1.  // Classes
2.  class Greeter {
3.      greeting: string;
4.      constructor(message: string) {
5.          this.greeting = message;
6.      }
7.      greet() {
8.          return "Hello, " + this.greeting;
9.      }
10. }
11.
12. let greeter = new Greeter("world");
13.
```

Typescript. Классы

```
1.  // Public, private, and protected modifiers
2.  class Animal {
3.      public name: string;
4.      public constructor(theName: string) { this.name = theName; }
5.      public move(distanceInMeters: number) {
6.          console.log(`${this.name} moved ${distanceInMeters}m.`);
7.      }
8.  }
9.
10. // Readonly, static
11. // Abstract
```

Generics

Typescript. Generics

```
1.  // Generic
2.  function echo<T>(arg: T): T {
3.      return arg;
4.  }
5.
```

Enums

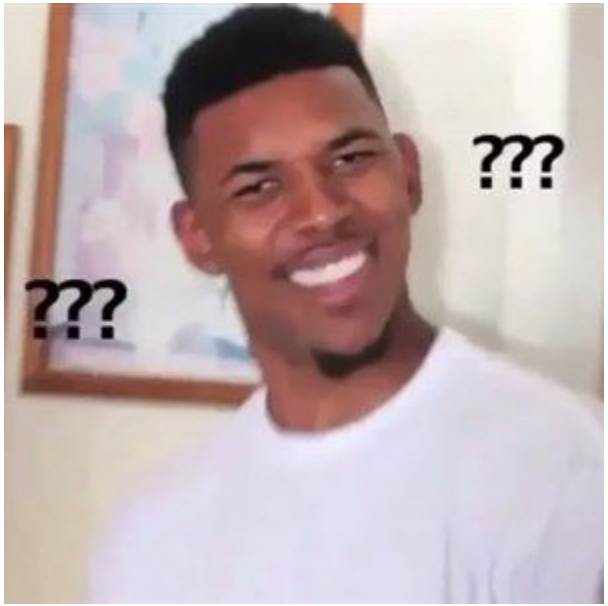
Typescript. Enums

```
enum Direction {
    Up,
    Down,
    Left,
    Right,
}

enum Direction {
    Up = "UP",
    Down = "DOWN",
    Left = "LEFT",
    Right = "RIGHT",
}

```

Modules

Оптимизация Frontend

Вопросы?



“Перерыв! (10 минут)

Препод (с)

Практика

Полезные ссылки

- https://www.typescriptlang.org/
- Серия коротких роликов про TS

Домашнее задание № 6

1. Разработать утилиту на TypeScript

Срок сдачи

18 апреля