**Plan**

Introduction (Meme Slide, Typescript Ideology)

1. Data Types

1. Principle of variable declaration
2. Boolean, string, number, [Any](https://www.typescriptlang.org/docs/handbook/basic-types.html#any)
3. [Null, Undefined](https://www.typescriptlang.org/docs/handbook/basic-types.html#null-and-undefined) and Never
4. [Object](https://www.typescriptlang.org/docs/handbook/basic-types.html#object) and Array
5. Void
6. [Tuple](https://www.typescriptlang.org/docs/handbook/basic-types.html#tuple)
7. [Enum](https://www.typescriptlang.org/docs/handbook/basic-types.html#enum)
8. [Type assertions](https://www.typescriptlang.org/docs/handbook/basic-types.html#type-assertions) (<type>, (str as string) )

2. Classes

3. Interfaces

**Introduction**

Today I want to speak about Typescript. What exactly Typescript is?

Typescript is a typed superset of Javascript that compiles to plain Javascript. Yeah i know it is hard to understand, in other words Typescript is a powerful feature that make life easier for Javascript-developers by implementing features from OOP-related programming languages.

**Data Types**

There are 4 primitive types like boolean, string, number and any. You can see at the example that every variable return its type and type any can be equal to any type.

Also, there are some types like null, undefined and never. From picture you can see that null and undefined is equal null and undefined and such type as never is equal to anything (really anything i have tried).

Next slide is about Object and array, array can be initialized by 2 different ways as you can see from example. Object is a unique type that can be equal only to Object type from Javascript.

Tuple is a new type in Typescript that helps to save the date in appropriate way. It is like an object but strictly depend on the position and types of elements inside.

Enum has the same logic like in javascript, the only difference here in Typescript is that for getting value from enums it is possible to use array indexes.

It is the way how to make type assertions, it is look like very similar to javascript, but there is also a possibility to make assertion with keyword as or diamond brackets.

Function types, i like this the most. With power of Typescript it is possible to define what types of parameters should be and what type of value should function return. It is like OOP.

**Class**

So Classes, classes in typescript look like the same as in javascript but there is a possibilities to define a types of properties inside the class, that i find totally usefull.

Next slide Class extension. I have made 3 classes Employee as the main Waiter and Manager. All the logic is the same as in javascript and by the way At this example we can see realization of Polymorphism.

Modifiers, it is a very powerful thing to make Incapculation and save the data from invasion. I have just remade the class Employee to see how modifiers work. So private, can be used only inside the class when it is need to transfer data to lower level getters can be used. protected is like private, but it means that this property or function can be used inside the class and inside children of this class. public mean that this property or function can be used anywhere without any conflicts.

Readonly, it is easy to understand what this modifier make, it hides the possibility to change the value of this variable, it is like const value but can be changed inside the class.

Static, is a modifier that helps to get properties or functions without initializing a class and by the way this property or function will be the same for all samples of this class.

Abstract class it is a class that have at least one abstract method. Like in example it can be useful for common logic of classes that extend parent class.

**Interface**

Is a new feature compared with javascript. It is a possibility to define some rules that should be followed by the class that this interface implements. Interfaces defines only main features that should class or object contain like properties or methods, but there is no logic inside the interfaces.

So for example Costable , the value that is being used as parameter in function countTaxes should contain property cost to get a right value. Exeption will be thrown if parameter of function doesnt satisfy all the rules in interface.

In interfaces as in classes there can be optional parameters that can be skipped.

Readonly has the same logic as variables have. It is possible to make an sample of interface, there is no possibility to change the value of this sample.

Class types. Interfaces can also describe the logic of classes, for example Calendar should have Date property and it is necessary to have. so when class implements an interface all the properties should be initialized in the class and logic of all methods should be also described.

Interface as parameter. It is possible to use one function for two or more classes that implemented one interface and call this function for all the samples of these classes.