TypeScript In-depth

INTRODUCTION

KUMARAN KANAGARAJ

@kkumaranraj

Why use TypeScript ?

- Superset of JavaScript
- Types add safety
- Types enable faster development
- Complies to plain JavaScript
- Cross-platform
- Open source

Course Modules

- TypeScript Basics
- Functions
- Interfaces
- Classes
- Modules and Namespaces
- Type Definitions
- Best Practices
- Code Reviews

TypeScript Basics

Overview

Declaring variables and constants

- var
- let
- const

Specifying types

Basic data structures

- enums
- arrays
- tuples

Demo

var versus let

Declaring variables and constants

Adding type annotations

Basic Types

Boolean

Number

String

Array

Enum

Any

Void

Functions

Overview

Functions in TypeScript versus JavaScript

Parameter types and return types

Arrow functions

Paramters

- Optional paramters
- Default paramters
- Rest paramters

Overloaded functions

Functions in TypeScript Versus JavaScript

	TypeScript	JavaScript
Types		No types
Arrow functions		Arrow functions (ES2015)
Required and optional parameters		All paramters are optional
Default paramters		Default paramters (ES2015)
Rest paramters		Rest paramters (ES2015)
Overloaded functions		No overloaded function

Demo

Declaring and using function types

Arrow functions

Declaring paramters

- Optional paramters
- Default paramters
- Rest paramters

Overloading functions

Interfaces

Overview

What is an interface ?

Declare interfaces

Interfaces for function types

Extending interfaces

Interfaces for class types

Interfaces ?

Contracts that define types

Complier enforces contracts via type checking

Collections of property and method definitions

Demo

Defining and using interfaces

Interfaces for function types

Extending interfaces

Implementing interface with classes

Classes

Overview

What is a class ?

Class members

- Constructors
- Properties
- Methods

Inheritance

Abstract classes

What is class ?

Templates for creating objects

Provides state storage and behaviour

Encapsulates reuasble functionality

Demo

Creating and using classes

Creating abstract classes

Defining and inheritance hierarchy

Modules and Namespaces

Overview

Modules versus Namespaces

Creating and using namespaces

Creating and using modules

Modules Versus Namespaces

Modules

Tools for organizing code

Browsers supported with module loader

Facilitates code reuse

Modules are the future!

Namespaces

Tools for organizing code

No special loader required

Prevents global namespace pollution

Best for smaller client applications

Demo

Using namspaces

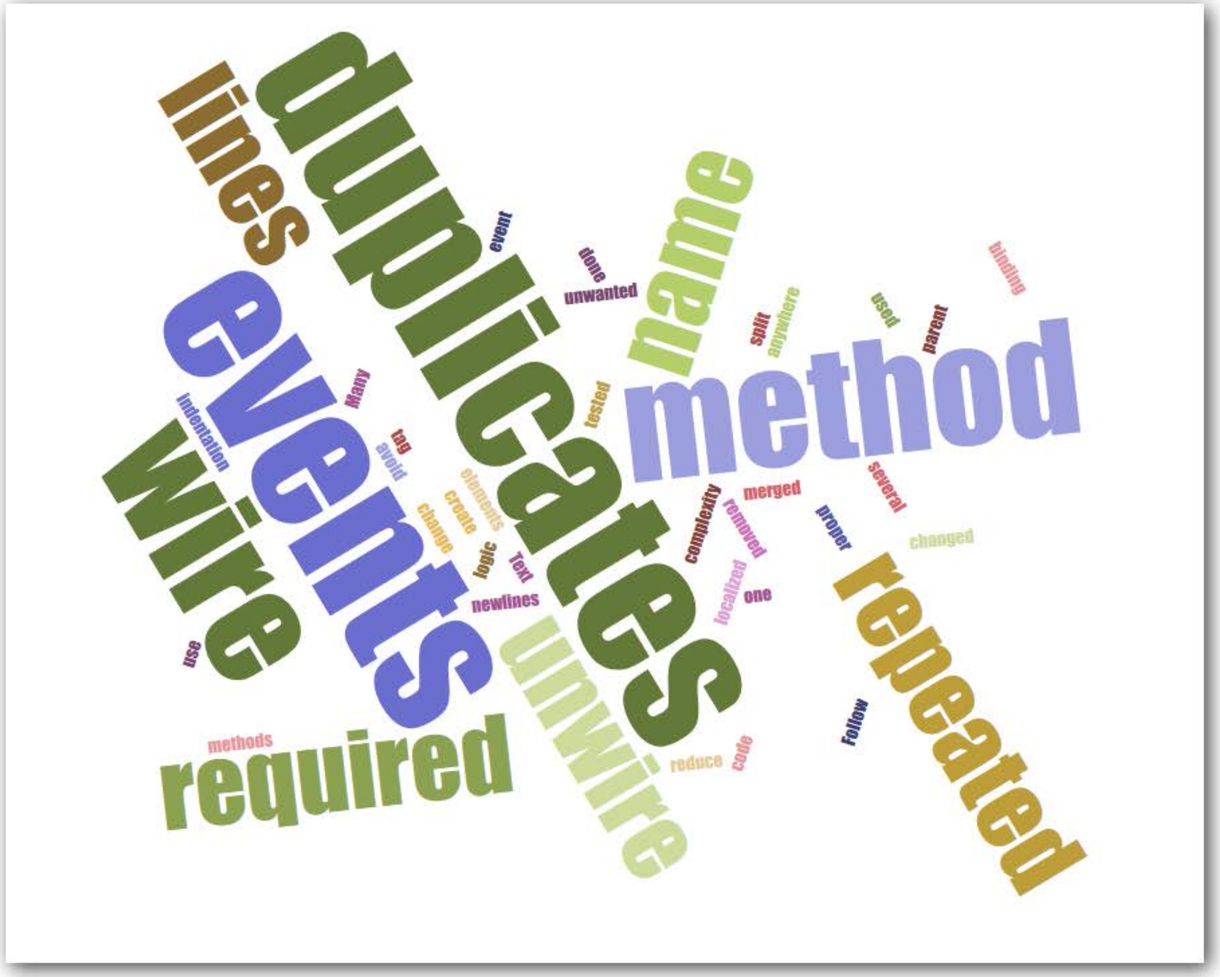
Using modules

Best practices and Clean Code

https://github.com/labs42io/clean-code-typescript

Demo

Code reviews



* is required?
* change method name
* reduce complexity
* split into several methods
* unwanted newlines can be removed
* Text should be localized
* Follow proper indentation
* avoid dup