**TYPESCRIPT от KOBEZZZA**

1. Вступление.

Разработчиком языка TypeScript является Андерс Хейлсберг, создавший ранее Turbo Pascal, Delphi и C#. Язык позволяет явно аннотировать использовать в программе типы.

**Встроенные типы:**

number // Double IEE754 64 бита в памяти

bigint // только целочисленные,

string // UTF-16, хранится как список, однако есть суррогатные пары

boolean // 0000 0001 0000 0000 1 байт

symbol

null // пустое значение

undefined // значение еще не присвоено

object // все остальное не примитив

цена абстракции в js очень высокая, будет работать медленнее чем например раст.

**Слабая и сильная типизация:**

1. В сильной мы не можем использовать операторы для операндов разных типов:

1 + 'ssds'

Js имеет неявную слабую динамическую типизацию.

**Enum быстро.**

enum Foo {

  A,

  B,

  C,

}

// будет

var Foo;

(function (Foo) {

    Foo[Foo["A"] = 0] = "A";

    Foo[Foo["B"] = 1] = "B";

    Foo[Foo["C"] = 2] = "C";

})(Foo || (Foo = {}));

// но по факту это вот это

const Foo{

    A: 0,

    B: 1,

    C: 2,

    0: 'A',

    1: 'B',

    2: 'C',

}

function sum(a: Foo, b: Foo): Foo{

    return a + b

}

sum(Foo.A, Foo.B)

**В реальности это используется примерно вот так:**

enum StatusCodes {

  NotFound = 404,

  ServerError = 500,

}

function sum(a: StatusCodes, b: StatusCodes){

    return a + b

}

sum(StatusCodes.ServerError, StatusCodes.NotFound)

StatusCodes.ServerError[500] === 'ServerError' // true

Enum мы не можем спред.

**Декораторы:**

class Foo {

  @once

  @deprecated()

  foo() {

    deprecated()

  }

}

function deprecated() {

  console.log('fn is deprecated')

}

function once(fn) {

однако это неправильно, наша функция once теперь будет принимать исходный объект в рамках которого этот метод объявлен, название используемого св-ва, объектов дескриптора объекта свойства:

class Foo {

  @once

  @deprecated

  foo() {

  }

}

function deprecated(target, key, desc) {

  // const fn = target[key]

const fn = desc.value

  return {

    ...desc,

    value: function () {

      console.log('fn is deprecated')

      return fn.apply(this, arguments)

    },

  }

}

function once(target, key, desc) {

  const NULL = {}

  let result = NULL

  // ссылка нашу исходную ф-цию

  // const fn = target[key]

const fn = desc.value

  return {

    ...desc,

    value: function () {

      if (result === NULL) {

        result = fn.apply(this, arguments)

      }

      return result

    },

  }

}

В es3 нет декораторов, поэтому мы собираем в es5 явно. Ts чтобы реализовать декораторы подключает свои доп модули. \_\_assign или \_\_decorate. Можно их также подключить через ts lib.

В ts всего две рантайм конструкции своих: enum и декораторы. У ts гибридная типизация, слабая типизация, динамическая типизация.

**Собственные типы с помощью классов:**

class Bla {

  bar: string = '121'

  foo(): number {

    return 1

  }

}

function getBla(a: Bla){

типа getBla(new Bla())

а если мы хотим передать ф-цию конструктор, то должны использовать typeof, но можно еще написать вот так: a: Bla[‘constructor’]