**Питання по TypeScript**

1. Основні особливості TypeScript

* Кросплатформний компілятор TypeScript може бути встановлений на будь-якій операційній системі, включаючи Windows, macOS і Linux.
* Особливості ES6: мова включає в себе більшість особливостей ECMAScript2015 (ES6)), наприклад, функції стріла.
* Об'єктно-орієнтована мова: TypeScript надає всі стандартні функції ООП, такі як класи, інтерфейси та модулі.
* Статична перевірка типів: TypeScript використовує статичний тип і допомагає перевіряти типи під час компіляції. Таким чином ви можете знайти помилки часу компіляції при написанні коду без запуску сценарію.
* TypeScript дозволяє вводити додатковий статичний текст.
* Маніпуляції з DOM: Ви можете використовувати TypeScript для маніпулювання DOM для додавання та видалення елементів веб-сторінки на стороні клієнта.

1. Переваги використання TypeScript

* TypeScript є більш виразною мовою, тобто він має менш синтаксичне безладдя.
* Легко налагоджуйте за допомогою вдосконаленого налагоджувача, який зосереджується на виявленні логічних помилок перед часом компіляції.
* Статична типізація полегшує читання TypeScript і більш структурована, ніж динамічна типізація JavaScript.
* Завдяки універсальній транспілізації можна використовувати на різних платформах як в клієнтських, так і в серверних проектах.

1. Які типи даних вбудовані в TypeScript?

* Числовий тип: використовується для представлення значень числового типу.
* Тип рядка: — послідовність символів, що зберігаються як стандарт кодування Юнікод UTF-16.
* Логічний тип: логічний двійковий перемикач, який має значення Справжній і false.
* Тип null: представляє змінну, значення якої невизначене.
* Тип Невизначена: Невизначений літерал, який є відправною точкою всіх змінних.
* Тип Порожнечу: Тип, призначений методами, які не мають значення, що повертається.

1. Що таке інтерфейс в TypeScript?

Це синтаксична конструкція, призначена для опису публічної частини об'єкта без реалізації (API). Інтерфейс визначається за допомогою ключового слова **interface** і може включати оголошення властивостей і методів, що використовують функцію або функцію arrow.

1. Що таке модулі в TypeScript?

Модулі в TypeScript - це сукупність пов'язаних змінних, функцій, класів та інтерфейсів.

По суті, ви можете думати про модулі як про контейнери, які містять все необхідне для виконання завдань. Модулі можна імпортувати, щоб легко обмінюватися кодом між проектами.

1. Як я можу використовувати TypeScript у фоновій розробці?

Щоб скористатися перевагами TypeScript для фонової розробки, ви можете використовувати його з Node.js.

1. Що таке претензії типу в TypeScript?

Твердження типу в TypeScript працює аналогічно перетворенню типів в інших Yaps, але без перевірки типів або реструктуризації даних, що можливо в таких мовах, як C# і Java. Затвердження типу не впливає на час виконання та використовується виключно компілятором.

Затвердження типу по суті є м'якою версією перетворення типу, яка пропонує компілятору розглядати змінну як певний тип, але не змушує її вписуватися в цю форму, якщо вона знаходиться в іншій формі.

1. Як створити змінну в TypeScript?

Для створення змінної в TypeScript існує три способи: використання функцій var, let, const.

Автор збірника рекомендує уникати використання змінної по можливості **var** тому що це може викликати проблеми при роботі з великими проектами.

**Let** це спосіб оголосити змінні за промовчанням у TypeScript. Порівняно з var ця функція зменшує помилки компіляції та покращує читабельність вашого коду.

**const** створює постійну змінну, значення якої не може змінитися. Функція використовує ті самі правила області, що й дозволятиі допомагає зменшити загальну складність програми.

1. Як викликати конструктор базового класу з дочірнього класу?

class Animal {

name: string;

constructor(theName: string) {

this.name = theName;

}

move(distanceInMeters: number = 0) {

console.log(`${this.name} moved ${distanceInMeters}m.`);

}}

class Snake extends Animal {

constructor(name: string) {

super(name);

}

move(distanceInMeters = 5) {

console.log("Slithering...");

super.move(distanceInMeters);}}

1. Як використовувати мікс(mix in) TypeScript?

Міксини (домішки) - це успадкування, яке працює у протилежному напрямку. Ці елементи мови дозволяють створювати класи, комбінуючи простіші налаштування часткових класів з попередніх класів.

1. Як перевірити null і undefined?

Для перевірки null і undefined можливе використання параметру juggle-check або strict-check.

1. Як зробити класи, визначені в модулі, доступними за межами модуля?

Щоб відкрити модулі для використання за межами модуля, можна скористатися ключовим словом export.

module Admin {

export class Employee {

constructor(name: string, email: string) { }

}

let alex = new Employee('alex', 'alex@gmail.com');

}

let nick = new Admin.Employee('nick', 'nick@yahoo.com');

1. Як перетворити рядок на число?

Щоб перетворити рядок на ціле або число з плаваючою комою, можна:

Як і в JavaScript, скористайтеся функціями **parseInt** або **parseFloat**.

Використовуйте унарний оператор, щоб перетворити рядок на найбільш відповідне число. Наприклад, "3" стане цілим числом, а "3,14" - числом з плаваючою комою. +

1. Що таке файли .map? Чому і як його можна використовувати?

.map — це файли карт, які показують, як вихідний код TypeScript був інтерпретований у корисний код JavaScript. Ця інформація дозволяє виявити будь-яку дивну поведінку компілятора, що спрощує налагодження.

Багато налагоджувачів, таких як Visual Studio або інструменти розробки Chrome, можуть використовувати ці файли для налагодження файлу TypeScript замість JavaScript.

1. Що таке класи в TypeScript? Як ви їх визначаєте?

Клас — це креслення, яке можна використовувати для створення екземплярів об'єктів, які мають точно таку саму конфігурацію, що й шаблон — ті самі властивості та методи. Клас визначає схему того, як об'єкт повинен виглядати і діяти, а потім реалізує цей об'єкт шляхом ініціалізації властивостей класу і визначення його методів.

Класи оголошуються за допомогою ключового слова **class**:

class Student {

studCode: number;

studName: string;

constructor(code: number, name: string) {

this.studName = name;

this.studCode = code;

}

1. Як TypeScript відноситься до JavaScript?

TypeScript - це синтаксична супермножина JavaScript з відкритим вихідним кодом, яка компілюється в JavaScript. Усі оригінальні бібліотеки та синтаксис JavaScript все ще працюють, але TypeScript додає додаткові параметри синтаксису та функції компілятора, яких у JavaScript не має.

TypeScript також може взаємодіяти з більшістю тих же технологій, що і JavaScript, таких як Angular і jQuery.

1. Що таке JSX у TypeScript?

JSX - це вбудований XML-подібний синтаксис, який дозволяє створювати HTML. TypeScript підтримує вбудовування, перевірку типів і компіляцію JSX безпосередньо в JavaScript.

1. Які режими JSX підтримує TypeScript?

TypeScript має вбудовану підтримку режимів preserve, react і react-native.

* **preserve** Зберігає JSX незмінним для подальшого перетворення.
* **react** не виконує перетворення JSX, а замість цього генерує метод react. createElement і відображає його у вигляді файлу з розширенням .js;
* **react-native** поєднує в собі властивості режимів Зберегти і Реагувати: підтримує всі JSX і виводить його у вигляді файлу з розширенням .js.

1. Які області доступні в TypeScript? Як це пов'язано з JavaScript?

* Глобальна сфера застосування: визначається за межами будь-якого класу і може бути використана в будь-якому місці програми;
* Сфера дії функції/класу: змінні, визначені у функції або класі, які можуть бути використані в будь-якому місці цієї області;
* Локальна область/область застосування блоку: Змінні, визначені в локальній області видимості, можуть бути використані в будь-якому місці блоку.

1. Що таке функції стрілки/ лямбди?

Функції зі стрілками є скороченим синтаксисом для визначення функціональних виразів анонімних функцій. Це схоже на лямбда-функції в інших мовах програмування. Функція arrow дозволяє пропустити ключове слово function і написати коротший код.

1. Поясніть, що таке параметри відпочинку та правила їх декларування

Параметри відпочинку дозволяють передавати функціям різну кількість аргументів (нуль і більше). Це корисно, коли немає інформації про те, скільки параметрів отримає функція. Усі аргументи після символу, що залишився ... зберігатиметься в масиві.

1. Що таке Директиви потрійного слешу?

Директиви з трьох скісних рисок - це однорядкові коментарі, які містять XML-тег для використання в якості директив компілятора. Кожна директива визначає, що завантажувати в процес компіляції. Директиви з трьома скісними рисками працюють лише у верхній частині файлу і будуть розглядатися як регулярні коментарі в будь-якому іншому місці файлу.

1. Які існують Директиви потрійного слешу?

* /// <reference path=”…” /> ---- є найбільш поширеною директивою, яка визначає залежність між файлами;
* /// <reference types=”…” /> ---- Визначає залежність для пакета.
* ///<reference lib=”…” /> --- дозволяє виключити вбудований файл "lib".

1. Що робить тип? Omit?

Omit — це форма типу обслуговування, яка спрощує перетворення поширених типів. Пропустити дозволяє створити тип, передаючи поточний Тип і вибір Ключіякий потрібно пропустити в новому типі.

Omit<Type, Keys>

1. Як перевантажити функцію?

Вам потрібно використовувати одне і те ж ім'я функції над вихідною функцією без дужок {} і змінити кількість і типи аргументів та / або тип повернутого значення.

function add(x: string, y: string): string;

function add(x: number, y: number): number {

return x + y;

}

1. До чого можна застосовувати декоратори?

Класи, властивості, методи та аргументи методу.

@MyClassDecorator

**export** **class** **Person** {

…

@MyPropertyDecorator myProperty: string;

}

1. Для чого потрібен тип " Record "?

Це дозволяє створити друковану карту

**let** Person = Record<string, number> = {};

Person.age = 25;

1. Коли використовується ключове слово " unknown"?

Якщо ви не хочете використовувати ключове слово будь-яке та / або не знаєте точного типу заздалегідь, але хочете призначити його пізніше.

**let** person: unknown = 'John';

**if** (**typeof** person === string) {

**let** name: string = person;}

1. Коли слід використовувати ключове слово "declare"?

Якщо ви використовуєте бібліотеку JavaScript, яка не оголошена у вашому проекті TypeScript.

declare const libraryName;

1. Для чого потрібен тип "Omit"?

Це новий тип, в якому ви можете вказати властивості, які будуть виключені з вихідного типу.

type Person = { name: string; age: number; location: string; };

type QuantumPerson = Omit<Person, 'location'>; // Аналогично следующей строке

QuantumPerson = { name: string; age: number; };