1. Главная задача TypeScript?

Надстройка над js, которая позволяет устранить проблемы JS, такие как: типизация, сложный для отладки код.

2. Может ли TypeScript исполняться в браузере?

Нет. Не может. Его нужно скомпилировать командой tsc в js.

3. Объясните процесс понижения уровня кода.

Понижение уровня кода осуществляется за счет упрощения написания кода программистом и скрытия от разработчика многих внутренних (системных) алгоритмов.

4. Какие настройки предоставляет TypeScript для управления уровнем строгости проверок?

tsconfig.json, установите флаг strict в секции compilerOptions в значение true

Из них две секции: Strict Checks и Linter Checks – содержат только опции тех самых флагов строгости. Помимо ещё часть интересующих нас сегодня флагов сокрыта в самой большой группе опций Advanced.

Группа Strict Checks. Пожалуй, флаги данной категории наиболее важные из всех. Вот они: strict, alwaysStrict, noImplicitAny, strictNullChecks, strictFunctionTypes, strictPropertyInitialization, noImplicitThis, strictBindCallApply.

5. Перечислите базовые типы данных, используемые в TypeScript.

В TypeScript имеются следующие базовые типы:

* **boolean**: логическое значение true или false
* **number**: числовое значение
* **string**: строки
* **Array**: массивы
* **кортежи**
* **Enum**: перечисления
* **Any**: произвольный тип
* **Symbol**
* **null и undefined**: соответствуют значениям null и undefined в javascript
* **Never**: также представляет отсутствие значения и используется в качестве возвращаемого типа функций, которые генерируют или возвращают ошибку

https://metanit.com/web/typescript/2.5.php

6. Каким образом в TypeScript аннотируются массивы?

Аннотирование нужно, чтоб определить статический тип данных в массивах и функциях.

Чтобы аннотировать тип массива, который вы используете, используйте определенный тип, за которым следует квадратная скобка : type[] :

let arrayName: type[];

Используйте аннотации TypeScript типа с синтаксисом : [type] , чтобы явно указать тип для переменной, функции, возвращаемого значения функции и т. д.

Источник: https://codezen.ru/osnovnoe-rukovodstvo-po-annotatsiyam-v-typescript

7. Что в TypeScript называется кортежем?

Кортежи (Tuples) также, как и массивы, представляют набор элементов, для которых уже заранее известен тип. В отличие от массивов кортежи могут хранить значения разных типов. Для определения кортежа применяется синтаксис массива:

// определение кортежа - кортеж состоит из двух элементов - строки и числа

let user: [string, number];

https://metanit.com/web/typescript/2.10.php

8. Как аннотировать функции с помощью TypeScript? Всегда ли это необходимо?

greeting = function (name: string) {

return `Hi ${name}`;

};

Используйте аннотации TypeScript типа с синтаксисом : [type] , чтобы явно указать тип для переменной, функции, возвращаемого значения функции и т. д.

Источник: <https://codezen.ru/osnovnoe-rukovodstvo-po-annotatsiyam-v-typescript>

9. Что в TypeScript называется объединением? Привести примеры использования.

Объединения или union не являются собственно типом данных, но они позволяют комбинировать или объединить другие типы. Так, с помощью объединений можно определить переменную, которая может хранить значение двух или более типов:

|  |  |
| --- | --- |
| 1  2  3  4  5 | let id : number | string;  id = "1345dgg5";  console.log(id); // 1345dgg5  id = 234;  console.log(id);  // 234 |

https://metanit.com/web/typescript/2.13.php

10. Объяснить псевдонимы типов, интерфейсы и их различия.

TypeScript позволяет определять псевдонимы типов с помощью ключевого слова **type**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | type id = number | string;    let userId : id = 2;  console.log(`Id: ${userId}`);  userId = "qwerty";  console.log(`Id: ${userId}`); |

Здесь для объединения number|string определяется псевдоним id. Далее мы можем использовать этот псевдоним для определения переменных.

### Расширение псевдонимов

Одни псевдонимы могут заимствовать или расширять код других. Для этого применяется операция **&**. Например:

|  |  |
| --- | --- |
| 1  2 | type Person = {name: string; age: number};  type Employee = Person & {company: string}; |

В данном случае псевдоним Employee расширяет псевдоним Person, добавляя к нему свойство company, которое представляет тип string. То есть фактически мы имеем дело с типом:

https://metanit.com/web/typescript/2.14.php

11. Объяснить утверждения типов. Как производить некорректные утверждения типов?

Утверждение типов - общее[¶](https://scriptdev.ru/guide/035/#-)

При разработке приложений на языках со статической типизацией время от времени может возникнуть нестыковка из-за несоответствия типов. Простыми словами, приходится работать с объектом, принадлежащим к известному типу, но ограниченному более специализированным (менее конкретным) интерфейсом.

Утверждение типа синтаксис[¶](https://scriptdev.ru/guide/035/#_2)

Одним из способов указать компилятору на принадлежность значения к заданному типу является механизм утверждения типа при помощи угловых скобок <ConcreteType>,

12. Какие типы в TypeScript называются литеральными? Привести примеры использования.

Литералы - это *фиксированные* значения, которые являются примитивами JavaScript.

Вы можете использовать строковый литерал в качестве типа. Например:

|  |  |
| --- | --- |
|  | let foo: 'Hello'; |

Они не очень полезны сами по себе, но могут быть собраны в тип объединение для создания мощной (и полезной) абстракции, например:

type CardinalDirection =

| 'North'

| 'East'

| 'South'

| 'West';

function move(

distance: number,

direction: CardinalDirection

) {

// ...

}

move(1, 'North'); // Okay

move(1, 'Nurth'); // Ошибка!

**13. Объяснить сужение типов. Способы сужения типов. Привести примеры**.

Рассмотрим интересный пример:

function printAll(strs: string | string[] | null) {

if (typeof strs === "object") {

for (const s of strs) {

// Object is possibly 'null'. Потенциальным значением объекта является 'null'

console.log(s);

}

} else if (typeof strs === "string") {

console.log(strs);

} else {

// ...

}

}

В функции printAll мы пытаемся проверить, является ли переменная strs объектом (массивом). Но, поскольку выражение typeof null возвращает object (по историческим причинам), мы получаем ошибку.

Таким образом, в приведенном примере мы выполнили сужение к string[] | null вместо желаемого string[].

Существует несколько различных конструкций, которые применяются в TypeScript для сужения:

* Защитник типов typeof
* Проверка на истинность (Truthiness narrowing)
* Сужение при проверке на равенство (Equality narrowing)
* Сужение типов с оператором in
* Сужение типов с instanceof
* Присваивания (Assignments)
* Анализ потока управления
* Использование предикатов типа (type predicates)
* Исключающие объединения (discriminated unions)

<https://it-dev-journal.ru/articles/suzhenie-tipov>

**14. Что называется сигнатурой вызова? Конструкторской сигнатурой?**

**15. Объяснить концепцию перегрузки функций в TypeScript. Привести пример.**

**16. Что в TypeScript называется объектным типом?**

**17. Как TypeScript подходит к сравнению типов?**

**18. Перечислите модификаторы свойств в TypeScript, назовите их особенности.**

**19. Объяснить сигнатуры индексов. Привести примеры использования.**

**20. Назовите особенности использования классов в TypeScript.**

**21. Объясните ключевые слова extends и implements в TypeScript. Назовите их различия.**

**22. Перечислите области видимости в TypeScript и назовите их особенности.**

**23. Объяснить концепцию обобщенных типов в TypeScript. Привести примеры.**

**24. Что в TypeScript называется ограничением обобщенного типа? Привести примеры**