**1.Объясните TypeScript и его использование?**

TypeScript — это язык программирования с открытым исходным кодом, разработанный и поддерживаемый компанией Microsoft, который является расширением JavaScript со статической типизацией. Расширяя JavaScript с помощью аннотаций типов и интерфейсов, он позволяет разработчикам выявлять ошибки в процессе компиляции, а не во время выполнения, тем самым обеспечивая более безопасный и эффективный код.

TypeScript также обладает высокой масштабируемостью, что позволяет разрабатывать крупные приложения с такими функциями, как пространства имён, декораторы и абстрактные классы. Одной из его наиболее значимых сильных сторон является бесшовная интеграция с экосистемой JavaScript и современными фреймворками, такими как Angular, React или Vue.js, что делает его бесценным инструментом для любого разработчика, стремящегося создавать удобные и надёжные веб-приложения.

**2.Можете ли вы назвать некоторые встроенные типы в TypeScript?**

В TypeScript есть несколько встроенных типов, которые помогают разработчикам создавать более безопасный и точный код. Вот некоторые из наиболее распространённых встроенных типов:

any: представляет любой тип значения и позволяет переменной хранить значения нескольких типов.

boolean: представляет логическое значение, которое может быть либо истинным, либо ложным.

число: представляет любое числовое значение, включая целые числа и числа с плавающей запятой.

строка: представляет собой последовательность символов, обычно используемую для хранения текстовых данных.

null: обозначает отсутствие значения, намеренно присвоенного переменной.

undefined: обозначает переменную, которой ещё не присвоено значение.

void: обозначает отсутствие типа, обычно используется в качестве возвращаемого типа для функций, которые не возвращают значение.

**3.Приведите пример дженериков в TypeScript?**

Дженерики в TypeScript позволяют создавать многократно используемые и гибкие компоненты без ущерба для безопасности типов. Они позволяют создавать функции, классы или интерфейсы, которые могут работать с разными типами, сохраняя при этом преимущества статической типизации. Вот пример использования дженериков в функции TypeScript:

**4.Расскажите мне о модулях в TypeScript.**

Модули в TypeScript позволяют организовывать и инкапсулировать код, упрощая разработку, поддержку и тестирование крупных приложений. Они позволяют разделять код на отдельные файлы, каждый из которых отвечает за определённый набор функций или одну задачу. Модули предназначены для управления зависимостями и предотвращения конфликтов имён за счёт использования отдельных пространств имён и импорта/экспорта.

В TypeScript есть два основных типа модулей:

-Внешние модули: также известные как «файловые модули», они представляют собой один модуль на каждый исходный файл. Когда вы создаёте новый файл TypeScript и работаете с ключевыми словами import и export, вы автоматически создаёте внешний модуль. Это наиболее часто используемые модули TypeScript.

-Внешние модули: используются для предоставления информации о типах для сторонних библиотек или существующих глобальных переменных. Обычно они объявляются с помощью файла .d.ts и могут предоставляться самими разработчиками библиотеки или создаваться сообществом. Синтаксис объявления модуля используется для определения внешнего модуля.

**5.Почему TypeScript может быть выбран вместо JavaScript?**

TypeScript обладает рядом преимуществ по сравнению с JavaScript, что делает его предпочтительным выбором для многих разработчиков, стремящихся создавать крупномасштабные и удобные в обслуживании приложения. Одним из наиболее заметных преимуществ TypeScript являются инструменты и интегрированные среды разработки, которые предоставляют богатую среду, помогающую легко находить ошибки. К другим преимуществам относятся:

Расширенное автозаполнение и IntelliSense: TypeScript обеспечивает расширенное автозаполнение, навигацию по коду и возможности рефакторинга благодаря улучшенной интеграции с такими редакторами, как Visual Studio Code, что делает процесс разработки более быстрым и эффективным.

Поддержка крупномасштабных приложений: такие функции, как классы, интерфейсы, обобщения и пространства имён в TypeScript, упрощают создание модульного кода, что делает его подходящим для крупных и сложных приложений.

Улучшенная читаемость кода: аннотации типов TypeScript и расширенные функции ООП вносят ясность в кодовые базы и упрощают понимание, отладку и поддержку кода, написанного другими разработчиками.

Строгие проверки на нулевые значения: TypeScript обеспечивает строгие проверки на нулевые значения, которые помогают обнаруживать нулевые или неопределённые значения во время компиляции, сокращая количество ошибок во время выполнения, связанных с отсутствием или недоступностью данных.

**6.Нужны ли файлы TypeScript для компиляции?**

Да, поскольку TypeScript не может быть интерпретирован браузерами. Для преобразования TypeScript в JavaScript необходима компиляция. А для компиляции нужен компилятор TypeScript. TypeScript обладает всеми функциями и возможностями JavaScript, а также некоторыми дополнительными функциями. Он упрощает работу разработчиков, предоставляет такие функции, как псевдонимы типов, абстрактные классы, кортежи, обобщения и т. д., и позволяет разработчикам добавлять в свои проекты безопасность типов.

**7.Перечислите некоторые преимущества TypeScript?**

TypeScript предлагает несколько преимуществ, которые упрощают процесс разработки, повышают качество кода и оптимизируют совместную работу. Некоторые из этих преимуществ:

-Статическая типизация: TypeScript добавляет в JavaScript статическую типизацию, которая выявляет ошибки, связанные с типами, во время компиляции, а не во время выполнения. Это позволяет разработчикам выявлять и устранять проблемы на ранних этапах разработки, сокращая количество ошибок и повышая общее качество кода.

-Удобство сопровождения кода: статические типы, интерфейсы и другие функции ООП в TypeScript способствуют повышению читаемости и удобства сопровождения кода, упрощая для разработчиков понимание и работу с общими кодовыми базами.

-Улучшенные инструменты и поддержка редакторов: TypeScript обеспечивает отличную интеграцию с популярными интегрированными средами разработки, что позволяет использовать расширенное автозаполнение, навигацию по коду и инструменты рефакторинга. Это улучшает процесс разработки и повышает производительность.

-Поддержка крупномасштабных приложений: функции TypeScript, такие как обобщения, пространства имён и модули, способствуют созданию модульного и масштабируемого кода, что делает его идеальным выбором для крупных и сложных приложений.

-Совместимость с экосистемой JavaScript: TypeScript полностью совместим с библиотеками, фреймворками и инструментами JavaScript, что обеспечивает его беспроблемную интеграцию в существующие рабочие процессы разработки.

**9.В чем разница между TypeScript и статически типизированным языком?**

TypeScript поддерживает необязательную статическую типизацию, что означает, что вы можете указать компилятору игнорировать тип переменной. Мы можем присваивать переменной значения любого типа, используя любой тип данных. Во время компиляции TypeScript не выполняет проверку на ошибки.

**10.Возможна ли интерполяция строк в TypeScript?**

Да, в TypeScript возможна интерполяция строк, которая также известна как шаблонные литералы или шаблонные строки. Она позволяет встраивать выражения в строковые литералы, используя обратные кавычки (`) вместо одинарных или двойных кавычек. Чтобы включить переменную или выражение в строку, используйте синтаксис ${выражение}. Эта функция позволяет создавать более читаемые и лаконичные строки, не прибегая к конкатенации строк.

**13.Объясните интерфейс на TypeScript?**

В TypeScript интерфейс — это фундаментальное понятие, которое позволяет определить форму, структуру и контракт объекта или класса без реализации его фактической функциональности. Интерфейсы позволяют определять пользовательские типы, указывая свойства и их соответствующие типы, которые должен иметь объект, а класс, реализующий интерфейс, должен соответствовать этим спецификациям. Интерфейсы также можно использовать с функциями для определения конкретных параметров и типов возвращаемых значений.

Интерфейсы могут содержать необязательные свойства, свойства только для чтения и сигнатуры методов, а также поддерживать расширение и реализацию нескольких интерфейсов.

**14.Объясните декораторы в TypeScript.**

Декораторы в TypeScript — это особый вид декларативного синтаксиса, который используется для добавления метаданных или изменения поведения классов, методов, свойств или параметров. Они обеспечивают удобный способ применения многократно используемой логики или поведения в нескольких частях кода без необходимости вручную изменять каждый целевой элемент. Декораторы создаются с помощью функций высшего порядка, которые принимают целевой элемент (например, класс или свойство) в качестве аргумента и возвращают его изменённую версию или выполняют с ним определённые действия.

**17.Что вы можете сделать в TypeScript, чтобы проверить null и undefined?**

В TypeScript null и undefined — это отдельные, но связанные типы, представляющие отсутствие или неинициализированное состояние значения. Чтобы проверить наличие значения null или undefined, можно использовать различные языковые конструкции и методы:

Проверка на равенство Используйте оператор равенства == или оператор неравенства != для одновременной проверки на null и undefined. Вы также можете использовать операторы строгого равенства === или строгого неравенства !== с последующей явной проверкой на null и undefined.

Оператор необязательной цепочки ?. может использоваться для доступа к свойствам и методам объекта, который может быть нулевым или неопределённым. Если объект нулевой или неопределённый, результатом выражения будет неопределённое значение.

const result = obj?.property; // результатом будет неопределённое значение, если obj нулевой или неопределённый

**18.Все ли объектно-ориентированные принципы поддерживаются TypeScript?**

Да, он поддерживает принципы ООП, такие как,

Наследование

Абстракция

Полиморфизм

Инкапсуляция

**21.Можно ли использовать TypeScript в серверной части и как?**

Помимо клиентской части или браузера, TypeScript можно использовать и на серверной части. TypeScript можно использовать на серверной части для создания серверных приложений и сервисов. Один из популярных способов использования TypeScript на серверной части — Node.js. С помощью TypeScript и Node.js можно создавать масштабируемые и удобные в обслуживании серверные приложения, пользуясь преимуществами статической типизации и других функций TypeScript.

**23.Можете ли вы объяснить разницу между String и строкой в TypeScript?**

В TypeScript String и string представляют текстовые данные, но имеют разное значение и используются по-разному:

string: это самый простой, предпочтительный и наиболее часто используемый тип для представления текстовых данных в TypeScript. Это примитивный тип, который автоматически присваивается строковым литералам. При использовании строкового типа TypeScript выполняет проверку типов и гарантирует, что все операции и присваивания допустимы для строк.

let text: string = 'Привет, машинописный текст!';

String: это встроенный глобальный объект JavaScript, который оборачивает простой строковый тип, предоставляя дополнительные методы и функции. Объект String выступает в качестве конструктора для создания новых строк и в качестве пространства имён для различных вспомогательных методов для работы со строками или их проверки. Однако в большинстве случаев нет необходимости использовать объект String, поскольку TypeScript и JavaScript автоматически преобразуют примитивные строки в объекты String при необходимости.

пусть обернутый текст будет строкой: String = new String('Обернутый TypeScript!');

На практике для работы с текстовыми данными в TypeScript обычно следует использовать строковый тип, так как он обеспечивает проверку типов и более простой и эффективный код. В то же время не следует использовать объект String, если только вам не нужна дополнительная функциональность, которую он предоставляет.

**27.Подскажите мне Компоненты TypeScript?**

TypeScript состоит из нескольких компонентов, которые работают вместе, предоставляя разработчикам мощные функции для создания приложений на JavaScript:

Язык− TypeScript расширяет возможности JavaScript, добавляя необязательную статическую типизацию, интерфейсы, классы, декораторы, пространства имён и многие другие функции, которых изначально нет в JavaScript.

Компилятор — у TypeScript есть специальный транспилятор (tsc), который преобразует код TypeScript в обычный JavaScript. Компилятор TypeScript обеспечивает безопасность типов, проверяет наличие ошибок и преобразует TypeScript в JavaScript, который может работать в разных браузерах и на разных платформах.

Язык-сервис: Язык-сервис TypeScript обеспечивает поддержку редакторов, такую как IntelliSense, автозавершение кода, подсветку синтаксиса и рефакторинг, для современных сред разработки, таких как Visual Studio, VSCode, WebStorm и т. д. Эти расширенные функции редактирования позволяют разработчикам эффективно и продуктивно писать код.

Определения типов: TypeScript использует файлы определений типов (.d.ts) для предоставления информации о типах внешних библиотек или кода JavaScript. Определения типов помогают разработчикам TypeScript получать доступ к информации о типах сторонних библиотек, таких как jQuery, React или Express, написанных на JavaScript. Репозиторий DefinitelyTyped — популярный источник определений типов для различных библиотек JavaScript.

Инструменты: TypeScript имеет надёжную интеграцию с популярными инструментами сборки, средствами запуска тестов, средствами выполнения задач и сборщиками, такими как Webpack, Rollup, Gulp, Grunt, Babel и Jest. Эти интеграции обеспечивают разработчикам TypeScript бесперебойный процесс разработки, тестирования и сборки.

29**.Перечислите шаги, которые вы будете использовать для компиляции файлов TypeScript?**

Чтобы скомпилировать файлы TypeScript, выполните следующие действия:

Установите TypeScript: сначала установите TypeScript глобально в вашей системе с помощью npm (менеджера пакетов Node.js), выполнив команду npm install -g typescript. Если у вас не установлен Node.js, вы можете скачать его с https://nodejs.org/en/download.

Напишите код на TypeScript: создайте файл TypeScript (с расширением .ts) и напишите в нём код на TypeScript. Например, создайте файл с именем app.ts и добавьте следующий код: const message: string = 'Привет, TypeScript!'; console.log(message);.

Скомпилируйте код TypeScript: запустите компилятор TypeScript (tsc) для файла TypeScript, выполнив команду tsc app.ts. Эта команда скомпилирует файл app.ts и создаст соответствующий файл JavaScript app.js в том же каталоге.

Выполните скомпилированный JavaScript: после успешной компиляции файлов TypeScript в JavaScript вы можете выполнить файлы .js с помощью Node.js или включить их в свои HTML-файлы для использования в браузере. Чтобы запустить скомпилированный файл JavaScript с помощью Node.js, выполните команду node app.js (или node ./dist/app.js, если вы используете файл tsconfig.json с параметром outDir).

**32.Можно ли автоматически компилировать .ts при внесении изменений в файл .ts в реальном времени?**

Да, вы можете автоматически компилировать файл TypeScript (.ts) всякий раз, когда происходят изменения в режиме реального времени, используя опцию компилятора TypeScript --watch . TSC (компилятор TypeScript) будет отслеживать изменения в файлах .ts и автоматически перекомпилировать их при каждом обнаружении изменений.

Чтобы использовать опцию --watch, откройте свой терминал или командную строку, перейдите в каталог, содержащий файлы .ts, и выполните следующую команду:

tsc --watch файл.ts

Замените ваш файл-file.ts на имя вашего файла TypeScript.

Если вы хотите отслеживать и компилировать несколько файлов .ts или весь проект, вы также можете указать параметр --watch в файле настроек tsconfig.json вашего проекта. Создайте файл tsconfig.json в корневом каталоге вашего проекта TypeScript (если у вас его ещё нет) и добавьте следующее:

**7.TypeScript поддерживает необязательные параметры в функции, можете ли вы объяснить, каким образом?**

В компиляторе TypeScript возникает ошибка, если функция вызывается без указания точного количества и типов параметров, указанных в сигнатуре функции. Чтобы объявить необязательный параметр, используйте символ ? после имени параметра в сигнатуре функции.

**13.Объясните, что такое файл tsconfig.json?**

Файл tsconfig.json — это файл конфигурации, используемый в проектах TypeScript для указания различных параметров и настроек компилятора. Он позволяет разработчикам управлять такими параметрами, как выходная папка для сгенерированного кода JavaScript, стратегия разрешения модулей, строгая проверка типов, сопоставление источников и функции, характерные для различных версий ECMAScript. Компилятор TypeScript использует этот файл конфигурации, чтобы понимать, как компилировать и преобразовывать код TypeScript в JavaScript.

**19. Объясните утверждения типа в TypeScript?**

Утверждения типов в TypeScript — это способ явного информирования компилятора о том, что вы, как разработчик, уверены в типе конкретной переменной или выражения. Утверждая тип, вы, по сути, переопределяете предполагаемый тип TypeScript, что даёт вам больше контроля над тем, как компилятор обрабатывает эту переменную или выражение. Утверждения типов не изменяют значение или поведение переменной во время выполнения, но они могут влиять на информацию о типе во время компиляции, обеспечивая дополнительные функции или предотвращая ошибки во время компиляции

**22.Объясните синтаксис “as” в TypeScript.**

Синтаксис «as» в TypeScript используется для создания утверждения типа, которое позволяет явно сообщить компилятору, что вы знаете фактический тип значения и хотите рассматривать его как конкретный тип. Это может быть полезно, когда компилятор TypeScript не может автоматически определить правильный тип или когда вам нужно выполнить преобразование типа во время разработки.

Утверждения типа с синтаксисом «as» не изменяют поведение базового значения во время выполнения, но предоставляют информацию о типе во время компиляции, которая может помочь выявить ошибки, связанные с типами.

**23.Что вы знаете о перечислении в TypeScript?**

В TypeScript перечисление (сокращение от enumeration) — это пользовательский тип данных, который позволяет определить набор именованных числовых констант, делая код более понятным и удобным для поддержки. Перечисления — это надёжный и выразительный способ работы с наборами значений, представляющими определённые категории, состояния или опции в вашей программе.

**29.Как вы можете создавать массивы только для чтения в TypeScript?**

Используя тип ReadonlyArray, мы можем определить массивы как доступные только для чтения. Таким образом, любая переменная, ссылающаяся на ReadonlyArray, не будет изменять ни один элемент массива.

**9.Объясните, что такое ключевое слово «type» в TypeScript и как его использовать?**

В TypeScript ключевое слово type используется для определения пользовательских псевдонимов типов многократного применения, которые могут представлять сложные типы или давать более понятное название существующему типу. Псевдоним типа может относиться к примитивным типам, типам объединений, типам пересечений, типам объектов, типам функций, массивам или любому другому допустимому типу TypeScript. Псевдонимы типов улучшают читаемость и удобство сопровождения кода и позволяют разработчикам со временем создавать более выразительные аннотации типов.