**JS**

**3 В чем разница между операторами "==" и "==="?**

<https://programmera.ru/uroki-po-javascript/raznitsa-mezhdu-operatorami-i/>

<https://developer.mozilla.org/ru/docs/Web/JavaScript/Equality_comparisons_and_sameness>

**6 В чем разница между методами call, apply и bind**

<https://medium.com/nuances-of-programming/%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%B2-bind-call-and-apply-%D0%B2-javascript-77ec312b1ec9#:~:text=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D1%8B%20call()%20%D0%B8%20apply,%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8E%2C%20%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D0%BE%D0%B9%20%D0%BC%D0%BE%D0%B6%D0%BD%D0%BE%20%D0%BE%D1%82%D0%BB%D0%BE%D0%B6%D0%B8%D1%82%D1%8C>.

**8 В чем разница между ключевыми словами «var», «let» и «const»?**

**9 Что такое модули в JS?**

**Под модулем подразумевается отдельный файл с JavaScript-кодом, который оказывает минимальное влияние как на Глобальный объект, так и на прочие модули, которые могут быть подключены.**

<script src="main.js" 
entry 
type="module"> 

**10 Что делают атрибуты async и defer у тега script?**

**Атрибут async**

при обнаружении <script async src="..."> браузер не останавливает обработку страницы на время загрузки скрипта с сервера, а спокойно обрабатывает и отрисовывает страницу дальше.

Когда скрипт будет загружен – браузер остановит процесс вывода страницы пользователю, (если конечна страница ещё не была полностью выведена на экран), и начнёт выполнять загруженный скрипт.

**Атрибут defer**

js-сценария так же выполняется асинхронно. НО есть два отличия от атрибута async.

Первое – браузер гарантирует, что относительный порядок скриптов с defer будет сохранён.

Второе отличие – скрипт с defer сработает, только когда весь HTML-документ будет обработан браузером.

**13 В чём разница между приватными и защищенными полями в JS?**

<https://learn.javascript.ru/private-protected-properties-methods>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes/Private_class_fields>

<https://javascript.info/private-protected-properties-methods>

**14 Что такое REPL?**

**REPL** (Read Eval Print Loop) — режим обработки команд с обязательным выводом результата в среду выполнения команды. Кароче это консоль в devtools браузера Chrome

**15 Как мы можем наследовать свойства родительского конструктора с помощью   
синтаксиса функций?**

**16 Для чего предназначены методы setTimeout и setInterval?**

<https://learn.javascript.ru/settimeout-setinterval#setinterval>

* + setTimeout позволяет вызвать функцию **один раз** через определённый интервал времени.
  + setInterval позволяет вызывать функцию **регулярно**, повторяя вызов через определённый интервал времени.

**17 Какой код нужно прописать в консоли,чтобы отключить все таймеры на странице?**

**18 Что такое анонимная функция?**

<https://learnjs.ru/lessons/anonymous-functions/>

**19 Что такое стрелочные функции? Отличаются ли они чем-нибудь от обычных?**

Берёт LexicalEnvironment из внешней функции. Не может использоваться как конструктор.

**20 С помощью какого метода можно отсортировать массив? Как отсортировать массив в обратном порядке?**

<https://medium.com/@stasonmars/%D0%BA%D0%B0%D0%BA-%D0%B4%D0%B5%D0%BB%D0%B0%D1%82%D1%8C-%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D1%83-%D0%B2-javascript-%D0%BF%D1%80%D0%B8-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D0%B8-sort-d492c97ccfdf#:~:text=%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0%20%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2%D0%BE%D0%B2%20%D0%B2%20JavaScript%20%D0%B4%D0%B5%D0%BB%D0%B0%D0%B5%D1%82%D1%81%D1%8F,%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%2C%20%D0%B5%D1%81%D0%BB%D0%B8%20%D0%BF%D0%BE%D0%BF%D1%8B%D1%82%D0%B0%D1%82%D1%8C%D1%81%D1%8F%20%D0%B7%D0%B0%D0%B9%D1%82%D0%B8%20%D0%B4%D0%B0%D0%BB%D1%8C%D1%88%D0%B5>.

Путём вызова встроенного в js-массив метода под названием sort, которому передают compare-функцию определяющую порядок сортировки. Compare-Функция возвращает принимает 2 аргумента и возвращает порядок этих аргументов относительно друг друга. Например, одно из этих значений -1, 0, 1

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort>

**TS**

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

noImplicitAny - Использование флага noImplicitAny или соответствующей настройки приводит к тому, что при обнаружении переменной с неявным типом any выбрасывается исключение

strictNullChecks - С данным флагом компилятор больше не позволяет передавать undefined или null туда, где ожидаются переменные других типов, если это не разрешено явно.

**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

**7 Что в TypeScript называется кортежем?**

Кортежи (Tuples) по сути являются массивами, т.е. представляют набор элементов, для которых уже заранее известен тип, только в отличие от массивов кортежи могут хранить значения разных типов и имеют определённую длину.

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

let user: [string, number];

Для присвоения значения применяется массив:

user = ["Tom", 28];

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

let id : number | string;

id = "1345dgg5";

**10 Объяснить псевдонимы типов, интерфейсы и их различия.**

<https://medium.com/nuances-of-programming/%D1%80%D0%B0%D0%B7%D0%BB%D0%B8%D1%87%D0%B8%D1%8F-%D0%BC%D0%B5%D0%B6%D0%B4%D1%83-%D0%BF%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D0%BD%D0%B8%D0%BC%D0%B0%D0%BC%D0%B8-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-%D0%B8-%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0%D0%BC%D0%B8-%D0%B2-typescript-4-6-992aaf59c97#:~:text=%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D0%B5%D1%82%20extends%20%D0%B4%D0%BB%D1%8F%20%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8,%D0%B8%20%D1%80%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%BE%D0%B2%20%D0%BC%D0%BE%D0%B6%D0%B5%D1%82%20%D0%BF%D1%80%D0%B8%D0%B3%D0%BE%D0%B4%D0%B8%D1%82%D1%8C%D1%81%D1%8F>.

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

<https://igorfonin.gitbook.io/typescript-book-ru/typescript-type-system/typeassertion>

const foo: any;

const bar = <string>foo

Либо

const bar = foo as String

Есть ещё двойное утверждение:

function handler(event: Event) {

const element = (event as any) as HTMLElement; // ok

}

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

<https://igorfonin.gitbook.io/typescript-book-ru/typescript-type-system/literals>

Литаральные типы - это когда литералы используются в качестве типа. Например, можем взять строковый литерал и представить его следующим образом:

let foo: 'Hello';

Здесь мы создали переменную с именем foo, которая принимает только одну переменную, буквальное значение которой равно Hello:

let foo: 'Hello';

foo = 'bar'; // Error: 'bar' не может быть назначен типу 'Hello'

Или так тоже можно сделать:

type CardinalDirection = 'North' | 'East' | 'South' | 'West';

type OneToFive = 1 | 2 | 3 | 4 | 5;

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

**function** padLeft(padding: number | string, input: string) {  
 **if** (**typeof** padding === "number") {  
 **return new** Array(padding + 1).join(" ") + input;  
 *// (parameter) padding: number*  
 }  
 **return** padding + input;  
 *// (parameter) padding: string*  
}

Выражение typeof padding === 'number' называется защитником или предохранителем типа (type guard). А процесс приведения определенного типа к более конкретной версии с помощью защитников типа и присвоений называется сужением типа (narrowing).

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

<https://habr.com/ru/companies/macloud/articles/561470/>

Простейшим способом описания типа функции является выражение (function type expressions). Такие типы похожи на стрелочные функции:

**function** greeter(fn: **(a: string) => void**) {  
 fn('Hello, World')  
}

**function** printToConsole(s: string) {  
 console.log(s)  
}

greeter(printToConsole)

В JS функции, кроме того, что являются функциями, которые можно вызвать, ещё к тому же являются объектами. А как объекты, они могут иметь собственные свойства. Но вот этот синтаксис не позволяет нам указать, что у функции есть какие-то ещё сво-ва кроме аргумента a типа string, и что функция возвращает пустоту:

type GreetFn = (a: string) => **void**

**function greeter**(fn: GreetFn) { // ... }

По этому для описания функций можно использовать так называемую сигнатуру вызова (call signature) в объектном типе:

type DescFn = {  
 **description**: string  
 (someArg: number): **boolean**  
}  
**function** doSomething(fn: DescFn) {  
 console.log(`Значением, возвращаемым ${fn.**description**} является ${fn(6)}`)  
}

Таким образом мы явно дали понять, что нам нужна не просто функция, а объект у которого есть свойство **description**. Но в тоже время мы не забыли и про то, что надо описать аргументы нашей функции и возвращаемое ею значение:

(someArg: number): **boolean**

Сигнатуры конструктора (construct signatures) похожи на сигнатуры методов, но их предназначение в том, чтобы указать на наличие в типе\классе определённого конструктора.

type SomeConstructor = { **new** (s: string): SomeObject }

**function fn**(ctor: SomeConstructor) {

**return new** ctor('Hello!')

}

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

Перегрузка функции (function overload)

<https://habr.com/ru/companies/macloud/articles/561470/>

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

<https://habr.com/ru/companies/macloud/articles/562054/>