1. Explain why `{ a: 1 } === { a: 1 }` is `false` in JavaScript

Ans: The “===” comparison operator is used to compare both types and values. Whenever an object is declared, JavaScript will store a unique reference to that object in memory even if the objects and properties are the same. This is why it translates to false as the reference will always be unique.

2. Describe the runtime difference(s) between `for await (const a of [p1, p2, p3]) { ... }` vs `[p1, p2, p3].forEach(async (p) { await p })`

Ans: “for await” would iterate over the promises in sequence. This will wait for one promise to resolve before moving to the next promise. The promises are resolved in a blocking fashion and in the order of the iteration. While “forEach” iterates through the promises concurrently. Each iteration has an async function which will resolve at different times in a non-blocking fashion. Simply put, the order is not preserved.

3. Explain the difference between nodejs and V8

Ans: V8 engine is a JavaScript engine that compiles JavaScript code into machine code for a computer’s processor to execute it. The V8 engine is open source and can be embedded inside a browser like chrome or any application. On the other hand, the NodeJs application is a runtime environment that is built on top of the V8 engine and it basically allows JavaScript to be executed outside of the web browser which is useful to build server side applications.

4. Using typescript, what is the difference between an enum and an object?

Ans: Enum in Typescript allows you to define a named constants. It maps a named constants to any value assigned. For example,

enum Gender {

Male = 1,

Female = 2

}

You can use the Gender enum as Gender.Male or Gender.Female. Enum also supports reverse mapping which you can call as Gender[1].

Objects on the other hand, is a key-value pair collection. The keys can be a string while the value can be of any type. Unlike enums, Objects do not support reverse mapping and it can have dynamic keys and values making it more flexibile. They also support type safety. For instance, the use of const while declaring an object makes it a readonly variable. This is shown in the example below.

const Gender = {

Male: 1,

Female: 2

}

5. Write a typescript variable declaration for a variable `a` that will guarantee that it is a property name or key of object `b`

Ans:

type B = {

Name: string

age: number

active: boolean

}

const b: B = {

Name: "John Doe",

age: 16,

active: true

}

const a:keyof B = "Name";

6. Describe some drawbacks or common pitfalls when using typescript in large applications

Ans:

1. Typescript can lead to complicated codes that might be difficult to understand especially when you have a lot of deeply nested types.
2. Typescript can inferred type might not be string as expected leading to bugs
3. Typescript can result to compatibility issues when integrating with a Third party library making developers to use “any” which defeats the purpose of Typescript.
4. Typescript in a large project can slow down compilation leading to slower build times.
5. Typescript can lead to code that can be hard to change which reduces flexibility and increases compilation time.

8. Demonstrate a nodejs function that increases the major garbage collection

Ans: refer to readme file

9. Write a jest mock of a class having a function `fetchAllRecords`, that will return a resolved promise immediately to array `[1, 2, 3]`.

Ans: refer to readme file