Nomes: Rafael Alves Andrade, Felipe Rossi e Vinícius Rodrigues **Turma:** 2TDS-B | **Tarde**

1) Explique com suas palavras:

a) Sim, em JS é possível manipular Strings como Arrays porque strings são iteráveis.

EXEMPLOS

- Dividir uma string em array com .split()
- Juntar uma Array em uma string com .join()

b) Math.Floor tem como função arredondar um valor para baixo, ou seja, 5.3 arredonda para 5; Math.Ceil() tem como função parecida com a Math.Floor, porém a Math.Ceil() arredonda um número independente do valor, ele vai arredondar para cima; Math.round() tem como função arredondar também, porém essa função depende de seu Decimal. Por exemplo se for maior ou igual a 0.5, arredondará para cima, caso seja menor que 0.5, arredondará para baixo

c) Usando a função Date(), pois assim você consegue trabalhar com horário e data específica

2) Explique com suas palavras:

a) Expressões Regulares são comuns em diversas linguagens de programação. Tem como função estabelecer um padrão que está sendo associado a uma string (Buscar alguma informação). Em JS, podem ser utilizados com match, replace, Search e split de alguma string

b) O método .test é um função que pertence ao RegExp, isso no JS. Esse método verifica se uma Expressão Regular encontra alguma correspondência em outra String, caso o método encontre algo na String é validado TRUE, caso ao contrário, é valida FALSE

3) Explique com suas palavras:

a) Uma Programação Assíncrona em JS é uma abordagem que executa o seu código(JS) linha por linha. Ele funciona através das funções de Callback, Promises, await/async, entre outras...

b) Single-Threaded afeta justamente porque o JS executa um código de uma vez só. Porém o Single-Threaded executa todos os códigos de uma vez só e isso pode sim gerar conflito.

c) O uso de Promises faz com que evite os problemas aninhados de call-backs e são funções que representam uma falha ou um sucesso de uma operação. A Async ou também await é uma implementação de Promises. Sim, existe uma relação direta entre os três, todos são formas de lidar com operações assíncronas em JS , mas com níveis diferentes de abstração e legibilidade.

d) No JS, o Event Loop é o mecanismo responsável por gerenciar a execução de código assíncrono de forma que não bloqueia, mesmo sendo JS, assim ele organiza QUANDO E COMO as tarefas serão executadas.

4) No console.log aparece a seguinte mensagem: **A data fornecida está a 1 dias de hoje.**

Como foi feito:

Utilizando como base o código já dado pelo professor, apenas chamei as duas funções criadas: **validar("2025-06-03", calcular);**

5) A ordem de exibição dos números é: 8, 3, 4, 7, 1, 6, 5, 2.
Ela é exibida dessa forma, pois:

- A função "**main()**" é chamada. Com isso, a linha 13 é executada "**console.log("8");**"
- Depois a linha 14 é executada e chama a função "**comDelay();**"
- Dentro da função "**comDelay();**" é executada a linha 2 "**console.log("3");**"
- Logo após é executada a linha 9 "**console.log("4");**"
- A linha 21 também é acionada em seguida "**console.log("7");**"
- Depois que os consoles.logs foram todos acionados dentro de suas funções (main e ComDelay), agora o programa irá ler a Promise e, por último, o setTimeout, que define a ordem de prioridade que será lido pelo programa. Promise e setTimeout são funções assíncronas (async).
- Após isso, as linhas 6, 7 e 8 serão lidas

```
Promise.resolve().then(() => {
        console.log("1");
    });
```

- Depois as linhas 15, 16 e 17:

```
Promise.resolve().then(() => {
        console.log("6");
    });
```

- Agora entraremos nos setTimeout, onde são definidas por milissegundos a ordem de execução (quanto maiores os milissegundos, menos prioridade o programa dará para executá-lo).

- Linhas 18, 19 e 20:

```
setTimeout(() => {
        console.log("5");
    }, 1000);
```

- E, por último, as linhas 3, 4 e 5:

```
setTimeout(() => {
        console.log("2");
    }, 2000);
```