**Atividade Avaliativa 02/06**

**1)** a- Em JS existe alguma possibilidade de manipulação de Strings associadas a Arrays? Como isso funciona? Dê exemplos:

R:é possível usar a propriedade de arrays em string mas o acesso pode ser imprevisível EX:

let texto = "HELLO WORLD";

let caracter = texto[0];

console.log(caracter); //H

mais isso não quer dizer strings são arrays, se nenhum caractere for encontrado, o array retorna undefined e o charAt() retorna string vazia.

Obs: alguns métodos de Array pode ser associados a Strings facilitando a manipulação, já que string é um conjunto de caracteres.

b) Explique a diferença entre Math.floor(), Math.ceil() e Math.round() no arredondamento de números em JS.

R: Math.floor() arredonda para baixo em direção ao números menores.

Math.ceil() retorna o menor número inteiro maior ou igual ao número

fornecido. Faz "arredondamento para cima", independentemente do sinal

do número.

Math.round() arredonda para o número inteiro mais próximo de um

número de ponto flutuante, com base no valor da parte decimal

c) Como objetos do tipo data são criados em JS? Em sua explicação faça uma demonstração capturando a data atual.

R: "let dataAtual = new Date();" ou assim passando uma data especifica "let data = new Date("2025-4-18");". para capturar a data atual e so fazer isso "let dataAtual = new Date();" e para exibir tem vários jeitos mais pode ser um "console.log(dataAtual);"

**2)**

a- O que é uma expressão regular em JS? Qual o seu propósito?

R:Normalmente o proposito esta associado a uma string quando precisamos de algum tipo de padrão, Exemplo: ela define especificações de entrada ou busca. em JavaScript elas são objetos, e podem ser utilizadas com os métodos match, replace, search, e split de uma

string.

b) Para que serve o método .test? Qual a sua relação com a validação de strings.

R:O método .test serve para validar um caractere feito em regex, ele analisa se o que foi enviado retornará uma string, com true ou false (um método boolean).

**3)**

a- O que é e como funciona Programação Assíncrona em JS?

R:Quando a Programação assíncrona em js e executada ela percorre o código linha por linha de cima para baixo, e vai executando instrução por instrução uma após a outra pq o js e single-threaded tento apenas uma theread principal.

b- Como o conceito de Single-Threaded pode afetar o

comportamento no fluxo de tarefas no JavaScript?

R: no javascript qualquer função que demore para ser concluída pode bloquear todas as outras tarefas de serem executadas pq o js é single-threaded.

c- Qual a diferença entre promises e async await?

R: A diferença e que a promessas e um objeto que diz se uma operação assíncrona foi um sucesso ou se deu falha. E async/await e apenas uma simples implementação de promessas.

Existe

alguma relação entre promises, async await e callbacks? Se

sim explique:

R: call-back e um passado como argumento para ser chamado depois

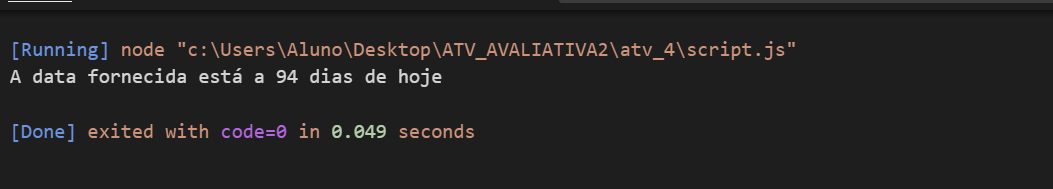
promise e objeto que representa algo que ainda não terminou, mas vai terminar no fututo(sucesso ou erro)

async/await forma moderna de escrever código assíncrono usando promise, então todas elas tem uma relação e são utilizadas juntas.

d- No contexto de JavaScript o que é event loop?

R:event loop e uma mecânica que faz o gerenciamento de execuções de tarefas, assim priorizando código síncrono e processando call-backs assíncronos em ordem, fazendo isso não bloqueia a thread principal.

**4**)



**5)**

Priemiro: O primeiro número a rodar foi o 8, pois ele é chamado no fim do código. "main()".

Segundo: número 3, ele é chamado dentro da função "comDelay()", que está sendo puxada pela main.

Terceiro: número 4, pois ele está dentro da função "comDelay()" também, mas ele respeita a ordem e vem depois do número 3.

Quarto: número 7, pois está dentro da main, mas após a função "comDelay()".

Quinto: o número 1, pois por se tratar de uma Promise, ele vem antes do setTime.

Sexto: o número 6, porque também é um Promise, mas vem depois do anterior.

Sétimo: número 5, porque no setTime ele foi definido para aparecer em um tempo menor do que o próximo número.

Oitavo: número 2, porque foi definido para ser exibido em 2 segundos (o mais demorado), por isso em último.