

Atividade 4 - Arraylist, Desestruturação e Espalhamento

/ /

fase-1-questao-1.js

→ alternativa (B)
→ A saída do código será 5, pois inicialmente a lista tem tamanho 4 pois contém 4 elementos, em seguida é executado 'lista.push(5)' adicionando um novo elemento (número 5) no final do array, aumentando seu tamanho em +1, ficando de tamanho 5.

fase-1-questao-2.js

→ A saída do código será ['maçã', 'banana'] (alternativa (c)), pois é utilizado 'frutas.pop()' onde o .pop() é responsável por remover o último elemento de um array removendo assim o elemento 'maça'.

fase-1-questao-3.js

→ A saída do código será [20, 30, 40] (alternativa (a)), pois é utilizado 'numeros.shift()' onde a função .shift() é responsável por remover o primeiro elemento de um array, removendo neste caso o elemento '10'.

fase-1-questao-4.js

→ A saída será [0, 1, 2, 3] (alternativa (b)) pois a função 'unshift()' adiciona um elemento no inicio do array que nesse caso foi utilizado para adicionar o elemento 0.

fase-1-questao-5.js

→ A saída será [6, 7] (alternativa (b)) pois é utilizado '.slice(1)' onde será criado um novo array a partir de um outro array, o número passado como parâmetro é a partir de que índice o novo array será criado.

fase-1-questao-6.js → alternativa(b)

→ A saída será [1, 4] pois está usando 'splice(1, 2)' onde irá remover os elementos contidos nos índices 1 e 2 do array.

fase-1-questao-7.js

→ A saída é '2' (alternativa c) pois é usado o 'indexOf('c')' que busca em qual índice está o elemento 'c', retornando 2.

fase-1-questao-8.js

→ A saída é [2, 4, 6] (alternativa b). Existe um array 'itens' e em seguida é criado um novo array utilizando a função 'map()'. O map cria um novo array a partir de um array anterior, neste caso, basta um array [4, 8, 12] e o map cria um novo array desses números divididos por 2 ('itens.map(x => x / 2)').

fase-1-questao-9.js

→ A saída é [9, 12] (alternativa c). Existe um array 'nums' contendo [3, 6, 9, 12], em seguida é utilizada a função 'filter' que cria um novo array a partir de outro array existente, o parâmetro utilizado foi ($m > 6$) para mostrar valores maiores que 6.

fase-1-questao-10.js

→ A saída é 12 (alternativa d). É utilizado o reduce onde o array será reduzido a um único valor, o parâmetro utilizado foi de soma todos os valores do array.

fase-1-questao-11.js

→ A saída é 'true' pois é utilizado 'in' para buscar um elemento. É buscado 'Lucas' que está no array.

fase 1 - questões 12.js

→ A saída é 'a-b-c' (alternativa c). O método join junta os elementos do array e entre eles coloca o parâmetro que foi passado.

fase 1 - questões 13.js

→ A saída é [1, 2, 3, 4, 5] (alternativa c). O método concat() junta um array a outro array e retorna um novo array.

fase 1 - questões 14.js

→ A saída é ['z', 'y', 'x'] (alternativa b). O método reverse() inverte a ordem dos elementos do array original.

fase 1 - questões 15.js

→ A saída é 3 (alternativa c). O método find() retorna o primeiro elemento do array que satisfaz a condição, nesse caso, a condição é um número maior que 2 e a primeira ocorrência no array é o 3.

fase 2 - questões 16.js

→ A saída é 37. O método filter() cria um novo array com todos os elementos que satisfazem a condição. O método map() cria um novo array com os resultados da chamada da função para cada elemento do array original. O método reduce() aplica uma função com um acumulador para o array se reduzido a um único elemento. Nesse caso, o array é filtrado para manter só os números ímpares, em seguida, cada um desses elementos é multiplicado por 3 e, por último, todos os elementos são somados e reduzidos a '37'.

fase 2 - questões 17.js

// código resumido

const arr = [10, 15, 22, 34, 45, 60];

let processados = 0;

```

for (let i = 0; i < arr.length; i++) {
    let m = arr[i];
    if (m % 3 === 0) {
        let metade = m / 2;
        processador += metade;
    }
}
console.log(processador);

```

fase-2-questao-18.js

→ O splice() modifica o array original, nesse caso, o splice remove 2 elementos apos o indice 1 e esses elementos removidos são guardados em 'n'. Uma forma de evitar isso é fazendo uma cópia do array, guardando ela em outra variável antes de manipula-la. Outra forma é usar o slice() que cria um novo array e não altera o original.

fase-2-questao-19.js

→ Para agrupar elementos do array é possivel usar o reduce().

```

let agrupador = pessoa.reduce((acumulador, pessoa) => {
    let cidade = pessoa.cidade;
    acumulador[cidade] = [];
    acumulador[cidade].push(pessoa);
    return acumulador;
});

```

fase-3-questao-20.js

→ O método find() retorna o primeiro elemento que satisfaz a condicão, nesse caso, retorna o primeiro 8 que aparece. O método filter retorna todos os elementos que satisfazem a condicão e retorna em formato de um novo array, nesse caso, retorna os dois 8 encontrados [8, 8]. O método some() retorna true ou false se pelo menos um dos elementos satisfaz a condição.

fase-3-questao-1.js

→ A saída é 'Ana { idade:28, cidade: 'SP' }' (alternativa a). É utilizada a desestruturação de objetos para retiver a propriedade 'nome' do objeto, depois é usado espalhamento (...) para manter o objeto sem perder as outras propriedades.

fase-3-questao-2.js

→ O resultado vai ser '1 3 [4,5]' (alternativa a). Na linha 2 temos 'const [a, , b, ...c] = numeros;', onde a pega o primeiro elemento (1), há um vazio entre as vírgulas que ignora o segundo elemento (2) e tem '...c' que pega o resto do array e espalha. Antes de '...c' tem 'b' que pega o terceiro elemento do array (3).

fase-3-questao-3.js

→ O resultado será 'R' (alternativa a). O código usa desestruturação para extrair a propriedade 'cidade'. Equivale a: let cidade = pessoa.endereco.cidade.

fase-3-questao-4.js

→ O resultado será '{a:1, b:3,c:4}' (alternativa b). Na linha 3 há 'const combinado = {...base, ...extra}', onde é espalhado os 2 objetos que se tornam um objeto só na variável 'combinado'. A propriedade b é 2 em 'base' e 3 em 'extra', → o que prevalece é o valor de 'extra' que é o valor depois e sobrescreve.

fase-3-questao-5.js

→ A saída será 25 (alternativa b). A função teste usa desestruturações para atribuir valores padrões com os valores máximos sejam passados. Quando a função é chamada 'teste({x:5})' é passado um valor para x e o y é usado o padrão

fase 3 - questao-6.js

→ alternativa c

- ~ O código imprime [1, 2, 3, 4, 5]. O código espalha os dois arrays `arr1` para formar um novo array e não faz adição mais um elemento

fase 3 - questao-7.js

- ~ A saída será 'escuro Arrial' (alternativa a). O valor da propriedade 'tema' do objeto é guardado numa nova variável chamada ('modo'), depois o "opções" espalha o restante do objeto. No final, modo vale "escuro" e opções vale { fonte: "Arrial", tamanho: 14}

fase 3 - questao-8.js

- ~ A saída é 32. A função soma recebe dois parâmetros e tem operador de espalhamento. Quando `resto` somar (...) valores ele expande o array 'valores' em seus elementos individuais. a é atribuído a 10, b é atribuído a 20, e o restante dos elementos do array vai para o array 'resto'. A função retorna 'a soma' de a e b + o comprimento do array resto.

fase 3 - questao-9.js

- ~ A saída será JS SQL (alternativa a). É desestruturado o objeto habilidades. h1 pega o valor 'JS', tem um espaço entre vírgulas que ignora o valor 'Python' e h3 pega o valor 'SQL'.

fase 3 - questao10.js

- ~ A saída é 99. O obj1 é espalhado no obj2. Ao modificar `obj2.b.x = 99`, é alterado o obj1 também.

fase-4-questao-11.js

→ A saída é `4 [5, 6]`. A desestruturação ignora o primeiro subarray `[1, 2]`, pega o segundo elemento (`4`) do segundo subarray `[3, 4]`, e o '... resto' coloca o último subarray `[5, 6]`.

fase-4-questao-12.js

→ A saída é escure { nome: 'Ana' } (alternativa a). A desestruturação extrai a propriedade 'Tema' do objeto 'preferencias', e o operador (...) espalha o restante das propriedades.

fase-4-questao-13.js

→ A saída é `3 99` (alternativa a). A função utiliza desestruturação para extrair a propriedade 'a' do objeto 'dados' e agrupa o restante das propriedades em um novo objeto resto. Resto é um novo objeto e não modifica dados.

fase-4-questao-14.js

→ A saída é `[2, 3, 1]`. A função fura usa desestruturações para pegar o primeiro elemento do array e agrupa o restante dos elementos em um novo array. O parâmetro y tem um valor padrão que cria um array contendo os elementos de resto seguidos por x.

fase-4-questao-15.js

→ A saída é `100 200`. Nesse caso, config2 é criado como uma cópia de config1. No entanto, a propriedade opcoes é um objeto animado. Quando altera o valor de z em config2.opcoes, isso também afeta config1.opcoes.