



Respostas da mini-teste PWeb 1

Aluno: Rhuan dos Santos Felix

Fase 1:

1. b), pois na lista já tem 4 elementos e depois adiciona mais 1
2. c), pois ao usar o método pop em reverso apaga o último da lista
3. a) [20, 30, 40], pois o método shift remove o primeiro elemento da lista
4. b), pois o método unshift(0) adiciona o número 0 no início do array
5. b), pois o método slice retorna uma cópia do vetor, indicando em que índice começar, neste caso, o índice 1.
6. b), pois o método splice corta os índices [1] e [2], criando uma nova lista com esses elementos e tirando eles da lista original, ficando apenas o primeiro e último índice.
7. c), pois a indexOf('c') vai retornar a posição que este elemento está, no caso: 2
8. b), pois o map itera o array itens e cada elemento será dividido por 2, retornando um novo array com esse cálculo
9. c), pois o filter itera todos e cria uma nova lista com os números maiores que 6
10. d), pois o reduce itera "arr", o acumulador "a" inicia em 0 e "b" é o valor corrente que vai somando com o acumulador.
11. a), pois o método includes verifica se o elemento tá no array, se tiver retorna True, se não, false
12. c), pois o join cria uma string e o que tiver dentro da parêntese, é o que vai separar os elementos, neste caso, o '-'.
13. c), pois ele (o método concat) vai juntar o array com [4,5]
14. b), pois o reverse inverte o orden das elementos
15. c), pois o método find retorna o primeiro elemento da lista que satisfaça a condição $n > 2$

Fase 2:



16. Primeiro a lista array é filtrada para os elementos que atendem a condição $m \% 2 == 1$ (os números que são dividíveis por 2 tem que ter resto 1), com isso é retornado um novo array que filtrado a partir deste array utiliza o map para multiplicar cada elemento por 3 e vai retornando uma lista com esses valores multiplicados por 3, para depois, a partir deste novo array em reduzir ele para um número; a redução (acumulador) começa com 10 e depois vai somando com os elementos corrente até chegar em 37, que é o resultado final.

17. Primeiro a lista é filtrada apenas com os elementos que não dividíveis por 3 e é gerado (retornado) um novo array com esse filtro. A partir deste novo array ele é transformado para um objeto a propriedade "original" é "metade", após isso, é retornado um novo array em que ele será reduzido pegando os valores da propriedade "metade" e vai somando com acumulador que começa em zero, tendo como resultado final 60.

18. O código modifica o array original quando pegamos "lista.splice", estamos pegando o endereço de memória e cortando a lista original, pegando os índices 1, 2 e criando um novo array com esses elementos, o antigo fica apenas com os índices 0 e 3, ficando [4, 16]. Uma forma de evitar essa mutação é criando uma nova variável e espalhar a lista original, uma outra forma é utilizando o método slice ao invés do splice.

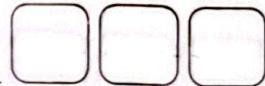
19.

20. O find retorna 8, pois ele pega apenas a primeira ocorrência que atende a condição e retorna; o filter retorna uma nova lista contendo os elementos que atendem a condição; a soma verifica se pelo menos um dos elementos atendem a condição, se sim, retorna true, se não, false.

Exercícios

1. a), pois destrutiva a lista com a propriedade nome, e a restante do objeto permanece que não espalhamento.

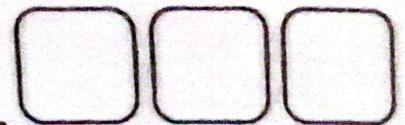
2. a), pois destrutiva a lista pegando o numero, e troca e a restante dos índices (sem espalhamento).



3. a), pois em `esta` desestrutura o objeto pessoa, pegando a propriedade "endereco", como também desestrutura o objeto dentro para pegar a propriedade "cidade".
4. b), pois quando expulsa os dois, ambos possuem a propriedade "b", fazendo com que a última execução sobrescreva as outras.
5. b), pois como eu passo apenas um dos parâmetros (`X`), ele vai substituir o valor padrão (`10`) e soma com o outro valor padrão da segunda propriedade.
6. c), pois expulsa os lista `arr1` e depois `arr2`, adicionando o `5` no final.
7. a), pois desestrutura o objeto pegando a propriedade `tema` e renomeando para `mota` e expulsa o restante do objeto.
8. b), pois a lista é expulsa e esse automaticamente é passado como parâmetro, fazendo com que "`a`" e "`b`" peguem respectivamente os dois primeiros índices e o resto pega o resto formando um novo array de dois elementos.
9. a), pois `arrayis` é desestruturado pegando a propriedade `hasildades` e desestruturando-o pegando a primeira, ultima índices.
10. b), pois não fazemos uma cópia completa do objeto, fazendo com que a propriedade `b` seja passada como referência.

Fase 4

11. b), pois destruímos a matriz, desestruturando a lista do segundo índice pegando o segundo elemento (`4`) e depois copiamos o resto da matriz com o espalhamento.
12. a), pois eu desestrutura o "obj" pegando a propriedade "dados" e desestruturando-o também, pegando a propriedade `preferencias` desestruturando-o também, pegando finalmente a propriedade `tema`, após isso, é expulso o restante do objeto dados (em que esse resto é a propriedade "mota").
13. a), pois quando passa "dados" como parâmetro de "preencher", ele desestrutura pegando "`a`" e espalhando (copiando) o resto, por isso que quando printa "dados.c" e "n[1].c" elas dão valores diferentes, pois a reatribuição que acontece dentro da função altera a cópia e não o original.
14. a), pois como não passa nenhum parâmetro para `y`, ele pega os mesmos nomes da desestruturação de primeiro parâmetro e inverte a ordem delas.



(S. c), pois na "config2" está espalhando a primeira propriedade e espalha incorretamente o objeto que está na propriedade "opções", fazendo com que quando atualiza o valor de "zoom" na "config2" não altere a "config1", pois estiver atualizando a propriedade copiada e não a original.