Explicações de resoluções - Funções em JS

Othavio Henrique Melancieri Amaral

1 — Neste exercício comecei criando a função “dobrar”, que recebe um número que é o que vai ser multiplicado por 2 e então retornando o resultado, depois o código recebe três números em variáveis diferentes e então vem a saída, que chama a função três vezes na mesma linha, uma para cada número que o usuário enviou como entrada.

2 — O mesmo estilo de algoritmo do exercício 1, criar a função, receber valores guardando-os em variáveis e a saída chamando a função criada. Na função que criei eu usei o operador embutido do *JavaScript*, o “**” para fazer cálculo de potenciação.

3 — Para esse aqui eu usei um método um tanto complicado, talvez teria um bem mais simples com matemática. A função “fundir” recebe os parâmetros “n1” e “n2” e os transforma em *strings* logo em seguida, guardando-os dentro das variáveis locais “primeiroDigito” e “ultimoDigito”, depois os valores dentro das variáveis são cortados com a função “Slice”, que eu aprendi na aula em que foram explicados *arrays*, porém eu não sabia até então que este método servia em valores *string* também, descobrindo por meio de pesquisas (sites especificados depois), as então *strings* são cortadas de uma forma que é pego apenas um dígito específico delas, já que foi feito para números de apenas dois dígitos, depois surge a variável local “n3” que recebe a soma das duas *strings* cortadas formando assim uma nova, que é então retornada. O resto do exercício segue o mesmo formato, entradas e com a chamada da função na própria saída. Sites usados para pesquisa:

- https://www.w3schools.com/jsref/jsref_slice_string.asp
- https://www.w3schools.com/jsref/jsref_tostring_number.asp

4 — Comecei mais uma vez pela criação da função, “calcularMedia” recebe quatro parâmetros, sendo um para cada nota e um para definir o tipo de cálculo, ela vai seguir rotas de processamento diferentes para retornar seus valores baseados no argumento “letra”, coloquei também uma mensagem de erro para ficar mais completo (caso “letra” não for A nem P). Depois é recebido as entradas de usuário e a saída chama a função, mesmo processo padrão. Site usado para relembrar o cálculo de média ponderada:

- https://brasilescola.uol.com.br/matematica/media-ponderada.htm#:~:text=Para%20calcular%20a%20m%C3%A9dia%20ponderada,soma%20pela%20soma%20dos%20pesos.

5 — Neste exercício a função “calculadoraSimples” recebe 3 parâmetros, os dois valores e a operação matemática a efetuar com eles, a função verifica qual é o dígito

operador e decide qual caminho de processamento e retorno seguir, se o usuário digitou qualquer coisa que não é reconhecida a função retorna uma mensagem de erro. O algoritmo depois de definir a função recebe os três valores do usuário e faz uma saída personalizada.

6 — Declarei a função “comentarSobreDiasRestantes” que recebe o argumento correspondente ao total de dias que restam no ano, assim a partir dele a função determina com base nas condições indicadas pelo exercício qual mensagem exibir, as mensagens são exibidas pela própria função, ela não retorna valores (como indicado).

7 — É só a definição de uma função que imprime uma mensagem no console. Depois de ser definida ela é chamada.

8 — A mesma lógica do exercício 7.

9 — Minha função recebe um número como argumento ele então é transformado em *string* e é retornada a propriedade *length* dele, desse modo que eu fiz, ele poderia funcionar com uma *string* recebida do usuário também. A propriedade *length* foi descobrida em pesquisas durante a outra lista de exercícios.

10 — Aqui na função eu tive que transformar o número recebido em *string* e usar um método de inverter *strings* para poder inverter um número. A linha “return numero.toString().split('').reverse().join('')” funciona da seguinte maneira: aqui a função retorna o número transformado em *string*, separado pelo “.split()”, que retorna um vetor com os dígitos da *string* como elementos, então “.reverse()” o inverte e logo depois “.join()” junta a versão invertida da lista de volta em uma *string*. Site consultado para pesquisa:

- https://pt.stackoverflow.com/questions/5943/como-inverter-uma-string-em-javascript

11 — Este exercício realmente me testou, fez “sair fumaça da cabeça”, e eu não consegui pensar sozinho em um método de adquirir um número aleatório e tive que me basear em um modelo que encontrei online em:

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

O modelo gera um número aleatório que vai entre o mínimo e o máximo, podendo ser igual ao mínimo, mas nunca igual ao máximo, o modelo originalmente gerava com decimais e eu tive que mudar isso para se encaixar no contexto de jogar um dado. O código principal funciona em um laço com uma quantidade bem exagerada, porém necessária de condicionais. Antes do laço começar, meu algoritmo sabe que é a primeira partida do jogador, isso por meio da variável *boolean* “primeiraRodada”, assim dá para aplicar as condições da primeira rodada descritas no exercício, depois é só

alternar o valor da variável e assim se analisa as condições de quando não é a primeira partida. Quando “primeiraRodada” carrega o valor de falso, o código “cai no else do else”, nas últimas condicionais possíveis, verificando apenas se o usuário tira 7 ou o ponto.

12 — A minha função ”escreverData” recebe como parâmetro “data”, que é uma *string* no formato DD/MM/AAAA (D para dia, M para mês e A para ano), a função se aproveita do método embutido “.slice()” (referência a ele nos sites usados para pesquisa do exercício 3) separando o dia, mês e ano em variáveis locais, depois de separar, inicializei uma variável *string* vazia para usar mais tarde, então vem um bloco de condicional que valida o mês como número e decide qual mês a variável “mesPorExtenso” vai se tornar. Se não for nenhum dos números esperados a função retorna “null”, após isso retorna uma *string* da maneira esperada. O código recebe a entrada de usuário e a usa como argumento no momento de chamar a função na saída.

13 — Ao começar a fazer este exercício eu já havia conhecido as funções “.split()” e “Math.random()” por pesquisas anteriores, no entanto eu tive que conhecer o “splice()” para fazer este exercício. A minha função funciona da seguinte forma: recebendo o argumento com a *string* a processar, ela primeiro separa todas as letras da palavra em um *array* por meio da função “split()” e então se inicia um laço “for” no estilo de posição em lista; dentro do laço primeiramente é verificado se a letra atual é um espaço ou uma tabulação, se for, é usada a função “splice()” para deletar um elemento na posição atual e a iteração para e pula para o próximo elemento (com a palavra-chave “continue”, que aprendi em Python); se um elemento passar por essa condição tendo ela como falsa, ele então passa pelo meu processo de aleatoriedade, em que se a função “Math.random()” (função que retorna um valor de 0 a 1) retornar um valor menor ou igual a meio, a letra atual passando pelo processo avança uma posição na lista, para fazer isso, primeiro salvei a letra em uma variável, então deletei ela com “splice()” e adicionei ela de volta uma posição à frente com a função “splice()” de novo, depois se o elemento tiver rodado o código até aqui ele verifica se “Math.random()” agora retorna um valor menor ou igual a 0.25, se sim, agora a letra fica maiúscula através de receber ela mesma, mas com o método “toUpperCase()”. Agora se não foi igual ou menor que meio em primeiro lugar, o código verifica se foi agora maior que 0.5, se sim ele faz o processo contrário com a letra, ele volta uma posição dela, da mesma maneira de que o código teria feito para avançar, agora é verificado se “Math.random()” dá um valor maior ou igual a 0.75, se sim, a letra fica minúscula por meio do método “toLowerCase()”, que faz a mesma coisa de “toUpperCase()” porém o processo contrário. Após o laço, ou seja, o processo todo de embaralhar, as letras bagunçadas são juntadas em outra palavra que a função retorna. Páginas web visitadas para pesquisa:

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String/toLowerCase
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String/toUpperCase

14 — Para este exercício eu fui só lendo a equação e montando o código dentro da função com seus respectivos argumentos, ela começa calculando o delta e depois o argumento “b” para a visão da função é multiplicado por –1, assim o delta ao usar “b” para seu cálculo usa-o enquanto ele está em sua forma normal, antes de ser multiplicado. Mais tarde é calculada a raiz de delta e o resultado é guardado em uma variável dentro da função onde tudo está acontecendo, para calcular a raiz quadrada eu utilizei o funcionamento embutido do operador “**” do JavaScript, usando-o junto com “0.5”, depois de seguir toda a lógica da fórmula, a função retorna o “x1” e o “x2” dentro de uma lista, já que funções só podem retornar um valor. O código recebe em variáveis as entradas do usuário e as usa na hora da saída quando a função é chamada, a saída é bem descritiva. Ao fazer este exercício eu fiquei um pouco confuso já que os resultados estavam dando “NaN”, ou seja, não números, mas a partir de uma calculadora eu concluí que isso era porque delta era negativo, ou seja, não existia as raízes reais. Calculadora usada para auxílio:

- https://bhaskaracalculator.netlify.app/

15 — O processo de fazer esse exercício foi parecido com o anterior, eu fui só montando a função com base na lógica de como se fosse resolver a fórmula manualmente, seguindo a mesma ordem. É como se fosse fazer um cálculo desses “na mão”, porém ao invés de substituir os termos por números você apenas vai seguindo a lógica deles através de variáveis e operações.

16 — O mesmo processo do exercício 7 e 8. Deixei uma mensagem honesta.