Basic Algorithm Scripting / JavaScript

Converter Celsius para Fahrenheit

A fórmula para converter de Celsius para Fahrenheit é a temperatura em Celsius vezes `9/5`, mais `32`.

Você tem uma variável `celsius` representando uma temperatura em Celsius. Use a variável `fahrenheit` já definida e atribua a ela a temperatura equivalente à temperatura Celsius indicada. Use a fórmula mencionada acima para ajudar a converter a temperatura em Celsius para Fahrenheit.

```
function convertToF(celsius)

{ let fahrenheit = celsius * (9 / 5) + 32; return
fahrenheit; } // Change the inputs below to test your code
convertToF(30);
```

Explicação :

Declare a variável fahrenheit.

Certifique-se de que a ordem correta das operações aritméticas seja seguida usando parênteses (()) quando necessário.

Inverter uma string

Inverta a string fornecida.

Você pode ter que transformar a string em um array antes de poder inverter.

Seu resultado deve ser uma string.

```
function reverseString(str) {

 let reversedStr = ""; for (let i = str.length - 1; i >= 0;
i--) { reversedStr += str[i]; } return reversedStr; }
function reverseString(str) { let reversedStr = ""; for (let
```

```
i = str.length - 1; i >= 0; i--) { reversedStr += str[i]; }
return reversedStr; }
```

Explicação : Começando no último caractere da string passada para a função, você constrói uma nova string reversedStr a partir de str.

Durante cada iteração do loop for, reversedStr é concatenado consigo mesmo e com o caractere atual.

Por fim, você retorna o valor final de reversedStr.

Solução 2 :

```
function reverseString(str) { return str .split("")
.reverse() .join(""); }
```

Explicação : Nosso objetivo é pegar a entrada, str, e devolvê-la ao contrário. Nosso primeiro passo é dividir a string por caracteres usando split(''). Observe que não deixamos nada entre aspas simples, isso diz à função para dividir a string por cada caractere.

Usar a função split() transformará nossa string em um array de caracteres, lembre-se disso enquanto avançamos.

Em seguida, encadeamos a função reverse(), que pega nosso array de caracteres e os reverte.

Finalmente, nós encadeamos join('') para juntar nossos caracteres novamente em uma string. Observe mais uma vez que não deixamos espaços no argumento para join, isso garante que a matriz de caracteres seja unida novamente por cada caractere.

Fatorar um número

Retorne o fatorial do inteiro fornecido.

Se o inteiro é representado com a letra `n`, o fatorial é o produto de todos os inteiros positivos menores ou iguais a `n`.

Fatoriais são frequentemente representados com a notação abreviada `n!`

Por exemplo: `5! = 1 * 2 * 3 * 4 * 5 = 120`

Apenas números inteiros maiores ou iguais a zero serão fornecidos para a função.

```
function factorialize(num)
{ let product = 1; for (let i = 2; i <= num; i++)
{ product *= i; }
 return product; }
 factorialize(5);
```

Como os valores de retorno da função serão sempre maiores ou iguais a 1, product é inicializado em um. Para o caso em que o número é 0, a condição do loop for será falsa, mas como product é inicializado como 1, ele terá o valor correto quando a instrução return for executada.

Para todos os números passados para a função que são maiores que um, o loop for simples incrementará i em um a cada iteração e recalculará o produto até o valor num.

Encontrar a palavra mais longa em uma string

Retornar o comprimento da palavra mais longa na frase fornecida.

Sua resposta deve ser um número.

```
function findLongestWordLength(str)
{ let words = str.split(' ');
 let maxLength = 0; for (let i = 0; i < words.length; i++)
{ if (words[i].length > maxLength)
{ maxLength = words[i].length; }
}
 return maxLength; }
```

Pegue a string e converta-a em uma matriz de palavras. Declare uma variável para acompanhar o comprimento máximo e faça um loop de 0 até o comprimento da matriz de palavras.

Em seguida, verifique a palavra mais longa comparando a palavra atual com a anterior e armazenando a nova palavra mais longa. No final do loop apenas retorne o valor numérico da variável maxLength.

Retornar os maiores números em arrays

Retorna um array que consiste no maior número de cada sub-array fornecido. Por simplicidade, o array fornecido conterá exatamente 4 sub-arrays.

Lembre-se: você pode iterar através de um array com um loop simples, e acesse cada membro com a sintaxe de array `arr[i]`.

```
function largestOfFour(arr) {
  const results = [];
  for (let i = 0; i < arr.length; i++) {
    let largestNumber = arr[i][0];
```

```
    for (let j = 1; j < arr[i].length; j++) {
      if (arr[i][j] > largestNumber) {
        largestNumber = arr[i][j];
      }
    }
    results[i] = largestNumber;
  }


  return results;
}


largestOfFour([[4, 5, 1, 3], [13, 27, 18, 26], [32, 35, 37,
39], [1000, 1001, 857, 1]]);
```

Explicação : Crie uma variável para armazenar os resultados como uma matriz.

Crie um loop externo para iterar pela matriz externa.

Crie uma segunda variável para conter o maior número e inicialize-a com o primeiro número. Isso deve estar fora de um loop interno para que não seja reatribuído até encontrarmos um número maior.

Crie o referido loop interno para trabalhar com os sub-matrizes.

Verifique se o elemento da submatriz é maior que o maior número armazenado atualmente. Em caso afirmativo, atualize o número na variável.

Após o loop interno, salve o maior número na posição correspondente dentro da matriz de resultados.

E, finalmente, retorne o referido array.

Confirmar o final

Verifique se uma string (primeiro argumento, `str`) termina com a sequência de caracteres de destino fornecida (segundo argumento, `target`).

Este desafio *pode ser resolvido* com o método `.endsWith()`, que foi introduzido na ES2015. Para este desafio, entretanto, gostaríamos que você usasse um dos métodos de substring JavaScript.

```
function confirmEnding(str, target) {
   return str.slice(str.length - target.length) === target;
}

confirmEnding("Bastian", "n");
```

Explicação : Primeiro usamos o método slice para copiar a string.

Para obter os últimos caracteres em str equivalentes ao comprimento do alvo, usamos o método slice.

O primeiro parâmetro dentro do método slice é o índice inicial e o segundo parâmetro seria o índice final.

Por exemplo, str.slice(10, 17) retornaria dar-me.

Neste caso, incluímos apenas um parâmetro que copiará tudo do índice inicial.

Subtraímos o comprimento de str e o comprimento de alvo, dessa forma, obteremos os últimos caracteres restantes equivalentes ao comprimento do alvo.

Por fim, comparamos o resultado de retorno do slice ao target e verificamos se eles possuem os mesmos caracteres.

Repetir uma string Repetir uma string

Repita uma string passada `str` (primeiro argumento), `num` vezes (segundo argumento). Retorne uma string vazia se `num` não for um número positivo. Para o propósito do desafio, *Não* use o método integrado `.repeat()`.

```
function repeatStringNumTimes(str, num) {
  let accumulatedStr = "";

 for (let i = 0; i < num; i++) {
    accumulatedStr += str;
  }

 return accumulatedStr;
}
repeatStringNumTimes("abc", 3);
```

Crie uma variável de string vazia para armazenar a palavra repetida.

Use um loop for ou for loop para repetir o código quantas vezes forem necessárias de acordo com num

Em seguida, adicionamos a string à variável criada na etapa um dentro do loop.

No final do loop, retorne a variável para a palavra repetida.

Truncar uma string

Trunque uma string (primeiro argumento) se ela for maior que o comprimento máximo da string (segundo argumento). Retorne a string truncada com `...` (reticências) ao final.

```
function truncateString(str, num) {
   if (str.length > num) {
    return str.slice(0, num) + "...";
  } else {
    return str;
  }
}
```

```
truncateString("A-tisket a-tasket A green and yellow
basket", 8);
```

Explicação = Começamos com uma simples instrução if para determinar um dos dois resultados...

Se o comprimento da nossa string for maior que o número que desejamos truncá-la, retornaremos uma fatia de nossa string começando no caractere 0 e terminando em num. Em seguida, anexamos nosso '...' ao final da string.

No entanto, se a situação acima não for verdadeira, isso significa que nosso comprimento de string é menor que nosso número de truncamento. Portanto, podemos apenas retornar a string.

Achar não é roubar

Crie uma função que olhe através do array `arr` e retorne o primeiro elemento dentro do array que passe pelo 'teste de verdade' ('truth test'). Isso significa que, dado um elemento `x`, o 'teste de verdade' é verdadeiro se `func(x)` é `true`. Se nenhum elemento passa no test, retorna `undefined`.

```
function findElement(arr, func) {
  let num = 0;


  for (let i = 0; i < arr.length; i++) {
    num = arr[i];
```

```
    if (func(num)) {
      return num;
    }
  }

  return undefined;
}
findElement([1, 2, 3, 4], num => num % 2 === 0);
```

Explicação : Challenge nos pede para olhar através de array. Isso é feito usando um loop for.

A variável num está sendo passada para a função, então nós a configuramos para cada índice em nosso array.

A função pré-definida já verifica cada número para nós, então se for “true”, retornamos esse número.

Se nenhum dos números no array passar no teste da função, retornamos undefined.

Identificar verdadeiro ou falso

Verifique se um valor é classificado como booleano primitivo. Retorna `true` ou `false`.

Os booleanos primitivos são `true` e `false`.

Obs : Typeof especifica qual o tipo de operador !!!!

```
function booWho(bool) {
  return typeof bool === "boolean";
}
```

```
booWho(null);
```

Usa o operador typeof para verificar se a variável é booleana. Se for, retornará true. Caso contrário, se for de qualquer outro tipo, retornará false.

Capitalizar o título de uma frase

Retorne a string fornecida com a primeira letra de cada palavra em letra maiúscula. Certifique-se de que o resto da palavra esteja em letras minúsculas.

Para o propósito desse exercício, você também deve capitalizar as palavras conectoras como `the` e `of`.

```
function titleCase(str) {
  const newTitle = str.split(" ");
  const updatedTitle = [];
  for (let st in newTitle) {
    updatedTitle[st] = newTitle[st][0].toUpperCase() + newTitle[st].slice(1).toLowerCase();
  }
  return updatedTitle.join(" ");


}


titleCase("I'm a little tea pot");
```

Explicação : Divida a string por espaços em branco e crie uma variável para rastrear o título atualizado. Em seguida, usamos um loop para transformar o primeiro caractere da palavra em maiúscula e o restante em minúscula. criando uma string concatenada composta pela palavra inteira em minúsculas com o primeiro caractere substituído por sua letra maiúscula.

Fatiar e emendar

Você está recebendo dois arrays e um índice.

Copie cada elemento da primeira matriz para a segunda matriz, em ordem.

Comece inserindo elementos no índice `n` do segundo array.

Retorne o array resultante. Os arrays recebidos devem permanecer os mesmos após a função ser executada.

```
function frankenSplice(arr1, arr2, n) {
  // It's alive. It's alive!
  let localArray = arr2.slice();
  for (let i = 0; i < arr1.length; i++) {
    localArray.splice(n, 0, arr1[i]);
    n++;
  }
  return localArray;
}
```

Explicação : Nosso objetivo é pegar todos os elementos de arr1 e inseri-los em arr2 começando com a posição de índice n. Ao mesmo tempo, devemos garantir que nem arr nem arr2 tenham sofrido mutação.

Usando a função slice() podemos criar uma réplica exata de arr2 e atribuir o resultado da operação a uma variável, localArray.

Agora que temos um array no qual podemos mudar, podemos iterar por todos os itens do primeiro array. Para cada item no primeiro array podemos usar a função splice() para inserir o item no índice n de localArray.

Incrementamos o índice n em um. Isso garantirá que cada item do arr1 seja inserido em localArray na posição de índice adequada.

Por fim, retornamos o localArray e encerramos a função.

Remover falsos

Remover todos os valores falsos de um array.

Valores falsos (falsy) em JavaScript são `false`, `null`, `0`, `""`, `undefined`, e `NaN`.

Dica: tente converter cada valor para um booleano.

```
function bouncer(arr) {
  let newArray = [];
  for (let i = 0; i < arr.length; i++) {
    if (arr[i]) newArray.push(arr[i]);
  }
  return newArray;
}
```

Explicação : Criamos um novo array vazio.

Usamos um ciclo for para iterar sobre todos os elementos do array fornecido (arr).

Usamos a instrução if para verificar se o elemento atual é verdadeiro 2,4k ou falso 3,6k.

Se o elemento for verdadeiro, nós o enviamos para o novo array (newArray). Isso resulta no novo array (newArray) contendo apenas elementos verdadeiros.

Retornamos o novo array (newArray).

Encontrar o local em um array

Retorne o menor índice em que um valor (segundo argumento) deve ser inserido no array (primeiro argumento) assim que tenha sido ordenado. O valor retornado deve ser um número.

Por exemplo, `getIndexToIns([1,2,3,4], 1.5)` deve retornar `1` porque é maior que `1` (índice 0), mas menor que `2` (índice 1).

Da mesma forma, `getIndexToIns([20,3,5], 19)` deve retornar `2` pois uma vez que o array foi sorteado se parecerá com `[3,5,20]` e `19` é menor que `20` (índice 2) e maior que `5` (índice 1).

```
function getIndexToIns(arr, num) {
  arr.sort((a, b) => a - b);


  for (let i = 0; i < arr.length; i++) {
    if (arr[i] >= num)
      return i;
  }


  return arr.length;
}
getIndexToIns([40, 60], 50);
```

Explicando : Primeiro, classifico o array usando .sort(callbackFunction) para classificá-lo do menor para o maior, da esquerda para a direita.

Então eu uso um loop for para comparar os itens na matriz a partir do menor. Quando um item na matriz é maior que o número com o qual estamos comparando, retornamos o índice.

Identificar mutações

Retorne `true` se a string no primeiro elemento do array contém todas as letras da string no segundo elemento do array.

Por exemplo, `["hello", "Hello"]`, deve retornar `true` porque todas as letras na segunda string estão presentes no primeiro, ignorando diferenças entre maiúsculos e minúsculos.

Os argumentos `["hello","hey"]` deve retornar `false` porque a string `hello` não contém o caracter `y`.

Por último, `["Alien", "line"]`, deve retornar `true` porque todas as letras em `line` estão presente em `Alien`.

```
function mutation(arr) {
  let test = arr[1].toLowerCase();
  let target = arr[0].toLowerCase();
  for (let i = 0; i < test.length; i++) {
    if (target.indexOf(test[i]) < 0) return false;
  }
  return true;
}
mutation(["hello", "hey"]);
```

Explicando : Primeiro fazemos as duas strings no array em minúsculas. teste irá conter o que estamos procurando no alvo.

Em seguida, percorremos nossos caracteres de teste e, se algum deles não for encontrado, retornaremos false.

Se todos forem encontrados, o loop terminará sem retornar nada e retornaremos true.

Dividir e agrupar arrays

Escreva uma função que divida um array (primeiro argumento) em grupos com o comprimento de `size` (segundo argumento) e os retorne como um array bidimensional.

```
function chunkArrayInGroups(arr, size) {
  // Break it up.
  let newArr = [];
  for (let i = 0; i < arr.length; i += size) {
    newArr.push(arr.slice(i, i + size));
  }
  return newArr;
}
chunkArrayInGroups(["a", "b", "c", "d"], 2);
```

Explicando :

Primeiro, criamos um array vazio arr2 onde armazenaremos nossos ‘pedaços’.

O loop for começa em zero, aumenta de tamanho a cada vez que passa pelo loop e para quando atinge arr.length.

Observe que esse loop for não percorre arr. Em vez disso, estamos usando o loop para gerar números que podemos usar como índices para dividir o array nos locais corretos.

Dentro do nosso loop, criamos cada pedaço usando arr.slice(i, i+size) e adicionamos esse valor a arr2 com arr2.push().

Por fim, retornamos o valor de arr2.