Projeto Marvin

FASE 06

Até quando preciso fazer isso?

Capítulo I

Instruções

- Somente esse material serve como referência. Não acredite em rumores.
- Fique atento, esse material pode mudar a qualquer hora antes da submissão.
- Os exercícios são cuidadosamente organizados em ordem de dificuldade, do mais fácil para o mais difícil. Esse mesmo raciocínio é aplicado para as correções, portanto não adianta completar om exercício mais difícil se os anteriores estão errados.
- Tenha certeza que possui as permições necessárias para acessar arquivos e comandos.
- Você precisa seguir os procedimentos de envio para todas as atividades.
- Suas atividades serão corrigidas e avaliadas pelos seus colegas.
- Exercícios em Shell devem ser funcionais através do /bin/bash
- Você não pode deixar nenhum outro arquivo na pasta, a não ser o que foi expressamente pedido.
- Tem uma pergunta? Pergunte ao seu colega à direita.Caso contrário, tente o seu colega à esquerda.
- Seu guia de referência se chama Google / man / Internet
- Procure por conteúdo no Youtube para introduzir ao assunto.
- Examine os exemplos cuidadosamente. Eles estão sempre corretos e podem pedir por detalhes que não foram mencionados explicitamente na atividade.
- A função não deve ser chamada no arquivo, apenas definida.
- Os arquivos devem ter compatibilidade
- “No início, o universo foi criado. Isso irritou profundamente muitas pessoas e, no geral, foi encarado como uma péssima idéia”.

Atenção

Evite travar seu navegador com laços infinitos

```
for (var i = 0; i > -3; i++) {
     //seu código
}
```

ao ser executado vai travar o navegador, pois nunca atende uma condição para finalizar.

Por esse motivo, recomendo fortemente utilizar o navegador Google Chrome para fazer essa etapa.

Caso trave, recomendo também abrir o gerenciador de tarefas do navegador, pois assim poderá finalizar o processo da aba travada e recarregar o editor. A tecla de atalho é shift + esc.

Capítulo II

Exercício 0: Aprendendo a contar

_M	Exercício: 00
Aprendendo a contar	
Diretório de submissão: fase06/ex00	
Arquivo a ser submetido: entreNumeros.js	

- Escreva uma função chamada “entreNumeros”;
- A função sempre recebe dois números inteiros como argumentos, e exibe no console os número no intervalo entre eles.

```
function entreNumeros(min, max) {
...
}
entreNumeros(14, 17)
```

ao ser executado:

```
$>14
$>15
$>16
$>17
```

Capítulo III

Exercício 01: Até a origem

_M	Exercício: 01
Até a origem	
Diretório de submissão: fase06/ex01	
Arquivo a ser submetido: ateZero.js	

- Escreva uma função chamada “ateZero”, que sempre recebe um número como argumento;
- A função deve retornar uma array com todos os números até o zero.
- Caso o número recebido seja negativo, deve retornar os números a partir dele até o 0;
- Caso o número recebido seja positivo, deve retornar os números a partir de 0 até ele.

Capítulo IV

Exercício 02: Quer dividir?

_M	Exercício: 02
Quer dividir?	
Diretório de submissão: fase06/ex02/	
Arquivo a ser submetido: por42.js	

- Escreva uma função chamada “por42”, que sempre recebe dois números como argumento;
- A função deve retornar o segundo múltiplo de 42 entre esses números (inclusos);
- Se não encontrar o número, deve retornar ***false*** e exibir “Não encontrado” no console;

```
function por42(num1, num2) {
...
}
conosole.log(por42(1, 84))
```

ao ser executado:

```
$> 84
```

Capítulo V

Exercício 03: Esse parente eu conheço

_M	Exercício: 03
Esse parente eu conheço	
Diretório de submissão: fase06/ex03/	
Arquivo a ser submetido: primo.js	

- Crie um função chamada “primo”, que sempre recebe um número como argumento;
- A função deve avaliar se o número é primo e retornar ***“Sim”*** ou ***“Não”*** propriamente;

Capítulo VI

Exercício 04: Muitos e muitos numeros

_M	Exercício: 04
Muitos e muitos numeros	
Diretório de submissão: fase06/ex04/	
Arquivo a ser submetido: somaPares.js	

- Escreva uma função chamada “somaPares”. Essa função sempre vai receber dois números - inteiros ou não - como argumento;
- A função deve retornar a soma de todos os números pares entre esses números (inclusos)

```
function somaPares(num1, num2) {
...
}
conosole.log(somaPares(1.33333, 4))
```

ao ser executado:

```
$> 6
```

Capítulo VII

Exercício 05: X O que que é? - é Xuxa!

_M	Exercício: 05
X O que que é? - é Xuxa!	
Diretório de submissão: fase06/ex05/	
Arquivo a ser submetido: x.js	

- Escreva uma função chamada “x”, que sempre recebe uma string como argumento;
- A função deve retornar o número de caracteres “x” (maiúsculo ou minúsculo) que o argumento tem;

Capítulo VIII

Exercício 06: É ou não é

_M	Exercício: 06
É ou não é	
Diretório de submissão: fase06/ex06/	
Arquivo a ser submetido: booleano.js	

- Escreva uma função chamada "**booleanos**", que sempre recebe uma array como argumento;
- A função deve retornar uma array apenas com todos os elementos, exceto do tipo **booleano** da array recebida, na mesma ordem

```
function booleanos(array) {
...
}
conosole.log(booleanos([true, 1, -10, "carro", false, 0]))
```

ao ser executado:

```
$> [1, -10, "carro", 0]
```

Capítulo IX

Exercício 07: Tudo ao contrátio

_M	Exercício: 07
Tudo ao contrátio	
Diretório de submissão: fase06/ex07/	
Arquivo a ser submetido: inverter.js	

- Escreva uma função chamada “**inverter**”, que sempre recebe uma array ou sttring como argumento;
- A função deve retornar o argumento, em ordem inversa;

```
function inverter(array) {
...
}
conosole.log(inverter(“peixe”))
```

ao ser executado:

```
$>exiep
```