



Projeto Marvin

FASE 04

Mundo condicional

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 um exercício mais difícil se os anteriores estão errados.
- Tenha certeza que possui as permissões 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.
- “No início, o universo foi criado. Isso irritou profundamente muitas pessoas e, no geral, foi encarado como uma péssima idéia”.
- Os arquivos devem ter compatibilidade

Capítulo II

Exercício 0: Aquelas cinco letras

_M	Exercício: 00
Aquelas cinco letras	
Diretório de submissão: fase04/ex00	
Arquivo a ser submetido: vogal.js	

- Escreva uma função chamada “vogal”, que sempre recebe uma letra minúscula como argumento
- A função deve avaliar se a letra recebida como argumento é uma vogal ou não e então retornar **true** ou **false** corretamente.

```
function vogal(letra) {  
  ...  
}  
console.log(vogal("a"))
```

ao ser executado:

```
$>true
```

```
function vogal(letra) {  
  ...  
}  
console.log(vogal("y"))
```

ao ser executado:

```
$>>false
```

Capítulo III

Exercício 01: Os números pares

_M	Exercício: 01
Os números pares	
Diretório de submissão: fase04/ex01	
Arquivo a ser submetido: pares.js	

- Escreva uma função chamada "pares", que sempre recebe dois números como argumento
- Caso **ambos** os números sejam pares, deve retornar ***"Os números são pares"***.
- Caso contrário, deve retornar ***"Os números não são pares"***.

Capítulo IV

Exercício 02: Vogal ou consoante?

_M	Exercício: 02
Vogal ou consoante?	
Diretório de submissão: fase04/ex02/	
Arquivo a ser submetido: vogalOuConsoante.js	

- Escreva uma função chamada “vogalOuconsoante”, que sempre recebe uma letra (maiúscula ou minúscula) como argumento
- Deve retornar **“Vogal”** caso a letra recebida seja uma vogal, ou **“Consoante”** caso seja uma consoante.

```
function vogalOuConsoante(letra) {  
  ...  
}  
console.log(vogalOuConsoante("v"))
```

ao ser executado:

```
$>Consoante
```

Capítulo V

Exercício 03: Os pares e os ímpares

_M	Exercício: 03
Os pares e os Ímpares	
Diretório de submissão: fase04/ex03/	
Arquivo a ser submetido: parOuImpar.js	

- Crie um função chamada “parOuImpar”, que sempre recebe um número como argumento.
- A função deve retornar **“Par”** caso o argumento seja um número par ou **“Ímpar”** caso o argumento seja um número ímpar.

Capítulo VI

Exercício 04: O começo do alfabeto

<code>_M</code>	Exercício: 04
O começo do alfabeto	
Diretório de submissão: fase04/ex04/	
Arquivo a ser submetido: abc.js	

- Escreva uma função chamada “abc”. Essa função sempre vai receber uma string como argumento.
- Caso a string comece com uma das três primeiras letras do alfabeto, deve retornar **true**. Caso contrário, deve retornar **false**.

Capítulo VII

Exercício 05: De dois em dois, começando do zero

_M	Exercício: 05
De dois em dois, começando do zero	
Diretório de submissão: fase04/ex05/	
Arquivo a ser submetido: parEPositivo.js	

- Escreva uma função chamada “parEPositivo”, que sempre recebe um número como argumento.
- A função deve retornar **“Sim”** caso o número recebido seja par e positivo, ou **“Não”** caso contrário.

Capítulo VIII

Exercício 06: Estamos no futuro, Marty

<code>_M</code>	Exercício: 06
Estamos no futuro, Marty	
Diretório de submissão: fase04/ex06/	
Arquivo a ser submetido: passadoOuFuturo.js	

- Escreva uma função chamada “passadoOuFuturo”, que sempre recebe um número correspondente a uma data válida em [EPOCH](#), incluindo os milissegundos;
- A função deve retornar “Passado” se a data recebida for antes do dia 21 de outubro de 2015;
- Caso o argumento seja igual ou posterior a essa data, deve retornar “Futuro”;

Capítulo IX

Exercício 07: Onde está meu número

_M	Exercício: 07
Onde está meu número	
Diretório de submissão: fase04/ex07/	
Arquivo a ser submetido: acharNumero.js	

- Escreva uma função chamada “acharNumero”, que sempre recebe um número como argumento;
- A função deve seguir a seguinte lógica:
 - Retotnar “a” caso o número seja menor do que 5;
 - Retornar “b” caso seja entre 5 (incluso) e 10 (incluso);
 - Retornar “c” caso seja entre 10 e 100 (incluso);
 - Retornar “d” caso seja maior do que 100;

```
function acharNumero(num) {  
  ...  
}  
console.log(acharNumero(5))
```

ao ser executado:

```
$>b
```

```
function acharNumero(num) {  
  ...  
}  
console.log(acharNumero(100))
```

ao ser executado:

```
$>c
```

Capítulo X

Exercício 08: Huston, temos um problema

_M	Exercício: 08
Huston, temos um problema	
Diretório de submissão: fase04/ex08/	
Arquivo a ser submetido: alfabetoMilitar.js	

- Escreva uma função chamada “alfabetoMilitar”, que recebe uma vogal como argumento;
- A função deve retornar o equivalente no alfabeto militar ([disponível aqui](#)).
- Caso o argumento recebido não seja uma vogal, deve retornar **false**;