
mdnm Exercícios


3dabbbe6 days ago🕒 History

..

README.md

Exercícios


6 days ago

index.js

Exercícios

6 days ago

⋮ README.md



## Exercício de casa 🏠

### Carro

Nesta sequência de exercícios criaremos um objeto carro que possuirá 6 membros, sendo 2 propriedades e *pele menos* 4 métodos. As propriedades deverão ser um boolean chamado **ligado** e um number chamado **velocidade**. Além dos indicadores de ligado/desligado e de velocidade, este carro deverá possuir métodos para **ligar**, **desligar**, **acelerar** e **desacelerar**.

**Vamos então aos passos para completar o exercício!!!**

a) Criar o objeto carro com apenas as duas propriedades (neste primeiro passo não é necessário criar os métodos):

- ligado** (boolean) : que de deverá ser inicializada com valor **false** (desligado).
- velocidade** (number) : que deverá ser inicializada com valor **0** (zero).

b) Acrescentar neste objeto carro um membro chamado **ligar** que possuirá como valor uma função que implementará a seguinte lógica:

- Verificar se o carro está ligado ou não.
  - Se o carro **já estiver ligado**, deverá imprimir uma mensagem no console dizendo: `Este carro já está ligado.`
  - Se não* (se o carro não estiver ligado), deverá alterar a propriedade **ligado** para **true**.

c) Acrescentar neste objeto carro um membro chamado **desligar** que possuirá como valor uma função que implementará a seguinte lógica:

- Verificar se o carro está ligado ou não.
  - Se o carro **já estiver desligado**, deverá imprimir uma mensagem no console dizendo: `Este carro já está desligado.`
  - Se não* (se o carro estiver ligado), deverá:
    - alterar a propriedade **ligado** para **false**.
    - atribuir valor **0** (zero) à propriedade **velocidade**.

d) Acrescentar neste objeto carro um membro chamado **acelerar** que possuirá como valor uma função que implementará a seguinte lógica:

- Verificar se o carro está ligado ou não.
  - Se o carro **não** estiver ligado, deverá imprimir uma mensagem no console dizendo: `Não é possível acelerar um carro desligado.`
  - Se não* (se o carro estiver ligado), deverá:
    - aumentar** em 10 o valor da propriedade **velocidade**

e) Acrescentar neste objeto carro um membro chamado **desacelerar** que possuirá como valor uma função que implementará a seguinte lógica:

- Verificar se o carro está ligado ou não.
  - Se o carro **não** estiver ligado, deverá imprimir uma mensagem no console dizendo: `Não é possível desacelerar um carro desligado.`
  - Se não* (se o carro estiver ligado), deverá:
    - diminuir** 10 do valor da propriedade **velocidade**

e) Em cada um dos 4 métodos (ligar, desligar, acelerar, desacelerar), quando e **apenas** quando alguma propriedade for alterada, imprimir no console uma mensagem mostrando o status atual do carro. Esta mensagem deverá seguir os seguinte formato: "Carro [ligado/desligado]. Velocidade: [velocidade]". Com isto, todos os métodos quando alterarem alguma das propriedades, imprimirá o status atual do carro.

Exemplos de mensagens

```
Carro desligado. Velocidade: 0.
Carro ligado. Velocidade: 0.
Carro ligado. Velocidade: 30.
```

f) Após construir todo o objeto com suas propriedades e métodos, deveremos executar os métodos na seguinte ordem:

- Desligar o carro
- Ligar o carro
- Ligar o carro
- Acelerar o carro
- Acelerar o carro
- Desacelerar o carro
- Desligar o carro
- Acelerar o carro
- Desacelerar o carro

As mensagens que deverão ser exibidas no console são:

```
Este carro já está desligado.
Carro ligado. Velocidade: 0.
Este carro já está ligado.
Carro ligado. Velocidade: 10.
Carro ligado. Velocidade: 20.
Carro ligado. Velocidade: 10.
Carro desligado. Velocidade: 0.
Não é possível acelerar um carro desligado.
Não é possível desacelerar um carro desligado.
```

Após executar os métodos, fique a vontade para testar as execuções de formas / ordens diferentes para testar como desejar! =)

---

### EXTRA

Estes itens desta seção EXTRA não são obrigatórios e só deverão ser feitos se estiver com tempo sobrando.

- Tente revisar os métodos para aplicar a técnica DRY (Don't Repeat Yourself) e excluir (quando possível) toda repetição de código igual ou muito semelhante.
- Faça uma segunda validação para permitir que o carro seja desligado **apenas** quando tiver em velocidade zero.
- Implemente a **função start/stop** no carro, de forma que quando o carro estiver desligado e for acelerado, ele antes de acelerar, liga o carro. O contrário deve ser aplicado para o desacelerar: se o carro for desacelerado e a **nova** velocidade for zero, ele deve ser desligado.

---

Preencha a checklist para finalizar o exercício:

☐ Resolver o exercício revendo a aula se necessário

☐ Adicionar as mudanças aos commits ( `git add .` para adicionar todos os arquivos ou `git add nome_do_arquivo` para adicionar um arquivo específico)

☐ Commitar a cada mudança significativa ou na finalização do exercício ( `git commit -m "Mensagem do commit"` )


☐ Pushar os commits na sua branch na origem ( `git push origin nome-da-branch` )

☐ Realizar o pull request

tags: [lógica](#) [módulo 1](#) [exercício de classe](#) [nodeJS](#) [funcao](#) [objetos](#)

© 2021 GitHub, Inc.

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#)



[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)