

tryber / sd-03-block9-project-zoo-functions

generated from [betrybe/sd-0x-project-zoo-functions](#)

☆ 0 stars 🍴 1 fork

☆ Star

👁 Watch ▼

< > Code

! Issues

🔗 Pull requests 61

▶ Actions

📁 Projects

📖 Wiki

🛡 Security

🔗 master ▼

...



mhamaji Fix codeclimate ...

✖ on 31 Mar 2020 ⌚ 5

[View code](#)

README.md



Boas vindas ao repositório do projeto de ES6 e Higher Order Functions!

Você já usa o GitHub diariamente para desenvolver os exercícios, certo? Agora, para desenvolver os projetos, você deverá seguir as instruções a seguir. Fique atento a cada passo, e se tiver qualquer dúvida, nos envie por *Slack*! #vqv 🚀

Aqui você vai encontrar os detalhes de como estruturar o desenvolvimento do seu projeto a partir desse repositório, utilizando uma branch específica e um *Pull Request* para colocar seus códigos.

Instruções para entregar seu projeto:

ANTES DE COMEÇAR A DESENVOLVER:

1. Clone o repositório

- `git clone https://github.com/tryber/sd-03-block9-project-zoo-functions.git`

- Entre na pasta do repositório que você acabou de clonar:

- `cd sd-03-block9-project-zoo-functions`

2. Crie uma branch a partir da branch `master`

- Verifique que você está na branch `master`

- Exemplo: `git branch`

- Se não estiver, mude para a branch `master`

- Exemplo: `git checkout master`

- Agora, crie uma branch onde você vai guardar os `commits` do seu projeto

- Você deve criar uma branch no seguinte formato: `nome-de-usuario-nome-do-projeto`

- Exemplo: `git checkout -b exemplo-zoo-functions`

3. Desenvolva a solução para os problemas no arquivo `src/zoo.js`. Você pode usar os arquivos do diretório `test` para verificar se a sua implementação está de acordo com o esperado;

4. Adicione as mudanças ao `stage` do Git e faça um `commit`

- Verifique que as mudanças ainda não estão no `stage`

- Exemplo: `git status` (deve aparecer listada a pasta *joaozinho* em vermelho)

- Adicione o novo arquivo ao `stage` do Git

- Exemplo:

- `git add .` (adicionando todas as mudanças - *que estavam em vermelho* - ao stage do Git)

- `git status` (deve aparecer listado o arquivo *joaozinho/README.md* em verde)

- Faça o `commit` inicial

- Exemplo:

- `git commit -m 'iniciando o projeto. VAMOS COM TUDO :rocket:'` (fazendo o primeiro commit)

- `git status` (deve aparecer uma mensagem tipo *nothing to commit*)

5. Adicione a sua branch com o novo `commit` ao repositório remoto

- Usando o exemplo anterior: `git push -u origin exemplo-zoo-functions`

6. Crie um novo `Pull Request` (PR)

- Vá até a página de *Pull Requests* do [repositório no GitHub](#)

- Clique no botão verde "New pull request"

- Clique na caixa de seleção "*Compare*" e escolha a sua branch **com atenção**
- Clique no botão verde "*Create pull request*"
- Adicione uma descrição para o *Pull Request* e clique no botão verde "*Create pull request*"
- **Não se preocupe em preencher mais nada por enquanto!**
- Volte até a [página de Pull Requests do repositório](#) e confira que o seu *Pull Request* está criado.

Correção automatizada

Você irá perceber que, ao realizar novos `commits` no seu *Pull Request*, eles serão automaticamente analisados pelo *CodeClimate* e pelo *TravisCI*.

Se atente para os comentários do *CodeClimate* que irão conter **dicas valiosas** de como deixar seu código melhor! Você pode também explorar o *TravisCI* para obter informações sobre quais *assertions* ainda precisa adereçar em seu projeto.

O que deverá ser desenvolvido

Requisitos do projeto

Você deverá implementar as funções que estão no `src/zoo.js` para passarem em cada um dos testes. O teste `test/animalsByIds.test.js`, por exemplo, testa a função `addEmployee`, que já está criada dentro do `src/zoo.js`, embora ainda não contenha lógica alguma. Para ver o que cada função precisa retornar basta ver o `assert` de cada um dos testes.

Utilize as novas funcionalidades do ES6 como arrow functions, template literals, spread operator, parâmetro rest, object destructuring, entre as outras. Utilize também as *High Order Functions*.

Antes de começar analise o arquivo `src/data.js`, para ver os dados que serão usados.

1- Implemente a função `animalsByIds`:

- Caso receba nenhum parâmetro, necessário retornar um array vazio
- Ao receber como parâmetro um único id, retorna os animais com este id
- Ao receber mais de um id, retorna os animais que têm um desses ids

2- Implemente a função `animalsOlderThan`:

- Ao passar o nome de uma espécie e uma idade, testa se todos os animais desta espécie possuem a idade mínima especificada

3- Implemente a função `employeeByName`:

- Sem parâmetros, retorna um objeto vazio
- Quando provido o primeiro nome do funcionário, retorna o objeto do funcionário
- Quando provido o último nome do funcionário, retorna o objeto do funcionário

4- Implemente a função `createEmployee`:

- Cria um novo colaborador a partir de objetos contendo informações pessoais, gerentes e animais gerenciados

5- Implemente a função `isManager`:

- Testa se o id passado é de um gerente

6- Implemente a função `addEmployee`:

- Adiciona um funcionário no fim da lista

7- Implemente a função `animalCount`:

- Sem parâmetros, retorna animais e suas quantidades
- Com o nome de uma espécie de animal, retorna somente a quantidade

8- Implemente a função `entryCalculator`:

- Retorna 0 se nenhum argumento for passado
- Retorna 0 se um objeto vazio for passado
- Retorna o preço total a ser cobrado dado o número de adultos, crianças e idosos

9- Implemente a função `animalMap`:

- Sem parâmetros, retorna animais categorizados por localização
- Com opções especificadas, retorna nomes de animais
- Com opções especificadas, retorna nomes de animais ordenados
- Com opções especificadas, retorna somente nomes de animais macho/fêmea

- Só retorna informações específicas de gênero se `includeNames` for setado

10- Implemente a função `schedule`:

- Sem parâmetros, retorna um cronograma legível para humanos
- Se um único dia for passado, retorna somente este dia em um formato legível para humanos

11- Implemente a função `oldestFromFirstSpecies`:

- Passado o id de um funcionário, encontra a primeira espécie de animal gerenciado pelo funcionário, e retorna um array com nome, sexo e idade do animal mais velho dessa espécie





12- Implemente a função `increasePrices`:

- Ao passar uma porcentagem, incrementa todos os preços, arredondados em duas casas decimais

13- Implemente a função `employeeCoverage`:

- Sem parâmetros, retorna uma lista de funcionários e os animais pelos quais eles são responsáveis
- Com o id de um funcionário, retorna os animais pelos quais o funcionário é responsável
- Com o primeiro nome de um funcionário, retorna os animais pelos quais o funcionário é responsável
- Com o último nome de um funcionário, retorna os animais pelos quais o funcionário é responsável

DURANTE O DESENVOLVIMENTO

-  **LEMBRE-SE DE CRIAR TODOS OS ARQUIVOS DENTRO DA PASTA COM O SEU NOME** 
-  **PULL REQUESTS COM ISSUES NO CODE CLIMATE NÃO SERÃO AVALIADAS, ATENTE-SE PARA RESOLVÊ-LAS ANTES DE FINALIZAR O DESENVOLVIMENTO!** 
- Faça `commits` das alterações que você fizer no código regularmente

- Lembre-se de sempre após um (ou alguns) `commits` atualizar o repositório remoto
- Os comandos que você utilizará com mais frequência são:
 - i. `git status` (para verificar o que está em vermelho - fora do stage - e o que está em verde - no stage)
 - ii. `git add` (para adicionar arquivos ao stage do Git)
 - iii. `git commit` (para criar um commit com os arquivos que estão no stage do Git)
 - iv. `git push -u nome-da-branch` (para enviar o commit para o repositório remoto na primeira vez que fizer o `push` de uma nova branch)
 - v. `git push` (para enviar o commit para o repositório remoto após o passo anterior)

DEPOIS DE TERMINAR O DESENVOLVIMENTO (OPCIONAL)

Para disponibilizar seu projeto para [Code Review](#), siga os passos a seguir:

- Vá até a página **DO SEU Pull Request**, adicione a label de "code-review" e marque seus colegas
 - No menu à direita, clique no *link* "**Labels**" e escolha a *label* **code-review**
 - No menu à direita, clique no *link* "**Assignees**" e escolha **o seu usuário**
 - No menu à direita, clique no *link* "**Reviewers**" e digite `students`, selecione o time `tryber/students-sd-03`

Se ainda houver alguma dúvida sobre como entregar seu projeto, [aqui tem um video explicativo](#).

⚠ Lembre-se que garantir que todas as *issues* comentadas pelo CodeClimate estão resolvidas! ⚠

REVISANDO UM PULL REQUEST



À medida que você e as outras pessoas que estudam na Trybe forem entregando os projetos, vocês receberão um alerta via Slack para também fazer a revisão dos Pull Requests dos seus colegas. Fiquem atentos às mensagens do "Pull Reminders" no Slack!

Use o material que você já viu sobre [Code Review](#) para te ajudar a revisar os projetos que chegaram para você.

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Contributors 3



jeanpsv Jean Paulo Silva Vasconcelos



antoniosb Antônio Augusto



mhamaji Mauricio Hamaji Homma

Languages

