tryber / **sd-03-block8-project-js-unit-tests**

0 stars **0** forks

Star Watch

Code Issues Pull requests 61 Actions Projects Wiki Secu

master

 **mhamaji** Fix circle code ... on 25 Mar 2020 **7**

View code

README.md

Boas vindas ao repositório do projeto de ES6 e Testes Unitários!

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 deste repositório, utilizando uma branch específica e um *Pull Request* para colocar seus códigos.

O que deverá ser desenvolvido

Você implementará várias funções para atender aos requisitos propostos e/ou testes unitários para garantir que as implementações das funções estão corretas.

Desenvolvimento

Este repositório contém um *template* de uma aplicação **NodeJS** (observe a existência do arquivo *package.json*). Após clonar o projeto e instalar as dependências (mais sobre isso abaixo), você não precisará realizar nenhuma configuração adicional. Todos os arquivos estritamente necessários para finalizar o projeto já estão criados, **não** sendo necessária a criação de outros arquivos. Você deverá completar as funções e testes unitários de forma a satisfazer os requisitos listados na próxima seção.

As funções a serem implementadas estão dentro da pasta `src` e seus respectivos testes estão na pasta `tests`. O nome dos arquivos também seguem uma ordem definida. Basicamente, os arquivos de teste possuem o nome do arquivo alvo (arquivo da funcionalidade) acrescido do nome `.spec.js`. Existirá um arquivo `src/blabla.js` que conterá a implementação de uma função e um arquivo `tests/blabla.spec.js` com os testes unitários referentes à função presente no arquivo `src/blabla.js`.

Cada função possui um bloco de comentários em suas primeiras linhas explicando qual é o trabalho que a função deve realizar.

Você só deve alterar os arquivos indicados nos requisitos. **Os arquivos que não estão indicados nos requisitos não devem ser alterados, ou sua avaliação poderá ser comprometida.**

Requisitos do projeto

1 - Implemente a função do arquivo `src/average.js`

2 - Implemente os casos de teste no arquivo `tests/numbers.js`

3 - Implemente a função do arquivo `src/vqv.js`

4 - Implemente os casos de teste no arquivo `tests/circle.spec.js`

5 - Implemente a função do arquivo `src/createStudent.js`

6 - Implemente os casos de teste no arquivo `tests/productDetails.spec.js`

7 - Implemente a função do arquivo `src/objCalculator.js`

8 - Implemente a função do arquivo `src/myCounter.js`

Agora prepare-se! Esse último requisito vai te guiar através de um longo e rico processo de desenvolvimento orientado a testes (Test Driven Development, ou TDD). Dará trabalho, mas vale a pena!

9 - Implemente os casos de teste no arquivo `tests/restaurant.spec.js` e as funções do arquivo `src/restaurant.js`

Instruções para entregar seu projeto

ANTES DE COMEÇAR A DESENVOLVER:

1. Clone o repositório

- `git clone git@github.com:tryber/sd-03-block8-project-js-unit-tests.git`.
- Entre na pasta do repositório que você acabou de clonar:
  - `cd sd-03-block8-project-js-unit-tests`

2. Instale as dependências

- `npm install`

3. 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 para qual você vai submeter os `commits` do seu projeto
  - Você deve criar uma branch no seguinte formato: `nome-de-usuario-nome-do-projeto`
  - Exemplo: `git checkout -b joaozinho-js-unit-tests`

4. Faça as alterações em, por exemplo, alguma das funções que precisam de implementação. Por exemplo, a `average.js` em `src/`:

```
const average = () => {
  // add your implementation here
}
```

```
module.exports = average
```

5. 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 listado o arquivo *src/last.js* em vermelho)
- Adicione o arquivo alterado 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 *src/last.js* 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*)

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

- Usando o exemplo anterior: `git push -u origin joaozinho-js-unit-tests`

7. 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

---

DURANTE 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)*

  vi. `npm test` *(executa todos os testes presentes na aplicação)*

  vii. `npm test path/to/file` *(executa apenas os testes presentes no arquivo path/to/file)*
  - exemplo: `npm test tests/average.spec.js`

  viii. `jest path/to/file` *(executa apenas os testes presentes no arquivo path/to/file)*
  - exemplo: `jest tests/average.spec.js`

DEPOIS DE TERMINAR O DESENVOLVIMENTO

Para **"entregar"** seu projeto, 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.

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 2

 **mhamaji** Mauricio Hamaji Homma

 **jeanpsv** Jean Paulo Silva Vasconcelos

Languages

- **JavaScript** 100.0%