Programa para criar projeto básico typescript versão: 0.2.0

```
#!/bin/sh

echo PROGRAMA: Cria project typescript Básico
echo AUTOR: Paulo Pacheco
echo DATA DA CRIAÇÃO: 01/06/2021
echo DATA DA ULTIMA ATUALIZAÇÃO: 08/06/2021
echo 0.2.0
echo .

# Passo 01 - Função para renomeá arquivo e apagar o arquivo anterior se
existir
rename(){
  #Parâmetros
  # Chamada: rename param1 param2
  fileName="$1";
  fileNameOld="$2";

# begin
    if [ -f $fileNameOld ]; then
        # Se $fileNameOld já existe então apaga
#        echo O arquivo $fileNameOld foi criado anteriormente!.
#        echo Tecle ENTER para apaga-lo.
#        read input
        rm $fileNameOld
    fi


    if [ -f $filename ]; then # Se $filename existe renomeia para
$fileNameOld
#        echo O arquivo $fileName foi criado anteriormente!.
#        echo Tecle ENTER para renomear para $fileNameOld.
#        read input
        mv  $fileName $fileNameOld
    fi
# end
}


# Passo 02 - Retorna o nome do projeto ou aborta o script
ifExistProject(){

# Se o parâmetro 01 não existe então fazer nome do project igual
./myproject
  if [ -z $project ]; # -z indica que a string $pasta existe e é vazia.
  then
    # echo  -z $pasta retorno true
```

```
    # read input
    project="./myproject"
  else
    # echo  -z $pasta retorno false
    # read input
    project="./$project"
  fi

  if [ -d $project ]; then # Se $pasta já existe então aborta execução.
      #echo O project $project foi criado anteriormente!. Tecle ENTER para
sair.
      #read input
      exit EEXIST #  aborta o script

  fi

}

# Passo 03 - Criar as pasta do projeto;
createProject(){

  ifExistProject # Se o projeto já existe essa função aborta

  # Cria a pasta do project passada no parâmetro 01
  mkdir $project

  # Cria as pasta para os códigos fontes do project
  mkdir ./"$project"/src      # Pasta raiz do código fonte
  mkdir ./"$project"/src/html # Pasta html de entrada de webpack
  mkdir ./"$project"/src/css  # Pasta css de entrada de webpack
  mkdir ./"$project"/src/js   # Pasta de saide js de tsc
  mkdir ./"$project"/src/ts   # Pasta de entrada ts de tsc
  mkdir ./"$project"/dist     # Pasta destino do webpack para ser publicada
na web.

}


# Passo 04 - Criar workspace para o projeto do vscode na pasta raiz do
projeto;
createWorkspace(){

cat >./"$project"/workspace.code-workspace"<<EOT
{
    "folders": [
        {
            "path": "."
        }
    ]
}
EOT

}
```

```
# Passo 05 - Adicionar o código .html no arquivo ./src/html/index.html;
createFileIndexHtml(){

# Criar arquivo inicial index.html
cat >"./"$project"/src/html/index.html"<<EOT

  <!DOCTYPE html>
      <html dir="ltr" lang="pt-br">

          <head>
              <meta http-equiv="content-type" content="text/html;
charset=utf-8" />

              <meta name="viewport" content="width=device-width, initial-
scale=1.0" />

              <title>Modelo de project typescript</title>

              <meta name="createDate" content="28/05/2021" />
              <meta name="createDateUpdate" content="25/05/2021" />
              <meta name="description" content="Todos project typescript
deve seguir essa sequência ao iniciar..." />
              <meta name="keywords" content="typescript,webpack" />

              <link type="text/css"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css"
                  rel="stylesheet" />

          </head>

          <body>
              <p>Alo mundo</p>
          </body>

      </html>

EOT

}


# Passo 06 - Adicionar o código .css no arquivo ./src/css/defaulttheme.css;
createFiledefaultThemeCss() {

cat >"./"$project"/src/css/defaulttheme.css"<<EOT
    html {
        scroll-behavior: smooth;
        /* scroll lento*/
    }

    body {
        margin: 0;
        font-family: Arial
```

```
    }

EOT
}

# Passo 07 - Adicionar o código .ts no arquivo ./src/ts/index.ts;
createIndexTs(){

cat >"./"$project"/src/ts/index.ts"<<EOT
   console.log('Alo mundo');
EOT

}

# Passo 08 - Executar o programa tsc-init para iniciar tudo que for preciso
para o projeto;
execTscInit(){
  cd $project
  tsc-init
}

# Passo 09 - Instalar pacotes.
installPackages(){

  # Instala plugin a ser usado por alterWebpackConfig()
  npm install --save-dev html-webpack-plugin
  npm install --save-dev html-loader

}


# Passo10 - Customizar o comportamento do webpack no arquivo webpack-
config.js;
alterWebpackConfig(){

  echo "... ";
  echo "Adiciona const path = require('path') em webpack.config.js;"
  echo "Adiciona a declaração \"const HtmlWebpackPlugin = require('html-
webpack-plugin')\" em webpack.config.js"


  subStrOrigem='module.exports = {'
  subStrDestino=" const path = require('path');\n const HtmlWebpackPlugin =
require('html-webpack-plugin'); \n\n module.exports = {"

  fileNameOld="./webpack.config.js.ant"
  fileName="./webpack.config.js"

  rename "$fileName" "$fileNameOld"

  sed "s/$subStrOrigem/$subStrDestino/g" $fileNameOld > $fileName

  echo ...
```

```
  echo Adiciona a propriedade "HtmlWebpackPlugin" em module.exports = {
webpack.config.js

  subStrOrigem='module.exports = {'
  subStrDestino="module.exports = \{plugins:\[new HtmlWebpackPlugin\(\
{filename:'index.html',template:'.\/src\/html\/index.html',inject:'body'\}\
)\],"

  fileNameOld="./webpack.config.js.ant"
  fileName="./webpack.config.js"

  rename "$fileName" "$fileNameOld"

  sed "s/$subStrOrigem/$subStrDestino/g" $fileNameOld > $fileName

  echo ...

  echo ...
  echo Alterar o arquivo webpack.config.js

  # Atualiza o arquivo de entrada de webpack.config.js
  subStrOrigem="entry: '.\/index.ts'"
  subStrDestino="entry: '.\/src\/ts\/index.ts'"
  fileNameOld="./webpack.config.js.ant"
  fileName="./webpack.config.js"
  rename "$fileName" "$fileNameOld"
  sed "s/$subStrOrigem/$subStrDestino/g" $fileNameOld > $fileName
}

# Passo11 - Customizar arquivo package.json;
alterPackage(){
  echo ...

  echo Atualiza script "dev" do package.json
  subStrOrigem="webpack-dev-server --inline --hot"
  subStrDestino="webpack serve --mode development --env development --hot -
-port 3000"
  fileNameOld="./package.json.ant"
  fileName="./package.json"
  rename "$fileName" "$fileNameOld"
  sed "s/$subStrOrigem/$subStrDestino/g" $fileNameOld > $fileName

  echo ...
  echo  Atualiza script "build" do package.json
  subStrOrigem="webpack -p"
  subStrDestino="webpack --mode='production'"
  fileNameOld="./package.json.ant"
  fileName="./package.json"
  rename "$fileName" "$fileNameOld"
  sed "s/$subStrOrigem/$subStrDestino/g" $fileNameOld > $fileName

}

# Passo 12 - Customizar arquivo tsconfig.json;
```

```
alterTsconfig(){
    echo ...

    echo Adiciona a propriedade "outDir" em "compilerOptions": {} do
tsconfig.json

    subStrOrigem='"compilerOptions": {'
    subStrDestino='"compilerOptions": {"outDir": ".\/src\/js",'

    fileNameOld="./tsconfig.json.ant"
    fileName="./tsconfig.json"

    rename "$fileName" "$fileNameOld"

    sed "s/$subStrOrigem/$subStrDestino/g" $fileNameOld > $fileName

    echo ...

}


# PROGRAMA PRINCIPAL
  # $1 é passado na execução de my-tsc-init.sh

project="$1"
createProject
createWorkspace
createFileIndexHtml
createFiledefaultThemeCss
createIndexTs
execTscInit
installPackages
alterWebpackConfig
alterPackage
alterTsconfig

echo fim do script
```