**Gulp**

**Índice**

1. Introdução
2. O que é?
3. Instalação
4. gulpfile.js
5. Importação de plugins
6. Tarefas
7. Executando
8. Funções privadas
9. Variáveis globais
10. Método src()
11. Método pipe()
12. Método dest()
13. Método watch()
14. Outros plugins
15. Desafio
16. Conclusão

**1. Introdução**

Dando continuidade à série de artigos visando atualizar os desenvolvedores WEB que possam ter ficado perdidos com o BUM de novas tecnologias que surgiram nos últimos anos (clique [aqui](https://github.com/eudiasrafael/artigo__introducao_ao_npm) e [aqui](https://github.com/eudiasrafael/artigo__introducao_ao_bower) para ver os artigos sobre NPM e Bower respectivamente), hoje irei abordar sobre o Gulp.

Vamos lá!

**2. O que é?**

Gulp.js (ou simplesmente Gulp) é um automatizador de tarefas baseado no NodeJS, isso significa que ele é capaz de executar para você todas aquelas tarefas braçais chatas que consomem tanto tempo (e paciência).

Existe uma gama absurda de tarefas que podem ser executadas, citarei algumas:

1. Compilar arquivos SASS, LESS, TypeScript, Cofee, Jade e etc...
2. Minificar arquivos
3. Unir arquivos
4. Remover arquivos temporários antigos
5. Trocar ocorrências em arquivos
6. Iniciar um determinado programa ou processo
7. Monitorar alterações nos arquivos para executar as tarefas
8. Etc….

Os limites do Gulp estão apenas na sua imaginação para como utilizar os plugins disponíveis para ele.

**3. Instalação**

Uma vez que o Gulp é baseado no NodeJS, ele deve ser instalado via NPM.

Caso não tenha o NPM e/ou não saiba utilizá-lo adequadamente, recomendo a leitura do meu artigo ["Introdução ao NPM"](https://github.com/eudiasrafael/artigo__introducao_ao_npm).

Uma vez tendo o NPM corretamente instalado e configurado no projeto, abra o diretório do projeto e execute os seguintes comandos:

npm install -g gulp

npm install --save-dev gulp

O primeiro comando instalará o gulp globalmente (necessário apenas uma vez na máquina), tornando-o disponível como um programa.

O segundo comando instalará o gulp como dependência de desenvolvimento do projeto.

**4. gulpfile.js**

Uma vez o gulp instalado (global e local), precisamos criar um arquivo na raiz do projeto chamado *"gulpfile"*.

Este é o arquivo principal do Gulp, onde ficarão todas as definições referentes as tarefas, tais como: Importação de plugins, variáveis globais, funções privadas e as tarefas em si.

**5. Importação de plugins**

A primeira coisa a ser escrita no *"gulpfile"* é a importação dos plugins a serem utilizados.

Trata-se de uma etapa crucial, uma vez que o Gulp por si só fornece somente uma estrutura para as tarefas serem executadas, cabendo aos plugins executá-las de fato.

Para importar um plugin é necessário primeiro instalá-lo no projeto. Veja um exemplo de instalação do plugin *"gulp-minify-css"*:

npm install --save-dev gulp-minify-css

Este comando irá baixar e instalar o pacote *"gulp-minify-css"* dentro do diretório do projeto.

Agora para importar este plugin ao *"gulpfile"*, basta adicionar o seguinte código no início do arquivo (na verdade, basta ser antes das tarefas que o utilizarem, mas para manter organizado, recomendo deixar sempre no início do arquivo).

var minifyCss = require("gulp-minify-css")

A função *"require"* irá importar os recursos do plugin *"gulp-minify-css"* para dentro da variável aqui chamada *"minifyCss*".

Para manter o código legível, recomendo manter o nome destas variáveis que contém os recursos dos plugins, com os mesmos nomes dos plugins.

Quem tem alguma experiência com NodeJS, irá reparar que esse código é na verdade…. NodeJS puro! Por isso mesmo, para que o NodeJS tenha acesso aos recursos do Gulp, é necessário também importá-los como a um plugin:

var gulp = require("gulp")

**6. Tarefas**

Para criar uma tarefa utilizamos o método *"task"* do plugin *"gulp"*, passando como primeiro parâmetro o nome da tarefa e no segundo, uma função anônima com a rotina da tarefa. Veja:

// Importações

var gulp = require("gulp")

var minifyCss = require("gulp-minify-css")

// Tarefas

gulp.task("minificarCSS", function() {

/\* Rotina da tarefa \*/

})

Vale informar que se o Gulp for chamado sem informar uma tarefa, o Gulp irá executar automaticamente a tarefa chamada *"default"*, recomendo criar uma tarefa com este nome que liste e detalhe as tarefas existentes.

**7. Executando**

Para executar uma tarefa implementada no *"gulpfile"*, acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

gulp minificarCSS

O Gulp irá executar a tarefa *"minificarCSS"*.

Para executar a tarefa *"default"*:

gulp

**8. Funções privadas**

Funções privadas são aquelas que não são acessíveis pela linha de comando, porém que podem ser evocadas por uma ou mais tarefas. Veja:

// Importações

var gulp = require("gulp")

var minifyCss = require("gulp-minify-css")

// Tarefas

gulp.task("minificarCSS", function() {

minhaFuncaoPrivada()

})

// Funções privadas

function minhaFuncaoPrivada() {

/\* Rotina da função privada \*/

}

Salientando que não é obrigatório chamar uma função privada dentro de uma tarefa, toda a rotina pode ser implementada diretamente na tarefa. Foi feito assim somente para mostrar como é possível declarar e chamar uma função privada.

**9. Variáveis globais**

Variáveis globais são aquelas que não são acessíveis pela linha de comando, porém que podem ser utilizadas por uma ou mais tarefas ou funções privadas. Veja:

// Importações

var gulp = require("gulp")

var minifyCss = require("gulp-minify-css")

// Variáveis globais

var cssDir = "src/css/\*\*/\*.css"

// Tarefas

gulp.task("minificarCSS", function() {

console.log(cssDir)

minhaFuncaoPrivada()

})

// Funções privadas

function minhaFuncaoPrivada() {

console.log(cssDir)

}

A tarefa *"minificarCSS*" irá imprimir duas vezes o conteúdo da variável privada *"cssDir"*, uma na tarefa em si e outra na função privada que foi evocada pela tarefa.

**10. Método src()**

O método *"src"* do gulp é o responsável por capturar os arquivos com os quais desejamos trabalhar.

Ele receberá uma string ou um array de strings, em ambos os casos as strings se referem ao endereço dos arquivos.

O endereço dos arquivos pode ser montado com expressão regular.

Veja:

// Importações

var gulp = require("gulp")

var minifyCss = require("gulp-minify-css")

// Variáveis globais

var cssDir = "src/css/\*\*/\*.css"

// Tarefas

gulp.task("minificarCSS", function() {

minhaFuncaoPrivada()

})

// Funções privadas

function minhaFuncaoPrivada() {

gulp.src(cssDir)

}

Passando o endereço *"src/css/\*\*/\*.css"*, pedimos ao Gulp para capturar todos os arquivos de qualquer nome (\*) e extensão css (.css) dentro de qualquer diretório de "src/" ou em sua raiz.

O método *"src"* irá retornar um *resource* dos arquivos, o qual disponibilizará alguns métodos para manipulação destes arquivos.

**11. Método pipe()**

O método *"pipe"* pertence ao *resource* retornado pelo *"src"*, é ele o responsável por executar uma determinada ação sobre os arquivos em questão.

Recebe como parâmetro uma chamada ao plugin desejado (o qual poder receber seus parâmetros também, sendo necessário consultar a documentação de cada plugin), veja:

// Importações

var gulp = require("gulp")

var minifyCss = require("gulp-minify-css")

// Variáveis globais

var cssDir = "src/css/\*\*/\*.css"

// Tarefas

gulp.task("minificarCSS", function() {

minhaFuncaoPrivada()

})

// Funções privadas

function minhaFuncaoPrivada() {

gulp.src(cssDir).pipe(minifyCss())

}

O *"pipe"* irá executar o método *"minifyCss"* responsável por minificar arquivos CSS, minificando todos os arquivos do resource do *"src"*.

Podemos utilizar o método *"pipe"* quanta vezes quisermos utilizando a concatenação para chegarmos no resultado desejado, por exemplo:

gulp.src(cssDir)

.pipe(minifyCss())

.pipe(outroPlugin())

.pipe(outroPlugin())

Neste caso mantive os *"pipes"* um abaixo do outro para maior legibilidade apenas.

**12. Método dest()**

Para completar o ciclo básico da tarefa, temos o método *"dest"* do Gulp, o qual irá salvar em arquivos o resultado dado pelo método *"pipe"*

Recebe como parâmetro uma string contendo o endereço do diretório onde os arquivos serão salvos.

A hierarquia de diretórios do arquivo será respeitada e o novo arquivo será salvo sobre a mesma hierarquia.

Veja:

// Importações

var gulp = require("gulp")

var minifyCss = require("gulp-minify-css")

// Variáveis globais

var cssDir = "src/css/\*\*/\*.css"

// Tarefas

gulp.task("minificarCSS", function() {

minhaFuncaoPrivada()

})

// Funções privadas

function minhaFuncaoPrivada() {

gulp.src(cssDir).pipe(minifyCss()).pipe(gulp.dest("dist/"))

}

O *"dest"* irá salvar todos os arquivos processados pelo *"pipe*" e salvar no diretório *dist/*.

Respeitando a hierarquia de diretórios, um arquivo que estava em *src/css/dir1/dir2/dir3/meuCss.css* terá sua minificação salva em *dist/css/dir1/dir2/dir3/meuCss.css*.

**13. Método watch()**

O método *"watch"* do Gulp é o responsável por uma mágica na qual ele monitora os arquivos de um dado diretório e executa uma ou mais tarefas quando um destes arquivos for alterado.

Ele receberá uma string ou um array de strings, em ambos os casos as strings se referem ao endereço dos arquivos.

O endereço dos arquivos pode ser montado com expressão regular.

Também recebe como segundo parâmetro um array contendo os nomes das tarefas a serem executadas.

Veja:

// Importações

var gulp = require("gulp")

var minifyCss = require("gulp-minify-css")

// Variáveis globais

var cssDir = "src/css/\*\*/\*.css"

// Tarefas

gulp.task("minificarCSS", function() {

minhaFuncaoPrivada()

})

gulp.task("monitorarCSS", function() {

gulp.watch(cssDir, [“minificarCSS”]);

})

// Funções privadas

function minhaFuncaoPrivada() {

gulp.src(cssDir).pipe(minifyCss()).pipe(gulp.dest("dist/"))

}

A tarefa *"monitorarCSS"* irá monitorar mudanças nos arquivos de *src/css/\*\*/*.css\* e, sempre que houver, executar a tarefa *"minificarCSS"*.

**14. Outros plugins**

Como podemos ver, o Gulp fornece somente uma estrutura básica para a execução das tarefas, cabendo aos plugins fazer a mágica de fato.

Assim como o *"minifyCss"* minifica os arquivos CSS, existem vários outros responsáveis por uma infinidade de ações.

Vejamos alguns:

| **Nome pacote** | **Nome no NPM** | **Descrição** |
| --- | --- | --- |
| GulpRename | gulp-rename | Renomeia o arquivo |
| GulpConcat | gulp-concat | Concatena o conteúdo dos arquivos |
| GulpSync | gulp-sync | Executa as tarefas na ordem adequada |
| GulpReplace | gulp-replace | Troca um termo por outro no conteúdo dos arquivos |
| GulpJade | gulp-jade | Compila código JADE transformando-o em HTML |
| GulpTypeScript | gulp-typescript | Compila código TypeScript transformando-o em JS |