

Gulp.js 22.07.2021

☰ Description	Wie man Gulp aufsetzt und konfiguriert.
☰ Tags	develop gulp terminal

Gulp.js Setup 22.07.2021

Node.js Versioncheck

Ist man sich nicht mehr sicher ob man **node**, **npm** und **npx** auf dem neusten Stand (überhaupt) besitzt kann man dies testen mit den unten stehenden Terminal Commands.

```
node --version
npm --version
npx --version
```

Node.js installieren oder updaten

Sollte man **node.js** noch nicht installiert haben, kann man sich die neuste Version [hier](#) herunterladen.

node.js updaten oder installieren ist auch mit dem opensource Paketmanager [Homebrew](#) möglich.

```
brew install node
brew upgrade node
```

Gulp global installieren

Hat man die vorherigen Schritte richtig befolgt, so kann man jetzt die **Gulp CLI**(command line utility) global per Terminal installieren.

```
npm install --global gulp-cli
```

Projektverzeichnis erstellen + package.json

In diesem Schritt erstellt man sich ein Verzeichnis und wechselt in jenes.

```
npx mkdirp my-project  
cd my-project
```

Im Anschluss erstellt man das package.json file per Terminal.

```
npm init
```

Gulp Paket in den devDependencies installieren

```
npm install --save-dev gulp
```

Nun sollte man noch die Version von gulp checken.

```
gulp --version
```

Gulpfile erstellen und konfigurieren

Nachdem man alle vorherigen Schritte richtig ausgeführt hat, kann man sich jetzt eine Datei mit dem Namen gulpfile.js erstellen, dieses muss sich im Root-Verzeichnis des Projekts befinden.

In diesem File erstellt man nun eine Grundfunktion.

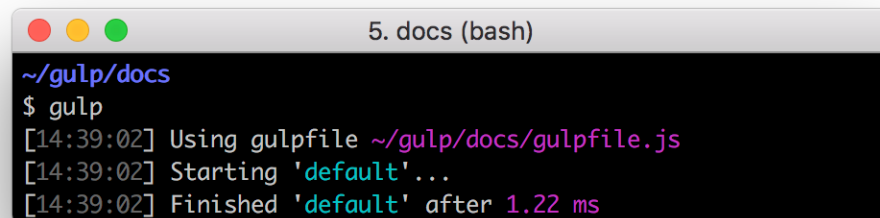
```
function Task(cb) {  
  // Beliebige Funktion einfügen.  
  cb();  
}  
  
exports.default = Task;
```

Oder man erstellt ein Verzeichnis mit dem Namen gulpfile.js, welches die Datei index.js oder andere Module beinhaltet.

Um die vorherigen Schritte zu testen benutzen wir diesen Command.

```
gulp  
oder  
gulp (name des Tasks bsw. default)
```

Das Resultat sollte dann so aussehen:

A terminal window titled "5. docs (bash)" showing the execution of the 'gulp' command. The prompt is "~/gulp/docs". The command "\$ gulp" is entered. The output shows three lines: "[14:39:02] Using gulpfile ~/gulp/docs/gulpfile.js", "[14:39:02] Starting 'default'...", and "[14:39:02] Finished 'default' after 1.22 ms".

```
~/gulp/docs  
$ gulp  
[14:39:02] Using gulpfile ~/gulp/docs/gulpfile.js  
[14:39:02] Starting 'default'...  
[14:39:02] Finished 'default' after 1.22 ms
```

Die einzelnen Elemente eines Gulpfiles

Um den Allgemeinen Aufbau zu erklären, wird im unten stehenden Code kommentiert, wie SASS und ein CSS minify per gulp ausgeführt wird, im default, watch und build.

Um zu checken welche Tasks bestehen, gibt man im Terminal den unten stehenden Code ein.

```
gulp --tasks
```

Zuweisung public Tasks

```
const gulp = require('gulp');
```

Zuweisung von Private Tasks

```
//Serie führt calls nacheinander aus  
//Parallel führt calls gleichzeitig aus  
{series, parallel} = require('gulp');
```

Integrieren von Plugins

```
//Plugins  
clean_css = require('gulp-clean-css'),  
rename = require('gulp-rename'),  
sass = require('gulp-sass')(require('sass')),
```

All diese Plugins müssen im package.json bei den dependencies eingetragen werden.

```
//package.json dependencie  
"devDependencies": {  
  "gulp": "^4.0.2",  
  "gulp-minify": "^3.1.0",  
  "gulp-rename": "^2.0.0",  
  "gulp-sass": "^5.0.0",  
  "sass": "^1.36.0"  
},  
"dependencies": {  
  "gulp-clean-css": "^4.3.0",  
  "gulp-imagemin": "^7.1.0"  
}
```

Erstellen der css Generatorfunktion

```
//Erstellt das .css des .scss  
createcss = function () {
```

```

return gulp
//.src() wählt das File aus, welches als Source dienen soll.
.src('./src/styles/style.scss')
//.pipe() setzt Pipeline, welches diverse Modulfunktionen auslösen kann.
.pipe(sass().on('error', sass.logError))
//erstellt das css File
.pipe(clean_css())
//erstellt ein .min.css File
.pipe(rename({suffix: '.min'}))
//.dest() definiert den Zielpath des geänderten Files
.pipe(gulp.dest('./dist/styles'));

```

Das Gulpfile als Ganzes

```

//Hier werden diverse Module und Methoden, den Variablen zugewiesen
//gulp wird bei jedem Gulpfile benötigt
const gulp = require('gulp'),
//Serie führt calls nacheinander aus
//Parallel führt calls gleichzeitig aus
    {series, parallel} = require('gulp'),
    {watch} = require('gulp'),
//Plugins
    clean_css = require('gulp-clean-css'),
    rename = require('gulp-rename'),
    sass = require('gulp-sass')(require('sass')),

//Erstellt das .css des .scss + minify
createcss = function () {
return gulp
//.src() wählt das File aus, welches als Source dienen soll.
.src('./src/styles/style.scss')
//.pipe() setzt Pipeline, welches diverse Modulfunktionen auslösen kann.
.pipe(sass().on('error', sass.logError))
//erstellt das css File
.pipe(clean_css())
//erstellt ein .min.css File
.pipe(rename({suffix: '.min'}))
//.dest() definiert den Zielpath des geänderten Files
.pipe(gulp.dest('./dist/styles'));

//gulp default / gilt als Default call.
exports.default = function () {
//Ruft die exports.watch Funktion
    exports.watch();
};

//gulp watch / Überwacht die per watch() definierten Files.
exports.watch = function () {
watch('src/styles/style.scss',
{ignoreInitial: false} ,

```

```
series(createcss,minifycss));  
};  
  
//gulp build / Führt die im build call aufgelisteten Funktionen in einer Serie aus.  
exports.  
  build = series(  
    createcss,  
    minifycss,  
  );
```

wichtig Elemente:

Tasks

Die task() Funktion ist depreciated

asynchrone JS Funktionen

Globs / Wildcards

**/*! sind Globbs, sie werden für das übereinstimmen eines Dateipfads verwendet.

Module / Plugins

Diese werde verwendet um GULP in kleinen Schritten zu erweitern.