Gulp-Article

/\*\*

\*Author : Sandeep Vedam

\*About : Gulp Article

\*\*/

Gulp is a build tool system or toolkit that helps to optimize your application flow.

Gulp is JavaScript task runnerer. It is open-source and available on GitHub. It helps you automate repetitive tasks such as minification, compilation, unit testing, linting, etc.

Especially Gulp for Optimizing the code and improving the performance of application.

Gulp provide different tasks so that we can easily automate our tasks in development workflow.

Gulp follows streams approach and readable format. we can write code easily and understandable way and we can expect different task outputs through pipe.

Gulp is faster than Grunt because of streams.

we can install all gulp plugins by npm package.

We should follow or love few Gulp packages. Please observe below packages that you might intrested.

\*Gulp : As i already explained about gulp

package installation : npm install gulp --save-dev

usage : var gulp = require('gulp');

website : https://www.npmjs.com/package/gulp

\*Gulp-help : It adds a default help task to gulp. we can create custom help message to our gulp tasks.

package installation : npm install gulp-help --save-dev

usage : var gulp = require('gulp-help')(require('gulp'));

gulp.task('lint', 'Lints all server side js', function () {

gulp.src('./lib/\*\*/\*.js')

.pipe(jshint())

.pipe(jshint.reporter("default"));;

});

website : https://www.npmjs.com/package/gulp-help

\*Gulp-jshint : JSHint scans a program written in JavaScript and reports about commonly made mistakes and potential bugs. The potential problem could be a syntax error, a bug due to implicit type conversion, a leaking variable or something else.

From the above task we can observe gulp.src('source files'). It will return a representation of files , that can be piped directly into plugins. Hence, we can take all those files and pipe them directly into jshint(). jshint is the function made available by the gulp-jshint plugin. This runs each file one by one through JSHint, and we then pipe the result of that through to the JSHint reporter, which is responsible for showing us the results.

package installation : npm install gulp-jshint --save-dev

usage : var jshint = require('gulp-jshint');

var gulp = require('gulp');

gulp.task('lint', function() {

return gulp.src('./lib/\*.js')

.pipe(jshint())

.pipe(jshint.reporter('YOUR\_REPORTER\_HERE'));

});

Reference : https://www.npmjs.com/package/gulp-jshint

\*Gulp-jslint : Jslint is code quality tool. It means that it organize the code in structured way. It reports syntactical mistakes. Gulp-jslint has different options to perform operations.

package installation : npm install gulp-jslint --save-dev

usage : var gulp = require('gulp');

var jslint = require('gulp-jslint');

gulp.task('default', function () {

return gulp.src(['source.js'])

.pipe(jslint({ /\* this object represents the JSLint directives being passed down \*/ }))

.pipe(jslint.reporter( 'my-reporter' ));

});

Reference : https://www.npmjs.com/package/gulp-jslint

\*Gulp-concat : It concatenates files.This will concat files by your operating systems newLine. It will take the base directory from the first file that passes through it.

Package installation : npm install --save-dev gulp-concat

usage : var concat = require('gulp-concat');

gulp.task('scripts', function() {

return gulp.src('./lib/\*.js')

.pipe(concat('all.js'))

.pipe(gulp.dest('./dist/'));

});

Reference : https://www.npmjs.com/package/gulp-concat

\*gulp-cssmin : It Minifies css.

Package installation : npm install --save-dev gulp-cssmin

usage : var gulp = require('gulp');

var cssmin = require('gulp-cssmin');

var rename = require('gulp-rename');

gulp.task('default', function () {

gulp.src('src/\*\*/\*.css')

.pipe(cssmin())

.pipe(rename({suffix: '.min'}))

.pipe(gulp.dest('dist'));

});

Reference : https://www.npmjs.com/package/gulp-cssmin

\*gulp-inject : It inject required files into your index.html. A stylesheet, javascript and webcomponent reference injection plugin for gulp. No more manual editing of your index.html

gulp-inject takes a stream of source files, transforms each file to a string and injects each transformed string into placeholders in the target stream files.

Package installation : npm install --save-dev gulp-inject

usage : var gulp = require('gulp');

var inject = require('gulp-inject');

gulp.task('index', function () {

var target = gulp.src('./src/index.html');

// It's not necessary to read the files (will speed up things), we're only after their paths:

var sources = gulp.src(['./src/\*\*/\*.js', './src/\*\*/\*.css'], {read: false});

return target.pipe(inject(sources))

.pipe(gulp.dest('./src'));

});

Reference : https://www.npmjs.com/package/gulp-inject

\*Gulp-bump : Bump any version in any file which supports semver versioning. It means that we can give version to npm package.If you are just requiring a bump for npm, consider using npm version

package installation : npm install gulp-bump --save-dev

usage : var gulp = require('gulp');

var bump = require('gulp-bump');

// Basic usage:

// Will patch the version

gulp.task('bump', function(){

gulp.src('./component.json')

.pipe(bump())

.pipe(gulp.dest('./'));

});

Reference : https://www.npmjs.com/package/gulp-bump

\*del : It will del required files.

package installation : npm install --save-dev del

usage : var gulp = require('gulp');

var del = require('del');

gulp.task('clean:mobile', function () {

return del([

'dist/report.csv',

// here we use a globbing pattern to match everything inside the `mobile` folder

'dist/mobile/\*\*/\*',

// we don't want to clean this file though so we negate the pattern

'!dist/mobile/deploy.json'

]);

});

Reference : https://www.npmjs.com/package/del

\*run-sequence : Run a series of dependent gulp tasks in order

package installation : npm install --save-dev run-sequence

usage : gulp.task('build', function(callback) {

runSequence('build-clean',

['build-scripts', 'build-styles'],

'build-html',

callback);

});

Reference : https://www.npmjs.com/package/run-sequence

\*gulp-util : It provides logging.It is used to handle the error messages. It provides methods. Please observe one of method from below usage.

package installation : npm install --save-dev gulp-util

usage : gutil.log('stuff happened', 'Really it did', gutil.colors.magenta('123'));

Reference : https://www.npmjs.com/package/gulp-util

\*gulp-load-plugins : Automatically load any gulp plugins in your package.json. This frees you up from having to manually require each gulp plugin.It Loads gulp plugins from package dependencies and attaches them to an object of your choice.

package installation : npm install --save-dev gulp-load-plugins

usage : For example i am taking package.json here

{

"dependencies": {

"gulp-jshint": "\*",

"gulp-concat": "\*"

}

}

Adding above plugins to gulp task

var plugins = require('gulp-load-plugins')();

Will result in the following happening (roughly, plugins are lazy loaded but in practice you won't notice any difference):

plugins.jshint = require('gulp-jshint');

plugins.concat = require('gulp-concat');

Reference : https://www.npmjs.com/package/gulp-load-plugins

\*jshint-stylish : It is stylish reporter. It will show highlight color on errors and warnings than normal reporter.

package installation : npm install --save-dev jshint-stylish

usage : gulp.task('default', () =>

gulp.src(['file.js'])

.pipe(jshint('.jshintrc'))

.pipe(jshint.reporter('jshint-stylish'))

);

Reference : https://www.npmjs.com/package/jshint-stylish

\*gulp-uglify : We can minify files with UglifyJS. By default mangle property is set to false to skip mangling(destroy) names.

package installation : npm install --save-dev gulp-uglify

usage : var uglify = require('gulp-uglify');

gulp.task('compress', function() {

return gulp.src('lib/\*.js')

.pipe(uglify())

.pipe(gulp.dest('dist'));

});

Reference : https://www.npmjs.com/package/gulp-uglify

\*browserify : browserify will support require('modules') in the browser by bundling up all js files and we can load single js file instead of all js files in our application.

package installation : npm install -g --save-dev browserify - I am installing browserify globally

usage : var browserify = require('browserify');

var gulp = require('gulp');

var source = require('vinyl-source-stream');

gulp.task('browserify', function() {

return browserify('./src/javascript/app.js')

.bundle()

//Pass desired output filename to vinyl-source-stream

.pipe(source('bundle.js'))

// Start piping stream to tasks!

.pipe(gulp.dest('./build/'));

});

We can manually bundle js files by running command - browserify src/app.js -o dist/bundle.js

src/app.js - source folder

dist/bundle.js - destination folder

Reference : http://browserify.org/

https://wehavefaces.net/gulp-browserify-the-gulp-y-way-bb359b3f9623?gi=42a7bc77767f - Imp

https://www.viget.com/articles/gulp-browserify-starter-faq

\*browserify-istanbul : A browserify transform for istanbul code coverage tool

Package installation : npm install --save-dev browserify-istanbul istanbul

usage : There are a few options available to browserify-istanbul when you use it from JavaScript. They are shown in the following code example:

var istanbul = require('browserify-istanbul');

// use without any options...

browserifyBundle.transform(istanbul);

// or with some options...

browserifyBundle.transform(istanbul({

// ignore these glob paths (the ones shown are the defaults)

ignore: ['\*\*/node\_modules/\*\*', '\*\*/bower\_components/\*\*', '\*\*/test/\*\*', '\*\*/tests/\*\*', '\*\*/\*.json'],

// by default, any paths you include in the ignore option are ignored

// in addition to the defaults. set the defaultIgnore option to false

// to only ignore the paths you specify.

defaultIgnore: true

}));

Reference : https://github.com/gotwarlost/istanbul#features

https://www.npmjs.com/package/browserify-istanbul

\*vinyl-source-stream : This just converts the js bundle into the type of stream gulp is expecting.

vinyl-source-stream convert the readable stream you get from browserify into a vinyl stream that is what gulp is expecting to get.

package installation : npm install --save-dev vinyl-source-stream

usage : Please explore above package (browserify)

Reference : https://www.npmjs.com/package/vinyl-source-stream

\*tsify : tsify is browserify plugin. This plugin is useful for compiling typescript.tsify will generate sourcemaps if the --debug option is set on Browserify.

Browserify transforms are not flexible enough to deal with multiple file outputs given a single file input, which means that any working Typescript compilation transform either skips the resolution step (which is necessary for complete type checking) or performs multiple compilations of source files further down the dependency graph.

tsify avoids this problem by using the power of plugins to perform a single compilation of the Typescript source up-front, using Browserify to glue together the resulting files.

package installation : npm install --save-dev tsify

usage : browserify()

.add('main.ts')

.plugin('tsify', { noImplicitAny: true })

.bundle();

Reference : https://www.npmjs.com/package/tsify

\*gulp-git : There’s an awesome git package for Gulp in gulp-git. It does everything. It commits, it tags, it pushes, and everything else

package installation : npm install --save-dev gulp-git

usage : gulp.task('tag', function () {

var pkg = require('./package.json');

var v = 'v' + pkg.version;

var message = 'Release ' + v;

return gulp.src('./')

.pipe(git.commit(message))

.pipe(git.tag(v, message))

.pipe(git.push('origin', 'master', '--tags'))

.pipe(gulp.dest('./'));

});

Reference :

https://www.npmjs.com/package/gulp-git

https://ponyfoo.com/articles/my-first-gulp-adventure

\*gulp-filter : Filter files in vinyl stream.It Enables you to work on a subset of the original files by filtering them using globbing. When you're done and want all the original files back you just use the restore stream.

package installation : npm install --save-dev gulp-filter

Reference and usage : https://www.npmjs.com/package/gulp-filter

\*gulp-tag-version : It will read the version of packages from(package.json or bower.json), and prefixes it with 'v' and tag the repositary. For example i am giving package("angularjs":"1.5.0") it will automatically prefix with 'v'. It looks like ("angularjs":"v1.5.0").

package installation : npm install --save-dev gulp-tag-version

usage : var gulp = require('gulp'),

tag\_version = require('gulp-tag-version');

// Assuming there's "version: 1.2.3" in package.json,

// tag the last commit as "v1.2.3"

gulp.task('tag', function() {

return gulp.src(['./package.json']).pipe(tag\_version());

});

Reference : https://www.npmjs.com/package/gulp-tag-version

\*gulp-nsp : nsp is the main command line interface to the Node Security Project. It allows for auditing a package.json file against the API. A gulp module that runs Node Security check.

package installation : npm install gulp-nsp --save

usage : var gulpNSP = require('gulp-nsp');

//To check your project

gulp.task('nsp', function (cb) {

gulpNSP({package: \_\_dirname + '/package.json'}, cb);

});

reference : https://www.npmjs.com/package/gulp-nsp

https://nodesecurity.io/tools