**Таск-менеджер**

Таск-менеджер — небольшое приложение, которое используется для автоматизации скучных и рутинных, но, от того, не менее важных, задач, которые приходится постоянно выполнять в процессе разработки проекта. Такие задачи включают в себя, к примеру, запуск модульных тестов, конкатенацию файлов, минификацию, препроцессинг CSS. Просто создав таск-файл, вы можете проинструктировать таск-менеджер, каким образом следует выполнить ту или иную задачу. И после этого вы можете заниматься своими делами. Довольно простая идея, которая позволяет сэкономить очень много времени, и помогает сохранять фокус на задачах, связанных непосредственно с разработкой проекта.

**GruntJS**

Большинство JS разработчиков уже используют какие-то инструменты компоновки для своих разработок, даже если не знают или не используют этот термин. Они объединяют файлы при разработке, уменьшают код JavaScript-а, чтобы ускорить загрузку страниц и конвертировать Sass, или уменьшают количество файлов в CSS для браузера, и много чего другого. Чаще всего это разные инструменты, что есть не очень удобно.

Grunt помогает управлять всеми этими шагами в одном месте и организовать сторонние компоненты.

Все дополнительные функции основаны на плагинах, так что вы можете настроить его для особенных потребностей вашего проекта без каких-либо наворотов. Grunt и все подключаемые модули добавляются в директорию вашего проекта, чтобы убедиться, что все, что нужно, чтобы построить и поддерживать ваш проект, всегда остается на месте. Эти модули можно либо загружать вместе с проектом, или же если платформы для разработки будут отличаться, то лучше все используемые модули хранить в файле, грант файле

Преимущества:

1. Уменьшение JavaScript и CSS,
2. интеграция предварительных процессоров для CSS и JavaScript таких, как Sass/Compass, Less and Stylus, CoffeeScript или LiveScript Сасс / Compass,
3. оптимизация размера файлов изображения PNG и JPEG и автоматическое встраивание изображений, и многое другое.
4. интегрирует оптимизации для Require.js и Google Closure Compiler
5. позволяет предварительной компиляции шаблонов Handlebars, Jade, Underscore, Mustache, Eco или Hogan.
6. Также уже есть плагины для наиболее распространенных каркасов для тестирования таких, как Jasmine, Mocha, QUnit и Cucumber.

Установка:

1. npm install grunt-cli –g
2. package.json
3. Файл настроек:

//Стандартный экспорт модуля в nodejs

module.exports = function(grunt) {

// Инициализация конфига GruntJS

grunt.initConfig({

//Настройки различных модулей GruntJS, их нужно предварительно установить через менеджер пакетов npm, или добавить в файл package.json перед запуском npm install

//Например проверка кода javascript с помощью утилиты jshint

jshint: {},

//Склеивание файлов

concat: {}

//И так далее

});

//Загрузка модулей, которые предварительно установлены

grunt.loadNpmTasks('grunt-contrib-jshint');

grunt.loadNpmTasks('grunt-contrib-concat');

//Эти задания будут выполнятся сразу же когда вы в консоли напечатание grunt, и нажмете Enter

grunt.registerTask('default', ['jshint', 'concat']);

};

1. Три различных метода для работы с Grunt:

* grunt.initConfig
* grunt.loadNpmTasks
* grunt.registerTask

Каждая задача может быть вызвана отдельно из командной строки, передав его имя в качестве параметра командной строки: Например, grunt jshint будет выполнять только задачи, направленные на контроль стиля программирования на соответствие стандартам вашего проекта. С grunt.registerTask вы определяете задачу по умолчанию для запуска, когда Grunt вызывается без параметров. Он выполняет перечисленные задачи в заданном порядке, в нашем случае jshint, concat and uglify.

GulpJS

Gulp.js это потоковый сборщик проектов на JS. Он использует Stream и действительно является очень быстрым. Gulp, как система, построена вокруг концепции потоковой передачи данных. Потоковая передача дает вам больше контроля над происходящим и избавляет вас от промежуточных папок и файлов. Вы передаете файл в gulp, а затем сохраняете результат в другой файл. Всё очень просто.

var gulp = require('gulp');

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

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

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

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

// Линтинг файлов

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

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

.pipe(jshint())

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

});

// Конкатенация и минификация файлов

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

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

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

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

.pipe(rename('all.min.js'))

.pipe(uglify())

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

});

// Действия по умолчанию

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

gulp.run('lint', 'minify');

// Отслеживаем изменения в файлах

gulp.watch("./src/\*.js", function(event){

gulp.run('lint', 'minify');

});

});

В gulp нет необходимости в плагине watch, потому как возможность реагировать на изменения в файлах уже включена в ядро. Это одна из тех функциональностей, которую вы ожидаете увидеть по умолчанию, а не посредством расширения возможностей с помощью плагина.

В дополнение, переименование файла в Grunt выполняется с помощью плагина uglify. Получается, что один и тот же плагин отвечает как за минификацию кода, так и за переименование файла на выходе. В gulp же каждый плагин выполняет одно простое действие, и отвечает только за него. Для переименования файла мы просто подключим плагин gulp-rename и добавим его в нашу задачу по минификации кода.

**Источники**

http://frontender.info/no-need-to-grunt-take-a-gulp-of-fresh-air/