# Gulp. Инструкция

06 DECEMBER 2017

1.**Watch-таск**. Watch-таск позволяет наблюдать за ситуацией в папках. Обычно при изменении файлов нам приходится заново запустить команду `gulp scripts`. Но watch-таск все делает за нас. Он наблюдает за изменениями, и если они состоялись, то вызывает необходимый таск.

```
// в этой секции подключаем все необходимые плагины
gulp.task('watch', function() {
    gulp.watch('js/main.js')
});
```

А теперь введите команду:

```
gulp watch
```

2.**Gulp-concat**. Отвечает за объединение файлов. Объединяет содержимое нескольких файлов в один. Подключается по принципу `gulp-uglify`:

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

Добавляем в начало в перечисление переменных:

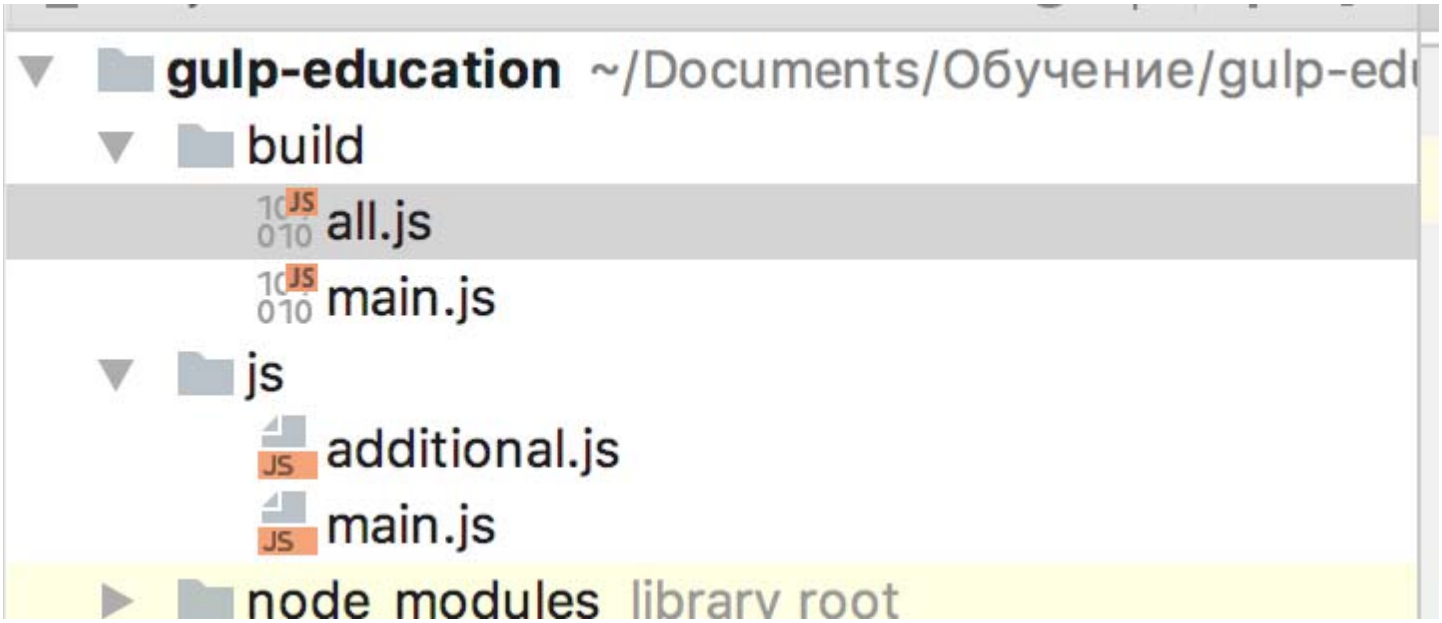
```
...
clean = require('gulp-clean')
...
```

И меняем таск `scripts`:

```
...
gulp.task('scripts', function () {
    gulp.src('./js/*.js')
        .pipe(concat('all.js'))
        .pipe(uglify())
        .pipe(gulp.dest('build'));
});
...
```

После изменения `gulpfile.js` всегда надо перезапускать `gulp scripts`.

3.**Gulp clean**. Выполнили таск, и у нас образовалось два выходных файла в папке `build` (старый и новый), что явно противоречит логике.




Чтобы очищать содержимое папки перед ее следующим конфигурированием, необходим плагин `gulp-clean`. Подключаем по старой схеме:

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

Добавляем в начале в перечисление переменных:

```
...
concat = require('gulp-concat')
...
```

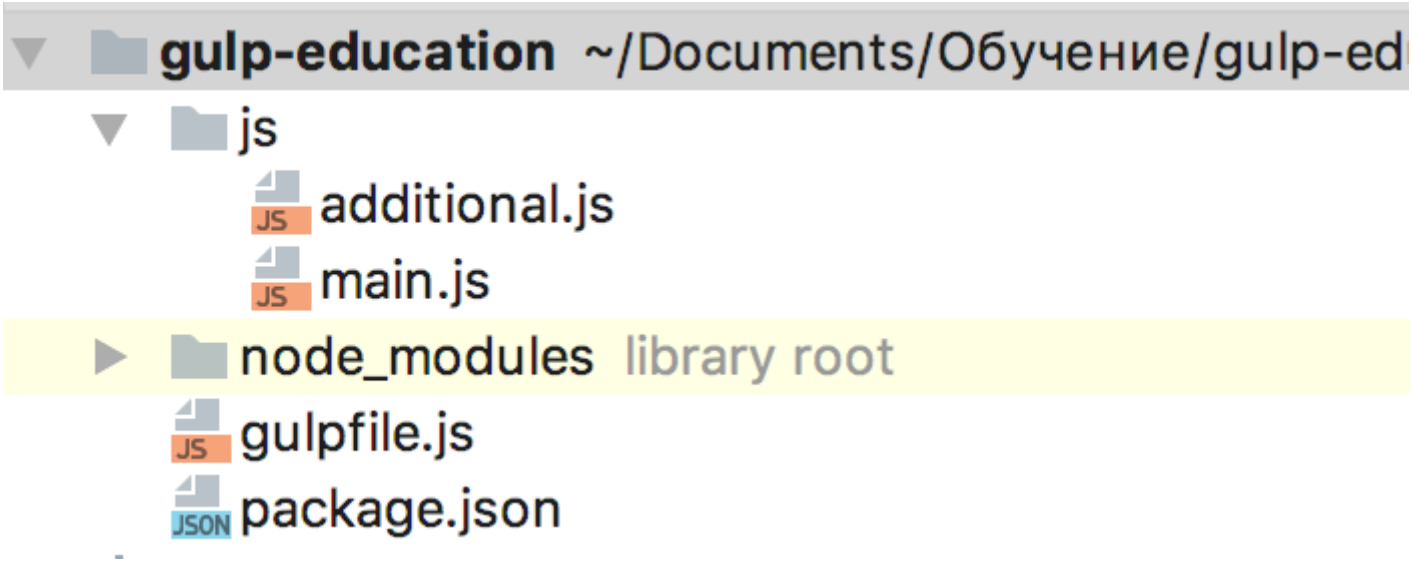
Пишем, что хотим очистить содержимое папки `build` перед добавлением в нее нового содержания:

```
...
gulp.task('clean', function () {
  return gulp.src('build/', {read: false})
    .pipe(clean());
});
...
```

Вызываем новый таск старым методом:

```
gulp clean
```

Все, теперь папки нет:




4.**Таски для CSS**. Мы работали с js-файлами. А теперь давайте провернем все то же самое с CSS. На этот раз за минификацию будет отвечать плагин `cssnano`, за объединение файлов - все тот же `concat`, и мы воспользуемся дополнительным плагином `autoprefixer` для добавления вендорных значений в CSS-стили (чтобы не писать *webkit-*, *-ms-* и др. свойства для каждого браузера, а указывать их один раз (*display: flex*, к примеру).

Посмотрим, что получилось:

```
npm install gulp-cssnano --save-dev
npm install --save-dev gulp-autoprefixer
```

```
...
cssnano = require('gulp-cssnano'),
autoprefixer = require('gulp-autoprefixer')
...
```

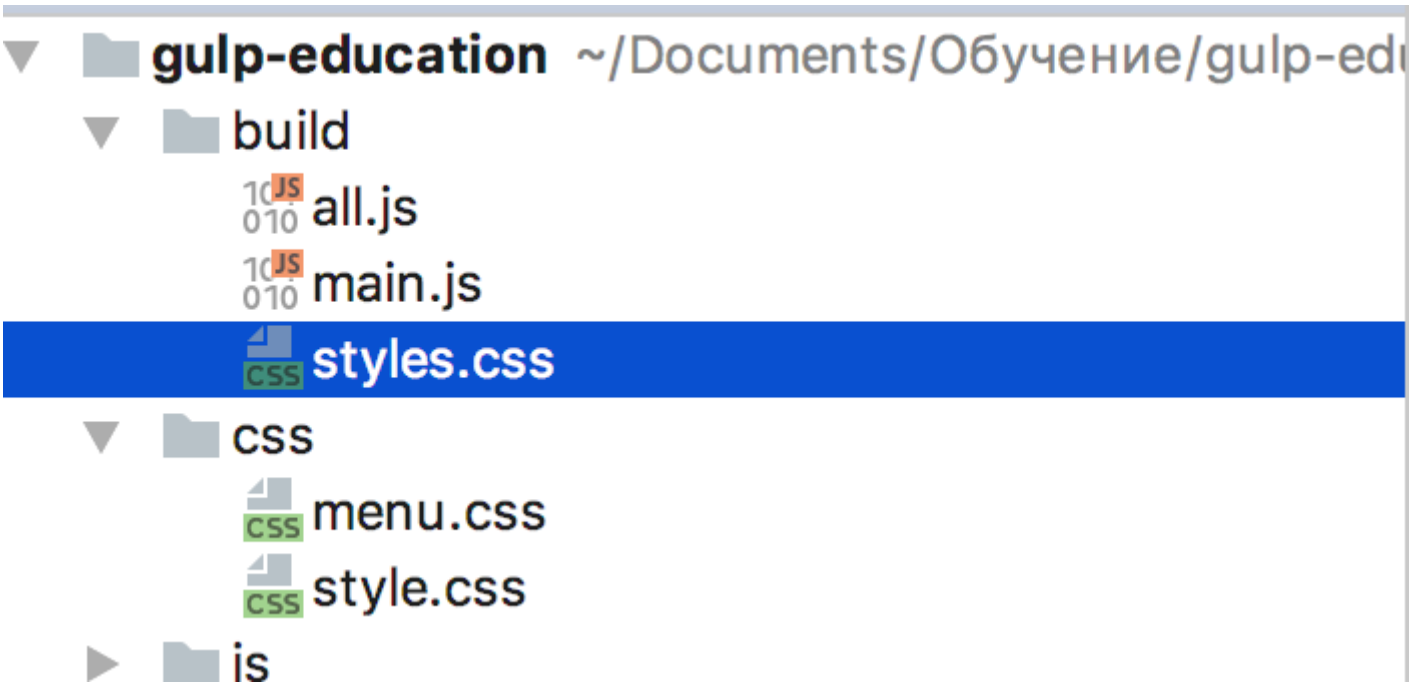
Внедряем таск для CSS:

```
...
gulp.task('styles', function () {
    gulp.src('./css/*.css')
        .pipe(autoprefixer())
        .pipe(concat('styles.css'))
        .pipe(cssnano())
        .pipe(gulp.dest('build'));
});
...
```

Вызываем необходимую задачу

```
gulp styles
```

УРА! Теперь в папке build образовался `styles.js`.




Добавим таск для перемещения HTML в `build`.

```
...
gulp.task('html', function() {
```

```
    gulp.src('./index.html')
        .pipe(gulp.dest('build/'))
});
...
```

5.**Default task**. Все предыдущие разы мы вызывали необходимый нам таск командами `gulp scripts/gulp styles/gulp watch`. Но можно создать один единый таск, объединяющий остальные. Посмотрим, как этом делается.

```
...
gulp.task('default',gulp.parallel('scripts', 'images', 'svg'));
...
```

Мы только что сказали: "Объедини в таск default две других задачи". Вместо `default` может быть любое имя. Но именно `default` сокращает доступ к задаче и позволяет вызывать ее единственным словом:

```
gulp // подхватится таск по умолчанию, т.е. default
```

6.**Синхронность и асинхронность**. Вспомним предыдущую статью, раздел "Скользкие моменты", часть 2.

*"Все таски выполняются сихронно, т.е. одновременно".*

Всегда ли это необходимо? Нет. Особенно, если мы работаем с таском `clean`.

Дело в том, что очищать содержимое папки необходимо строго до того, как новые файлы будут сконфигурированы. Для этой цели есть следующая конструкция:

```
...
gulp.task('default', gulp.series('clean', gulp.parallel('scripts', 'im
...
```

Которая говорит: "Создай таск `default` таким образом, чтобы мы сначала выполнили задачу `clean`, а уже после нее приступили к таскам `scripts` и `styles`".

Запускаем

```
gulp
```

И все работает!

7.**BrowserSync**. Обычно таски в `gulpfile` пишутся не ради тренировки, а для запуска реальных проектов. Но чтобы увидеть проект, нам необходимо поднять сервер. Давайте сделаем это!

Подключаем необходимый плагин для *Gulp*:

```
npm install browser-sync gulp --save-dev
```

Добавляем его в наш список плагинов:

```
...
browserSync = require('browser-sync').create();
...
```

Создаем задачу для запуска сервера:

```
...
gulp.task('browser-sync', function() {
    return browserSync.init({
        server: {
            baseDir: './build/'
        },
        port: 3000,
        host: 'localhost',
        logPrefix: 'frontend',
        open: true
    });
});
...
```

Добавляем новый таск в `default`.

```
...
gulp.task('default', gulp.series('clean', gulp.parallel('scripts', 'in
...
```

... Но отображать нам пока нечего =( . Давайте изменим наш главный файл `index.html`:

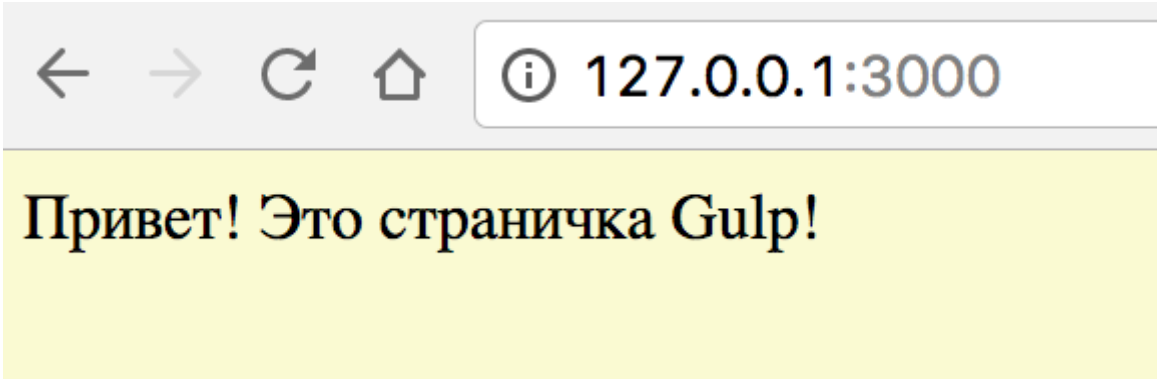
```
...
<!DOCTYPE html>
<html>
<head>
    <title>Test Gulp project</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="./styles.css">
</head>
<body>

Привет! Это страничка Gulp!

<script src="./all.js"></script>

</body>
</html>
...
```

И запустим наш основной таск `gulp`. УРА!!! Теперь мы видим результат:




8.И напоследок: отредактируем наш watch-таск, который с момента его добавления претерпел значительные изменения:

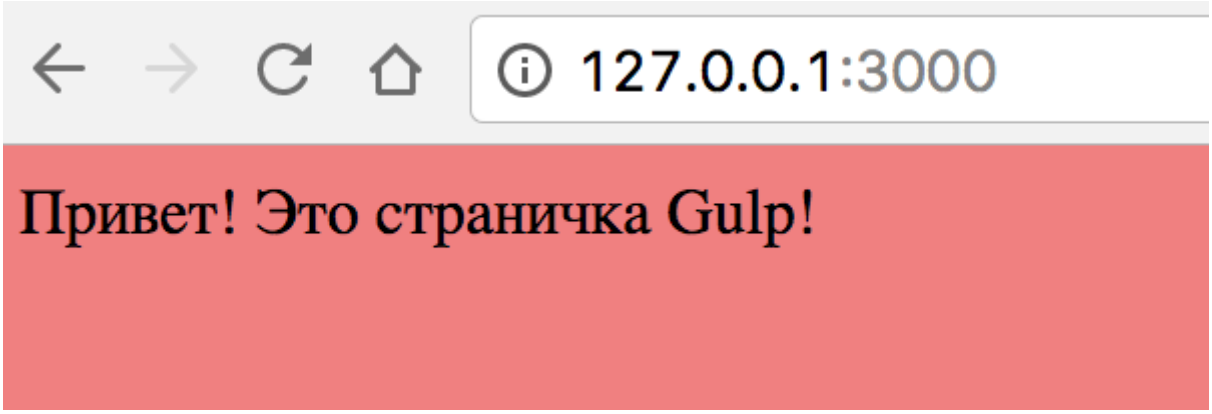
```
...
gulp.task('watch', function() {
    gulp.watch('./js/*.js', gulp.series('scripts'));
    gulp.watch('./css/*.css', gulp.series('styles'));
    gulp.watch('./*.html', gulp.series('html'));
    gulp.watch('./*.images', gulp.series('images'));
    gulp.watch('./*.svg', gulp.series('svg'));
```

```
});
...
```

Запускаем watch-таск в отдельном терминале в той же папке, где запускали gulp:

```
gulp watch
```

Попробуйте поиграться со стилями в css, обновите страничку и оцените результат (без перезапуска команды `gulp`):




На этом ВСЕ!

Приведем полный код `gulpfile.js`:

```
...
var gulp = require('gulp'),
    uglify = require('gulp-uglify'),
    concat = require('gulp-concat'),
    imagemin = require('gulp-imagemin'),
    svgmin = require('gulp-svgmin'),
    clean = require('gulp-clean'),
    cssnano = require('gulp-cssnano'),
    autoprefixer = require('gulp-autoprefixer'),
    browserSync = require('browser-sync').create();


gulp.task('scripts', function () {
    gulp.src('js/*.js')
        .pipe(concat('all.js'))
        .pipe(uglify())
        .pipe(gulp.dest('build'));
```

```
});


gulp.task('images', function () {
    gulp.src('images/*')
        .pipe(imagemin())
        .pipe(gulp.dest('build/images'))
});

gulp.task('svg', function () {
    gulp.src('svg/*')
        .pipe(svgmin())
        .pipe(gulp.dest('build/svg'));
});

gulp.task('clean', function () {
    return gulp.src('build/', {read: false})
        .pipe(clean());
});

gulp.task('styles', function () {
    gulp.src('./css/*.css')
        .pipe(autoprefixer())
        .pipe(concat('styles.css'))
        .pipe(cssnano())
        .pipe(gulp.dest('build'));
});

gulp.task('html', function() {
    gulp.src('./index.html')
        .pipe(gulp.dest('build/'))
});

gulp.task('browser-sync', function() {
    return browserSync.init({
        server: {
            baseDir: './build/'
        },
        port: 3000,
        host: 'localhost',
        logPrefix: 'frontend',
        open: true
    });
});

gulp.task('watch', function() {
    gulp.watch('./js/*.js', gulp.series('scripts'));
    gulp.watch('./css/*.css', gulp.series('styles'));
    gulp.watch('./*.html', gulp.series('html'));
```

```
    gulp.watch('./*.images', gulp.series('images'));
    gulp.watch('./*.svg', gulp.series('svg'));
});

gulp.task('default', gulp.series('clean', gulp.parallel('scripts', 'im

...
```

## Домашнее задание

Внедрите все вышеописанное в свой проект.



## @Afelua

Read more posts by this author.

*Moscow*

## Share this post