

عنوان:	Gulp #1
نویسنده:	م محمد شریفی
تاریخ:	۱۱:۱۰ ۱۳۹۴/۰۶/۱۶
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
گروه‌ها:	JavaScript, Gulp, Grunt

**Gulp** ابزاری ست که شما را در انجام دادن کارهای مختلف توسعه‌ی وب، در سمت Front-end کمک می‌کند و اغلب برای کارهایی هم‌چون موارد ذیل بکار می‌رود:

- راه اندازی یک وب سرور
- بارگذاری مجدد مرورگر به صورت خودکار بعد از ذخیره‌ی هر فایل
- تبدیل پیش پردازنده‌های CSS مانند LESS, SASS به CSS
- بهینه سازی فایل‌های asset شامل CSS,JS و همچنین عکسها

و در طی این سری آموزشی، همه‌ی آنها را پوشش خواهیم داد.



gulp.js

The streaming build system

البته این‌ها تنها چیزهایی نیستند که گالپ می‌تواند آنها را انجام دهد. اگر علاقمند باشید، می‌توانید یک سازنده‌ی سایت ایستا را با گالپ درست کنید! گالپ واقعا قدرتمند است و در این سری آموزش‌ها قرار است با آن آشنا شویم و از آن استفاده کنیم. قبل از اینکه کار با گالپ را شروع کنیم، به این می‌پردازیم که چرا باید گالپ را از میان ابزارهای مشابه انتخاب کرد.

## چرا گالپ؟

ابزارهای مشابه گالپ، تحت عنوان نام "build tools" یا ابزارهای ساخت شناخته می‌شوند. دو مورد از معروف‌ترین آنها Gulp و Grunt هستند. اما در سویی دیگر، ابزاری به نام [Boroccoli](#) داریم که تمرکزش را بر روی کامپایل فایل‌های asset گذاشته است. تفاوت اصلی بین آنها، چگونگی پی‌کربندی workflow تان با آن است.



Gulp در مقایسه با گرانت، پی‌کربندی مختصر و آسان‌تری دارد و سریع‌تر نیز اجرا می‌شود. اگر می‌خواهید بیشتر در مورد تفاوت Grunt و Gulp بدانید مراجعه کنید به [+](#) ، [+](#) و [+](#).

## نصب گالپ

برای نصب گالپ باید NodeJS را بر روی سیستم خود نصب داشته باشید. برای اینکار به [سایت ند جی اس](#) مراجعه کنید و آن را دانلود کرده و نصب کنید. پس از این‌کار حالا در خط فرمان سیستم عامل خود، دستور زیر را برای نصب گالپ وارد کنید:

```
sudo npm install -g gulp
```

**npm install** دستوری است که با استفاده از Node Package Manager یا **npm** یک پکیج گالپ را از مخزن‌های Node دانلود و بر روی سیستم شما نصب می‌کند. توجه کنید که فقط کاربران مک و لینوکس می‌توانند از **sudo** استفاده کنند. برای ویندوز باید خط فرمان خود را به صورت Run As Admin باز کنید؛ در غیر این صورت ممکن است در هنگام نصب، با خطا مواجه شوید. **-g** یک دستور پرچم است که باعث نصب گالپ به صورت global بر روی سیستم شما می‌شود تا در کل سیستم شما، دستور گالپ در دسترس باشد و نه منحصر در مسیر نصب کنونی. بسته به سرعت اینترنت شما، زمانی طول خواهد کشید و در نهایت برای اینکه مطمئن شوید که گالپ بر روی سیستم شما نصب شده‌است، دستور زیر را در خط فرمان وارد کنید:

```
gulp -v
```

اگر خروجی مشابه زیر را مشاهده کردید، نصب گالپ با موفقیت انجام شده‌است:

```
CLI version 3.9.0
```

Local version 3.9.0 در قسمت بعدی به ساخت یک پروژه با گالپ خواهیم پرداخت و درک مفاهیم گفته شده در این جلسه را که شاید برایتان گنگ باشد، به صورت عملی کار خواهیم کرد.

## نظرات خوانندگان

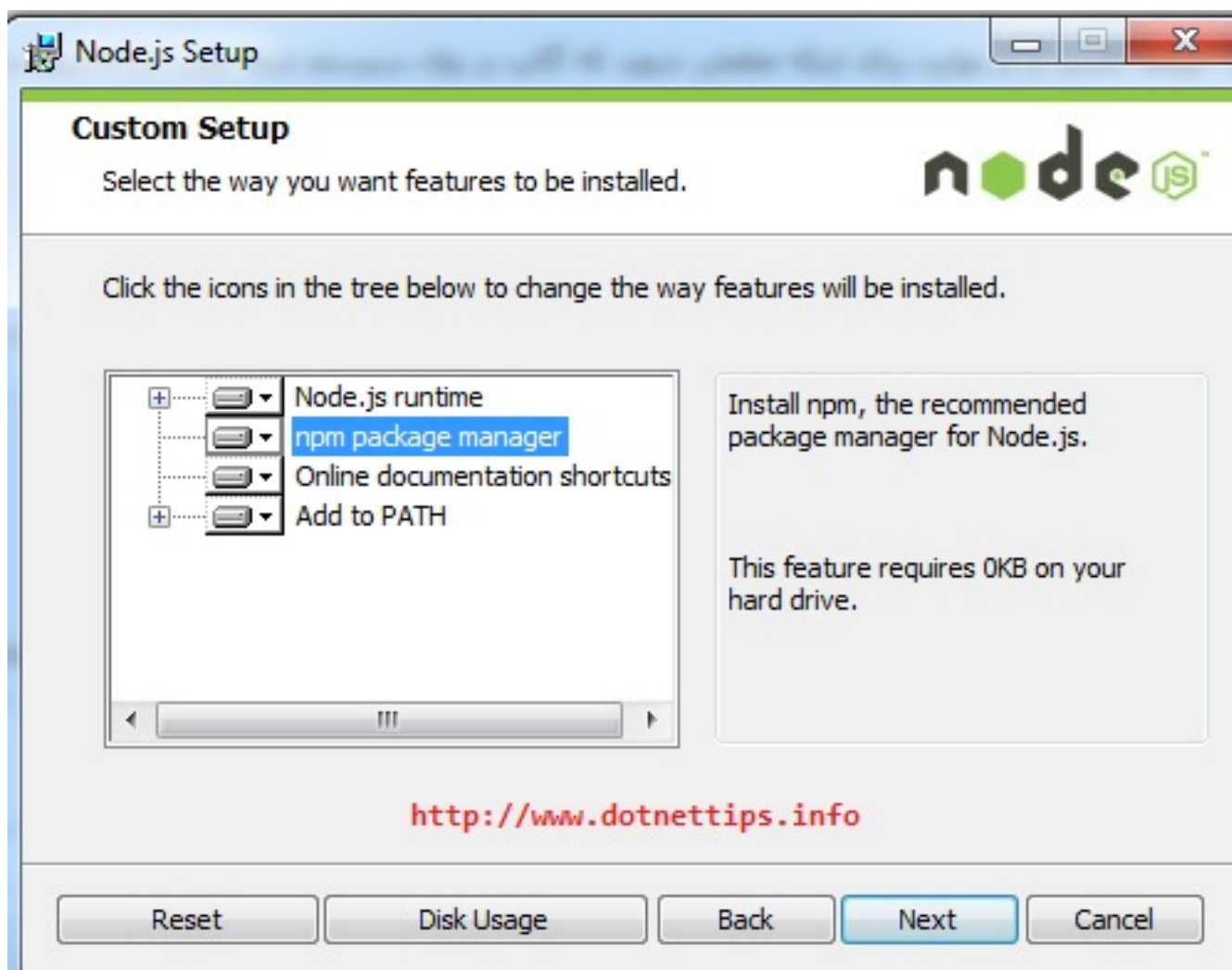
نویسنده:

محمود راستین

تاریخ:

۱۸:۴۰ ۱۳۹۴/۰۷/۰۲

مرسی بابت این مقاله. در زیر یک توضیحاتی در مورد اطمینان از نصب nodeJs دادم  
در همون مسیری که در مقاله ذکر شده NodeJs رو دانلود و نصب کنید. توجه داشته باشید در هنگام نصب حتما npm package manager رو انتخاب کنید :



برای اینکه بدونیم NodeJs رو سیستم نصب هستش باید دستور زیر رو در خط فرمان اجرا کنیم :

```
node -v
```

خروجی باید معادل آخرین نسخه نصبی باشه یعنی به این صورت :

```
v4.1.1
```

و برای اینکه اطمینان داشته باشیم npm هم نصب هستش کافیه دستور زیر رو اجرا کنیم :

```
npm -v
```

و خروجی ، مثل دستور قبلی باید نسخه npm رو نمایش بده ، یعنی چیزی مثل خروجی زیر :

```
2.14.4
```

برای تست نهایی و اجرای یک فایل js برای اینکه اطمینان داشته باشیم همه چیز خوب پیش رفته ، یک فایل js با نام dotnettips\_info.js درست کنید و محتوای اون رو به این صورت بنویسید :

```
console.log('Hello www.dotnettips.info');
```

و دستور زیر رو به خط فرمان صادر کنید :

```
node dotnettips_info.js
```

و خروجی زیر رو باید ببینید :

```
Hello www.dotnettips.info
```

و در نهایت چیزی همانند زیر باید داشته باشید :

```
D:\>node -v
v4.1.1

D:\>npm -v
2.14.4

D:\>node dotnettips_info.js
Hello www.dotnettips.info
```

عنوان:	Gulp #2
نویسنده:	م محمد شریفی
تاریخ:	۱۰:۴۵ ۱۳۹۴/۰۶/۱۷
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
گروه‌ها:	JavaScript, Twitter Bootstrap, bower, Gulp

در [قسمت قبلی](#) بحث کردیم که گالپ چیست و چه کاربردی دارد و در نهایت آن را بر روی سیستم خود نصب کردیم. در این مقاله و مقالات بعد می‌خواهیم کار خود را با راه اندازی یک workflow برای بوت استرپ، روند شخصی سازی آن را بسیار آسان و لذت بخش‌تر کنیم. امیدوارم که برای ادامه‌ی این بحث هیجان انگیز آماده باشید!

## ساخت پروژه گالپ

ابتدا یک پوشه‌ی دلخواه به نام **project** را درست کنید. سپس خط فرمان خود را به این مسیر تغییر دهید و در نهایت دستور زیر را وارد کنید:

```
npm init
```

این دستور برایمان یک فایل *package.json* می‌سازد تا هم مشخصات پروژه را مثل نام، ورژن، نام توسعه دهنده، مخزن و ... مشخص کنیم و هم وابستگی‌های آن را، تا توسعه دهندگان دیگر، هنگام استفاده از پروژه، با مشکل مواجه نشوند و فقط با اجرای دستور `npm install` تمام وابستگی‌های پروژه را نصب کنند. حتما مشاهده کرده‌اید که این دستور چند سوالی را از شما می‌پرسد. برای نمونه من آنها به این صورت پاسخ می‌دهم و در نهایت از من یک تایید می‌گیرد که **yes** را می‌زنم.

```
name: (Gulp-RTLbootstrap-fontawesome)
version: (1.0.0)
description: An Awesome workflow
entry point: (index.js) index.html
test command: test
git repository: https://github.com/mmdsharifi/gulp-rtlBootstrap-fontawesome.git
keywords: gulp, rtlbootstrap, persian bootstrap
author: Mohammad Sharifi
license: (ISC) MIT
```

الان فایل *package.json* درست شده و چون ما در این پروژه می‌خواهیم از گالپ استفاده کنیم، یکی از وابستگی‌های پروژه‌ی ما گالپ خواهد بود. بدین معنی که اگر بخواهیم پروژه را توسعه دهیم و گالپ نصب نشده باشد، با مشکل مواجه می‌شویم.

## نصب گالپ

در خط فرمان دستور زیر را وارد کنید تا گالپ در این پروژه نصب شود.

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

تفاوتی آن با دستوری که در مقاله‌ی قبلی اجرا کردیم، این است که این دستور فقط گالپ را در مسیر جاری در فولدر `node_modules` نصب می‌کند و **save-dev** آن را به وابستگی‌های توسعه‌ی پروژه در فایل *package.json* اضافه می‌کند. گام بعدی، ساخت فایلی است که گالپ به آن نیاز دارد تا با استفاده از آن، تسک‌ها و کارهایی را که برایش نوشته‌ایم، از آن بخواند و اجرا کند.

## ایجاد فایل *gulpfile.js*

این فایل را یا به صورت دستی ایجاد کنید یا با خط فرمان با دستور ذیل:

```
touch gulpfile.js
```

و حالا آن را در ویرایشگر مورد علاقه‌ی خود باز کنید. من از ویرایشگر [Atom](#) استفاده می‌کنم. اما notepad هم کفایت می‌کند.

## نوشتن اولین تسک گالپ

در ویراشگر خط زیر را می‌نویسیم:

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

require به Node می‌گوید که به فولدر node\_modules برای پکیج gulp نگاه کند. زمانیکه آن را پیدا کرد، آن را به متغیر gulp انتساب می‌دهیم تا از تابع‌های گالپ بتوانیم استفاده کنیم. حال می‌خواهیم اولین تسک خود را بنویسیم:

```
gulp.task('task-name', function() {
  // Stuff here
});
```

گالپ دارای [۴ تابع](#) task,src,dest,watch است که با آنها آشنا خواهیم شد. task یک کار را برای گالپ تعریف می‌کند و [سه پارامتر](#) دارد. اولی نام تسک، دومی (اختیاری) وابستگی این تسک (بدین معنا که اول باید این وظیفه اجرا شود، سپس تسک جاری) و در نهایت تابع درون تسک‌ها را می‌نویسیم. برای مثال:

```
gulp.task('hello', function() {
  console.log('Hello Gulp !');
});
```

فایل را ذخیره کنید. می‌خواهیم به گالپ بگوییم که این وظیفه را انجام دهد. کافی است **gulp hello** را در خط فرمان وارد کنیم. نتیجه به صورت زیر خواهد بود:

```
mmdsharifi@mmdsharifi ~/Work/Gulp-RTLbootstrap-fontawesome $ gulp hello
[09:10:44] Using gulpfile ~/Work/Gulp-RTLbootstrap-fontawesome/gulpfile.js
[09:10:44] Starting 'hello'...
hello Gulp !
[09:10:44] Finished 'hello' after 83 μs
```

البته که تسک‌هایی که برای گالپ می‌نویسیم، کاراتر از این است؛ برای مثال:

```
gulp.task('task-name', function () {
  return gulp.src('source-files') // Get source files with gulp.src
    .pipe(aGulpPlugin()) // Sends it through a gulp plugin
    .pipe(gulp.dest('destination')) // Outputs the file in the destination folder
});
```

با استفاده از متد src به گالپ می‌گوییم که مسیر مبدأ فایل‌ها، برای انجام تسک کجا است و dest هم بعد از انجام تسک، فایل‌های خروجی را به مقصد مشخص می‌برد. متد pipe یک تابع ند جی اس است که مطابق [مستندات](#) خودش متدی است که تمام [جریان](#)‌های قابل خواندن را واکنشی می‌کند و به مسیری [که به صورت آرگومان به عنوان مقصد] داده شده، هدایت می‌کند. شاید در ابتدا نوشتن تسک برایتان کمی پیچیده باشد، اما بعد از ساخت اولین پروژه با گالپ، خواهید دانست که تسک نوشتن برای گالپ کاری بسیار آسان و شیرین است!

[مخزن پروژه](https://github.com/mmdsharifi/gulp-rtlBootstrap-fontawesome) در گیت هاب : <https://github.com/mmdsharifi/gulp-rtlBootstrap-fontawesome>  
نام کامیت این قسمت: Init commit

در مقاله بعدی gulp را در کنار [bower](#) بکار خواهیم برد. بهتر است مطالعه‌ای در مورد bower نیز انجام دهید. (پیشنهاد: [+ و +](#))

## نظرات خوانندگان

نویسنده: محمود راستین  
تاریخ: ۱۹:۹ ۱۳۹۴/۰۷/۰۲

تشکر بابت این مقاله. فقط یک نکته رو بیان کنم که فرمان برای پلتفرم ویندوز نیست. اگر دستور :

```
touch gulpfile.js
```

رو صادر کنیم با خطای زیر رو به رو میشیم :

```
'touch' is not recognized as an internal or external command, operable program or batch file.
```

برای ایجاد فایل خالی در ویندوز باید اینکارو بکنیم :

```
echo $null >> gulpfile.js
```

این دستور همانند دستور touch هستش. با این دستور فایل gulpfile.js در مسیر پروژه ساخته میشه.

نویسنده: م محمد شریفی  
تاریخ: ۹:۲۳ ۱۳۹۴/۰۷/۰۳

خواهش می‌کنم.

بله درسته دستور touch در لینوکس به درستی کار می‌کند.

سپاس از اشتراک گذاری این نکته.

در [قسمت اول](#) گالپ را معرفی کردیم و در [مقاله قبلی](#) به نوشتن اولین تسک با گالپ پرداختیم. در این قسمت می‌خواهیم با نصب bower، پروژه‌ی workflow بوت استرپ راستچین شده را انجام دهیم.

نصب bower



[bower](#) یک مدیریت پکیج سمت Front end است و از مزایای استفاده از آن می‌توان به موارد زیر اشاره کرد:

ساده کردن تعریف وابستگی‌های منابع پروژه با تعریف یک فایل **bower.json**

نیازی به commit کردن وابستگی‌های پروژه نیست.

با ذکر ورژن مربوط به وابستگی یا محدوده‌ی قابل قبول برای آن، به روز رسانی منابع به سادگی با یک دستور انجام می‌شود.

**وابستگی‌های وابسته به یک منبع را نیز نصب می‌کند.** برای مثال زمانیکه بوت استرپ را به عنوان وابستگی پروژه تعریف می‌کنیم،

وابستگی آن یعنی jquery را چون در فایل **bower.json** بوت استرپ تعریف شده‌است، به صورت خودکار دانلود می‌کند.

در نهایت افراد هم تیمی یا توسعه دهندگان دیگر به راحتی با زدن دستور **bower install** تمام وابستگی‌های پروژه را می‌توانند نصب کنند.

برای نصب آن کافی است دستور زیر را بزنید و بعد از نصب نیز دستور خط دوم را در مسیر پروژه وارد کنید تا یک فایل

**bower.json** را برایمان بسازد. برای اینکار به سوال‌هایی که می‌پرسد باید جواب دهیم. تنها نکته‌ای که قابل ذکر است، پاسخ به

سوال **what types of modules does this package expose ?** است که باید گزینه‌ی Node را انتخاب کنید.

```
sudo npm install -g bower
bower init
```

حال می‌خواهیم وابستگی‌های پروژه را نصب کنیم که عبارتند از **bootstrap-sass,fontawesome,bootstrap-rtl**:

```
bower install bootstrap-sass-official --save
bower install fontawesome --save
bower install bootstrap-rtl --save
```



نکته : عبارت `--save` وابستگی مربوطه را به `bower.json` اضافه می‌کند. اگر نصب با موفقیت صورت گرفته باشد، پکیج‌های مربوطه را می‌توانید در فولدر `bower_components` در `root` پروژه مشاهده کنید.

### نصب پلاگین‌های مورد نیاز gulp

ما می‌خواهیم که بوت استرپ و نگارش `sass` آن‌را کامپایل کنیم و همچنین وابستگی‌های `bower` پروژه را از طریق گالپ نصب کنیم تا نیازی به زدن `bower install` نباشد و توسعه دهنده‌ی پروژه فقط با زدن `npm install`، تمام وابستگی‌های پروژه‌ی ما را نصب کند. می‌توان تمام پلاگین‌ها را پشت سر هم با یک دستور نصب کرد و یا به صورت جداگانه این کار را انجام داد.

```
sudo npm install gulp gulp-ruby-sass gulp-notify gulp-bower --save-dev
```

نکته: پلاگین `gulp-notify` به منظور نشان دادن خطاها در ترمینال است؛ تا در صورت وجود اشتباه در کامپایل فایل‌های `Sass`، کل روند گالپ متوقف نشود.

نکته ۲: برای اینکه کامپایل `sass` انجام شود نیاز به `Ruby` دارید. برای ویندوز می‌توانید از [روبی اینستالر](#) استفاده کنید.

### نوشتن تسک‌ها برای گالپ

به قسمت مهم و هیجان انگیز کار رسیدیم! همان طور در مقاله قبلی گفتیم، ابتدا باید ماژول‌هایی را که نصب کردیم، `include` کنیم به این صورت:

```
var gulp = require('gulp'),
    sass = require('gulp-ruby-sass'),
    notify = require('gulp-notify'),
    bower = require('gulp-bower');
```

برای اینکه دسترسی به مسیرهای مهم پروژه آسان‌تر شود، آن‌ها را درون یک شیء نگه داری می‌کنیم.

```
var config = {
  sassPath: './resources/sass',
  bowerDir: './bower_components'
}
```

تسک `baor` را اضافه می‌کنیم تا کار `bower install` را خودکار کنیم. مزیت این کار این است اگر یک هم تیمی، پکیج جدیدی را در حین توسعه‌ی پروژه نصب کرد، بدون اینکه لازم باشد تا در جایی از پروژه، بقیه را از آن مطلع کنید، فقط با زدن `gulp` خیالتان راحت شود که تمام کارهایی که باید انجام دهید، گالپ برایتان انجام می‌دهد.

```
// create a task to do bower install
gulp.task('bower', function() {
  return bower()
    .pipe(gulp.dest(config.bowerDir))
});
```

در گام بعدی، تسک `جاوا اسکریپت` را اضافه می‌کنیم. یعنی جی کوری و فایل `bootstrap.js` را به مسیر `public/js` می‌آوریم. فولدر `public` برای جدا سازی فایل‌های نهایی از فایل‌های توسعه است و به همین صورت برای فونت آیکن‌های `fontawesome`.



```
// Copy js files to public folder
gulp.task('js', function() {
  return gulp.src([config.bowerDir + '/bootstrap-sass-official/assets/javascripts/bootstrap.min.js',
    config.bowerDir + '/jquery/dist/jquery.min.js'
  ])
  .pipe(gulp.dest('./public/js'));
});

// Copy fontawesome icons to public/fonts folder
gulp.task('icons', function() {
  return gulp.src(config.bowerDir + '/fontawesome/fonts/**/*.*)
  .pipe(gulp.dest('./public/fonts'));
});
```

برای سی اس اس هم تسک مربوطه اش را به صورت زیر می نویسیم



```
gulp.task('css', function() {
  return sass(config.sassPath + '/style.scss', { // Our custom sass
    style: 'compressed', // minify css
    loadPath: [ // load paths to easily use imports in resources/sass
      './resources/sass',
      config.bowerDir + '/bootstrap-sass-official/assets/stylesheets', // bootstrap sass files
      config.bowerDir + '/fontawesome/scss' // awesome icons sass files
    ]
  });
});
```

حال تعریف می‌کنیم که اگر خطایی در حین کامپایل رخ داد، آن را به ما نشان دهد و در نهایت فایل کامپایل و فشرده شده را در مسیر خروجی قرار می‌دهیم. کدها را به صورت زیر به روز می‌کنیم

```
gulp.task('css', function() {
  return sass(config.sassPath + '/style.scss', { // Our custom sass
    style: 'compressed', // minify css
    loadPath: [ // load paths to easily use imports in resources/sass
      './resources/sass',
      config.bowerDir + '/bootstrap-sass-official/assets/stylesheets', // bootstrap sass files
      config.bowerDir + '/fontawesome/scss' // awesome icons sass files
    ]
  })
  .on('error', notify.onError(function(error) {
    return 'Error: ' + error.message;
  }))
  .pipe(gulp.dest('./public/css'));
});
```

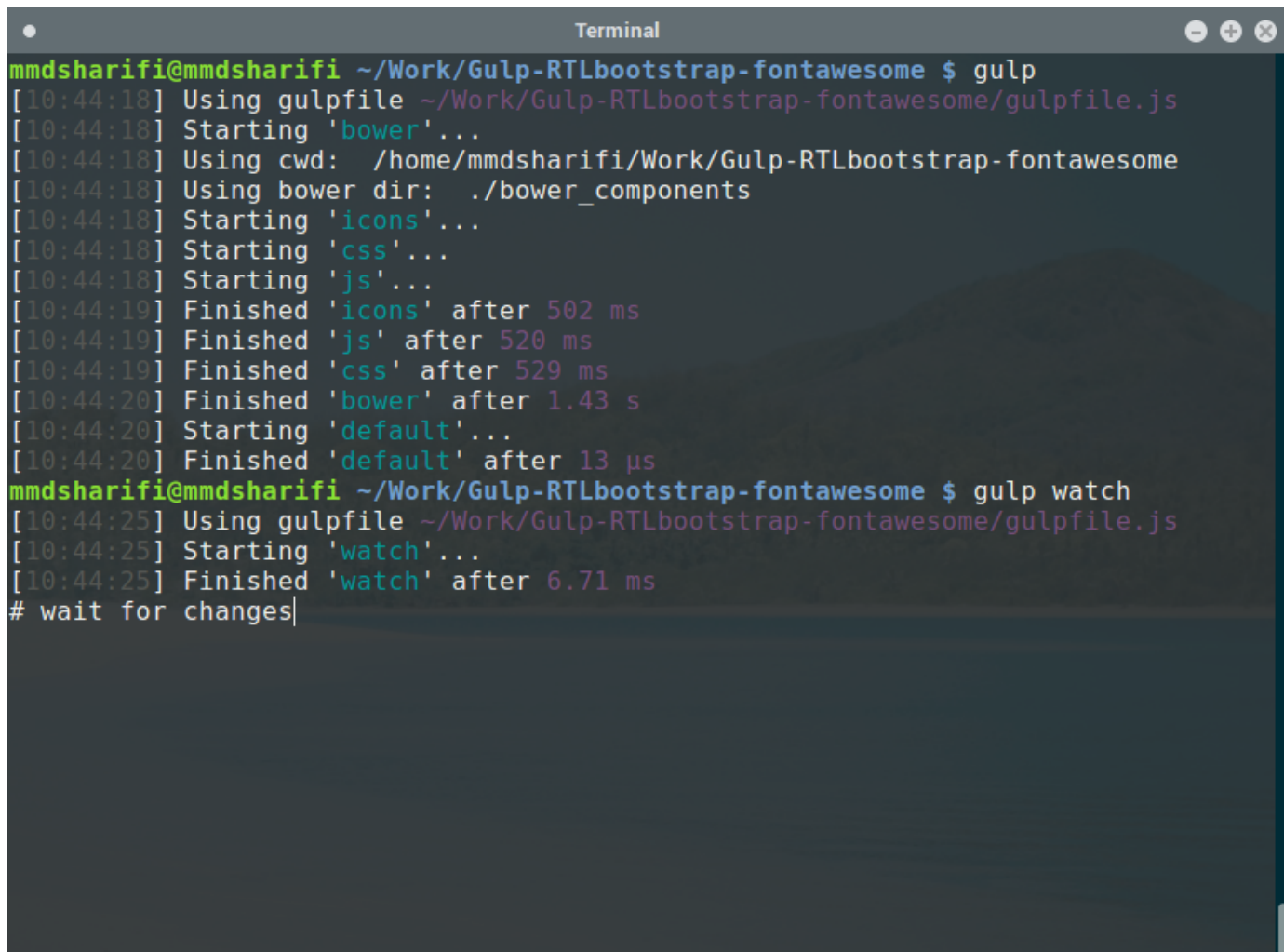
اینجا ما می‌خواهیم که کارها را خودکار سازی کنیم تا با تغییر و ذخیره‌ی مجدد فایل‌های سس، تسک سی اس اس را انجام دهد. برای این کار کدهای زیر را اضافه می‌کنیم

```
// Rerun the task when a file changes
gulp.task('watch', function() {
```

```
gulp.watch(config.sassPath + '/*.scss', ['css']);
});
```

در نهایت برای سادگی می‌توانیم مجموعه‌ای از وظایف را در یک تسک تعریف کنیم تا به راحتی و با زدن تنها یک دستور در ترمینال، کارها خودکار سازی شوند

```
// Run this task with : gulp
// OR gulp default
gulp.task('default', ['bower', 'icons', 'css', 'js']);
```

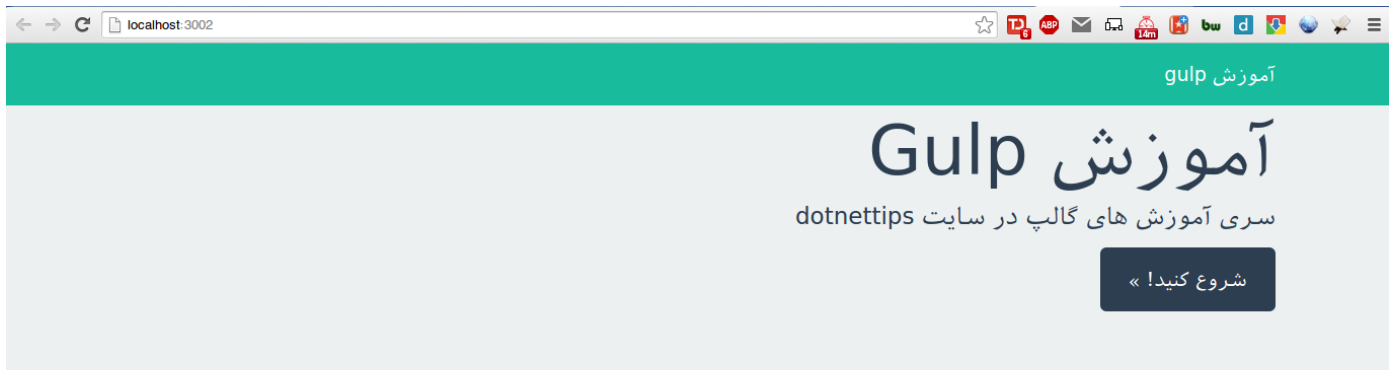


```
Terminal
mmdsharifi@mmdsharifi ~/Work/Gulp-RTLbootstrap-fontawesome $ gulp
[10:44:18] Using gulpfile ~/Work/Gulp-RTLbootstrap-fontawesome/gulpfile.js
[10:44:18] Starting 'bower'...
[10:44:18] Using cwd: /home/mmdsharifi/Work/Gulp-RTLbootstrap-fontawesome
[10:44:18] Using bower dir: ./bower_components
[10:44:18] Starting 'icons'...
[10:44:18] Starting 'css'...
[10:44:18] Starting 'js'...
[10:44:19] Finished 'icons' after 502 ms
[10:44:19] Finished 'js' after 520 ms
[10:44:19] Finished 'css' after 529 ms
[10:44:20] Finished 'bower' after 1.43 s
[10:44:20] Starting 'default'...
[10:44:20] Finished 'default' after 13 µs
mmdsharifi@mmdsharifi ~/Work/Gulp-RTLbootstrap-fontawesome $ gulp watch
[10:44:25] Using gulpfile ~/Work/Gulp-RTLbootstrap-fontawesome/gulpfile.js
[10:44:25] Starting 'watch'...
[10:44:25] Finished 'watch' after 6.71 ms
# wait for changes
```

بسیار خوب؛ ما توانستیم پایه‌ی ورک فلو یمان را بسازیم. در مقاله‌ی بعدی از پلاگین‌های دیگری برای بهینه سازی کارهایمان کمک خواهیم گرفت  
[مخزن گیت هاب](#)

commit : [Update :gulp file and bower.json](#)

همانطور که در [مقاله ی قبلی](#) پایه ی ورک فلو ی خود را راه اندازی کردیم، در این مقاله می‌خواهیم با طراحی یک صفحه، با بوت استرپ شخصی سازی شده، در عمل با کارایی گالپ آشنا شویم.  
دمو پایانی:



قسمت اول	قسمت دوم	قسمت سوم
Gulp ابزاری ست که شما را در انجام دادن کارهای مختلف توسعه ی وب، در سمت Front-end کمک می‌کند و اغلب برای کارهایی هم‌چون موارد ذیل بکار می‌رود:	در این مقاله و مقالات بعد می‌خواهیم کار خود را با راه اندازی یک workflow برای بوت استرپ، روند شخصی سازی آن را بسیار آسان و لذت بخش‌تر کنیم. ...	در قسمت اول گالپ را معرفی کردیم و در مقاله قبلی به نوشتن اولین تسک با گالپ پرداختیم. در این قسمت می‌خواهیم با نصب bower، پروژه ی workflow بوت استرپ راستچین شده را انجام دهیم.
شروع کنید! »	شروع کنید! »	شروع کنید! »

## به هنگام سازی مرورگر و بارگذاری مجدد به صورت خودکار

یکی از موارد فوق العاده تکراری در هنگام توسعه ی وب، برای یک توسعه دهنده سمت کاربر (Front end Developer) ریلود کردن مرورگر است. همچنین تست وب سایت یا آپلود در موبایل و سایر دستگاه‌ها، متداول است. با پلاگین گالپ می‌توان این مشکل را به صورت بهینه‌ای حل کرد.

### نصب

برای نصب دستور زیر را در مسیر پروژه، در ترمینال سیستم عامل خود وارد کنید.

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

می‌دانیم برای استفاده از یک پلاگین باید توسط متد require آن را به یک متغیر انتساب دهیم؛ به صورت زیر:

```
var gulp = require('gulp'),
    sass = require('gulp-ruby-sass'),
    notify = require('gulp-notify'),
    browserSync = require('browser-sync'), // Add browser syncs plugin
    bower = require('gulp-bower');
```

توجه کنید هر پلاگینی که اضافه می‌کنید، باید تسک مربوط به آن را بنویسیم تا بتوانیم از آن استفاده کنیم. برای این پلاگین فقط مشخص کردن مسیر root سرور کافی است.

```
gulp.task('browserSync', function() {
  browserSync({
    server: {
      baseDir: './' //our server root
    }
  });
});
```

حال می‌خواهیم با زدن `gulp watch`، تمام کارهای ما به صورت خودکار انجام شوند. اما این دستور که در جلسه‌ی قبل آن را تعریف کردیم، فقط منتظر انجام یک تغییر است. تسک `watch` را به گونه‌ای تغییر می‌دهیم که ابتدا تسک‌های `brower sync`، `css`، `css` انجام شوند (به دلیل اینکه باید ابتدا، سرور راه اندازی شود) سپس گالپ منتظر تغییرات باشد و آنها را اعمال کند.

```
gulp.task('watch', [ 'css','browserSync'], function() {
})
```

تسک‌های `html,css,browserSync` قبل از تسک `watch` اجرا می‌شوند. طبق مستندات، این پلاگین یکی از توابع `API` [متد watch](#) است و کار آن همانند متد مشابهی در گالپ است. آن را برای ریلود خودکار مرورگر استفاده می‌کنیم.

```
// Rerun the task when a file changes
gulp.task('watch', ['html', 'css','browserSync'], function() {
  gulp.watch(config.sassPath + '/*.scss', ['css']);
  gulp.watch(config.htmlPath, ['html'] )
  browserSync.watch("./*.html").on("change", browserSync.reload); // browserSync watch task
});
```

می‌خواهیم بعد از کامپایل، فایل‌های `sass` هم مرورگر دوباره بارگذاری شوند. کد زیر را به انتهای تسک `css` اضافه می‌کنیم:

```
.pipe(browserSync.reload({
  stream: true
}));
```

بسیار خوب با انجام این کارها پلاگین باید به درستی کار کند.

## شخصی سازی بوت استرپ

برای شخصی سازی بوت استرپ کافی است ابتدا فایل‌های `sass` بوت استرپ و `FontAwesome` را در `style.scss` ایمپورت کنیم؛ به این صورت:

```
@import "bootstrap";
@import "font-awesome";
```

حال دستور `gulp` را می‌زنیم. با اینکار فایل `style.scss` کامپایل می‌شود. می‌خواهیم یک فونت فارسی و یک قالب فلت را به پروژه‌امان اعمال کنیم. من فایل‌ها را اضافه کرده‌ام و شما با یک نگاه می‌توانید، چیزی را که گفتم درک کنید.

```
@import "fonts-fa";
@import "variable";
@import "bootstrap";
@import "font-awesome";
@import "rtl.scss";
@import "typography";
```

نکته : سعی کنید برای استایل هر قسمت، یک فایل مجزا درست کنید؛ مانند مثال بالا که در پروژه لحاظ شده. برای توسعه‌ی پروژه، ابتدا مخزن گیت هاب را فورک کرده و با زدن دستورات زیر کار خود را آغاز کنید:

```
sudo npm install
```

```
gulp
```

```
gulp watch
```

```
mmdsharifi@mmdsharifi ~/Work/Gulp-RTLbootstrap-fontawesome $ gulp watch
[12:58:37] Using gulpfile ~/Work/Gulp-RTLbootstrap-fontawesome/gulpfile.js
[12:58:37] Starting 'css'...
[12:58:37] Starting 'browserSync'...
[12:58:37] Finished 'browserSync' after 43 ms
[BS] Access URLs:
-----
    Local: http://localhost:3002
  External: http://192.168.1.7:3002
-----
    UI: http://localhost:3003
  UI External: http://192.168.1.7:3003
-----
[BS] Serving files from: ./
[12:58:39] Finished 'css' after 1.9 s
[12:58:39] Starting 'watch'...
[12:58:39] Finished 'watch' after 37 ms
[13:00:41] Starting 'css'...
[BS] 1 file changed (style.css)
[13:00:43] Finished 'css' after 1.62 s
[BS] File changed: index.html
[BS] File changed: index.html
[13:05:52] Starting 'css'...
[BS] 1 file changed (style.css)
[13:05:54] Finished 'css' after 1.83 s
[13:06:43] Starting 'css'...
[BS] 1 file changed (style.css)
[13:06:44] Finished 'css' after 1.69 s
[13:09:03] Starting 'css'...
```

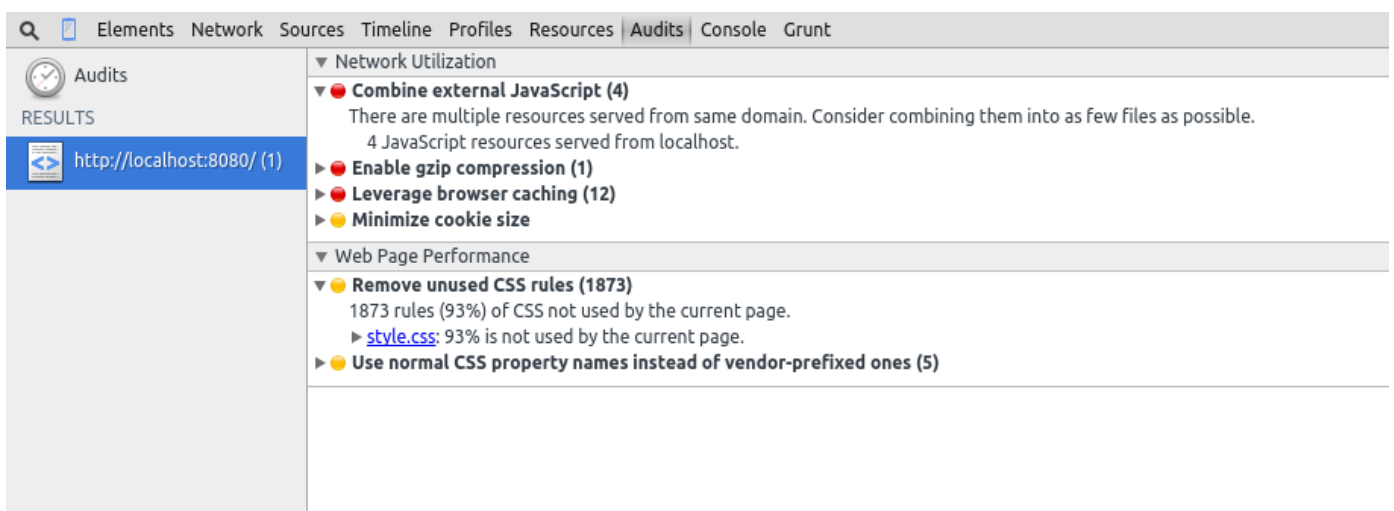
[مخزن گیت هاب](#) : کامیت :

[Add : browserSync plugin and index.html](#)

در [مقالات قبلی](#) به طور کامل با گالپ آشنا شدیم و گفتیم که می‌تواند ما را در بهینه سازی ورک فلویمان کمک کند. در این قسمت یاد خواهیم گرفت که چگونه تجربه‌ی کاربری بهتری را از سرعت بارگذاری سایتمان ایجاد کنیم.

### افزایش کارآیی Performance وب با گالپ

برای اینکه بفهمیم چه کارهایی می‌تواند سایت یا اپلیکیشن ما را کاراتر کند، از Developer tools با زدن Ctrl+Shift+I درون گوگل کروم، کار خود را شروع می‌کنیم. به برگه‌ی Audits می‌رویم و دکمه‌ی Run را با تنظیمات پیش فرض می‌زنیم. نتایج آن بعد از اندکی صبر، برای من به صورت شکل زیر است:



ما قصد داریم بدانیم گالپ چه ابزاهایی را برای راه حل‌های داده شده توسط مرورگر دارد؟

### ۱- کنار هم قرار دادن و فشرده کردن فایل‌های جاوا اسکریپت

پلاگین‌های گالپ برای اینکار، [gulp-concat](#) و [gulp-uglify](#) هستند. آنها را در مسیر ریشه‌ی پروژه نصب می‌کنیم.

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

بعد از نصب، فایل gulpfile.js را به صورت زیر ویرایش و تسک js را به آن اضافه می‌کنیم.

```
// first load all required js files
// concat them in to script.min.js
// and minify it.
gulp.task('js', function() {
  return gulp.src([
    config.bowerDir + '/jquery/dist/jquery.min.js', // این فایل وابستگی فایل‌های زیر است
    config.bowerDir + '/materialize/dist/js/materialize.min.js',
    './resources/js/app.js'
  ])
  .pipe(concat('script.min.js'))
  .pipe(uglify())
  .pipe(size())
  .pipe(gulp.dest('./public/js'));
});
```



خروجی:

```
mmdsharifi@mmdsharifi ~/Work/Lenus $ gulp js
[13:27:20] Using gulpfile ~/Work/Lenus/gulpfile.js
[13:27:20] Starting 'js'...
[13:27:23] all files 206.05 kB
[13:27:23] Finished 'js' after 3.18 s
```

فشرده کردن جاوا اسکریپت، حجم فایل‌ها را ۳۰ تا ۹۰ درصد [کاهش](#) می‌دهد.

## ۲- حذف سکتورهای بدون استفاده css

همانگونه که در عکس اول آمده، ۹۳ درصد سکتورها در این صفحه بلا استفاده هستند! و این یعنی کاهش فوق العاده زیاد حجم فایل فشرده شده css. به طور معمول، توسعه دهندگان ۸۵ درصد حجم فایل css خود را می‌توانند با این کار [کاهش](#) دهند (البته بیشتر این اتفاق هنگام استفاده از فریک ورک‌هایی مانند bootstrap, ... می‌افتد) برای این کار از پلاگین [gulp-uncss](#) استفاده می‌کنیم. نصب:

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

سپس تسک مربوطه را می‌نویسیم:

```
gulp.task('css', function() {
  return sass(config.sassPath + '/style.scss', {
    style: 'compressed',
    loadPath: [
      './resources/sass',
      config.bowerDir + '/materialize/sass'
    ]
  })
  .on('error', util.log)
  .pipe(size())
  .pipe(uncss({
    html: ['./index.html', './posts.html']
  }))
  .pipe(gulp.dest('./public/css'))
  .pipe(size())
  .pipe(connect.reload());
});
```

نتیجه:

```
mmdsharifi@mmdsharifi ~/Work/Lenus $ gulp css
[14:24:58] Using gulpfile ~/Work/Lenus/gulpfile.js
[14:24:58] Starting 'css'...
[14:25:01] all files 136.68 kB
[14:25:03] all files 17.34 kB
[14:25:03] Finished 'css' after 4.46 s
```

نتیجه فوق العاده است! ۸۷ درصد [کاهش](#) حجم css! اما ممکن است بعضی از استایل‌های شما توسط javascript به صفحه تزریق شوند. در این صورت نباید سکتورهای لازم را حذف کرد و آنها را داخل آرایه‌ی ignore قرار می‌دهیم. برای این منظور، تسک بالا را به صورت زیر به روز رسانی می‌کنیم.

```

gulp.task('css', function() {
  return sass(config.sassPath + '/style.scss', {
    style: 'compressed',
    loadPath: [
      './resources/sass',
      config.bowerDir + '/materialize/sass'
    ]
  })
  .on('error', util.log)
  .pipe(size())
  .pipe(uncss({
    html: ['./index.html', './posts.html'],
    timeout: 2000, // wait for load js files
    ignore: [
      ".waves-ripple ",
      ".drag-target",
      "#sidenav-overlay",
      ".waves-effect",
      ".waves-effect .waves-ripple",
      ".waves-effect.waves-pinck .waves-ripple",
      ".waves-block.waves-light"
    ]
  }))
  .pipe(minifyCss())
  .pipe(size())
  .pipe(gulp.dest('./public/css'))
  .pipe(connect.reload());
});

```

بعد از اینکار حجم فایل css من کمی افزایش پیدا کرد ولی بعد از فشرده کردن، نهایتاً به حدود ۱۴KB رسید و این یعنی ۸۷ درصد کاهش حجم فایل css؛ تنها بعد از حذف سکلتورهای اضافی و فشرده کردن آنها. می‌توانید [پلاگین‌های بیشتری](#) را در اینجا ببینید و استفاده کنید.

[Gist](#)