

با توجه به رشد روز افزون وب و مراحل تکامل برنامه نویسی آن ، نیاز به ابزارهایی که نصب ، به روز رسانی و مدیریت کتابخانه‌ها و ابزارهای جانبی استفاده شده در پروژه‌ها را بطور خودکار انجام دهند بیش از پیش احساس میشود. [Bower](#) یکی از ابزارهایی است که برای کمک به این امر معرفی شده است.



Bower چیست؟

Bower یک package manager برای فن آوری‌های سمت کلاینت است. توانایی نصب ، جستجو و حذف کتابخانه‌های

مزایا :

نصب ابزارها و کتابخانه‌ها توسط یک خط فرمان!

به جای اینکه در سایتهای مختلف ورژن کتابخانه‌ها را پیگیری کنید و update شدن یا نشدن آنها را بررسی نمایید(مثلا آیا jQuery مورد استفاده در پروژه ، آخرین نسخه است؟) ، با استفاده از Bower در کمترین زمان ممکن این کار را انجام دهید. نصب آفلاین. وقتی کتابخانه ای برای اولین بار نصب شود کش شده و دفعات بعد برای نصب همان کتابخانه(و البته همان ورژن) از کش استفاده خواهد کرد.(مگر اینکه کاربر صراحتا کش را خالی کرده باشد).
نصب کتابخانه‌های وابسته. اگر کتابخانه ای وابسته به کتابخانه‌های دیگر باشد (مثل وابستگی Twitter Bootstrap به jQuery)، بطور خودکار وابستگی‌ها نیز نصب می‌گردند.

مراحل نصب :

قبل از نصب باید دو ابزار زیر در سیستم نصب شده باشند:

[Nodejs](#)

[Git](#) : برخی از کتابخانه‌ها باید از مخزن Git واکنشی شوند.

نصب Bower :

در خط فرمان دستور زیر را اجرا نمایید:

```
npm install -g bower
```

دستور بالا Bower را بصورت global نصب خواهد کرد و اکنون میتوان کتابخانه‌های مختلف را نصب نمود.

نصب کتابخانه ها:

برای نصب کتابخانه‌ها از دستور زیر استفاده می‌شود:

```
bower install <package>
```

برای مثال برای نصب کتابخانه angularjs باید دستور زیر را اجرا نمود:

```
bower install angular
```

یا jQuery:

```
bower install jquery
```

ممکن است نیاز باشد تا ورژن خاصی از یک کتابخانه را نصب کنید که در این صورت باید مانند مثال زیر عمل کرد:

```
bower install <package>#<version>
```

```
bower install jquery#1.7.0
```

دستور فوق نسخه 1.7.0 jQuery را نصب خواهد کرد.

پس از اجرای دستور، در مسیر جاری فولدری به نام bower_components ایجاد شده و کتابخانه‌ها در آن قرار می‌گیرند.

```
bower_components/  
jquery/  
  README.md  
  bower.json  
  component.json  
  composer.json  
  jquery-migrate.js  
  jquery-migrate.min.js  
  jquery.js  
  jquery.min.js  
  jquery.min.map  
  package.json
```

و در نهایت نحوه استفاده:

```
<script type="text/javascript" src="bower_components/jquery/jquery.js"></script>
```

جستجو در کتابخانه ها:

Bower امکان جستجو در کتابخانه‌های ثبت شده را می‌دهد. مثال:

```
bower search bootstrap
```

```
Search results:
```

```
bootstrap git://github.com/twbs/bootstrap.git
```

```
angular-bootstrap git://github.com/angular-ui/bootstrap-bower.git
```

```
sass-bootstrap git://github.com/jlong/sass-twitter-bootstrap.git
```

عنوان:	Gulp #2
نویسنده:	م محمد شریفی
تاریخ:	۱۰:۴۵ ۱۳۹۴/۰۶/۱۷
آدرس:	www.dotnettips.info
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!

<https://github.com/mmdsharifi/gulp-rtlBootstrap-fontawesome> در گیت هاب : [مخزن پروژه](#) در گیت هاب :
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'));
});
```

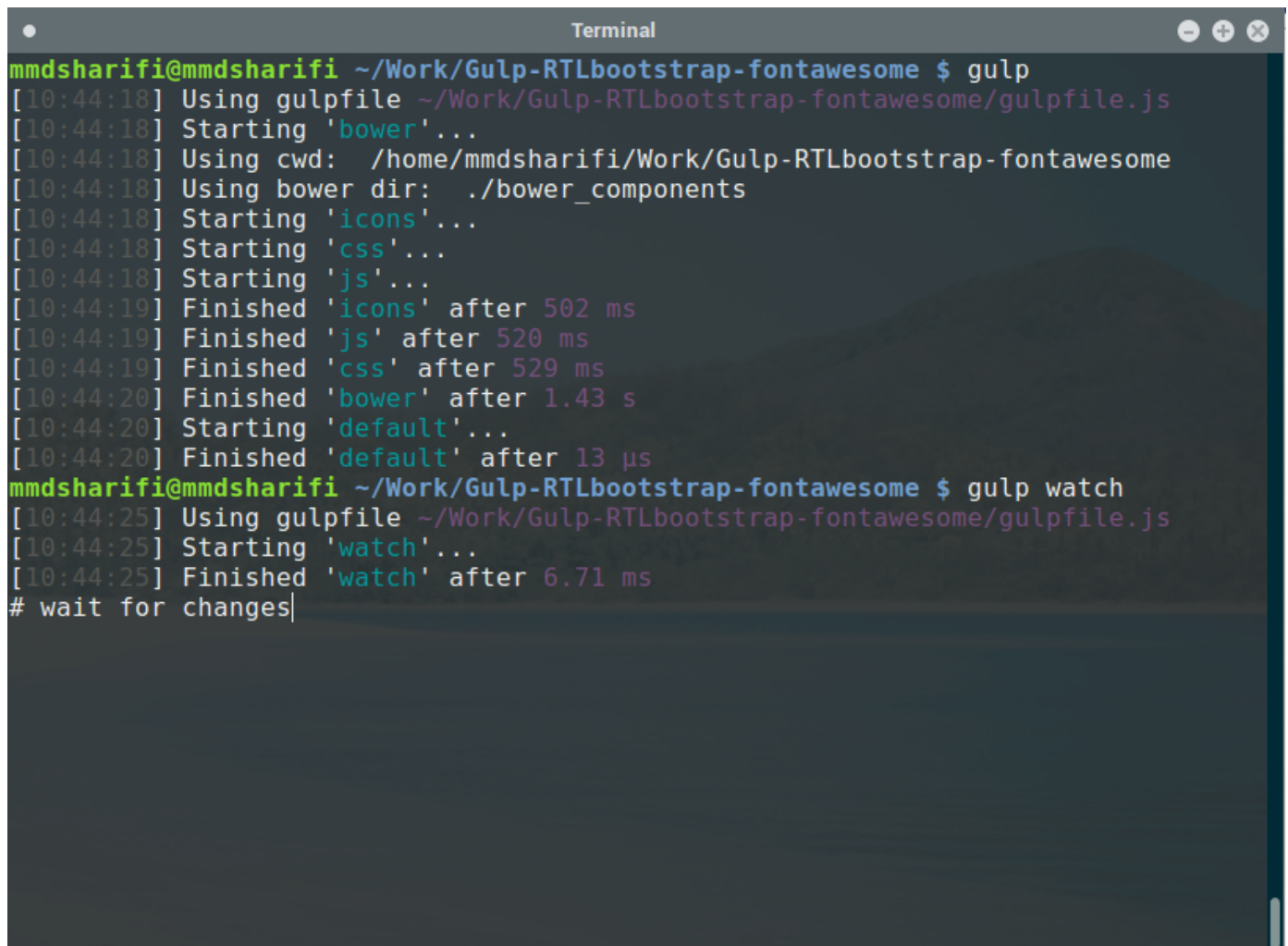
اینجا ما می‌خواهیم که کارها را خودکار سازی کنیم تا با تغییر و ذخیره‌ی مجدد فایل‌های سس، تسک سی اس اس را انجام دهد. برای این کار کدهای زیر را اضافه می‌کنیم

```
// 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](#)

عنوان:	استفاده از bower در visual studio
نویسنده:	مهرداد کاهه
تاریخ:	۱۳۹۴/۰۷/۱۷
آدرس:	www.dotnettips.info
گروه‌ها:	JavaScript, Git, Visual Studio, bower

اگر از آن دسته افرادی هستید که با پکیج‌های مختلف و پروژه‌های مختلف تحت کلاینت سر و کار دارید و همچنین اطلاعات چندانی نسبت به NodeJs ندارید (مثل خود من)، حتما به پروژه‌هایی در Github برخوردید که نیازمند نصب وابستگی‌ها از خط فرمان bower و یا npm هستند.

بعد از مطالعه‌ی مطلب [آشنایی با bower](#) این نیاز ایجاد شد تا در پروژه‌هایی که قرار است درون Visual Studio اجرا شوند، وابستگی‌های bower چگونه می‌توانند مدیریت شوند.

خوشبختانه Microsoft این امکان را ایجاد کرده تا شما بتوانید پروژه‌هایی را که وابستگی‌هایشان درون bower تعریف شده را نیز درون Visual Studio حل و فصل کنید. در ادامه تمامی این مراحل، قدم به قدم اضافه تشریح شده است. قابل ذکر است که هر سه package manager معروف bower، npm و Nuget در ویژوال استدیو 2015 به صورت توکار موجود هستند. [جزئیات بیشتر در مستندات مایکروسافت](#)

معرفی پکیج Bower

همانطور که در مقاله [آشنایی با bower](#) نیز اشاره شد، bower یک package manager برای تکنولوژی‌ها و کتابخانه‌های کلاینت است. این package manager بر روی Git اجرا می‌شود. همانطور که می‌دانید تمامی پکیج‌ها نیز از Git دریافت می‌شود. اما حال اینکه چگونه می‌توان از این package manager در سمت Visual studio بدون نصب NodeJs و Git استفاده کرد، با پکیج توسعه داده شده Bower مایکروسافت رفع شده است. جزئیات این پکیج را می‌توانید در [NuGet](#) مطالعه کنید.

شروع کار با Bower

برای آغاز، یک پروژه‌ی web Application ایجاد می‌کنیم. من Empty را انتخاب و ریسورس‌های MVC را هم اضافه کردم. حال در بخش Package Manager Console دستور زیر را اجرا کنید:

```
Install-Package Bower
```

پس از نصب وابستگی‌ها و خود bower خروجی package manager console به صورت زیر خواهد بود:

```
PM> Install-Package Bower
Attempting to resolve dependency 'Node.js (≥ 0.10.28)'.
Attempting to resolve dependency 'NoGit (≥ 0.0.8)'.
Installing 'Node.js 0.10.36'.
Successfully installed 'Node.js 0.10.36'.
Installing 'NoGit 0.0.9'.
Successfully installed 'NoGit 0.0.9'.
Installing 'Bower 1.3.11'.
Successfully installed 'Bower 1.3.11'.
Adding 'Node.js 0.10.36' to WebApplication5.
Successfully added 'Node.js 0.10.36' to WebApplication5.
Adding 'NoGit 0.0.9' to WebApplication5.
Successfully added 'NoGit 0.0.9' to WebApplication5.
Adding 'Bower 1.3.11' to WebApplication5.
Successfully added 'Bower 1.3.11' to WebApplication5.
```

مشاهده می‌کنید که فولدر bin. به پروژه‌ی شما اضافه شده است.

حال درون صفحه‌ی cmd (توجه کنید cmd، نه package manager console) به آدرس پروژه (نه solution) با دستور زیر منتقل شوید:

```
cd <Project Location>
```

که به جای project location آدرس فایل پروژه را قرار می‌دهیم. شکل زیر نمایانگر این مسیر است:

```
Command Prompt
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\mehrdad>cd C:\Users\mehrdad\Documents\visual studio 2013\Projects\SimpleBower\SimpleBower
```

با اجرای دستور زیر bower.json را به پروژه اضافه می‌کنیم:

```
bower init
```

```
C:\Users\mehrdad\Documents\Visual Studio 2013\Projects\SimpleBower\SimpleBower>bower init
? name: (SimpleBower)
```

مشاهده می‌کنید که پس از دستور bower init مواردی که قرار است درون bower قرار گیرد، مقدار دهی می‌شوند. من مقادیر را به صورت زیر (حالت‌های پیش فرض) تکمیل کردم:

```
C:\Users\mehrdad\Documents\Visual Studio 2013\Projects\SimpleBower\SimpleBower>bower init
? name: SimpleBower
? version: 0.0.0
? description:
? main file:
? what types of modules does this package expose?
? keywords:
? authors:
? license: MIT
? homepage:
? set currently installed components as dependencies? Yes
? add commonly ignored files to ignore list? No
? would you like to mark this package as private which prevents it from being accidentally published to the registry? (y? would you like to mark this package as private which prevents it from being accidentally published to the registry? No
{
  name: 'SimpleBower',
  version: '0.0.0',
  license: 'MIT'
}
```

حال باید تا اینجای کار یک فایل bower.json برای شما در روت پروژه ساخته شده باشد. حال بیاید اولین اسکریپت رفرنس را به پروژه اضافه نماییم. من قصد دارم تا با دستور زیر JQuery را به پروژه اضافه کنم:

```
bower install jquery
```

پکیج JQuery به صورت زیر دانلود می شود و در پوشه‌ی bower_component در روت پروژه قرار می گیرد.

```
C:\Users\mehrdad\Documents\Visual Studio 2013\Projects\SimpleBower\SimpleBower>bower instal jquery
bower jquery#*      not-cached git://github.com/jquery/jquery.git#*
bower jquery#*      resolve  git://github.com/jquery/jquery.git#*
bower jquery#*      download https://github.com/jquery/jquery/archive/2.1.4.tar.gz
bower jquery#*      extract  archive.tar.gz
bower jquery#*      resolved git://github.com/jquery/jquery.git#2.1.4
bower jquery#~2.1.4  install  jquery#2.1.4

jquery#2.1.4 bower_components\jquery
```

به همین صورت شما می توانید تمامی نیازمندی های پروژه را از Git با استفاده از bower package manager دریافت کنید.