

در [پست‌های قبلی](#) با مفهوم ng-app آشنا شدید. دایرکتیو ng-app برای استفاده از راه انداز خودکار فریم ورک Angular (معروف به auto-bootstrap) استفاده می‌شود. در حالت پیش فرض، به ازای هر سند Html فقط می‌توان یک ماژول در Angular تعریف کرد. در سند مربوطه اولین المانی که دارای دایرکتیو ng-app باشد به عنوان عنصر ریشه در نظر گرفته می‌شود و تمام عناصر تعریف شده در محدوده این دایرکتیو قابل استفاده برای ماژول مورد نظر خواهد بود. سایر عناصر حتی اگر ng-app یکسان داشته باشند نادیده گرفته می‌شوند.

ابتدا یک مثال زیر را به روش auto-bootstrap بررسی می‌کنیم:

```
<div ng-app="myApp">
  <div ng-controller="myController as ctrl">
    <span>ng-app #1</span> {{ctrl.firstName}} {{ctrl.lastName}}
  </div>
</div>

<div ng-app="myApp">
  <div ng-controller="myController as ctrl">
    <span>ng-app #2</span> {{ctrl.firstName}} {{ctrl.lastName}}
  </div>
</div>

@section scripts
{
  <script type="text/javascript" src="~/scripts/Modules/module8.js"></script>
}
```

در کنترلر مورد نظر نیز تعاریف به صورت زیر خواهد بود:

```
var app = angular.module('myApp', []);
app.controller('myController', function ()
{
  this.firstName = "Masoud";
  this.lastName = "Pakdel";
});
```

در مثال بالا دو تگ div وجود دارد که به صورت مشترک با استفاده از دایرکتیو ng-app به یک ماژول اشاره می‌کنند. طبق گفته‌ها بالا در روش auto-bootstrap اولین عنصری که دارای دایرکتیو ng-app باشد به عنوان محدوده ماژول مورد استفاده قرار خواهد گرفت در نتیجه سایر المان‌ها (در اینجا منظور تگ div دوم است) نادیده گرفته خواهند شد. پس خروجی به صورت زیر می‌شود:



اما اگر قصد داشته باشیم که در یک سند html دو نقطه شروع تعریف کنیم در حالی که هر کدام از یک منبع داده استفاده نمایند باید bootstrap برنامه را به صورت دستی تعیین کرد. برای این کار کافیه از دستور angular.bootstrap به صورت زیر استفاده نماییم:

### پیاده سازی مثال بالا

```
<div id="myAppContainer1">
  <div ng-controller="myController as ctrl">
    <span>ng-app #1</span> {{ctrl.firstName}} {{ctrl.lastName}}
  </div>
</div>

<div id="myAppContainer2">
  <div ng-controller="myController as ctrl">
    <span>ng-app #2</span> {{ctrl.firstName}} {{ctrl.lastName}}
  </div>
</div>

@section scripts
{
  <script type="text/javascript" src="~/scripts/Modules/module8.js"></script>
}
```

اولین تغییر مورد نظر این است که، دایرکتیو ng-app حذف شد و به جای آن id برای تگ div تعیین کردیم. در فایل کنترلر مورد نظر نیز تغییر زیر را اعمال می‌کنیم:

```
var app = angular.module('myApp', []);
app.controller('myController', function ()
{
  this.firstName = "Masoud";
  this.lastName = "Pakdel";
});

angular.bootstrap(document.getElementById("myAppContainer1"), ["myApp"]);
angular.bootstrap(document.getElementById("myAppContainer2"), ["myApp"]);
```

با استفاده از دستور angular.bootstrap می‌توان بر اساس id تعیین شده تگ مورد نظر در سند را به دست آورد و ماژول مورد نظر را به آن نسبت داد.

خروجی مثال بالا:



برخلاف حالت قبل هر دو نقطه شروع به یک منبع داده اشاره می‌کنند و محدودیت حالت قبل برطرف می‌شود.

## نظرات خوانندگان

نویسنده: ابوالفضل رجب پور  
تاریخ: ۱۹:۴ ۱۳۹۲/۱۰/۱۷

سلام و تشکر  
بنظرم جای نمونه برنامه‌های کوچکی که گام به گام توضیح داده بشن خالیه. خیلی خوب و کاربردی مفهوم جا می‌افته. مثلاً نمونه‌ی  
معروف todo

نویسنده: مسعود پاکدل  
تاریخ: ۲۳:۳ ۱۳۹۲/۱۰/۱۷

مثال بالا نمونه ای از یک برنامه کوچک است. بهتر است هر نکته و سرنخ در طی یک پست ارائه شود به جای ذکر تاریخچه،  
هستی و چیستی مسائل در طی یک پست.

DOM در حالت عادی بسیار نامرتب است. همچنین با افزودن کلاس‌های CSS، کد HTML به مراتب نامرتب‌تر از قبل می‌شود. بوت استرپ نیز شامل تعداد زیادی از کلاس‌های CSS می‌باشد که برای انجام وظایف خاصی به HTML اضافه می‌شوند. روش متداول استفاده از بوت استرپ

### Embedد کردن کلاس‌های CSS بوت استرپ به صورت مستقیم درون HTML

اغلب فریم‌ورک‌ها، از لحاظ معنایی یا semantic، دارای مشکل هستند. اگر به سورس HTML صفحاتی که با این نوع از فریم‌ورک‌ها ساخته شده باشند نگاهی بیندازید با حجم زیادی از کلاس‌هایی مانند `<div class="row">` و یا `<div class="col-sm">` مواجه خواهید شد. نوشتن کدهای HTML به این صورت از لحاظ معنایی اشتباه است. مثلاً اگر بنا به دلایلی سازندگان بوت استرپ تصمیم بگیرند نام کلاس‌های را در نسخه بعدی این فریم‌ورک تغییر دهند (مانند تغییر نام کلاس‌ها در نسخه‌ی 3 بوت استرپ)، و یا اگر در آینده بخواهید از یک فریم‌ورک دیگر در سایت‌تان استفاده کنید. باید این تغییرات را در تمام صفحات سایت‌تان اعمال کنید؛ در نتیجه اینکار زمان زیادی را از شما صرف می‌کند.

راه‌حل؟

### استفاده از CSS preprocessors

بوت استرپ، از [Less](#) برای اینکار استفاده می‌کند. Less در واقع یک CSS preprocessor نوشته شده با جاوا اسکریپت است که قابلیت اجرا در مرورگر را دارد. Less امکانات زیادی، از قبیل استفاده از توابع، متغیرها، Mixins و ... را در اختیار شما قرار می‌دهد. در واقع هدف از Less، نگهداری آسان و قابلیت توسعه فایل‌های CSS می‌باشد. در این حالت شما کدهای CSS خود را درون فایل‌هایی با پسوند Less می‌نویسید. در این حالت بجای پیوست کردن کلاس‌های بوت استرپ در کد HTML، آن را درون استایل‌شیت پیوست خواهید کرد. همانطور که عنوان شد، بوت استرپ با Less نوشته شده است. فایل‌های Less بوت استرپ را می‌توانید از مخزن کد [گیت‌هاب](#) آن دانلود نمائید یا اینکه از طریق [نیوگت](#) می‌توانید آن را نصب کنید:

```
PM> Install-Package Twitter.Bootstrap.Less
```

کار با Less خیلی ساده است. به عنوان مثال در کد زیر یک کلاس با نام loud داریم که استایل‌هایی را به آن اعمال کرده‌ایم. اکنون جهت استفاده مجدد از این استایل‌ها برای کلاسی دیگر، نیاز به نوشتن مجدد آن نیست. کافی همانند یک تابع در هر کلاسی آن را فراخوانی کنیم:

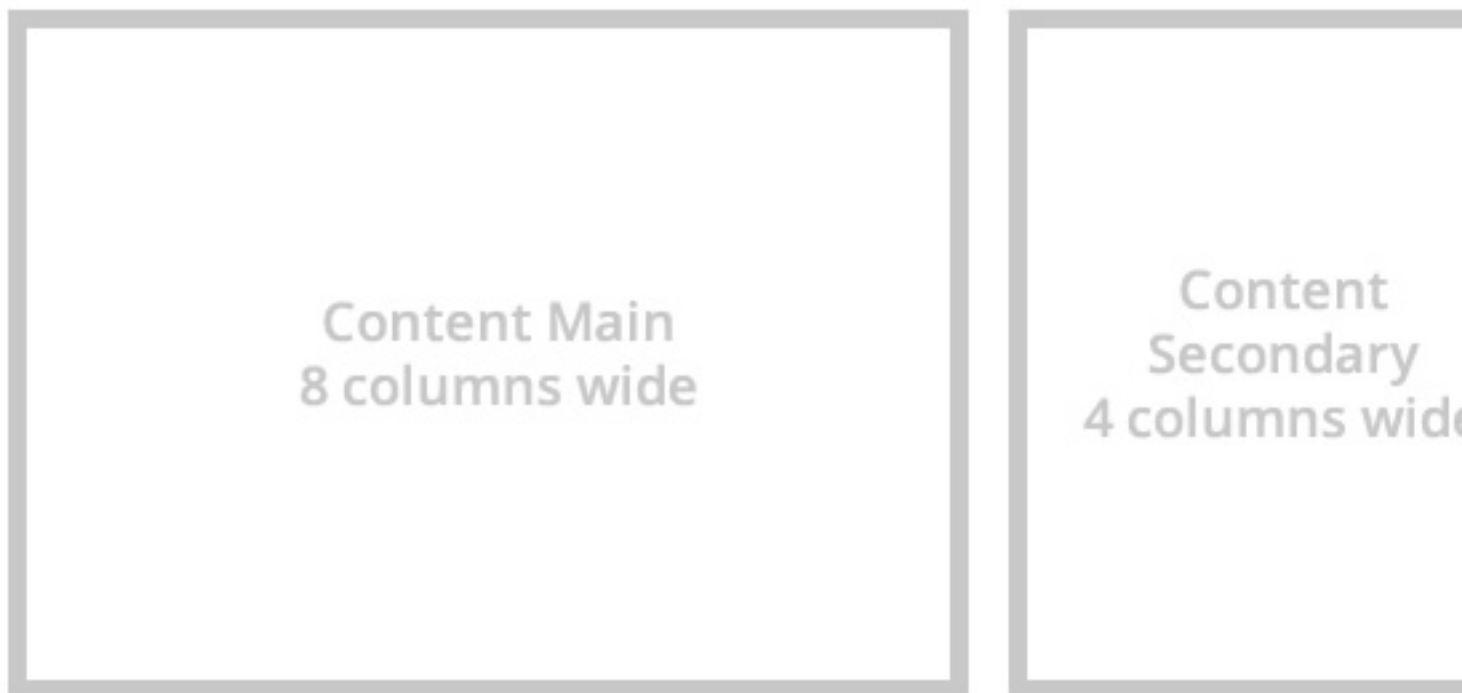
```
.loud {
  color: red;
}

// Make all H1 elements loud
h1 {
  .loud;
}
```

**نکته:** در Visual Studio 2012 Update 2 به بعد به صورت توکار از فایل‌های Less [پشتیبانی](#) می‌شود. (توسط پلاگین [Web Essentials](#))

### استفاده از Mixins

با استفاده از Mixins می‌توانیم عناصر داخل صفحات‌مان را به صورت [Semantic](#) تعریف نمائیم. به عنوان مثال می‌خواهیم با استفاده از سیستم گرید بوت استرپ، ساختاری مانند تصویر زیر را داشته باشیم:



در حالت معمول با استفاده از کلاس‌های CSS بوت استرپ می‌توانیم اینکار را انجام دهیم:

```
<div class="container">
  <div class="row">
    <div class="col-md-8">
      Content - Main
    </div>
    <div class="col-md-4">
      Content - Secondary
    </div>
  </div>
</div>
```

کد فوق را بهتر است به این صورت بنویسیم:

```
<div class="wrapper">
  <div class="content-main">
    Content - Main
  </div>
  <div class="content-secondary">
    Content - Secondary
  </div>
</div>
```

در بوت استرپ از [Less Mixins](#) جهت اعمال استایل‌هایی مانند row و column می‌توانیم استفاده کنیم. به طور مثال برای اعمال استایل به کلاس‌های فوق می‌توانیم به این صورت عمل کنیم:

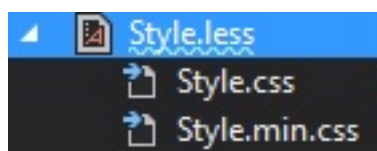
```
// Core variables and mixins
@import "variables.less";
@import "mixins.less";
.wrapper {
  .make-row();
}
.content-main {
```

```
.make-lg-column(8);
}
.content-secondary {
  .make-lg-column(3);
  .make-lg-column-offset(1);
}
```

کد فوق بعد از کامپایل به کد زیر تبدیل خواهد شد:

```
.wrapper {
  margin-left: -15px;
  margin-right: -15px;
}
.content-main {
  position: relative;
  min-height: 1px;
  padding-left: 15px;
  padding-right: 15px;
}
@media (min-width: 1200px) {
  .content-main {
    float: left;
    width: 66.66666666666666%;
  }
}
.content-secondary {
  position: relative;
  min-height: 1px;
  padding-left: 15px;
  padding-right: 15px;
}
@media (min-width: 1200px) {
  .content-secondary {
    float: left;
    width: 25%;
  }
}
@media (min-width: 1200px) {
  .content-secondary {
    margin-left: 8.333333333333333%;
  }
}
```

همانطور که قبلاً عنوان شد ویژوال استودیو به راحتی توسط افزونه Web Essentials از فایل‌های Less پشتیبانی می‌کند. در نتیجه کامپایل فایل‌های Less داخل ویژوال استودیو توسط این افزونه به راحتی قابل انجام می‌باشد. یک قابلیت جالب دیگر در رابطه با فایل‌هایی Less، تولید نسخه‌های CSS عادی و فشرده نهایی توسط افزونه Web Essentials می‌باشد. به طور مثال شما می‌توانید نسخه minified شده را به Layout تان اضافه کنید. بعد از هربار تغییر در فایل Less این فایل نیز به روز خواهد شد:



همچنین برای دیگر اجزای بوت‌استرپ نیز می‌توانید به این صورت عمل کنید:

```
<!-- Before -->
<a href="#" class="btn danger large">Click me!</a>

<!-- After -->
<a href="#" class="annoying">Click me!</a>

a.annoying {
  .btn;
  .btn-danger;
```

```
.btn-large;  
}
```

خب، با استفاده از این حالت، کدهای HTML به صورت مرتب‌تر، قابل انعطاف‌تر و همچنین از لحاظ معنایی (Semantic) استاندارد خواهند بود. بنابراین با آمدن یک فریم‌ورک جدید، به راحتی امکان سوئیچ کردن برای ما میسر و آسان‌تر از قبل خواهد شد.

## نظرات خوانندگان

نویسنده: سعید شیرزادیان

تاریخ: ۱۳۹۳/۰۴/۰۳ ۰:۰

سلام. مطلب جالبی بود. آیا قابلیت فوق برای حالت بدون Less امکان دارد. من در یک پروژه از بوت استرپ استفاده کردم و یکسری تغییرات فارسی نیز در آن اعمال شده است. راهکاری دارید؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۴/۰۳ ۰:۵۷

نسخه‌ی LESS راست به چپ بوت استرپ هم موجود است .  
ضمناً نگارش 3.2 بوت استرپ قرار است که به صورت رسمی RTL پشتیبانی کند.

نویسنده: حسن محمدی

تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۳:۳۵

با سلام و تشکر از سایت بی نهایت خوبتون  
مقاله عالی بود اگر امکانش بود به تمپلیت کوچک سایتو بوت استرپ و less بگید بی نهایت ممنون میشم. من خودم تو استفاده از less خیلی مشکل دارم خیلی‌ها هم مثل خودم می‌شناسنم. من آپدیت 2 ویژوال استودیو 2013 و webessenial رو نصب کردم تا فایل‌های less رو برام باز کرد.  
فقط bootstrap.less رو بدون مشکل باز میکنم و نسخه css رو میاره ولی مثلاً هر فایل دیگه ای مثل alerts.less رو باز میکنم پیغام خطا میده که مثلاً variable @alert-padding is undefined یا چیزهای دیگرو میگه undefined به نظرتون مشکل از چیه؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۳:۴۵

مشکلی ندارد . نسخه‌ی less بوت استرپ از چندین و چند فایل تشکیل شده که نهایتاً از کامپایل این‌ها یک فایل نهایی تولید می‌شود. برای مثال اگر فایل [bootstrap.less](#) را باز کنید، ترتیب import فایل‌ها قابل ملاحظه هستند. ابتدا فایل [variables.less](#) ذکر شده، چون مقادیر پیش فرض متغیرهای سایر فایل‌ها در آن موجود است و همینطور الی آخر. فایل نهایی که سبب کامپایل تمام تغییرات خواهد شد، فایل [bootstrap.less](#) است.

نویسنده: حسن محمدی

تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۳:۵۸

ممنون که پاسخ دادید من اگر بخوام مثلاً navbar رو تغییر بدم باید bootstrap.less رو ویرایش کنم یا navbar.less ?  
بعد چیکار کنم که خطا برطرف شه و این ترتیبو بفهمم و مثلاً navbar.less رو باز میکنم دیگه خطا نده؟  
ممنون

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۴:۲۶

- هیچکدام. فایل navbar.less تشکیل شده از یک سری متغیر. این متغیرها در فایل variables.less مقدار دهی شده‌اند. بنابراین برای تغییر آن باید فایل variables.less ویرایش شوند (در این فایل، Navbar را جستجو کنید).  
- یک کپی از فایل اصلی bootstrap.less را مثلاً به نام test.less ایجاد کنید (با همان محتوا). المان‌های مختلف آن را حذف کنید تا به حداقل وابستگی‌هایی که برای کامپایل navbar.less نیاز است برسید:

```
// Core variables and mixins
@import "variables.less";
@import "mixins.less";
```



```
// Core CSS
@import "forms.less";

// Components
@import "navbar.less";

// Utility classes
@import "utilities.less";
```