سبک‌های کامپوننت

برنامه‌های Angular به کمک استاندارد CSS سبک بندی می‌شوند. این یعنی شما می‌توانید از تمام چیزهایی که درباره‌ی استایل شیت های CSS، انتخابگرها، قوانین و پرس و جوهای واسطه‌ای می دانید مستقیماً در برنامه‌های Angular استفاده کنید.

علاوه بر این Angular می‌تواند به کمک کامپوننت ها، سبک‌های کامپوننت را دسته بندی کند تا بتوان نسبت به استایل شیت های معمولی از طراحی پیمانه‌ای‌تری بهره برد.

در این صفحه به چگونگی بارگیری و استفاده از این سبک‌ها می‌پردازیم.

برای اجرا یا دانلود مثال به این لینک مراجعه کنید.

استفاده از سبک‌های کامپوننت

برای تمامی کامپوننت هایی که در Angular می‌نویسید، نه تنها می‌توانید یک قالب HTML را تعریف کنید، بلکه می‌توانید سبک‌های CSS ای را در کنار این قالب ایجاد کنید تا بتوانید تمامی انتخابگرها، قوانین و پرس و جوهای واسطه‌ای مورد نیازتان را مشخص کنید.

یکی از راه‌هایی که می‌توانید این کار را انجام دهید این است که ویژگی styles را داخل متادیتای کامپوننت تنظیم کنید. این ویژگی آرایه‌ای از رشته‌هایی را می‌گیرد که شامل کد CSS باشند. معمولاً شما مانند مثال زیر تنها یک رشته را به این ویژگی می‌دهید:

src/app/hero-app.component.ts

```
@Component({
 selector: 'app-root',
 template: `
 <h1>Tour of Heroes</h1>
 <app-hero-main [hero]="hero"></app-hero-main>
 `,
 styles: ['h1 { font-weight: normal; }']
```

```
})
export class HeroAppComponent {
/*... */
}
```

حیطه‌ی سبک

سبک‌هایی که داخل متادیتای component@ تعیین می‌شوند تنها داخل قالب این کامپوننت به کار می‌آیند.

سبک‌ها نه توسط هیچ کامپوننت تودرتویی در قالب و نه توسط هیچ محتوای طرح ریزی شده‌ای در کامپوننت به ارث برده نمی‌شوند.

در این مثال، سبک h1 تنها بر روی HeroAppComponent اعمال می‌شود و در هر جای دیگری از برنامه کاری با تگ‌های HeroMainComponent و <h1> ندارد.

این محدودیت گستره یا حیطه یکی از ویژگی‌های پیمانه‌ای بودن سبک بندی است.

- می‌توانید از اسم کلاس‌های CSS و انتخابگرهایی استفاده کنید که از نظر منطقی بیشترین مطابقت را با زمینه‌ی هر یک از کامپوننت ها دارند.
- انتخابگرها و اسامی کلاس‌ها در محل کامپوننت قرار دارند و تعارضی با انتخابگرها و کلاس‌های استفاده شده در بخش‌های دیگر برنامه ندارند.
- هر تغییری که در بخش‌های دیگر برنامه بر روی سبک‌ها ایجاد شوند تاثیری بر سبک‌های کامپوننت نخواهند گذاشت.
- می‌توانید به صورت مشترک، مکان کد CSS هر یک از کامپوننت ها را به همراه کد HTML و تایپ اسکریپت کامپوننت تعیین کنید تا بتوانید ساختار پروژه‌ی خود را تمیز و منظم کنید.
- بدون نیاز به جستجوی کل برنامه برای پیدا کردن مکان کدهای دیگر مورد استفاده، می‌توانید کد CSS کامپوننت را حذف کنید یا تغییر دهید.

انتخابگر های خاص

سبک‌های کامپوننت دارای چند انتخابگر خاص از world of shadow DOM style scoping (که در این لینک در وبسایت W3C به آن پرداخته شده است) هستند. در بخش‌های زیر این انتخابگرها را شرح می‌دهیم.

:host

برای هدف قرار دادن سبک‌های موجود در عنصری که میزبان کامپوننت هستند، می‌توانید از انتخابگر شبه کلاس:host استفاده کنید (این کار برخلاف هدف قرار دادن عناصر داخل قالب کامپوننت است).

src/app/hero-details.component.css

```
:host {
 display: block;
 border: 1px solid black;
}
```

انتخابگر:host تنها راهی است که می‌توان به کمک آن عنصر میزبان را مورد هدف قرار داد. به کمک انتخابگرهای دیگر نمی‌توانید از داخل کامپوننت به عنصر میزبان دسترسی پیدا کنید زیرا این انتخابگر بخشی از قالب خود کامپوننت نیست. عنصر میزبان داخل قالب کامپوننت مادر قرار دارد.

برای به کارگیری سبک‌های میزبان به صورت شرطی با لحاظ کردن انتخابگر دیگر داخل پرانتز و بعد از:host از فرم تابعی استفاده کنید.

مثال زیر بار دیگر عنصر میزبان را هدف قرار داده است، اما تنها وقتی که این عنصر کلاس سی اس اس active را نیز داشته باشد.

src/app/hero-details.component.css

```
:host(.active) {
 border-width: 3px;
}
```

:host-context

گاهی اوقات استفاده از سبک‌ها بر اساس برخی از شرایط خارج از view یک کامپوننت می‌تواند مفید باشد. برای مثال فرض کنید می‌خواهید ظاهر کامپوننت خود را تغییر دهید در این صورت می‌توانید از کلاستم CSS در عنصر <body> سند استفاده کنید.

برای این کار از انتخابگر شبه کلاس:host-context() استفاده کنید. این انتخابگر درست مانند فرم تابعی :host() عمل می‌کند. انتخابگر:host-context() در تمامی نیاکان عنصر میزبان کامپوننت تا ریشه‌ی سند به دنبال یک کلاس CSS می‌گردد. این انتخابگر زمانی که در کنار انتخابگرهای دیگر استفاده شود، می‌تواند کمک شایانی به شما بکند.

در مثال پایین سبک background-color در تمامی عناصر داخل کامپوننت استفاده شده است. تنها در صورتی که یکی از عناصر نیاکان دارای کلاس سی اس اس theme-light باشد.

src/app/hero-details.component.css

```
:host-context(.theme-light) h2 {
 background-color: #eef;
}
```

/deep/، <<< وng-deep:: (منسوخ شده)

سبک‌های کامپوننت معمولاً تنها در HTML داخل قالب خود کامپوننت کاربرد دارند.

برای آن که به اجبار سبکی را از طریق درخت کامپوننت فرزند داخل تمامی view های کامپوننت فرزند اعمال کنید از ترکیب کننده‌ی ضد سایه‌ی فرزند /deep/ استفاده کنید. این ترکیب کننده تا هر عمقی از کامپوننت های تو در تو جواب می‌دهد که هم در فرزندان view و هم در فرزندان محتوای کامپوننت می‌توان از آن استفاده کرد.

مثال زیر تمامی عناصر <h3> از عنصر میزبان گرفته تا تمامی عناصر فرزند آن که در DOM قرار دارند را هدف قرار می‌دهد.

src/app/hero-details.component.css

```
:host /deep/ h3 {
 font-style: italic;
}
```

ترکیب کننده‌ی /deep/ اسامی مستعار <<< و ng-deep:: را نیز دارد.

از /deep/، <<< و ng-deep:: تنها در کنار کپسوله سازی view شبیه سازی شده استفاده کنید. حالت شبیه سازی شده‌ی کپسوله سازی view، حالت پیش فرض و پرکاربردترین حالت آن است. برای اطلاعات بیشتر به بخش «کنترل کپسوله سازی view» مراجعه کنید.

ترکیب کننده‌ی فرزند ضد سایه منسوخ شده است و دیگر توسط ابزارها و مروگرهای اصلی پشتیبانی نمی‌شود. به همین دلیل ما نیز قصد داریم دیگر در Angular (برای /deep/، <<< و ng-deep::) پشتیبانی خود را متوقف کنیم. تا آن زمان بهتر است ازng-deep:: استفاده شود زیرا سازگاری بیشتری با این ابزارها دارد.

بارگیری سبک‌های کامپوننت

روش‌های متعددی برای اضافه کردن سبک‌ها به کامپوننت وجود دارد:

- با تنظیم متادیتای styles یا styleUrls
- کدنویسی درون خطی در HTML قالب
- با import کردن CSS

قوانین حیطه بندی که در گذشته بیان شد، در هر سه الگوی بارگیری بالا صدق می‌کنند.

سبک‌ها در متادیتای کامپوننت

می‌توانید به دکوراتور Component@ ویژگی آرایه‌ی styles را اضافه کنید.

هر یک از رشته‌های موجود در این آرایه برخی از CSS های مورد نیاز این کامپوننت را تعریف می‌کنند.

src/app/hero-app.component.ts (CSS inline)

```
@Component({
 selector: 'app-root',
 template: `
<h1>Tour of Heroes</h1>
<app-hero-main [hero]="hero"></app-hero-main>
`,
```

```
 styles: ['h1 { font-weight: normal; }']
})
export class HeroAppComponent {
/*... */
}
```

توجه: این سبک‌ها تنها در این کامپوننت کاربرد دارند. آن‌ها نه توسط هیچ کامپوننت تو در تویی در قالب و نه توسط هیچ محتوای طرح ریزی شده‌ای در کامپوننت به ارث برده نمی‌شوند.

زمانی که شما کامپوننت را به کمک پرچم --inline-style ایجاد می‌کنید، دستور سی آل آی ng generate یک Angular component آرایه‌ی خالی styles را تعریف می‌کند:

```
ng generate component hero-app --inline-style
```

فایل‌های سبک در متادیتای کامپوننت

با اضافه کردن ویژگی styleUrls به دکوراتور Component@ می‌توانید سبک‌ها را از فایل‌های خارجی CSS بارگیری کنید:

```
src/app/hero-app.component.ts (CSS in file)
@Component({
 selector: 'app-root',
 template: `
 <h1>Tour of Heroes</h1>
 <app-hero-main [hero]="hero"></app-hero-main>
 `,
 styleUrls: ['./hero-app.component.css']
})
export class HeroAppComponent {
/*... */
}
```

src/app/hero-app.component.css

```
h1 {
 font-weight: normal;
}
```

توجه: سبک‌های موجود در این فایل تنها در این کامپوننت کاربرد دارند. آن‌ها نه توسط هیچ کامپوننت تو در تویی در قالب و نه توسط هیچ محتوای طرح ریزی شده‌ای در کامپوننت به ارث برده نمی‌شوند.

می‌توانید بیش از یک فایل سبک و یا حتی ترکیبی از چند styles و styleUrls را تعیین کنید.

زمانی که از دستور سی آل آی ng generate component Angular بدون پرچم --inline-style استفاده می‌کنید، این دستور یک فایل سبک خالی را برای شما ایجاد می‌کند و در styleUrls تولید شده‌ی کامپوننت به این فایل اشاره می‌کند.

```
ng generate component hero-app
```

سبک‌های درون خطی قالب

می‌توانید سبک‌های CSS را به صورت مستقیم و با قرار دادن آن‌ها داخل تگ‌های <style> داخل قالب HTML تعبیه کنید.

src/app/hero-controls.component.ts

```
@Component({
selector: 'app-hero-controls',
template: `
<style>
button {
background-color: white;
border: 1px solid #777;
}
</style>
<h3>Controls</h3>
<button (click)="activate()">Activate</button>
`
})
```

تگ‌های لینک قالب

می‌توانید تگ‌های <link> را داخل قالب HTML کامپوننت نیز بنویسید.

src/app/hero-team.component.ts

```
@Component({
selector: 'app-hero-team',
template: `
<!-- We must use a relative URL so that the AOT compiler can find the stylesheet -->
<link rel="stylesheet" href="../assets/hero-team.component.css">
<h3>Team</h3>
<ul>
<li *ngFor="let member of hero.team">
{{member}}
</li>
</ul>`
})
```

زمانی که در حال ساخت CLI هستید، مطمئن شوید که فایل سبک لینک شده میان asset ها لحاظ شده باشد تا همان طور که در CLI wiki شرح داده شده است، داخل سرور کپی شود.

بعد از این که مطمئن شدید این فایل لحاظ شده است، CLI چه آدرس href تگ لینک نسبتی با ریشه‌ی برنامه یا با فایل کامپوننت داشته باشد و چه نداشته باشد، stylesheet را لحاظ می‌کند.

CSS @imports

می‌توانید با استفاده از قانون @import استاندارد CSS، فایل‌های CSS را داخل فایل‌های CSS ایمپورت کنید. برای جزئیات بیشتر به وبسایت MDN و بخش @import مراجعه کنید.

در این حالت URL با فایل CSS ای که شما داخل آن به import کردن پرداخته‌اید نسبت دارد.

src/app/hero-details.component.css (excerpt)

```
/* The AOT compiler needs the `./` to show that this is local */
@import './hero-details-box.css';
```

فایل‌های سراسری و خارجی سبک

زمانی که به کمک CLI در حال ساخت بخش‌های برنامه‌ی خود هستید، می‌توانید angular.json را به گونه‌ای پیکربندی کنید که تمامی asset های خارجی از جمله فایل‌های خارجی سبک را شامل شود.

فایل‌های سبک سراسری را در بخش styles که توسط فایل سراسری styles.css به صورت پیش فرض از قبل پیکربندی شده است، ثبت کنید.

برای اطلاعات بیشتر به CLI wiki مراجعه کنید.

فایل‌های سبک غیر CSS

اگر به کمک CLI در حال ساخت برنامه‌ی خود هستید، می‌توانید فایل‌های سبک را در sass، less یا stylus بنویسید و مانند مثال زیر این فایل‌ها را همراه با پسوندهای مناسب (.styl،.less،.scss) داخل متادیتای Component.styleUrls@ مشخص کنید:

```
@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.scss']
})
...
```

فرآیند ساخت CLI پیش پردازنده‌های CSS مرتبط را اجرا می‌کند.

زمانی که فایل کامپوننتی را به کمک ng generate component ایجاد می‌کنید، CLI به صورت پیش فرض یک فایل سبک CSS خالی (.css) را از خود خارج می‌کند. همان طور که در CLI wiki توضیح داده شده است، می‌توانید CLI را به صورت پیش فرض بر روی پیش پردازنده‌های CSS ترجیحی خود پیکربندی کنید.

رشته‌های سبک اضافه شده به آرایه‌ی Component.styles@ باید در CSS نوشته شوند، زیرا CLI نمی‌تواند بر روی سبک‌های درون خطی، پیش پردازنده‌ها را اعمال کند.

کپسوله سازی view

همان طور که قبلاً نیز بیان شد، سبک‌های CSS کامپوننت داخل view کامپوننت کپسوله می‌شوند و تحت تأثیر بخش‌های دیگر برنامه قرار نمی‌گیرند.

برای آن که کنترل کنید که این کپسوله سازی در هر یک از کامپوننت ها چگونه اتفاق بیفتند، می‌توانید در متادیتای کامپوننت حالت کپسوله سازی view را تنظیم کنید. این حالت‌ها را می‌توانید در زیر مشاهده کنید:

- کپسوله سازی ویوی ShadowDom برای متصل کردن یک DOM سایه به عنصر میزبان کامپوننت از پیاده سازی DOM سایه‌ی بومی مرورگر استفاده می‌کند (به بخش ShadowDom سایت MDN مراجعه کنید). بعد از این کار view کامپوننت را داخل این DOM سایه قرار می‌دهد. سبک‌های کامپوننت داخل DOM سایه وجود دارند.
- کپسوله سازی ویوی Native از نسخه‌ی منسوخ پیاده سازی DOM سایه‌ی بومی مرورگر استفاده می‌کند. درباره‌ی تغییرات آن بیشتر بخوانید.
- کپسوله سازی ویوی Emulated (پیش فرض) با پیش پردازش و (تغییر اسم) کد CSS رفتار DOM سایه را تقلید می‌کند تا گستره‌ی CSS به view کامپوننت برسد. برای اطلاعات بیشتر به پیوست 1 مراجعه کنید.
- None به این معنی است که Angular هیچ کپسوله سازی view ای ندارد. Angular CSS را به سبک‌های سراسری اضافه می‌کند. قوانین حیطه بندی، جداسازی و اصول حفاظتی که قبلاً بیان شدند در این جا صدق نمی‌کنند. این کار اساساً هیچ تفاوتی با paste کردن سبک‌های کامپوننت در HTML ندارد.

برای تنظیم حالت کپسوله سازی کامپوننت ها، داخل متادیتای کامپوننت از ویژگی encapsulation استفاده کنید:

src/app/quest-summary.component.ts

```
// warning: few browsers support shadow DOM encapsulation at this time
encapsulation: ViewEncapsulation.Native
```

کپسوله سازی ویوی ShadowDom تنها در مرورگرهایی جواب می‌دهد که به صورت بومی از ShadowDom پشتیبانی می‌کنند (به بخش ShadowDom v1 در وبسایت Can I use مراجعه کنید). سطح پشتیبانی همچنان محدود است، به همین دلیل است که در اغلب موارد کپسوله سازی ویوی Emulated حالت پیش فرض است و استفاده از آن توصیه می‌شود.

بررسی CSS تولید شده

در مواقعی که از کپسوله سازی شبیه سازی شده استفاده می‌کنید، Angular تمامی سبک‌های کامپوننت را پیش پردازش می‌کند، به گونه‌ای که این سبک‌ها به قوانین حیطه بندی استاندارد CSS سایه نزدیک شوند.

داخل DOM یک برنامه‌ی Angular در حال اجرا که در آن کپسوله سازی شبیه سازی شده فعال است به هر یک از عناصر DOM تعدادی صفت اضافی متصل شده است:

```
<hero-details _nghost-pmm-5>
 <h2 _ngcontent-pmm-5>Mister Fantastic</h2>
 <hero-team _ngcontent-pmm-5 _nghost-pmm-6>
 <h3 _ngcontent-pmm-6>Team</h3>
 </hero-team>
</hero-detail>
```

این صفات دو نوع دارند:

- عنصری که در کپسوله سازی بومی یک میزبان DOM سایه باشد، صفت تولید شده‌ی آن _nghost خواهد بود. این حالت معمولاً در عناصر میزبان کامپوننت اتفاق می افتد.
- عنصری که داخل view کامپوننت قرار دارد، دارای صفت _ngcontent است. این صفت تشخیص می‌دهد که این عنصر به کدام DOM سایه‌ی شبیه سازی شده‌ی میزبان تعلق دارد.

مقادیر دقیق این صفات اهمیتی ندارند. زیرا این مقادیر به صورت خودکار تولید می‌شوند و شما هیچ وقت در کد برنامه به آن‌ها اشاره نمی‌کنید. اما سبک‌های کامپوننت تولید شده که در بخش <head> مربوط به DOM قرار دارند، این مقادیر را هدف قرار می‌دهند:

```
[_nghost-pmm-5] {
 display: block;
 border: 1px solid black;
}

h3[_ngcontent-pmm-6] {
 background-color: white;
```

```
 border: 1px solid #777;
}
```

این سبک‌ها به گونه‌ای پس پردازش شده‌اند که هر یک از انتخابگرها با انتخابگرهای صفت _nghost یا _ngcontent تکمیل شوند. این انتخابگرهای اضافی قوانین حیطه بندی بیان شده در این صفحه را فعال می‌کنند.