

Advanced Styles & Bootstrap

UI-Angular



מה נלמד היום?

- Angular Global Styles
- View Encapsulation
 - None
 - Emulated
 - ShadowDOM
- Style Binding
- Bootstrap in Angular



Angular Global Styles

- ישנן מספר דרכים להוסיף סגנונות עיצוב גלובליים ליישום Angular:
 - ייבוא ל-index.html
 - הוספה ב-angular.json
- למדנו גם איך להוסיף סגנונות עיצוב מקומיים ב-component
 - *component-name.component.css*



Index.html הוספת סגנון גלובלי

Index.html

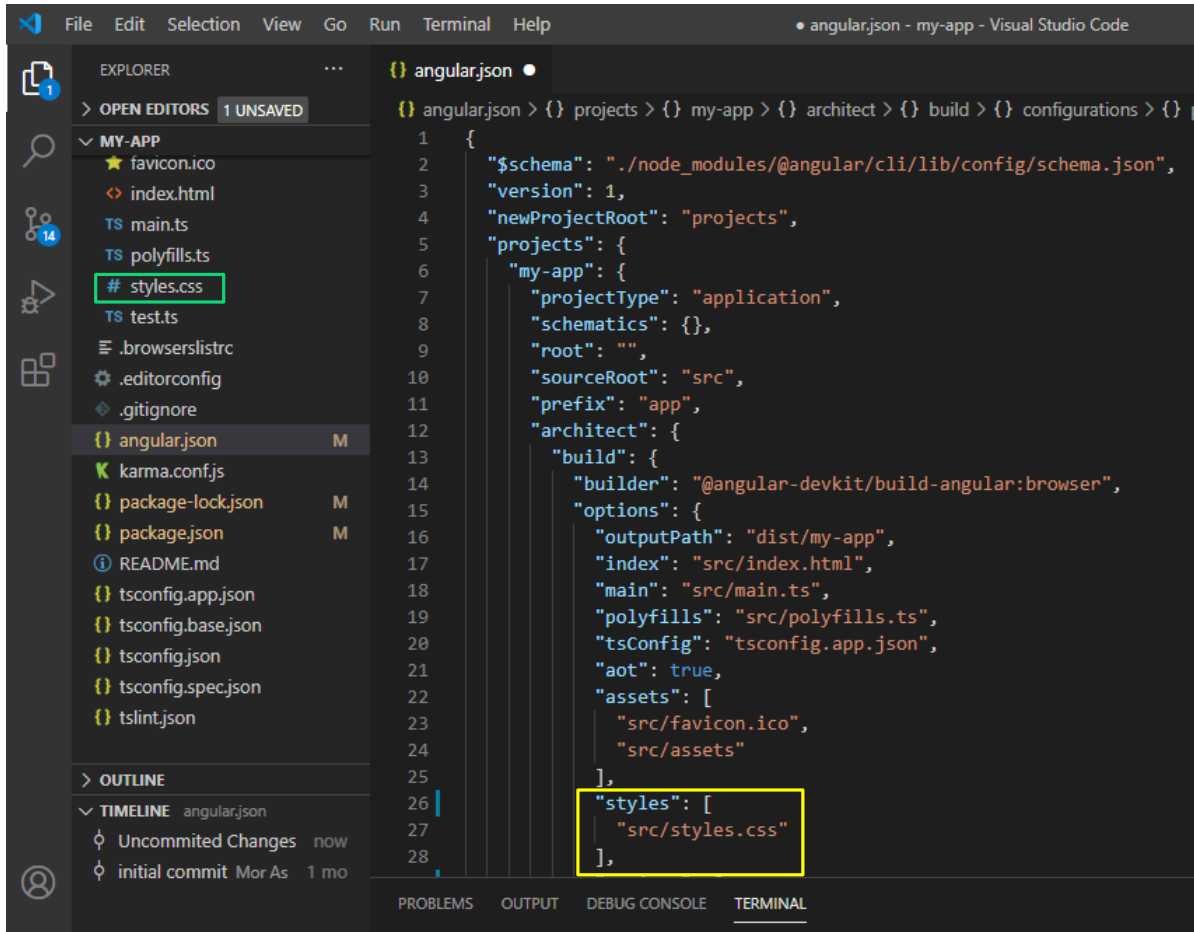
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title> CSS Example </title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="assets/css/morestyles.css">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

- ניתן להוסיף מס' קבצי css משלנו
- ניצור קובץ css חדש בשם morestyles בתיקייה assets/css
- נגדיר בקובץ סגנונות חדשים
- נוסיף את הקובץ ל-index.html
- כעת הסגנונות מוחלים על כל היישום

Angular Global Styles

angular.json

הוספת סגנון גלובלי

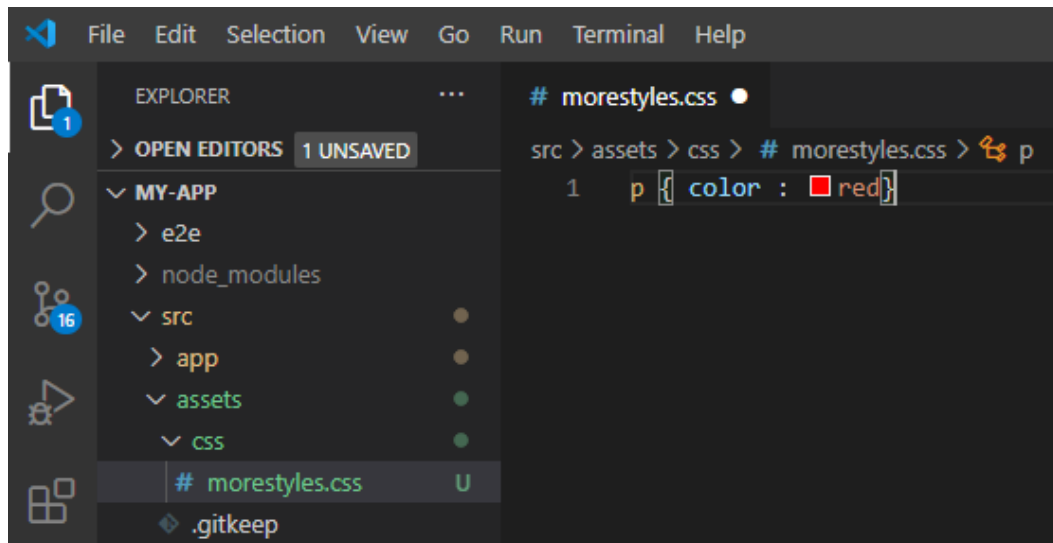


```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "my-app": {
      "projectType": "application",
      "schematics": {},
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/my-app",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "aot": true,
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/styles.css"
            ],
            "scripts": []
          },
          "configurations": {
            "production": {
              "optimization": true,
              "outputPath": "dist/my-app",
              "sourceMap": false,
              "namedChunks": false,
              "aot": true,
              "extractLicenses": true,
              "vendorChunk": false,
              "buildOptimizer": true
            }
          }
        }
      }
    }
  }
}
```

- עם יצירת יישום angular ב-angular cli נוצר קובץ בשם angular.json
- בקובץ זה קיים מערך סגנונות בשם styles ובתוכו נוכל להגדיר קבצי css שיוחלו בצורה גלובלית על היישום כולו.
- קובץ נוסף שנוצר באופן אוטומטי עם יצירת היישום הוא styles.css והוא מוגדר באופן ברירת מחדל במערך styles.

angular.json הוספת סגנון גלובלי

- ניתן להוסיף מס' קבצי css משלנו למערך styles
- ניצור קובץ css חדש בשם morestyles בתיקייה src/assets/css
- נגדיר בקובץ סגנונות חדשים
- נוסיף את הקובץ למערך styles
- כעת הסגנונות מוחלים על כל היישום

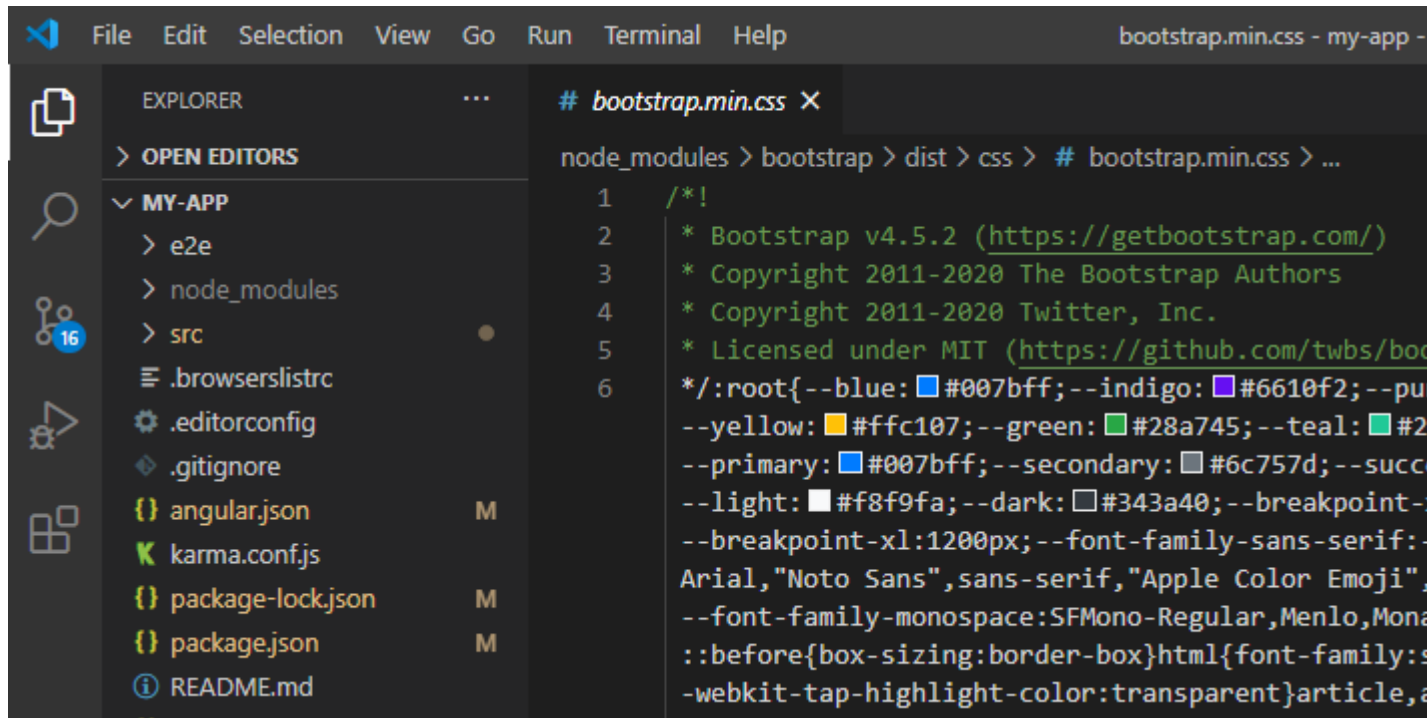


```
"styles": [  
  "src/styles.css",  
  "src/assets/css/morestyles.css"  
],
```

הוספת סגנון גלובלי

angular.json

- ניתן להוסיף מס' קבצי css חיצוניים למערך styles



```
"styles": [  
  "src/styles.css",  
  "src/assets/css/bootstrap.min.css"  
],
```

ואם לא לגמרי גלובלי?

- אם נרצה שלקומפוננטה יהיה סגנון משלה המוסתר מכל חלקי היישום?
- אם נרצה לשלוט ברמת החשיפה של הסגנון?



*Scoping Your Styles
in Angular With
ViewEncapsulation*

View Encapsulation



- מגדיר כיצד יישום Angular מסתיר (אם בכלל) סגנונות קומפוננטה משאר חלקי היישום.
- מגדיר כיצד הסגנונות המוגדרים ב-template משפיעים על חלקי היישום האחרים.
- אחד המושגים הבסיסיים בתכנות מונחה עצמים (OOP) הוא כימוס. בהקשר של מחלקות, כל הנתונים והמתודות הפועלות על הנתונים נשמרים פרטיים במחלקה אחת.

View Encapsulation

```
enum ViewEncapsulation {  
  Emulated: 0  
  Native: 1  
  None: 2  
  ShadowDom: 3  
}
```

■ 3 גישות:

- ViewEncapsulation.None
- ViewEncapsulation.Emulated
- ViewEncapsulation.ShadowDOM

View Encapsulation

הוספת View Encapsulation לקומפוננטה

TS app.component.ts •

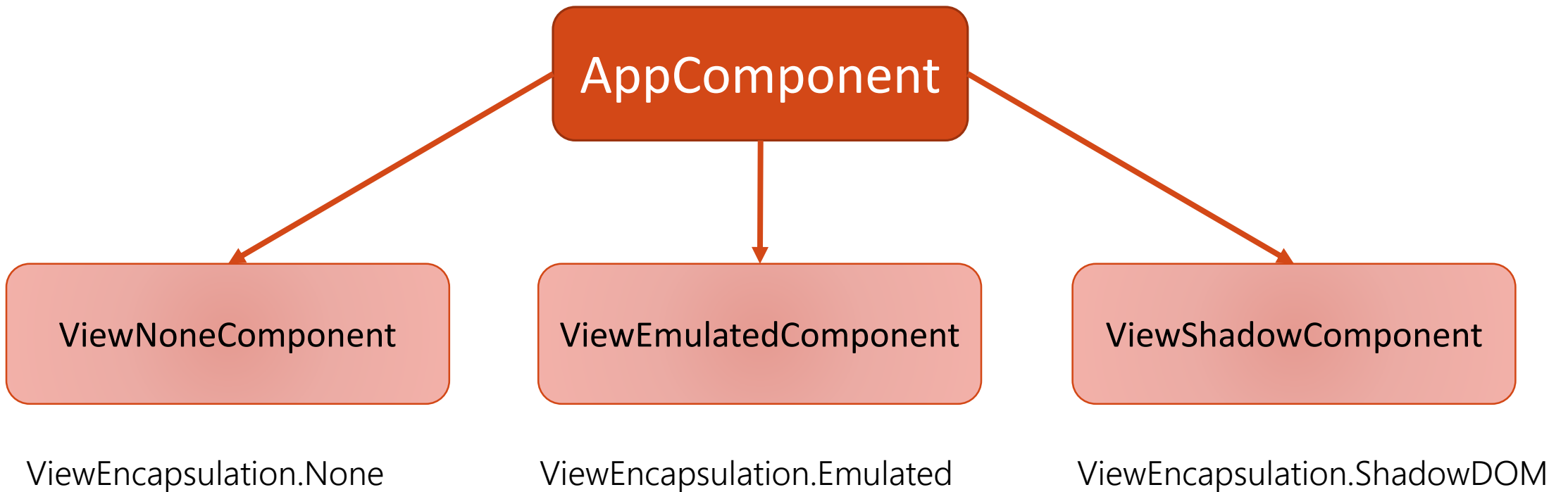
src > app > TS app.component.ts > ...

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'Angular UI'
10 }
```

```
import { Component, ViewEncapsulation } from '@angular/core';
```

```
@Component({
  selector: 'app-root',
  template: `<p>Using Emulator</p>`,
  styles: ['p { color:red}'],
  encapsulation: ViewEncapsulation.Emulated //The default
//encapsulation: ViewEncapsulation.None
//encapsulation: ViewEncapsulation.ShadowDOM
})
```

הוספת View Encapsulation לקומפוננטה



ViewEncapsulation.None

- נשתמש כאשר אין צורך בכימוס.
- הסגנונות המוגדרים בקומפוננטה אחת משפיעים על האלמנטים בקומפוננטות אחרות.

viewNone.component.ts

```
import { Component, ViewEncapsulation } from '@angular/core';

@Component({
  selector: 'app-none',
  template: `

I am not encapsulated and in blue (ViewEncapsulation.None) </p>`,
  styles: ['p { color:blue}'],
  encapsulation: ViewEncapsulation.None
})
export class ViewNoneComponent {
}


```

View Encapsulation

ViewEncapsulation.None

app.component.html

```
<h1>{{title}}</h1>  
<p>I am a paragraph in green</p>
```

```
<app-none></app-none>
```

Styles.css

```
p {color: green;}
```

View Encapsulation in Angular

I am a paragraph in green

I am not encapsulated and in blue (ViewEncapsulation.None)

ViewEncapsulation.None



מסקנות:

- הסגנונות המוגדרים בקומפוננטה זולגים לשאר הקומפוננטות
- הסגנון שהוגדר בקומפוננטה "עוקף" את הסגנון שהוגדר גלובלית ב-styles.css

ViewEncapsulation.Emulated

- לא כימוס אמיתי, אך דומה (חיקוי, אמולציה)
- ישנה הפרדה בין סגנונות css של קומפוננטה לשאר הקומפוננטות

viewEmulated.component.ts

```
import { Component, ViewEncapsulation } from '@angular/core';

@Component({
  selector: 'app-emulated',
  template: `<p>I am now encapsulated using ViewEncapsulation.Emulated </p>`,
  styles: ['p { color:red}'],
  encapsulation: ViewEncapsulation.Emulated
})
export class ViewEmulatedComponent {
}
```


View Encapsulation

ViewEncapsulation.Emulated

app.component.html

```
<h1>{{title}}</h1>  
<p>I am a paragraph in green</p>  
  
<app-none></app-none>  
<app-emulated></app-emulated>
```

Styles.css

```
p {color: green;}
```

View Encapsulation in Angular

I am a paragraph in green

I am not encapsulated and in blue (ViewEncapsulation.None)

I am now encapsulated using ViewEncapsulation.Emulated

ViewEncapsulation.Emulated

View Encapsulation in Angular

I am a paragraph in green

I am not encapsulated and in blue (ViewEncapsulation.None)

I am now encapsulated using ViewEncapsulation.Emulated

The screenshot shows the Chrome DevTools 'Elements' panel with the following DOM structure:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>ViewEncapsulation</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <style type="text/css">...</style>
    <style>...</style>
    <style>p { color:blue}</style>
    <style>p[_ngcontent-c2] { color:red}</style>
  </head>
  <body>
    <app-root _ngghost-c0 ng-version="7.1.4">
      <h1 _ngcontent-c0>View Encapsulation in Angular</h1>
      <p _ngcontent-c0>I am a paragraph in green</p>
      <app-none _ngcontent-c0>...</app-none>
      <app-emulated _ngcontent-c0 _ngghost-c2>
        <p _ngcontent-c2>I am now encapsulated using ViewEncapsulation.Emulated</p>
      </app-emulated>
    </app-root>
```

Annotations:

- CSS rule from the ViewEmulatedComponent
- _ngcontent-c2 attribute is added by angular
- _ngcontent-c2 attribute inserted to emulate shadow DOM

ViewEncapsulation.Emulated



מסקנות:

- הסגנון המוגדר בקומפוננטה לא משפיע על שאר הקומפוננטות
- הסגנון שמוגדר בקומפוננטה הוא רק בסקופ של הקומפוננטה
- עבודת אנגולר מאחורי הקלעים:
 - אנגולר מוסיפה שדה מזהה של סגנון הקומפוננטה ומשייכת אותו לאלמנט html הרלוונטי (זה שהוגדר בקומפוננטה)
- אין כימוס אמיתי כי עדיין הסגנונות חשופים (אך ללא השפעה) לשאר היישום

ViewEncapsulation.ShadowDOM

- כימוס אמיתי. נוצר עץ סגנונות במקביל ל-DOM הנקרא ShadowDOM
- ישנה היררכיה של הסגנונות בין אלמנטי אב, בן, אחים...

viewShadow.component.ts

```
import { Component, ViewEncapsulation } from '@angular/core';

@Component({
  selector: 'app-shadowdom',
  template: `<p>I am encapsulated inside a Shadow DOM ViewEncapsulation.ShadowDom</p>`,
  styles: ['p { color:brown}'],
  encapsulation: ViewEncapsulation.ShadowDom
})
export class ViewShadowdomComponent {
}
```

View Encapsulation

ViewEncapsulation.ShadowDOM

app.component.html

```
<h1>{{title}}</h1>  
<p>I am a paragraph in green</p>  
  
<app-none></app-none>  
<app-emulated></app-emulated>  
<app-shadowdom></app-shadowdom>
```

Styles.css

```
p {color: green;}
```

View Encapsulation in Angular

I am a paragraph in green

I am not encapsulated and in blue (ViewEncapsulation.None)

I am now encapsulated using ViewEncapsulation.Emulated

I am encapsulated inside a Shadow DOM ViewEncapsulation.ShadowDom

ViewEncapsulation.ShadowDOM

View Encapsulation in Angular

I am a paragraph in green

I am not encapsulated and in blue (ViewEncapsulation.None)

I am now encapsulated using ViewEncapsulation.Emulated

I am encapsulated inside a Shadow DOM ViewEncapsulation.ShadowDom

```
<!doctype html>
<html lang="en">
  <head>...</head>
  <body>
    <app-root _ngghost-c0 ng-version="7.1.4">
      <h1 _ngcontent-c0>View Encapsulation in Angular</h1>
      <p _ngcontent-c0>I am a paragraph in green</p>
      <app-none _ngcontent-c0>...</app-none>
      <app-emulated _ngcontent-c0 _ngghost-c2>...</app-emulated>
      <app-shadowdom _ngcontent-c0>
        <#shadow-root (open)>
          <style>...</style>
          <p { color:blue}</p>
          <style>p[_ngcontent-c2] { color:red}</style>
          <style>p { color:brown}</style>
          <p>I am encapsulated inside a Shadow DOM ViewEncapsulation.ShadowDom</p>
        </#shadow-root>
      </app-shadowdom>
    </app-root>
  </body>
</html>
```

Styles are also copied from parent and sibling component

The Shadow DOM starts at #shadow-root

It is rendered independently from the rest of the document

app-shadowdom becomes Shadow host, to which Shadow DOM is attached

Style copied from the ShadowDomComponent

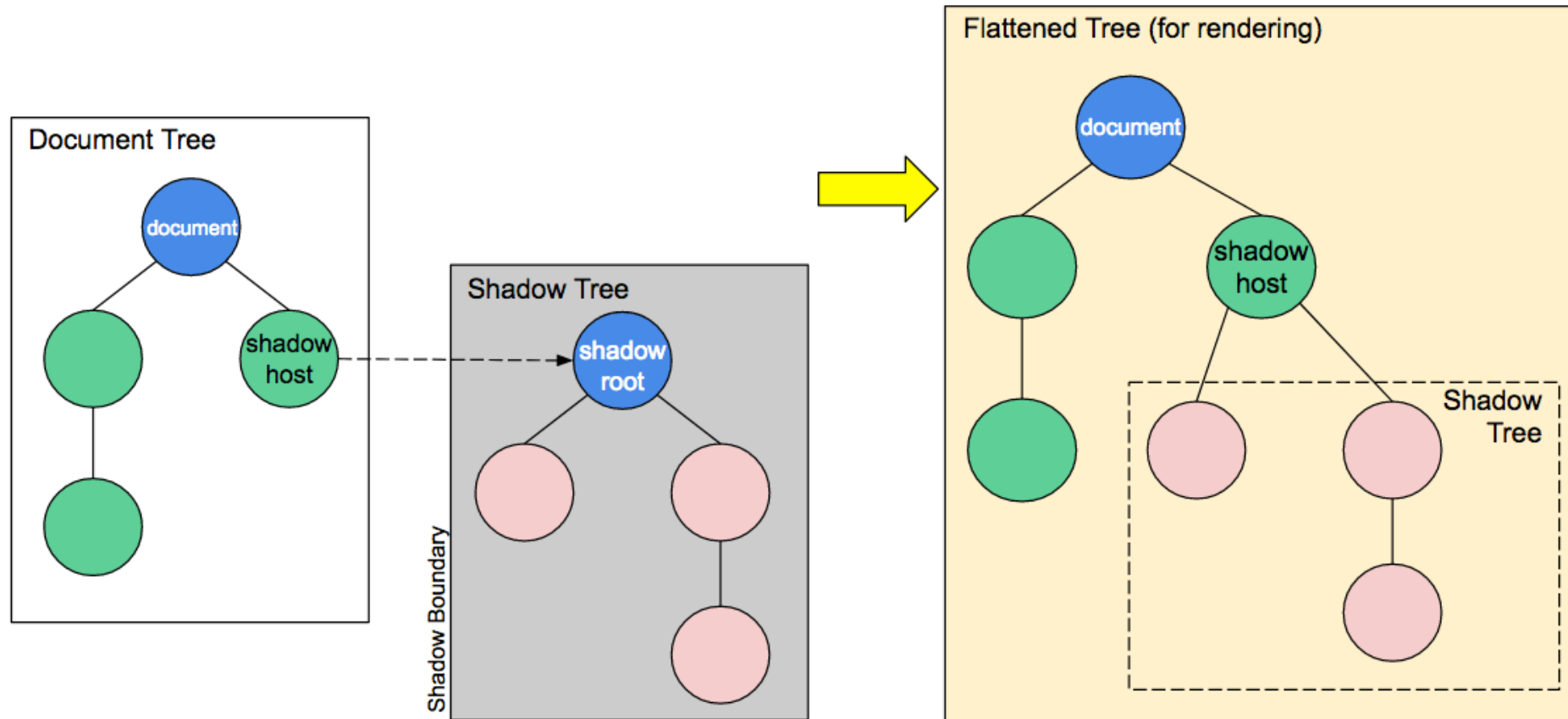
ViewEncapsulation.ShadowDOM



מסקנות:

- הסגנון המוגדר בקומפוננטה לא משפיע על שאר הקומפוננטות
- הסגנון שמוגדר בקומפוננטה הוא רק בסקופ של הקומפוננטה
- עבודת אנגולר מאחורי הקלעים:
 - יצירת shadowDOM עבור הקומפוננטה והגדרת תגית הקומפוננטה בתור shadow-host
- כל המידע אודות סגנונות הקומפוננטה מצוי אך ורק תחת ה-shadowDOM שעיבודו מתבצע בנפרד מה-DOM ולכן מדובר בכימוס אמיתי

ViewEncapsulation.ShadowDOM



Style Binding

ניתן להגדיר inline style של אלמנט html בדומה לשיטה בה הגדרנו Property Binding ■

```
[style.style-property] = "style-value"
```

```
<p [style.color]='red'> Give me red </p>
```

```
<button [style.border]='5px solid yellow'> Save </button>
```

Style Binding

- ניתן להגדיר באופן דינאמי עם משפטי תנאי

```
[style.style-property] = "condition"
```

```
<button [style.color]="status=='error' ? 'red': 'blue'"> Button 1 </button>
```

Style Binding

- ניתן להגדיר בעזרת פונקציה

```
[style.style-property] = "func()"
```

Component class

```
getColor() {  
  return 'yellow';  
}
```

app.component.html

```
<button [style.color]="getColor()"> Button 2 </button>
```

Style Binding

- ניתן להגדיר מספר עיצובים בו זמנית

```
<p [style.color]="getColor()"
  [style.font-size.px]="20"
  [style.background-color]="status=='error' ? 'red': 'blue'">
  paragraph with multiple styles
</p>
```

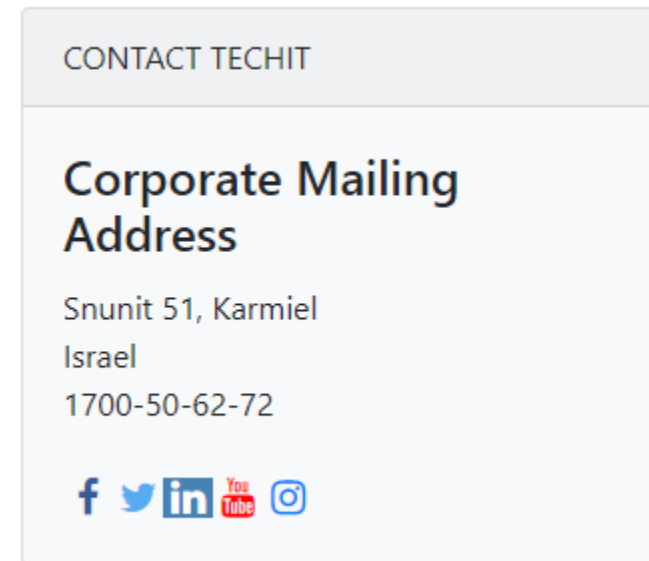
Bootstrap

- Bootstrap שוחררה בשנת 2011 על ידי טוויטר, אחרי שהתחילה כספרייה סגורה שנוצרה לשימוש פנימי בטוויטר
- סביבת עבודה (framework) בקוד פתוח לצד לקוח, שמכילה אוסף של כלים רספונסיביים ליצירת אפליקציות ואתרים
- 3 חלקים:
 - CSS, רכיבי ממשק משתמש ו-Javascript



Bootstrap

```
<div class="card-header">CONTACT TECHIT</div>
<div class="card-body">
  <h4 class="card-title">Corporate Mailing Address</h4>
  <p class="card-text">Snunit 51, Karmiel <br>
    Israel <br>
    1700-50-62-72 <br><br>
    <a href="#" class="fa fa-facebook"></a>
    <a href="#" class="fa fa-twitter"></a>
    <a href="#" class="fa fa-linkedin"></a>
    <a href="#" class="fa fa-youtube"></a>
    <a href="#" class="fa fa-instagram"></a>
  </p>
</div>
```

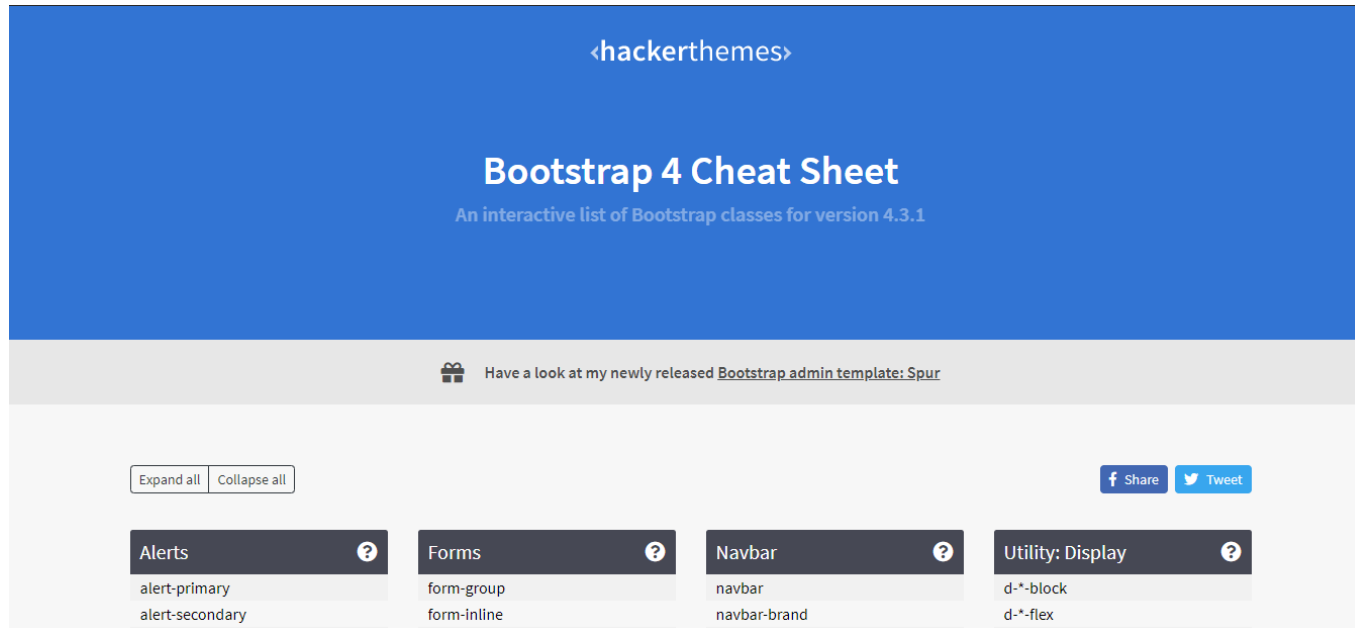


Adding Bootstrap

מחלקות Bootstrap

<https://hackerthemes.com/bootstrap-cheatsheet/>

האתר מכיל את המחלקות הקיימות ב-Bootstrap



Adding Bootstrap

מחלקות Bootstrap

Alerts ? <ul style="list-style-type: none">alert-primaryalert-secondaryalert-successalert-infoalert-warningalert-dangeralert-lightalert-darkalert-linkalert-dismissiblealert-heading	Forms ? <ul style="list-style-type: none">form-groupform-inlineform using the gridform-controlform-control-lgform-control-smform-control-fileform-control-plaintextform-control-rangeform-checkform-check-inlinedisabled itemsreadonly	Navbar ? <ul style="list-style-type: none">navbarnavbar-brandnavbar with formnavbar-textnavbar-dark bg-darknavbar-lightnavbar fixed-topnavbar fixed-bottomnavbar sticky-topcollapse navbar-collapsenavbar-togglernavbar-expand-*	Utility: Display ? <ul style="list-style-type: none">d-*.blockd-*.flexd-*.inlined-*.inline-blockd-*.inline-flexd-*.noned-*.tabled-*.table-celld-print-...
Badges ? <ul style="list-style-type: none">badgebadge-pillbadge-primarybadge-secondarybadge-successbadge-infobadge-warningbadge-danger	Form Input Groups ? <ul style="list-style-type: none">input-groupinput-group-prependinput-group-appendinput-group-sminput-group-lgcheckbox	Pagination ? <ul style="list-style-type: none">paginationpage-item disabledpage-item activepagination-lgpagination-sm	Utility: Flexbox ? <ul style="list-style-type: none">flex-*.columnflex-*.column-reverseflex-*.rowflex-*.row-reverseflex-*.nowrapflex-*.wrapflex-*.wrap-reverseflex-fillflex-*.grow-1flex-*.grow-0
		Popover ?	

Adding Bootstrap

מחלקות Bootstrap

Alerts ?

alert-primary
alert-secondary
alert-success
alert-info
alert-warning
alert-danger
alert-light
alert-dark
alert-link
alert-dismissible
alert-heading

Forms ?

form-group
form-inline
form using the grid
form-control
form-control-lg
form-control-sm
form-control-file
form-control-plaintext
form-control-range
form-check
form-check-inline

Navbar ?

navbar
navbar-brand
navbar with form
navbar-text
navbar-dark bg-dark
navbar-light
navbar fixed-top
navbar fixed-bottom
navbar sticky-top
collapse navbar-collapse
navbar-toggler

Utility: Display ?

d-*-block
d-*-flex
d-*-inline
d-*-inline-block
d-*-inline-flex
d-*-none
d-*-table
d-*-table-cell
d-print-...

Utility: Flexbox ?

Code snippet

Copy

```
<div class="alert alert-success" role="alert">  
  <strong>Well done!</strong> You successfully read this  
  important alert message.  
</div>
```

Preview

X

Well done! You successfully read this important alert message.

←

alert-success

→

התקנת Bootstrap

ניתן להוסיף את Bootstrap ליישום angular ב-3 אופנים:

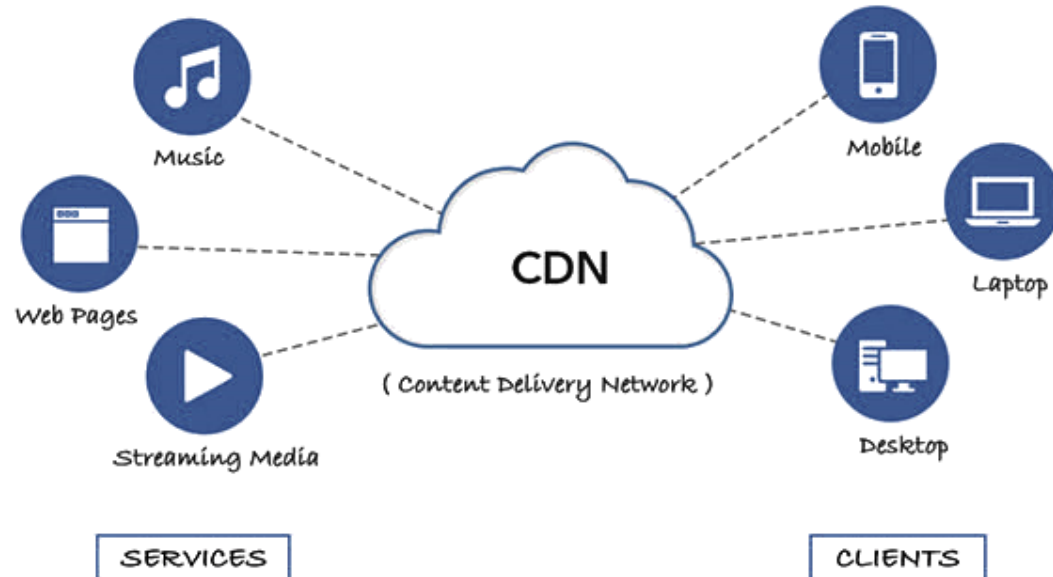
- הוספה ל-index.html ע"י שימוש ב-CDN*
- ייבוא ל-styles.css ע"י שימוש ב-CDN*
- התקנה באמצעות angular cli



*CDN (content delivery network)

התקנת Bootstrap

- הוספה ל-index.html ע"י שימוש ב-CDN
 - CDN (content delivery network) הינה רשת של שרתים ברחבי העולם, אשר מטרתם היא האצה של אתרים קיימים אשר משתמשים בה.
 - לרוב שרתי CDN יאחסנו תמונות וקבצי וידאו של האתר, אך במקרים מיוחדים הם לרוב יכילו גם את קבצי האתר הסטטיים כגון HTML, CSS, JavaScript



התקנת Bootstrap

- הוספה ל-index.html ע"י שימוש ב-CDN

Index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>BootstrapCSSExample</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
    integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
    crossorigin="anonymous">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

התקנת Bootstrap

- ייבוא ל-styles.css ע"י שימוש ב-CDN

Styles.css

```
@import "https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
```

התקנת Bootstrap

- התקנה באמצעות angular cli
- במקום להריץ מה-CDN אנחנו מתקינים Bootstrap באופן מקומי

שלב 1: הרצת הפקודה לאחר יישום Angular

```
npm install --save bootstrap
```

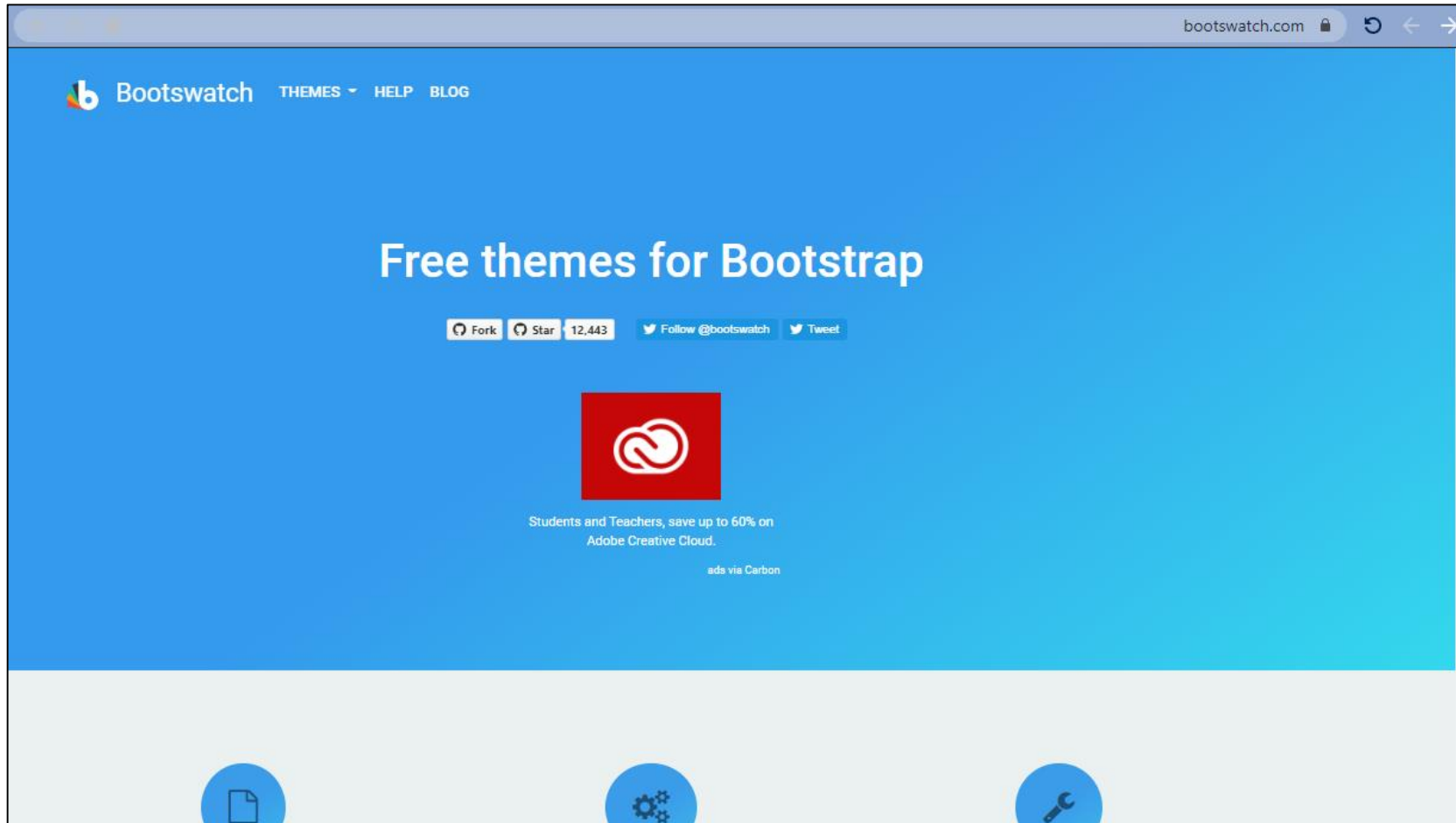
שלב 2: הוספת Bootstrap למערך styles ב-angular.json

```
"styles": [  
  "node_modules/bootstrap/dist/css/bootstrap.min.css",  
  "src/styles.css"  
],
```

Adding Bootstrap

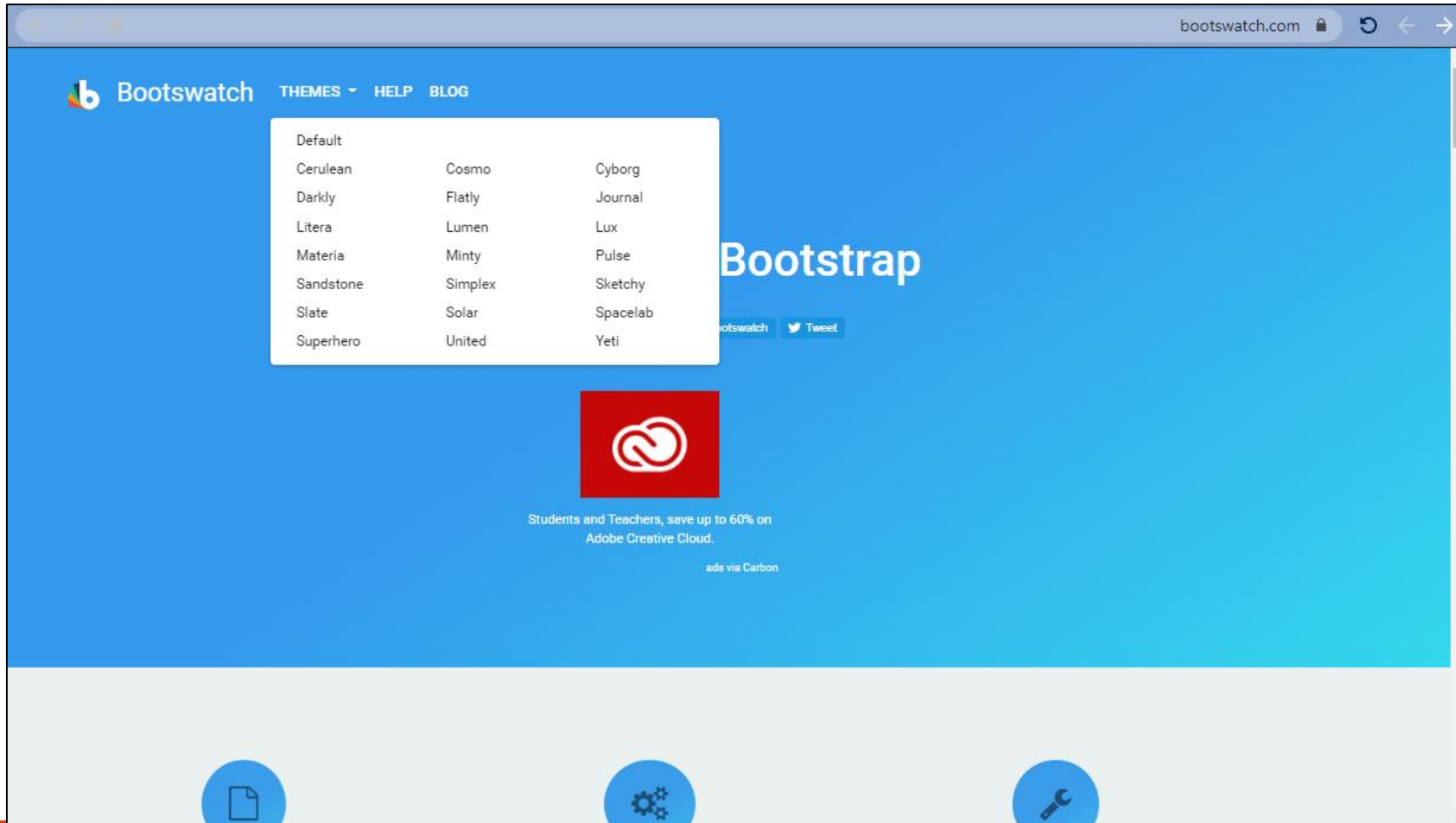
<https://bootswatch.com/>

Bootswatch



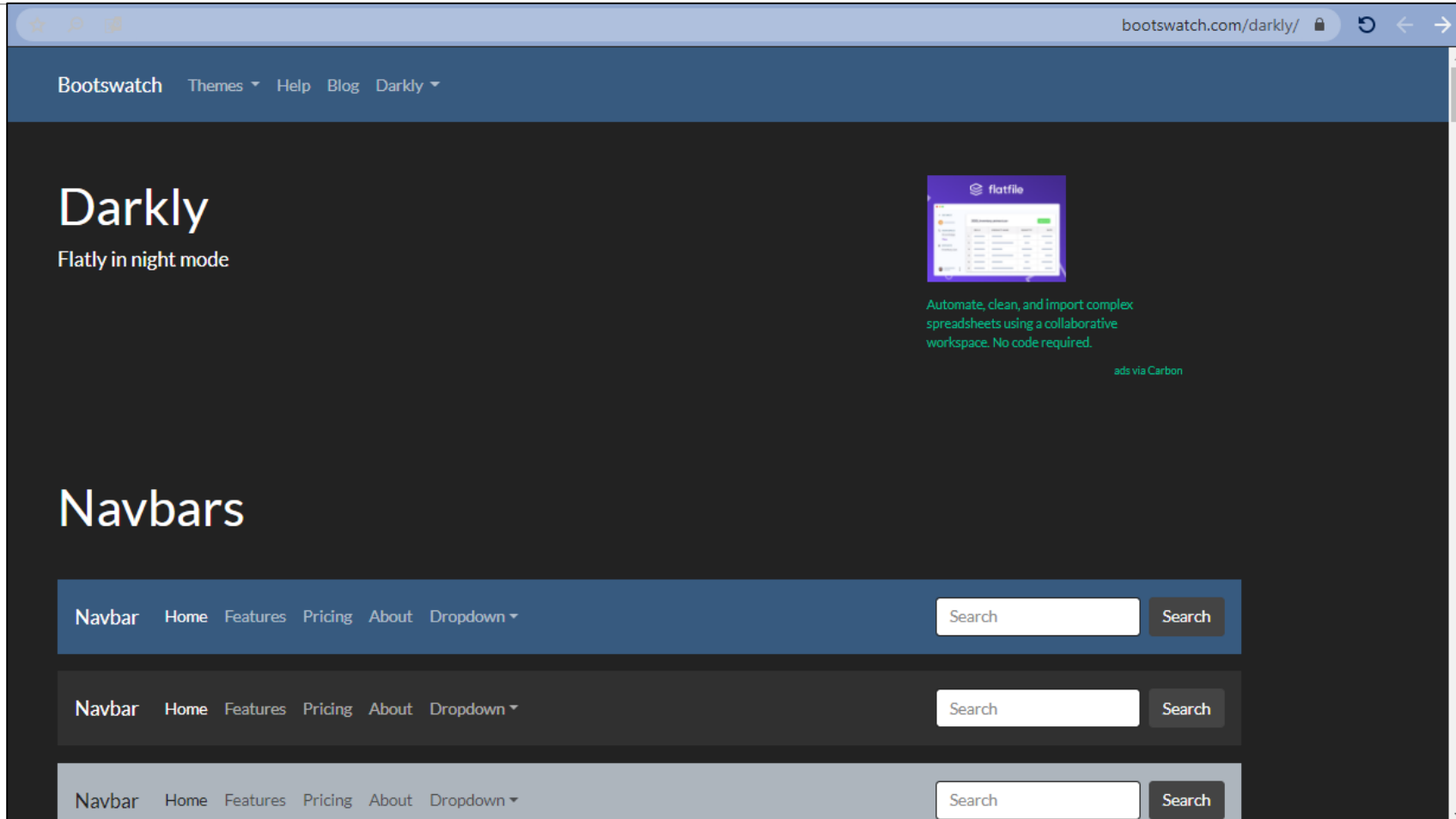
Adding Bootstrap

Bootswatch



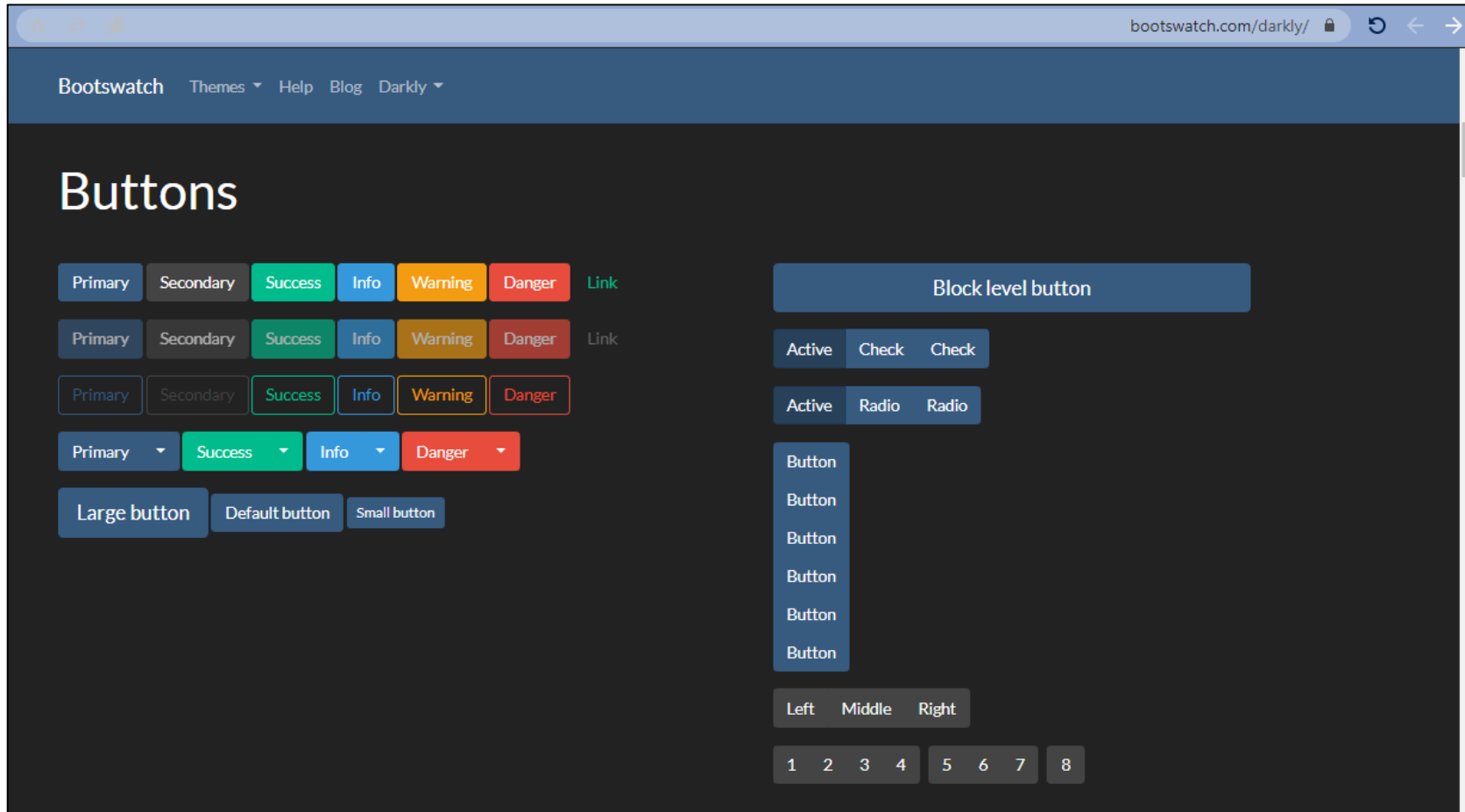
Adding Bootstrap

Bootswatch



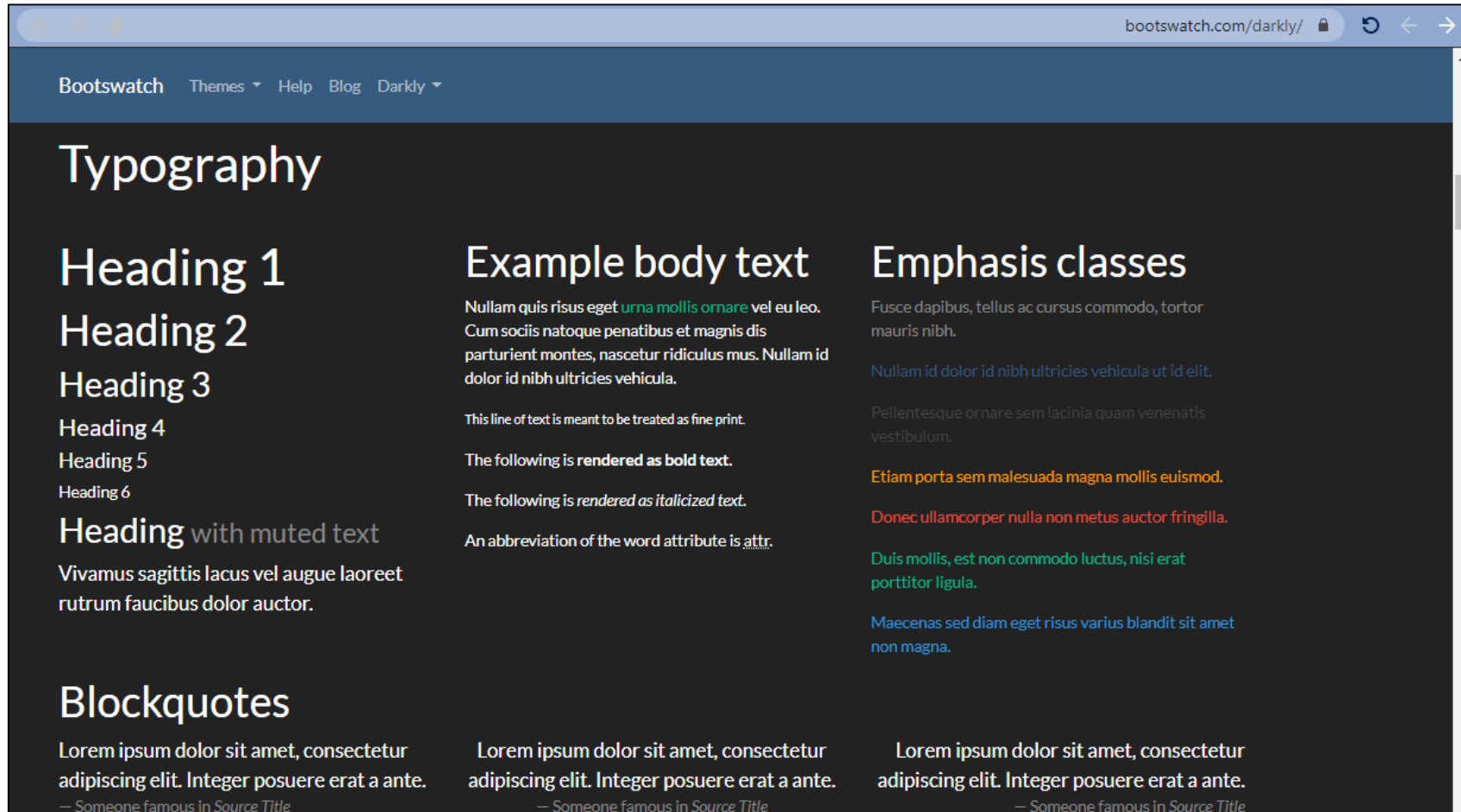
Adding Bootstrap

Bootswatch



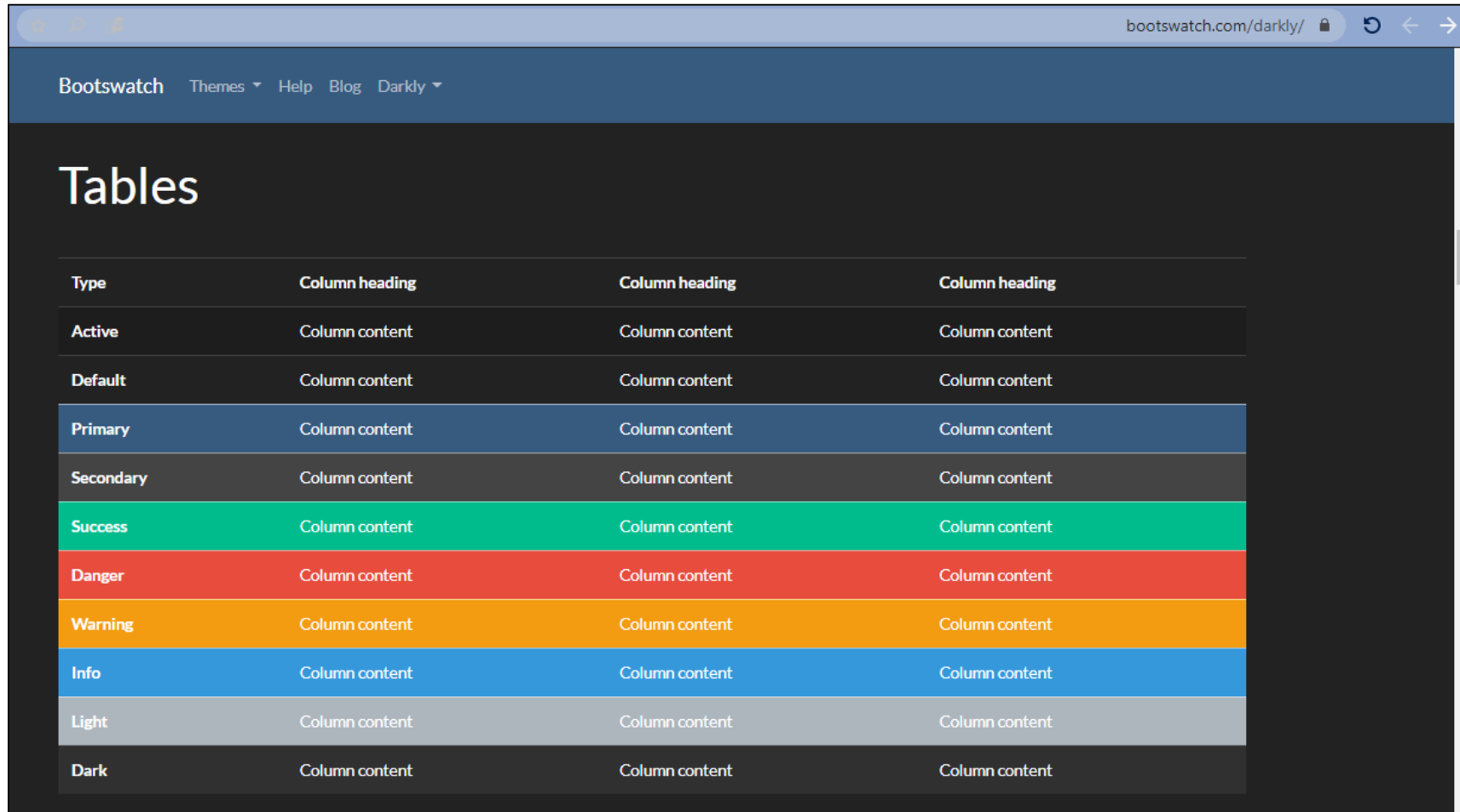
Adding Bootstrap

Bootswatch



Adding Bootstrap

Bootswatch



The screenshot shows the Bootswatch website in a web browser. The page title is 'Tables'. It displays a table with 4 columns: 'Type', 'Column heading', 'Column heading', and 'Column heading'. The table rows represent different Bootstrap table styles, each with a unique background color. The 'Active' row is highlighted in dark blue. The 'Default' row is white. The 'Primary' row is blue. The 'Secondary' row is dark gray. The 'Success' row is green. The 'Danger' row is red. The 'Warning' row is orange. The 'Info' row is light blue. The 'Light' row is light gray. The 'Dark' row is dark gray.

Type	Column heading	Column heading	Column heading
Active	Column content	Column content	Column content
Default	Column content	Column content	Column content
Primary	Column content	Column content	Column content
Secondary	Column content	Column content	Column content
Success	Column content	Column content	Column content
Danger	Column content	Column content	Column content
Warning	Column content	Column content	Column content
Info	Column content	Column content	Column content
Light	Column content	Column content	Column content
Dark	Column content	Column content	Column content