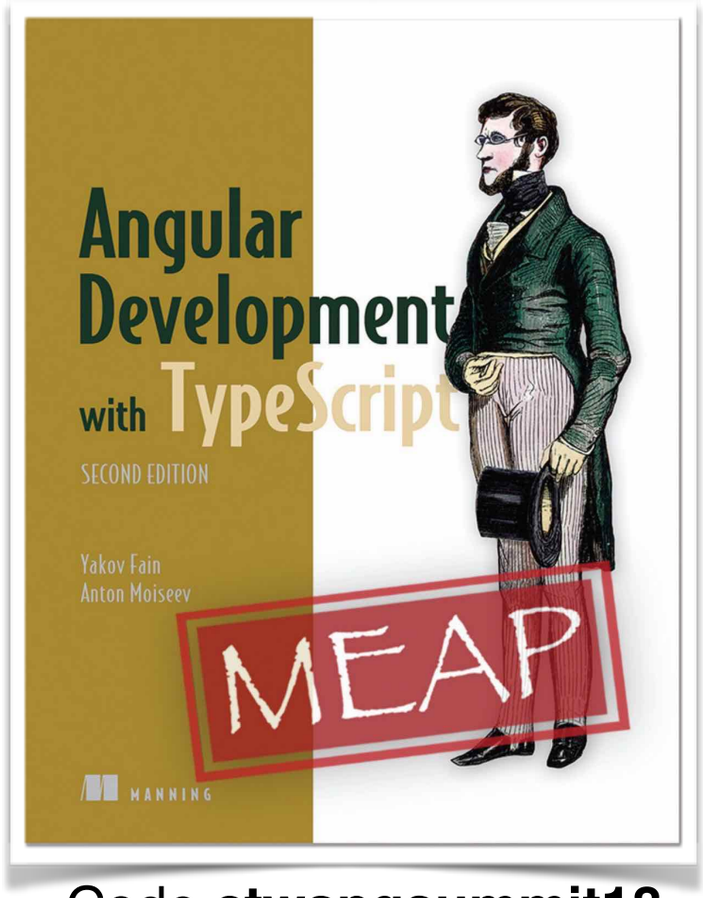
# Create a modern-looking UI with Angular Material and Flex Layout libraries

Yakov Fain, Farata Systems

# About myself

- Work for Farata Systems
- Angular consulting and training
- Java Champion
- Co-authored two editions of the book “Angular Development with TypeScript”
- Working on a book “Get Programming with TypeScript”




Code **ctwangsummit18**
40% off all books
at manning.com

@yfain

# Agenda

- Intro to the Flex Layout library
- Intro to the Angular Material library
- Bringing together Flex Layout and Angular Material

# Responsive Web Design (RWD)

- RWD - UI layout adopts to the device screen width
- CSS Media Queries - specify different styles depending on the screen width
- Read about RWD at http://alistapart.com/article/responsive-web-design

# Implementing RWD in your app

- CSS Flexbox: https://mzl.la/2nSqhlG
- CSS Grid: https://mzl.la/2HrHXgI
- Bootstrap grid system: https://bit.ly/2HEnxly
- Angular Flex Layout: https://github.com/angular/flex-layout

# Flex Layout Library

# Angular Flex Layout library

- UI layout engine for implementing RWD without the need to write CSS media queries

- `ObservableMedia` notifies your app about the current viewport width

- Produces cross-browser compatible CSS

**Docs**: https://github.com/angular/flex-layout/wiki

# Container Directives

- `fxLayout` - use the CSS Flexbox row (default) or column layout for arranging child elements

- `fxLayoutAlign` - aligns child elements in a particular way, e.g. `fxLayoutAlign = "center"`

- `fxLayoutGap` - controls space between child elements so you can create grid-like layouts

**Live demos:** https://bit.ly/2izkPWa

# Child Directives

- `fxFlex` - controls the amount of space that the child takes within the parent container

- `fxFlexAlign` - selectively change the child's alignment within the parent container

- `fxFlexOrder` - change the order of the child element within the parent container

# Flex Layout suffixes

- **xs** - less than 599px width
- **sm** - from 560px to 959px
- **md** - from 960px to 1279px
- **lg** - from 1280px to 1919px
- **xl** - from 1920px to 5000px
- **lt** - less than
- **gt** - greater than

For example, `flexLayout.sm, fxLayout.lt-md`

# Dynamically changing styles

- `fxShow.xs`
- `fxHide.lt-md`
- `ngClass.sm`
- `ngStyle.md`

# Adding Flex Layout library

- npm i @angular/flex-layout
- Add it to your app's module

```
import {FlexLayoutModule} from '@angular/flex-layout';

@NgModule({
  imports: [
    FlexLayoutModule
    ...
  ]
})
export class AppModule {}
```

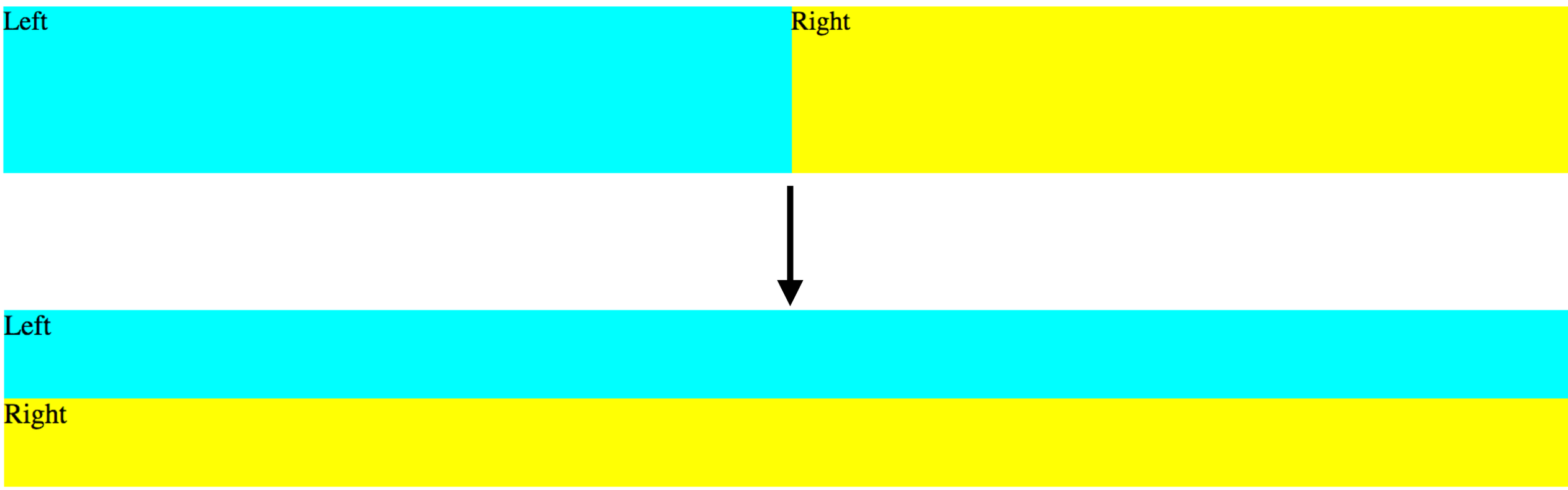
# Using Flex Layout directives

```
@Component({
  selector: 'app-root',
  styles: [`
    .parent { height: 100px; }
    .left   { background-color: cyan; }
    .right  { background-color: yellow; }
  `],
  template: `
    <div fxLayout="row" class="parent" fxLayout.lt-md="column" >
      <div fxFlex class="left">Left</div>
      <div fxFlex class="right">Right</div>
    </div>
  `
})
export class AppComponent {}
```

# Demo

- ng serve --app flex-layout

- Change the window width and the layout may change too




- Allocate 30% and 70% to child divs e.g. `fxFlex="30%"`

# The ObservableMedia service

- Subscribe to `ObservableMedia` to get notifications about screen width changes
- Useful for showing more or less content depending on the screen width
- You can change the order of the UI elements depending on the screen size

# Subscribing to ObservableMedia

```
@Component({
  selector: 'app-root',
  template: `<h3>Watch the breakpoint activation messages in the console.</h3>
  <span *ngIf="showExtras | async">Showing extra info on medium screens</span>
  `
})
export class AppComponent {
  showExtras: Observable<boolean>;

  constructor(private media: ObservableMedia) {

    this.showExtras = this.media.asObservable()
      .pipe(map(mediaChange => {
            console.log(mediaChange.mqAlias);

            return mediaChange.mqAlias==='md'? true:false;
          })
      );
  }
}
```

# Using isActive() API

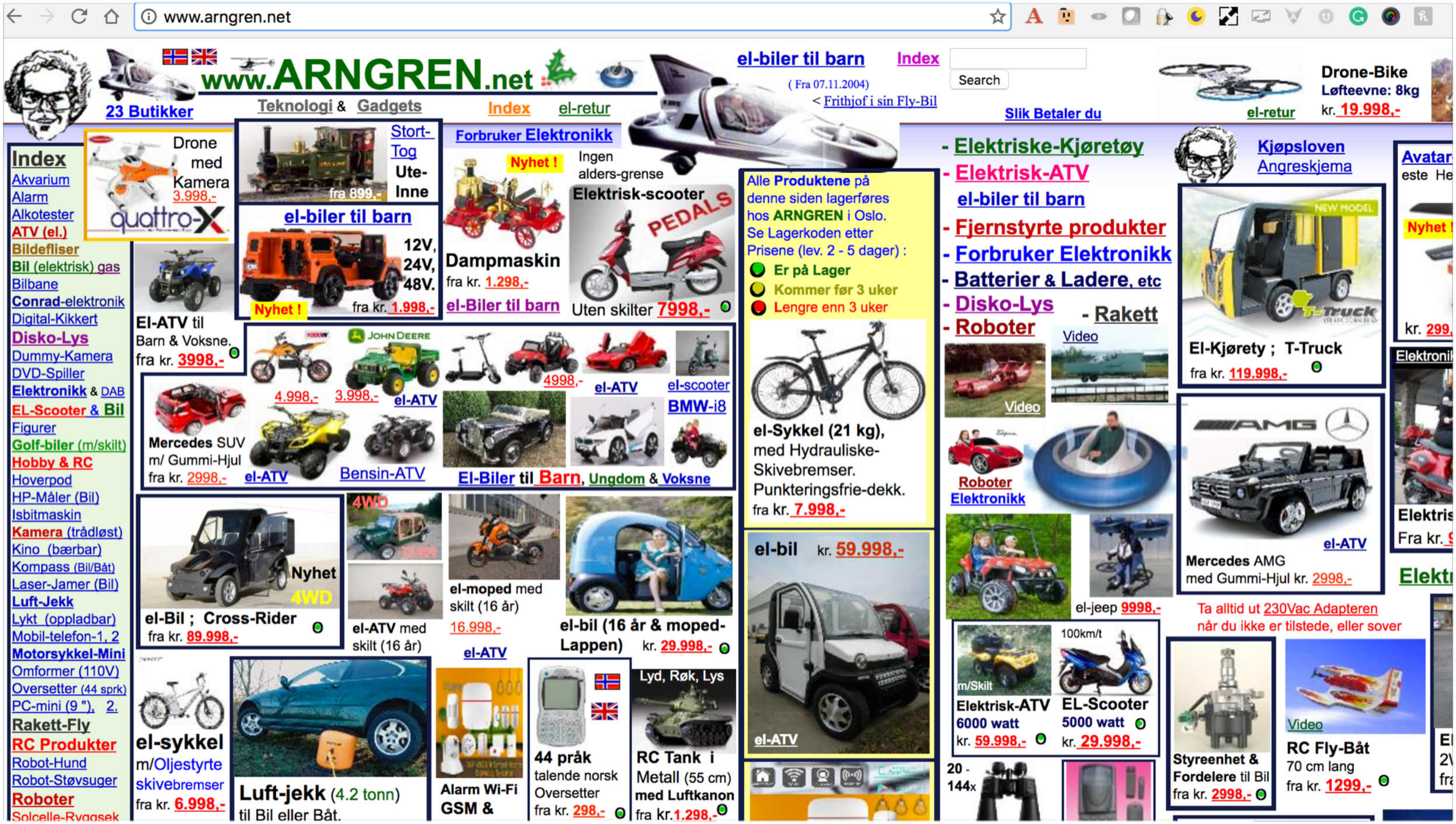
```
import {Component} from '@angular/core';
import {ObservableMedia} from '@angular/flex-layout';

@Component({
  selector: 'app-root',
  template: `<h3>Using the ObservableMedia.isActive() API</h3>
  <span *ngIf="this.media$.isActive('md')">Showing extra info on medium screens
  </span>
  `
})
export class AppComponent {
  constructor(public media$: ObservableMedia) {}
}
```

# Demo

1. ng serve --app observable-media - o
2. Open the browser's console to see the values of `mqAlias`
3. Change the window from to become small (`sm`)
4. The text "Showing extra info on medium screens" disappears
5. Change the `log()` statement to print `mediaChange` to see the applied media query
6. Replace the code with the commented version located at the bottom. It uses the `isActive()` API

# Angular Material Library

# Designed in 1995 in Norway (still alive)



# What’s Material Design?

Material design is visual language (a spec) that defines the classic principles of good design.

https://material.io/guidelines

# Palettes

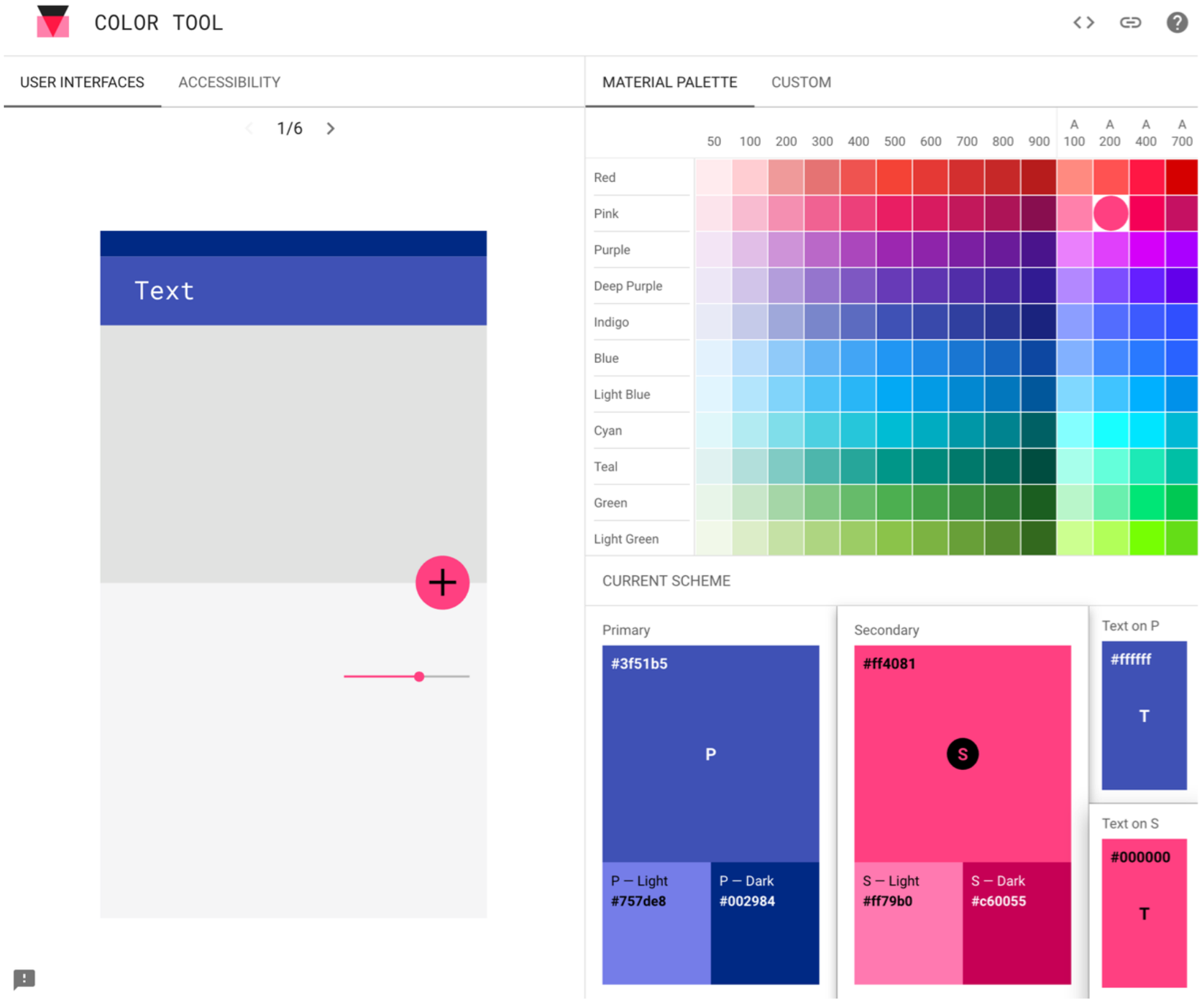
https://material.io/guidelines/style/color.html#color-color-palette

Style – Color

| Light Blue | | Cyan | | Teal | |
|---|---|---|---|---|---|
| 500 | #03A9F4 | 500 | #00BCD4 | 500 | #009688 |
| 50 | #E1F5FE | 50 | #E0F7FA | 50 | #E0F2F1 |
| 100 | #B3E5FC | 100 | #B2EBF2 | 100 | #B2DFDB |
| 200 | #81D4FA | 200 | #80DEEA | 200 | #80CBC4 |
| 300 | #4FC3F7 | 300 | #4DD0E1 | 300 | #4DB6AC |
| 400 | #29B6F6 | 400 | #26C6DA | 400 | #26A69A |
| 500 | #03A9F4 | 500 | #00BCD4 | 500 | #009688 |
| 600 | #039BE5 | 600 | #00ACC1 | 600 | #00897B |
| 700 | #0288D1 | 700 | #0097A7 | 700 | #00796B |
| 800 | #0277BD | 800 | #00838F | 800 | #00695C |
| 900 | #01579B | 900 | #006064 | 900 | #004D40 |
| A100 | #80D8FF | A100 | #84FFFF | A100 | #A7FFEB |
| A200 | #40C4FF | A200 | #18FFFF | A200 | #64FFDA |
| A400 | #00B0FF | A400 | #00E5FF | A400 | #1DE9B6 |
| A700 | #0091EA | A700 | #00B8D4 | A700 | #00BFA5 |

| Green | Light Green | Lime |
|---|---|---|

# Primary and accent colors




https://material.io/color

# Angular Material

- A library of UI components that follow the Material Design spec
- Features: https://github.com/angular/material2#available-features
- Documentation: https://material.angular.io
- Demo app: https://material2-demoapp.firebaseapp.com

# Themes

- A theme is a collection of palettes (colors)
- Pre-built themes are in the dir `@angular/material/prebuilt-themes`
- You can create custom themes

# Palettes

- **Primary** - main colors
- **Accent** - secondary colors
- **Warn** - errors
- **Foreground** - colors for text and icons
- **Background** - component backgrounds

# Adding pre-built themes

- either in HTML:

```
<link href="node_modules/@angular/material/
prebuilt-themes/indigo-pink.css" rel="stylesheet">
```

- or in the global styles.css:

```
@import '../node_modules/@angular/material/
prebuilt-themes/indigo-pink.css';
```

# Adding Angular Material to a CLI project

- Create new Angular CLI project:

  ng new myMaterialProject

- Open it in your IDE
- Install Angular Material:

  npm i @angular/material @angular/cdk

- Add `BrowserAnimationsModule` to `@NgModule imports`
- Add modules for the material components you need to `@NgModule` imports, e.g. `MatProgressBarModule`
- Include one of the pre-built themes to **styles.css**:

```
@import '~@angular/material/prebuilt-themes/deeppurple-amber.css';
```

# Add `MatProgressBarModule`

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { MatProgressBarModule } from '@angular/material'
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    MatProgressBarModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Don't import the entire `MaterialModule` (deprecated), import only what you use, e.g. `MdToolbarModule`

# Sass

- Sass is an alternative to writing CSS that provides additional features:
  - Variables
  - Mixins
  - Modules
  - Nesting
  - Inheritance
- Sass provides two syntaxes - Sass and Scss. Scss is a super set of CSS.
- A Sass preprocessor converts Sass into CSS during the build.
- Angular CLI supports Sass out-of-the-box.

Now CSS also supports variables

# Sass Examples

Declaring a variable:

```
$ngs-brand-font: 'Abril Fatface', cursive;
```

Importing modules:

```
@import './palette';
```

Using mixins:

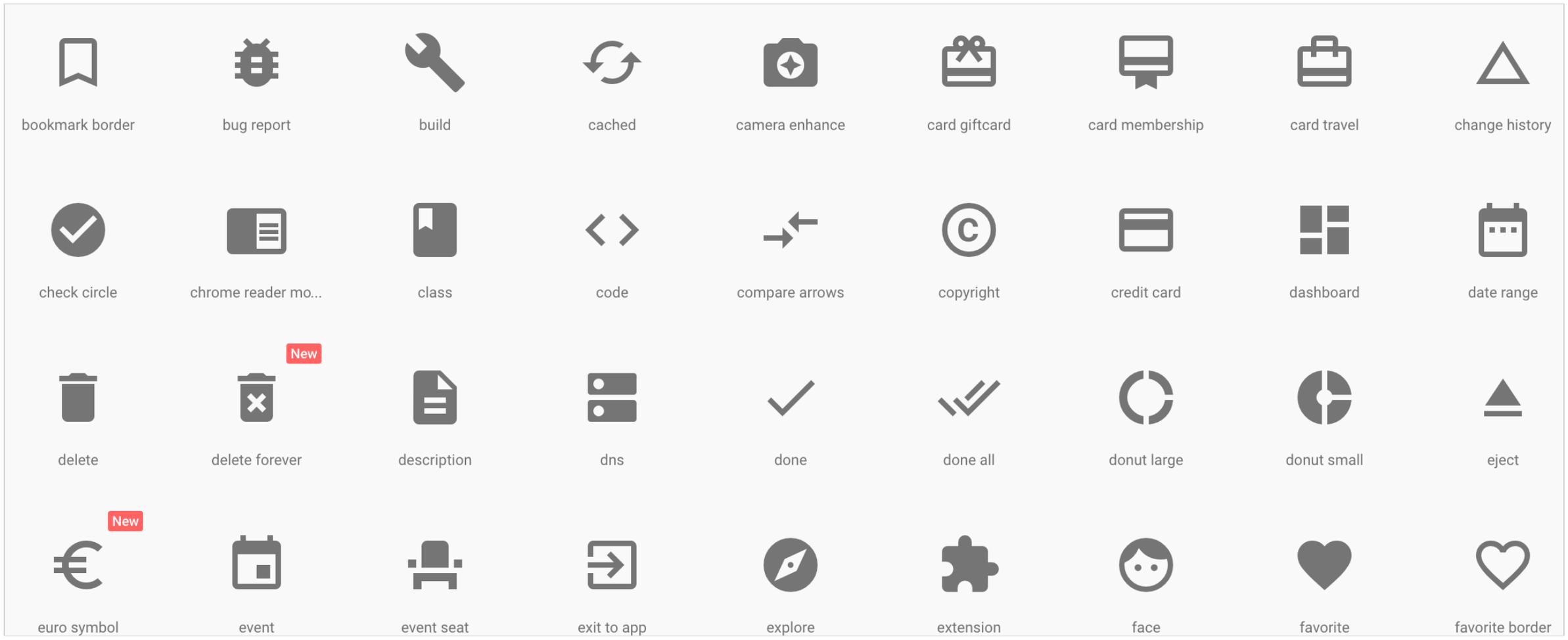
```
@include angular-material-theme($ngs-theme);
```

# Sass nesting example

Nesting the **<ul>** and **<a>** style selectors inside ** **

```
div {
  ul {
    margin: 0;
  }
  a {
    display: block;
  }
}
```

# Material icons




https://material.io/icons

# Adding material icons to your app

To use Material icons with Angular Material components, add the following to your index.html:

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet">
```

```
<button mat-icon-button>
  <mat-icon>menu</mat-icon>
</button>
```

# Using Angular material components

```
<mat-toolbar color="primary">
  <a [routerLink]="['/']">Home</a>
  <div class="fill"></div>

  <button mat-icon-button>

    <mat-icon>more_vert</mat-icon>
  </button>
</mat-toolbar>
```

Component

Directive

The icon name
from material icons

# Demo (start)

- Create a new project:
  ng new hello-material

- Open the new project in your IDE

- npm i @angular/material @angular/cdk

- ng serve -o

- Add the code below to your app.module.ts:

```
import {MatToolbarModule} from ‘@angular/material'
import {BrowserAnimationsModule} from '@angular/platform-browser/animations';
 ...
    imports: [
       MatToolbarModule,
       BrowserAnimationsModule]
```

# Demo (cont)

- Add a theme to style.css:

```
@import '~@angular/material/prebuilt-themes/deeppurple-amber.css';
```

- Replace the content of app.component.html with this:

```
<mat-toolbar color="primary">Hello Material!</mat-toolbar>
```

- The browser renders:


- Change the color to `accent`

- The browser renders:

# Demo (end)

- Add material icons to the <head> section of index.html:

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
```

- Add `MatIconModule` and `MatButtonModule` to `@NgModule` imports

- Add a button to the toolbar in app.component.html:

```
<mat-toolbar color="primary">Hello Material!
  <button mat-icon-button>
    <mat-icon>menu</mat-icon>
  </button>
</mat-toolbar>
```

- The app renders:


- Change the icon from `menu` to `more_vert`

- The app renders:

# Creating custom themes

- A theme is based on the Material Design spec
- Theming guide: https://material.angular.io/guide/theming
- A theme is composed from these palettes:
  primary, accent, warn, foreground, background
- You can define custom colors for palettes

# Use Sass to define a custom theme

- Import the `mat-core()` mixin, which includes common reusable styles
- Define a theme by composing palettes
- A theme object is created with either `mat-light-theme()` or `mat-dark-theme()`

# Defining vars in the _theme.scss partial

```scss
@import '~@angular/material/theming'; // Includes the Material Design palettes

// For each palette, you specify a default, lighter, and darker hue.
$nga-primary: mat-palette($mat-cyan);
$nga-accent:  mat-palette($mat-cyan, A200, A100, A400);

// The warn palette is optional (defaults to red).
$nga-warn:    mat-palette($mat-red);

// Create the theme object (a Sass map containing all of the palettes).
$nga-theme:      mat-light-theme($nga-primary, $nga-accent, $nga-warn);

$nga-background: map-get($nga-theme, background);
$nga-foreground: map-get($nga-theme, foreground);

// Typography is a way to set levels for headings, font size, etc.
// Details here: https://bit.ly/2qWTfm6
$nga-typography: mat-typography-config();
```

# Sample styles.scss

```scss
@import ‘./theme';  // Import vars from partial

@import url('https://fonts.googleapis.com/icon?family=Material+Icons');
@import url('https://fonts.googleapis.com/css?family=Titillium+Web:600');
@import url('https://fonts.googleapis.com/css?family=Abril+Fatface');

@include mat-core();

// Include theme styles for core and each component used in your app.
// Alternatively, you can import and @include the theme mixins for each
component that you are using.
@include angular-material-theme($nga-theme);


// Global styles.

body {
  font-family: mat-font-family($nga-typography);
  line-height: mat-line-height($nga-typography, body-1);
  font-size: mat-font-size($nga-typography, body-1);
  height: 100%;
  margin: 0;
}
```

# Bringing together Flex Layout and Angular Material in a sample ngAuction app

## More items to consider:

# Teapot

Impress your guests with Teapot by Kitchen Stuff. Teapot holds extremely hot liquids and pours them from the spout. Use the handle, shown on the left, so your fingers don't get burnt while pouring.

**$210.00** LAST BID

PLACE BID $215

NgAuction

# Teapot

Impress your guests with Teapot by Kitchen Stuff. Teapot holds extremely hot liquids and pours them from the spout. Use the handle, shown on the left, so your fingers don't get burnt while pouring.

$210.00 LAST BID

PLACE BID $215

## More items to consider:










NgAuction

# Teapot

Impress your guests with Teapot by Kitchen Stuff. Teapot holds extremely hot liquids and pours them from the spout. Use the handle, shown on the left, so your fingers don't get burnt while pouring.

$210.00 LAST BID

PLACE BID $215

## More items to consider:

# Code review of ngAuction

Source code: https://bit.ly/2vMSx0k

# Thank you!

- Sources of the Flex Layout samples:
  https://bit.ly/2Hpd2FY

- Blog: yakovfain.com

- Twitter: @yfain