Building Web Applications

Mendel Rosenblum

Good web applications: Design + Implementation

Some Design Goals:

- Intuitive to use
  - Don't need to take a course or read a user manual
- Accomplish task accurately and rapidly
  - Provide needed information and functionality
- Users like the experience
  - Joy rather than pain when using the app

The hardest part of good web applications is the **design**

Outside the scope of this course (and instructor)!

Good user interface principles are encoded in the toolkits and style guides

Some guiding design principles for Web Apps

- Be consistent

  Cognitive load less for the user

- Provide context

  User shouldn't get lost in the app

- Be fast

  Don't make the user wait

Consistency: Style guides & design templates

- Web apps should have a **style guide -** Covers the look and feel of the app
  - Style - Color schemes, animation, icons, images, typography, writing
  - User interactions - Menu, buttons, pickers, dialog boxes, tables, lists, ...
  - Layout - Structure, toolbars, content, responsiveness
- Patterns - If you do something multiple places do it the same way
  - Aided by reusable implementation components
  - Error handling, navigation, notifications, etc.
- Design templates - Follow a familiar structure
  - Example: Master-detail template

Style Guide Example: Material Design from Google

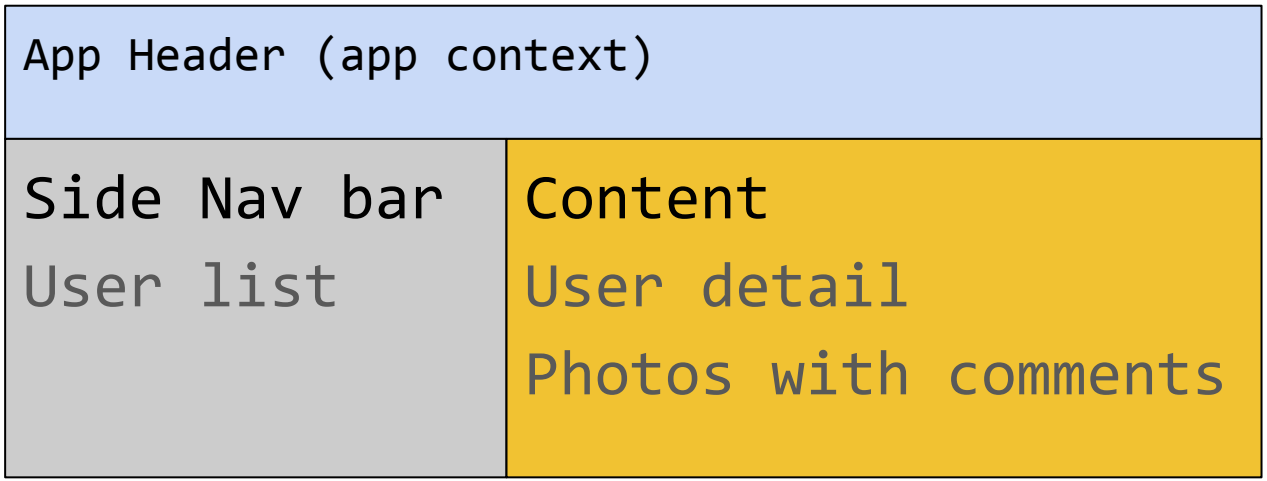
- Used in Google apps (e.g Android, web apps)
  - Influence by publishing (paper and ink) enhance with technology (3D look)
  - Focus on traditional print issues: grids, space, typography, scale, color, imagery
  - Heavy use of animation to convey action
- Dictates many aspect of design
  - Structure and layouts
  - User interface
  - Common patterns

Front-end web frameworks

- Popular example: Bootstrap
  - CSS style sheets
    - Design templates
    - Grid layout system with responsive support (breakpoints, etc.)
    - Element styling
  - HTML components
    - Buttons, menus, toolbars, lists, table, forms, etc.
  - JavaScript
    - Modals, transitions, dropdowns, etc.
    - Originally jquery based
- Angular Material
  - CSS style sheets and Angular directives for implementing Material design spec

Example: Use Angular Material for a Photo App

- Use an Master-Detail template layout
  - Users with Photos with Comments
- Classic layout:



Use grid to layout app

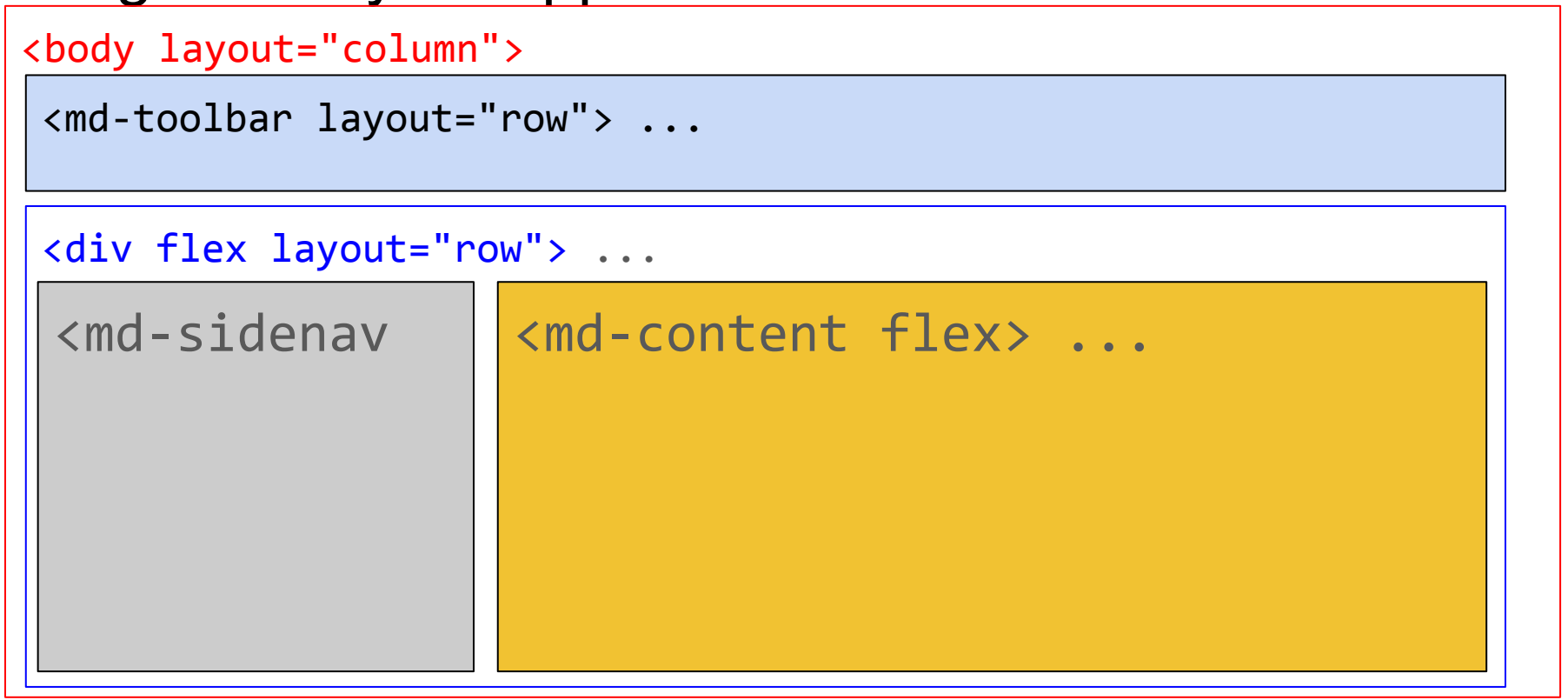
```
<body layout="column">
   <md-toolbar layout="row">
       ...
  </md-toolbar>
  <div flex layout="row">
       <md-sidenav>
            ...
       </md-sidenav>
       <md-content flex>
          ...
      </md-content>
   </div>
</body>
```

```
<!-- Body is a single column with 2 rows
   <!-- Row #1 is the header


   <!-- Row #2 has two columns
      <!-- Column #1 is the side nav bar


   <!-- Column #2 is the content area (flex)
```

Use grid to layout app



Deep linking support

To support bookmarking and sharing we can use `ngRoute` to load the views

The `md-content` contents can be the `ngRoute`'s `ng-view` directive

```
<md-content flex>
    <div ng-view></div>
</md-content>
```

The `md-sidenav` component can just use links to view

```
<a ng-href="#!/photos/...
```

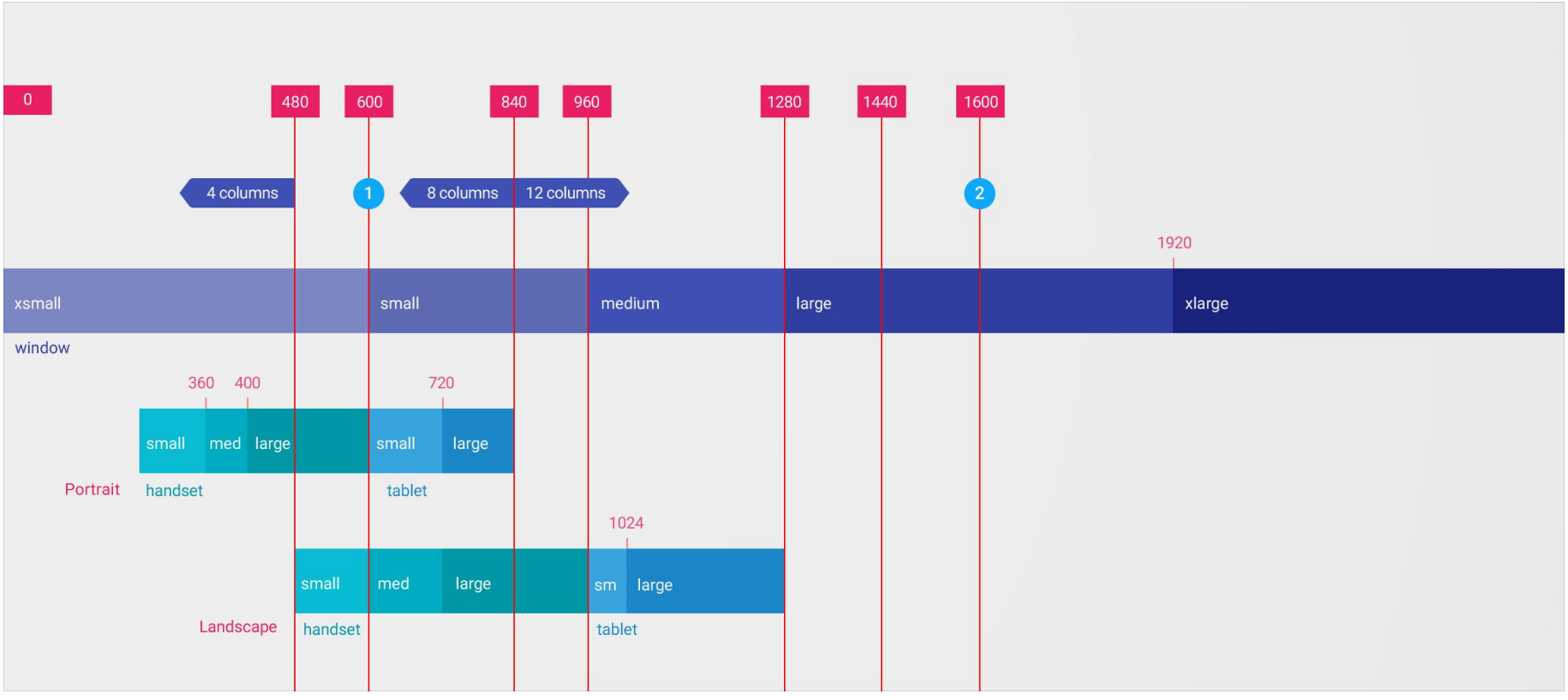
Responsive Design support

- Uses CSS flexbox - Relative sizing handles changes (flex attribute)

  `<md-content flex> ...` -- Smaller widths will have smaller content area

- Use CSS breakpoints to handle big differences

Breakpoint sizes: xs, sm, md, lg, xl



Angular Material Responsive Support

- Conditional HTML support with `hide/show`

```
<md-button hide-gt-sm
<md-menu show-lg
```

- Query from JavaScript with `$mdMedia`

```
<md-sidenav md-is-locked-open="$mdMedia('gt-sm')"
```

- Override `layout` and `flex` attributes

```
<div layout="row" layout-sm="column"  ...
<div flex="50" flex-gt-lg="75" ...
```

Photo App on Mobile

- Make the sidenav start closed on small devices

```
<md-sidenav md-is-locked-open="$mdMedia('gt-sm')"
            md-component-id="users"
            ng-click="toggleUserList()" >
</md-sidenav>
```

- Make a button in the toolbar for opening the nav bar

```
<md-button hide-gt-sm ng-click="toggleUserList()" >
    <md-icon md-svg-icon="menu" ></md-icon>
</md-button>
```

```
toggleUserList = function() { $mdSidenav("users").toggle(); }
```

Accessible Rich Internet Applications (ARIA)

- Add text descriptions for things that need it

```
<a aria-label="Photo of user {{user.name}}" ng-href=...
```

```
<img aria-label="{{photo.description}}"
```

- Need to add it to `md-button`, etc.

Internationalization (I18N)

- Users want different: text, dates, numbers, currencies, and graphics
- Ultimately need a level of indirection. Consider: `<h1>Getting Started</h1>`

  Could use:

```
<h1>{{i18n.GettingStarted}}</h1>
<h1 translate>Getting Started</h1>
<h1>{{"Getting Started" | translate}}</h1>
```

- Not applied to user generated content

```
<h1>Hello {{person.firstName}}</h1>
```

Testing the web app

- Unit testing
  - Each test targets a particular component and verifies it does what it claims it does
  - Requires mock components for the pieces that component interacts with
  - Example: Load an angular component (controller, directive, etc.) and run tests against it
    - Need to mock everything these touch (DOM, angular services, etc.)
- End-to-End (e2e) testing
  - Run tests against the real web application
  - Scripting interface into browser used to drive web application
  - Example: Fire up app in a browser and programmatically interact with it.
    - WebDriver interface in browsers useful for this
- Metric: Test Coverage
  - Does every line of code have a test?

Much useful functionality available for our app

md-menu-bar, md-menu, md-tabs, md-dialog, md-select

md-radio-button, md-checkbox, md-button

md-autocomplete

md-tooltip

md-datepicker

$mdToast, $mdBottomSheet