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, MaterializeCSS
  - CSS style sheets and Angular directives for implementing Material design spec
Example: Use Material Design for a Photo App

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

- Classic layout:

<table>
<thead>
<tr>
<th>App Header (app context)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Side Nav bar</td>
</tr>
<tr>
<td>User list</td>
</tr>
<tr>
<td>Content</td>
</tr>
<tr>
<td>User detail</td>
</tr>
<tr>
<td>Photos with comments</td>
</tr>
</tbody>
</table>
Angular Material - Use grid to layout app

```html
<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)
Materialize CSS w/React - Use grid to layout app

<TopBar /> <!-- Top bar across the top
<div className="row"> <!-- Row with List & either Detail or Photo
   <UserList />  <!-- className="col s4 z-depth-2" - 4 col wide
   <div className="center-align">
      <UserDetail ... /> <!-- className="col s7" - 7 columns wide
      or
      <UserPhotos ... /> <!-- className="col s7" - 7 columns wide
   </div>
</div>
Use grid to layout app

```html
<body layout="column">
    <md-toolbar layout="row"> or <TopBar /> ...
</body>
```

```html
<div flex layout="row"> or class="row" ...
    <md-sidenav
        or
        class="col s4"
    </md-sidenav>
    <md-content flex> ...
    Or
    class="col s7"
</div>
```
Deep linking support - AngularJS ngRoute

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:

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

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

```html
<a ng-href="#!/photos/...`
```
Deep linking support - React Route

To support bookmarking and sharing we can use React Route to load the views.

The content div can be the React Route Switch

```jsx
<Switch>
  <Route path="/users/:userId" component={UserDetail} />
  <Route path="/photos/:userId" component={UserPhotos} />
  <Route path="/users/" component={UserList} />
</Switch>
```

The UserList sidebar can just use links to view

```jsx
<Link to="/photos/57231f1a30e4351f4e9f4bd8">
  Photos of User Ellen Ripley
</Link>
```
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
Material Responsive Support

- Conditional HTML support with `hide/show`
  `<md-button hide-gt-sm` or `class="hide-on-med-and-up"`
  `<md-menu show-lg` or `class="show-on-large"`

- 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
  ```html
  <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
  ```html
  <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:
  ```html
  <h1>{{i18n.GettingStarted}}</h1>
  <h1 translate>Getting Started</h1>
  <h1>{{"Getting Started" | translate}}</h1>
  ```
- Not applied to user generated content
  ```html
  <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