Reasons for compiling this document

1. Prevent miscommunication about the intended outcome of the refactor.
2. Allow developers to give feedback on whether this is an improvement.
3. Identify any cases that may have been missed in the oral discussion.

Goals

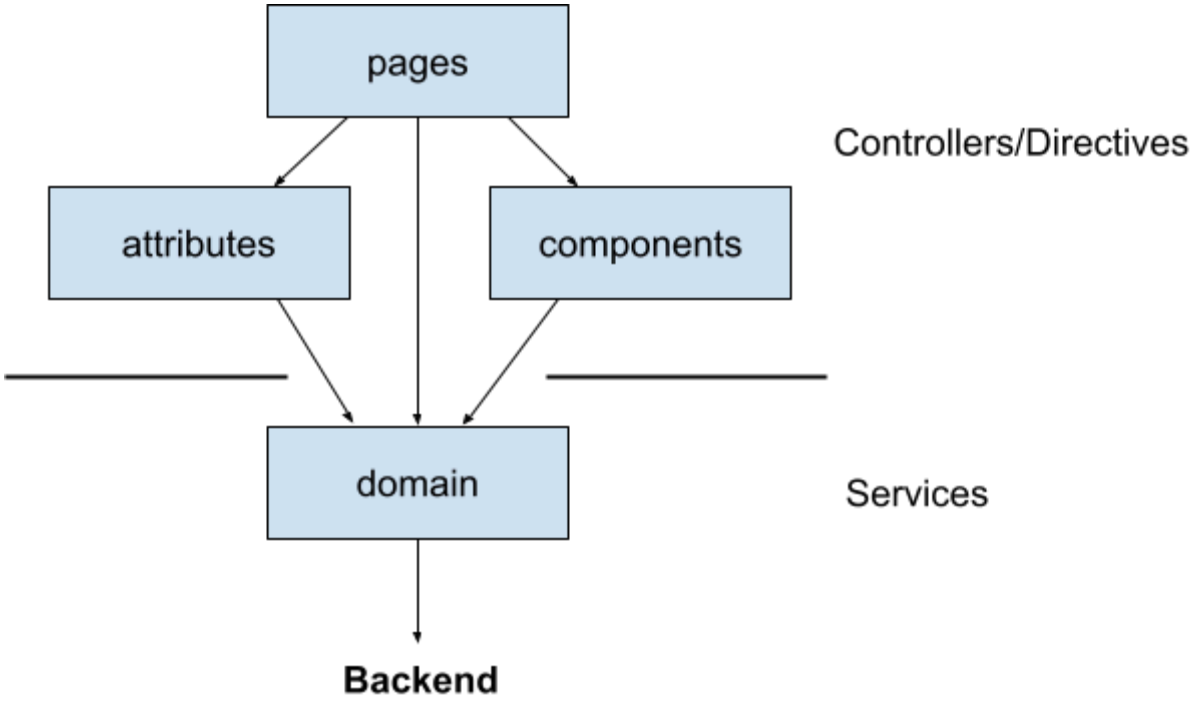
1. To enhance the readability and discoverability of the frontend.
2. To ease the creation of new tests.
3. Improve test coverage significantly.
4. Reduce tight couplings throughout the code base.
5. Follow good engineering principles by ensuring files have single responsibility, deterministic naming, and deterministic locating.

Guidelines

1. JavaScript file names use CamelCase.
2. Directory names and HTML file names use under_scores.
3. Shared services go in the lowest common ancestor (LCA) of the files which depend on them -- unless a minority of files in the LCA's subtree depend on them, in which case they should go into 'common' instead.
4. Controllers and directives should be thin. While doing the refactor, it is fine to break existing controllers into additional services/directives and add these to the relevant folders.
5. Any services which communicate to backend controllers using $http must end with ‘BackendApiService’ and only those services may correspond to backend controllers.
6. Directive names in Angular should not end in “Directive” despite their corresponding filenames ending in directive. For instance, ‘CollectionNodeListDirective’ would have its directive called ‘collectionNodeList’.
7. Services may go in pages/ if their only purpose is to maintain state that’s specific to a particular page. However, any services that are used by more than one page should go in domain/. Services may not go in components/.

Definitions of top-level folders

- attributes: These are reusable HTML attributes used in directives and controllers.
- components: These are reusable directives (HTML) tags used in other directives and controllers.
- domain: This represents all of the data structures and services which represent the frontend functionality.
- pages: This is a structure which reflects the actual frontend structure and layout that users of Oppia see and use (e.g. the gallery page, exploration editor, etc.)



Migration plan

1. Prerequisites: bring in #1171 (widget-answer-views-rebased).
2. Pull individual components/services into their individual files.
3. Move all files to the new structure. (For this and the previous stage, stage, don't bother about internal refactoring, etc. The aim of getting this step done as quickly as possible is to enable different people to do the refactoring/tests for each component in parallel.)
4. In parallel (for each component): add tests, and then make the component nice. Tests should be written prior to the refactor.

Moving files to the new structure

1. Move the stuff that's currently in pages/ one level down.
2. Create a new pages/ folder and move lots of stuff into it.
3. Do the rest of the refactor piecemeal.

Structure of the contents of core/templates/dev/head:

- components // Shared directives go here
  - alerts
    - AlertMessageDirective.js
  - attribution_guide
    - attribution_guide_directive.html
    - AttributionGuideDirective.js
  - create_button
    - CreateActivityButtonDirective.js
    - create_activity_button_directive.html
  - embed_modal

        - embed_exploration_modal_directive.html
        - ExplorationEmbedButtonService.js
    - gadget
        - GadgetDirective.js
        - gadget_directive.html
        - GadgetPanelDirective.js
        - gadget_panel_directive.html
    - share
        - SharingLinksDirective.js
        - sharing_links_directive.html
    - forms
        - HtmlSelectDirective.js
        - html_select_directive.html
        - image_uploader_directive.html
        - ImageUploaderDirective.js
        - ObjectEditorDirective.js
        - Select2DropdownDirective.js
        - SchemaFormBuilder.js
            - Split this into different JS files for constituent services and directives. All filters can be placed inside a FormsFilter.js file.
        - schema_form_builder.html
            - Ditto (compared to SchemaFormBuilder.js).
    - loading
        - LoadingDotsDirective.js
        - loading_dots_directive.html
    - side_navigation_bar
        - side_navigation_bar_directive.html
        - SideNavigationBarDirective.js
    - summary_tile
        - CircularImageDirective.js
        - circular_image_directive.html
        - CollectionSummaryTileDirective.js
        - collection_summary_tile_directive.html
        - ExplorationSummaryTileDirective.js
        - exploration_summary_tile_directive.html
    - profile_link
        - profile_link_image_directive.html
        - ProfileLinkImageDirective.js
        - profile_link_text_directive.html
        - ProfileLinkTextDirective.js
    - ratings
        - RatingDisplayDirective.js
        - rating_display_directive.html
        - RatingFromFrequenciesDirective.js
        - rating_from_frequencies_directive.html
        - RatingFromValueDirective.js
        - rating_from_value_directive.html

        - RatingVisibilityService.js
        - rating_visibility_service.html
        - RatingComputationService.js
- domain
    - collection
        - CollectionBackendApiService.js
        - CollectionNodeObjectFactory.js
        - CollectionObjectFactory.js
        - CollectionPlaythroughObjectFactory.js
        - CollectionRightsBackendApiService.js
        - CollectionUpdateService.js
        - CollectionValidationService.js
        - SkillListObjectFactory.js
        - WritableCollectionBackendApiService.js
    - dashboard
        - DashboardBackendApiService.js
    - editor
        - undo_redo
            - ChangeObjectFactory.js
            - UndoRedoService.js
                - Maintains a stack of undo/redo actions
                - Actions registered by name as functions (do and undo functions)
                - Each function is associated with angular.copy'd data
                    - This data is a commit-change JSON object
                - The current 'change list' for the exploration can be reconstructed by looking at the current action stack stored within the undo-redo service
                - 'Do' function passed into an 'action call' of the change stack service is invoked immediately; 'undo' is invoked at a later time based on a primary 'undo' function as part of the change stack service
                - Functionality to persist the change stack service
            - MementoRepositoryService.js (maybe)
                - Handles storing separate instances of a domain object property and bind to them, such that one variable represents the current value of the property and the second variable represents the transient/proposed state of the property (this one is changed by the calling code)
                - The service stores multiple bindings simultaneously
        - access
            - EditabilityService.js
        - navigation
            - EditorContextService.js

            - TabRouterService.js
                - Binds incoming URLs to tabs
            - TabManagerService.js
                - Binds a name to setup/teardown functions for initializing and deinitializing tabs of the editor
                - Client-provided data may be passed during a switch-state
                - Example to bind a tab: tabManagerService.bind('EditorTab', setupEditor, teardownEditor); // All bindings happen in ExplorationEditor.js, CollectionEditor.js, etc. on init of the page
                - Example to switch to a tab: tabManagerService.switchTo('EditorTab', clientVariable);
    - exploration
        - ExplorationObjectFactory.js
            - This wraps the data obtained from the backend in a JS object so that it can be operated on by services in the frontend.
            - [Analogous to exp_domain.py]
        - StateObjectFactory.js
        - ExplorationBackendApiService.js
            - Connects to backend controllers for serving/storing explorations. We might actually need multiple of these for each backend controller corresponding to an exploration.
            - Has methods like: fetch(), load(), which return items from the backend (converted first to ExplorationObjects using ExplorationObjectFactory.js).
        - ExplorationGraphService.js
        - ExplorationSnapshotBackendApiService.js
        - ExplorationStatisticsBackendApiService.js
        - ExplorationWarningService.js
        - ExplorationDataService.js
            - Maintains the single instance of the exploration object loaded for the entire exploration editor
            - This data is retrieved from backend API services located in the domain folder
            - Used by both learner and editor views
            - Has methods like: get(), which returns a copy of the object currently stored by this service in the frontend.
        - ExplorationRightsObjectFactory.js
        - ExplorationRightsBackendApiService.js
        - ExplorationRightsDataService.js
    - summary

        - ExplorationSummaryBackendApiService.js
            - Gets exp summaries given search query
            - Gets exp summaries by list of exp ids
            - Gets exp summaries by a user id ('my explorations')
        - ExplorationSummaryObjectFactory.js
        - ExplorationSummariesDataService.js
    - utilities
        - StopwatchObjectFactory.js
        - UrlInterpolationService.js
        - IframeEventMessengerService.js
        - DateTimeFormatService.js
        - DebouncerService.js
        - DeviceInfoService.js
        - ExtensionTagAssemblerService.js
        - FocusService.js
        - HtmlStringService.js
        - StringFilters.js // This has multiple filters in it.
        - ValidationService.js
        - WindowDimensionsService.js
        - WarningsService.js
    - history
        - HistoryDataService.js
        - VersionsTreeService.js
        - CompareVersionsService.js
    - parameter
        - ExpressionInterpolationService.js
        - ExpressionEvaluatorService.js
        - ExpressionParserService.js
    - suggestion
        - ...
    - feedback_thread
        - FeedbackThreadObjectFactory.js
        - FeedbackThreadBackendApiService.js
        - FeedbackThreadsDataService.js
            - ThreadStatusDisplayService goes into this; remove actual styling from it (conversion from status to style happens in feedback tab controller)
- pages (one folder per base URL)
    - base.html
    - footer.html
    - footer_js_libs.html
    - header_css_libs.html
    - header_js_libs.html
    - oppia.css
        - Split into constituent CSS files and place alongside HTML files which use them.
    - about
        - about.html

        - About.js
    - admin
        - Admin.js
        - admin.html
    - collection_editor
        - CollectionEditor.js
        - collection_editor.html
        - CollectionEditorNavbarBreadcrumbDirective.js
        - collection_editor_navbar_breadcrumb_directive.html
        - CollectionEditorNavbarDirective.js
        - Collection_editor_navbar_directive.html
        - CollectionEditorStateService.js
        - collection_editor_pre_publish_modal_directive.html
        - collection_editor_save_modal_directive.html
        - editor_tab
            - CollectionEditorTabDirective.js
            - collection_editor_tab_directive.html
            - CollectionNodeCreatorDirective.js
            - collection_node_creator_directive.html
            - CollectionNodeEditorDirective.js
            - collection_node_editor_directive.html
            - CollectionSkillListDirective.js
            - collection_skill_list_directive.html
            - CollectionLinearizerService.js
        - history_tab
            - CollectionHistoryTabDirective.js
            - Collection_history_tab_directive.html
        - settings_tab
            - CollectionDetailsEditorDirective.js
            - collection_details_editor_directive.html
            - CollectionSettingsTabDirective.js
            - collection_settings_tab_directive.html
        - statistics_tab
            - CollectionStatisticsTabDirective.js
            - collection_statistics_tab_directive.html
    - collection_player
        - collection_player.html
        - CollectionPlayer.js
        - Collection_node_list_directive.html
        - CollectionNodeListDirective.js
    - contact
        - contact.html
    - **dashboard**
        - Dashboard.js
        - dashboard.html
        - create_activity_modal_directive.html
        - upload_activity_modal_directive.html
        - **CollectionCreationService.js**

        - **ExplorationCreationService.js**
    - error
        - disabled_exploration.html
        - Error.js
        - error.html
    - exploration_editor
        - exploration_editor.html
        - ExplorationEditor.js
        - save_exploration_modal_directive.html
        - SaveExplorationModalDirective.js
        - publish_exploration_modal_directive.html
        - PublishExplorationModalDirective.js
        - **CodemirrorMergeviewDirective.js**
        - help_modal_directive.html
        - post_publish_modal_directive.html
        - welcome_modal_directive.html
        - ParamChangesEditorDirective.js
        - Param_changes_editor_directive.html
        - ValueGeneratorEditorDirective.js
        - state_diff_modal_directive.html
        - editor_tab
            - editor_tab.html
            - EditorTab.js
            - DeleteStateModalDirective.js
            - delete_state_modal_directive.html
            - StateNameEditorDirective.js
            - state_name_editor_directive.html
            - StateParameterChangeDirective.js
            - state_parameter_change_directive.html
            - StateEditorDirective.js
            - state_editor_directive.html
            - StateInteractionEditorDirective.js
            - state_interaction_editor_directive.html
            - StateStatisticsDirective.js
            - state_statistics_directive.html
            - **StateGraphLayoutService.js**
            - StateGraphVisualizationDirective.js
            - state_graph_visualization_directive.html
            - state_responses_editor
                - StateResponsesEditor.js
                - state_responses_editor.html
                - **AnswerGroupEditorDirective.js**
                - **answer_group_editor_directive.html**
                - **ClassifierPanelDirective.js**
                - **classifier_panel_directive.html**
                - **FallbackEditorDirective.js**
                - **fallback_editor_directive.html**
                - **OutcomeDestinationEditorDirective.js**

        - **outcome_destination_editor_directive.html**
        - **OutcomeEditorDirective.js**
        - **outcome_editor_directive.html**
        - **OutcomeFeedbackEditorDirective.js**
        - **outcome_feedback_editor_directive.html**
        - **ResponseHeaderDirective.js**
        - **response_header_directive.html**
        - **RuleEditorDirective.js**
        - **rule_editor_directive.html**
        - **RuleTypeSelectorDirective.js**
        - rule_type_selector_directive.html
    - feedback_tab
        - FeedbackTab.js
        - Feedback_tab.html
        - ThreadTableDirective.js
        - thread_table_directive.html
        - CreateThreadModalDirective.js
        - create_thread_modal_directive.html
    - history_tab
        - HistoryTab.js
        - history_tab.html
        - RevertModalDirective.js
        - revert_modal_directive.html
        - **VersionDiffVisualizationDirective.js**
        - **version_diff_visualization_directive.html**
    - preview_tab
        - preview_tab.html
        - PreviewTab.js
        - preview_set_parameters_modal_directive.html
    - settings_tab
        - settings_tab.html
        - SettingsTab.js
        - make_exploration_community_owned_directive.html
        - MakeExplorationCommunityOwnedDirective.js
        - nominate_exploration_modal_directive.html
        - NominateExplorationModalDirective.js
    - statistics_tab
        - statistics_tab.html
        - StatisticsTab.js
        - state_statistics_modal_directive.html
        - StateStatisticsModalDirective.js
        - BarChartDirective.js
- exploration_player
    - exploration_player.html
    - ExplorationPlayer.js
    - learner_local_nav.html
    - LearnerLocalNav.js
    - AnswerClassificationService.js

        - LearnerParamsService.js
        - StateTransitionService.js
        - StopwatchProviderService.js
        - answer_feedback_pair_directive.html
        - AnswerFeedbackPairDirective.js
        - exploration_skin_directive.html
        - ExplorationSkinDirective.js
        - progress_dots_directive.html
        - ProgressDotsDirective.js
    - forum
        - forum.html
    - library
        - Library.js
        - library.html
        - LibraryFooter.js
        - search_bar_directive.html
        - SearchBarDirective.js
        - search_results_directive.html
        - SearchResultsDirective.js
        - ActivityTilesInfinityGridDirective.js
        - activity_tiles_infinity_grid_directive.html
    - moderator
        - moderator.html
        - Moderator.js
    - notifications_dashboard
        - notifications_dashboard.html
        - NotificationsDashboard.js
    - preferences
        - preferences.html
        - Preferences.js
    - privacy
        - privacy.html
    - profile
        - profile.html
        - Profile.js
    - signup
        - signup.html
        - Signup.js
    - splash
        - Splash.js
        - splash.html
    - teach
        - teach.html
        - Teach.js
    - terms
        - Terms.html
    - thanks
        - thanks.html

        - Thanks.js
- attributes
    - AngularHtmlBindDirective.js
    - CustomPopoverDirective.js
    - FocusOnDirective.js
    - MathjaxBindDirective.js
    - SelectOnClickDirective.js
- Oppia.js // Contains all initializations/setup, such
            // as $interpolateProvider and Angular configs.

Notes

- The above structure does not include any files in extensions/.
- Add a test to ensure that there are no collisions in filenames within a single page (or in 'common') -- i.e. no two files anywhere in the subtree share the same name. Maybe this test could enforce additional aspects of the naming scheme, too.
- Learner view "play state" should be kept combined and isolated so that it may be persisted for later use. Users should be able to pause their progress in an exploration and continue where they left off later. This should be filed as a separate issue from the refactor and also needs to have some investigation as to how this will be affected by collections.
- Constants and animations may go in any JS files where they are reasonable.
- Extract controller and service bodies from their respective controller and factory calls. This is similar to the following style guide principles:
    - https://github.com/johnpapa/angular-styleguide#style-y032
    - https://github.com/johnpapa/angular-styleguide#style-y052
- Large/expensive logic is allowed to go into a directive if the logic is related to the *link* or *compile* aspects of a directive, such as in #1329.
- Whether a directive should use aThe 'GLOBALS' pattern will be replaced with a series of Angular constant creation calls instead.
    - Ex: oppia.constant('ALLOWED_GADGETS', JSON.parse('{{ALLOWED_GADGETS|js_string}}'));
- explorationContextService.js, UrlService.js
    - Replace with globals served from the backend controllers which specify whether it's in iframe, the page context, and the exploration ID.
- Remove all duplicate URLs and have feconf.py (or some other, better file) be the single source of truth for all URLs (these URLs are passed to the frontend and specified as Angular constants; then they are used in conjunction with the UrlInterpolationService)
- Refactor state graph directive.
- Rename /explore to /exploration & redirect
- Rename /create to /exploration_editor & redirect
- Rules for the linter wherein it crawls through entire frontend and ensures following:

    - For every frontend filename which ends in '_directive.html' have a corresponding '...Directive.js' file with the converted camelcase prefix
    - All HTML filenames are underscore and lowercase.
    - All JS filenames are UpperCamelCase.
    - All JS filenames end in either Service, Directive, Filters, unless they correspond to a controller.
    - Each folder and subfolder in pages/ has one HTML and related JS file named after the folder (e.g. exploration_editor -> exploration_editor.html and ExplorationEditor.js)
    - Ensure throughout entire frontend there are no duplicate file names
- Refactor existing developer documentation and create high-level concept diagrams to help explain the architecture and emphasize key services or components which contribute to some vertical slice of Oppia (such as interactions, rules, the exploration editor, etc.)

- controller, link, or compile construct should follow the standards defined by Angular, which is explained well here: http://stackoverflow.com/questions/12546945.

Error directive:

```
<error-messages value="data" validator="function"
has-error="isValidForm"></error-messages>
```

Template:

```
<span ng-if="!errorMessage"><[errorMessage]></span>
```

JS:

```
for (var i = 0; i < $scope.value.length; i++) {
  $scope.$watch($scope.value[i], …
```

The error directive sends the error message to an underlying ErrorService to help with form save buttons as well as grabbing the error message in order to display it.

Todos