About You

Why are you interested in working with Oppia, and on your chosen project?
I always wanted to work on a real-world project. I came to know about Oppia from my peers in my college as they have been past contributors to Oppia. First of all, I was fascinated by the mission of Oppia that is "*provide high-quality education to those who lack access to it*". The second thing that encourages me to contribute to Oppia is helpful and friendly mentors. Also the awesome work culture like regular meets, organized tasks make it different from any other open-source organization.

Project Chosen - Solve all typescript and webpack issues in the codebase

I learned about TypeScript in detail by contributing to Oppia for the past 5 months. And now TypeScript has become one of my favorite programming languages. One of the best things about Oppia's codebase is that it is completely written in TypeScript. Also as TypeScript is a strongly typed language it helps in earlier detection of errors which speeds up development. But currently there are some issues with the codebase like in some places many variables are defined as any which hinders the earlier detection of errors. So I want to make the Oppia Codebase error free and strongly typed which would also encourage developers to contribute to the project. Also this project will allow me to learn more about the language and webpack.

Prior experience
I have been coding in Python, TypeScript for the past 1.5 years. I have worked with various frontend and backend frameworks, you can see my portfolio here. I have also worked with various other open source projects that can be seen in my github profile. I also have experience in developing web apps completely from scratch using frontend frameworks like Angular, Vue.Js etc. and developing REST API's using backend frameworks like Django, DRF etc. I am also experienced in problem solving and data structures.

Apart from that I have been working in Oppia for the past 5 months and have become quite familiar with the codebase.

Some of my PR's are

- Migrate ClassroomBackendApiService to angular
- Fix part of #6351: Add custom typing for PencilCodeEmbed
- Fix #8126: Removed the error message for intermediate fraction expressions
- Fix part of #6240: Add e2e tests for testing navigation buttons in an exploration
- Migrate ExplorationFeaturesBackendApiService to angular and add tests

My complete list of PR's can be seen here.

I have also reported some issues like

- Irrelevant error message on rejecting or accepting the suggestion
- The voiceover error message doesn't get displayed in every tab
- Input box overflow while adding question to a skill

My complete list of issues can be seen here.

Contact info and timezone(s)

Email: mittalnishant14@outlook.com (Email)
mittalnishant14@gmail.com (Hangouts)

Phone: +91 9417907862

Timezone: Indian Standard Time (IST)

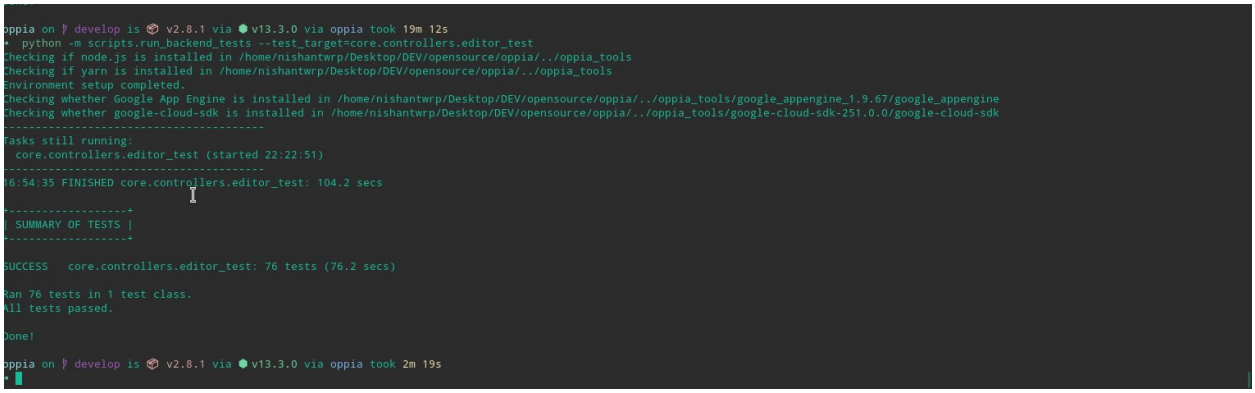
Time commitment

I plan to spend 45-50 hrs or more on the project per week.

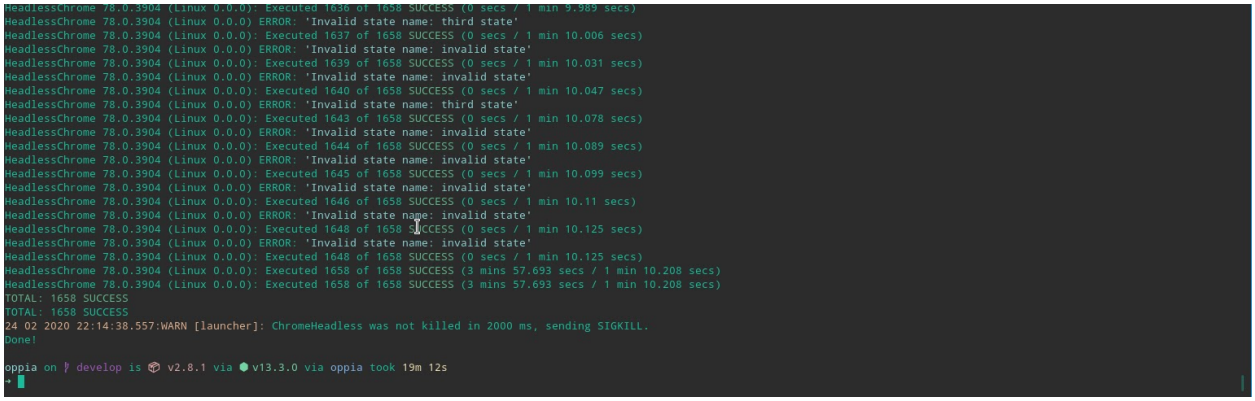
Essential Prerequisites

Answer the following questions:

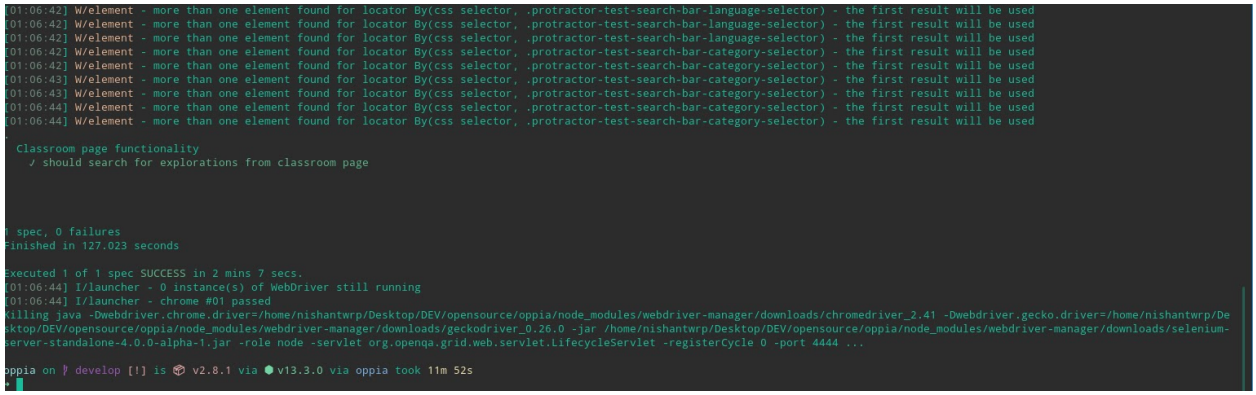
- *I am able to run a single backend test target on my machine. (Show a screenshot of a successful test.)*



- *I am able to run all the frontend tests at once on my machine. (Show a screenshot of a successful test.)*



- *I am able to run one suite of e2e tests on my machine. (Show a screenshot of a successful test.)*



Other summer obligations

I have no commitments in the summer.

Communication channels

I am planning to communicate through emails or hangouts or whatever means of communication is preferred by the mentors on a weekly or daily basis to discuss the progress of the project.

Project Details

Product Design

There are 5 mini projects in this project

- Remove all custom typings for typescript and any types (clubbed two parts in the idea's page)
- Remove all "any" types from the codebase
- Replace third_party and remaining <script> imports with webpack
- Reduce the overall time for webpack compilation
- Add documentation on all typescript and webpack errors and solutions on how to fix them.

I also would like to include a small mini project in my proposal

- Adding typescript checks to the pre_push_hooks.

When the project is complete

- The third party libraries in the typings directory would have proper interfaces defined for them.
- The keys of the response of all the api endpoints would be made camel cased.
- Type guards (explanation in the technical design) will be introduced which will make defining the interfaces for variables with more than one type possible.
- Any will be removed for the codebase. (There may be 3-4 files where the any remains, the reason is explained in the technical design section)
- A check in the eslint would be added that prohibits explicit defining of any.
- A check would be added in the lint checks that prohibits the use of any.
- Proper documentation would be added on instructions for defining the interfaces.
- We won't need to generate a separate *thid_party.js* file for loading third party libs.
- Many of the script and third party imports will be loaded with webpack.
- Webpack compilation speed would be increased. Type checker will be removed from the webpack as it takes a lot of time and its outcome is used nowhere.
- We may be using webpack 5.0.(If released)
- Documentation will be added on dealing with the typescript errors. (Detailed explanation in the technical design section.)
- The existing documentation on webpack in the Oppia Wiki would be improved. And the working of our current webpack config would be explained.
- Typescript checks would be added to the pre push hook. (Here by typescript checks I mean the pre-existing typescript checks.)

Remove all custom typings for typescript and any types (clubbed two parts in the idea's page)

The type definitions will be added for the third party libraries similar to the one added in this PR.

Remove all "any" types from the codebase

The all “any” types in the codebase can be classified into four categories(The files are listed in the technical design)

- Ones with issue #7176 - This issue is caused because the backend responds with a dict that is in snake_case and we can’t define an interface for the dict because tslint favours camelCasing instead of snake_casing. So, the solution to this issue was to change the case of the keys of the dict before sending the response.

  To deal with the first type, first the case of the response of all the api endpoints will be changed to camelCase. Then the types of all the any can be declared.

- Ones with issue #7165 - These any types are the ones that have the issue 7165 mentioned in their comments.

  The second type can be further classified in the following types
  - Variables can be of multiple types - The custom type guards will be introduced (details in technical design) that will allow the types of the variables to be declared.

  - Warning Type - There are some warnings defined as any in the spec files and thus, a simple interface can be declared for them.

  - Object Factory Upgradation - These are the files in which variables were awaiting angular upgradation of some ObjectFactories before their type could be defined but now those objectFactories have been upgraded and their types can be declared straight away or is similar to the *Ones with issue #7176.* In case there is any Object Factory that is not upgraded and is blocking the type defining I will first upgrade it and then add the type definitions.

    - Unknown - The types of these variables will need to be researched using the instruction described below.

- Others - These are the any types that are typed as any but do not have any comment associated with them. The any for the third type for almost all the types is declared by me in my recent PR’s #8823 and #8785. The method for the remaining files can be seen in the technical design section below.

- Ones in the typings directory - These are the custom typing that are added for the third party libraries. They have some variables that are typed as any.

In general, the steps that are followed to determine the types for the variables are

- Look for the use of the variable in that file itself. Sometimes the type of the variable can be inferred from that file itself.

- Perform a global search related to the function or the variable and see what kind of values are passed in the variable. Mostly the type of the variable is known in this step.

- Start the server and use console.log() to be more clear about the actual type of values of the variable.

Removing any from the codebase and adding proper interfaces and types would help the developers in the following way

- Clean code. This allows developers to write more robust code and maintain it, resulting in better, cleaner code.
- A clean code makes it easy for other devs to understand the code, and find out what's really happening like what all arguments are passed in that function, what does it return etc.
- Avoid errors while writing code.

When the mini-project is completed, the use of **any** after that should be stopped. I plan to achieve this by adding a **check** for the use of any in the **lint checks** and **eslint.** As mentioned in the technical design below there are some places where the type should really be any. There would be an excluded files section in the check, so that this check doesn't validate those files. The Technical Design and mock screenshots of this lint check can be found here. Here is a screenshot on how the lint check would look.

It should also be ensured that it is easy for developers to follow the conventions that I am planning to put in place. For that I will add proper documentation for adding the type definitions in the **Oppia Wiki**. The exact plan for the type definitions documentation can be found in the Technical Design.

Note - We can't add a check for implicit declaration of any because the codebase is not migrated to angular yet. That is, directives and controllers are not yet migrated to Angular. We have to wait for them to be migrated before we add a check for implicit declaration of any. I think we can prohibit implicit declaration of any after the codebase is migrated to angular.

Replace third_party and remaining <script> imports with webpack

In the mini-project "Replace third_party and remaining <script> imports with webpack" the script tags included in the *mainpage.html* files would be removed and we will use webpack to load those dependencies. This would allow the libraries to be loaded only when they are needed. This would help in boosting the web app's performance. There will be a *third-party-imports* directory in the *core/templates* directory that would contain the require() imports for these third_party libs.

The generated *third_party.js* file will no longer be needed. And all the code related to it in the build script can be removed.

Reduce the overall time for webpack compilation

In the mini-project "Reduce the overall time for webpack compilation" I plan to introduce things like *cache-loader, efficient devTools* to the webpack, which would decrease the build time along with the consequent build times during the development. Rest of the details are present in the technical design section.

Add documentation on all typescript and webpack errors and solutions on how to fix them

In the mini project "Add documentation on all typescript and webpack errors and solutions on how to fix them" I plan to create a documentation covering **all of the errors** that the devs may majorly come across while working on Oppia, This includes the instructions to **add type definitions** described above. This also included the **webpack errors** which may be faced by the devs and an **explanation** of the webpack config that we use. I also plan to **create a google sheet** similar to this sheet for reporting the typescript and webpack errors faced by devs and So, the documentation can be updated accordingly.

Adding typescript checks to the pre_push_hooks

This small mini-project includes adding typescript checks to the start script and the pre push hook. Currently the webpack has a plugin that does the type checking but no weightage in given to its outcome i.e. devs are allowed to push even when the type checks in the webpack are failing. To be honest, I **didn't** even know the webpack had a type checker before studying its config.

This project will include the typescript checks in the script and will not allow the devs to push until the typescript errors are fixed.

Technical Design

Remove all Custom Typings for Typescript

Currently there is a file (typings/third-party-defs.d.ts) which lists the *third-party-libraries* we use that don't have type definitions. The goal of this mini project is to write type definitions for these libraries and create a file for each library in the *typings* folder itself and remove this file.

```
1  // This file should contain declaration for third party libraries for which
2  // type definitions are not present.
3  declare var BlobBuilder: any;
4  declare var MIDI: any;
5  declare var Guppy: any;
6  declare var Sk: any;
7  declare var MathExpression: any;
8  declare var PencilCodeEmbed: any;
9  declare var WaveSurfer: any;
```

I created a pr writing type definitions for PencilCodeEmbed (PR #8712) and a doc containing my approach for writing type definitions for third party libraries.

The basic steps required for adding type definitions are

- Finding the source code of the library - This had to be searched manually i.e. search for the file where the library is imported. For example for pencil-code-embed I found (extensions/interactions/pencilcode.html) file with the following content.

```
1   <script src="https://pencilcode.net/lib/pencilcodeembed.js"></script>
2
```

- Create a file for writing the type definitions - A file in the typings directory with name <library-name>-defs.d.ts

- Writing the type definitions - For writing custom definitions, existing type definitions in DefinitelyTyped and this guide can be used as a reference. The steps which I followed while writing the Type Definitions for PencilCodeEmbed are
    - First I noticed that only a class i.e. PencilCodeEmbed is used in our codebase. And only that class is exported by the library as we can see here.

```
355     if (global.define && global.define.amd) {
356       // Support AMD-style loading, if present.
357       define(function() {
358         return {
359           PencilCodeEmbed: PencilCodeEmbed
360         };
361       });
362     } else {
363       // Otherwise, just set the global symbol.
364       global.PencilCodeEmbed = PencilCodeEmbed
365     }
```

    - Then we need to define the constructor for the class, as we can see here only four variables are defined in the class, so we can begin by defining them.

```
112     function PencilCodeEmbed(div) {
113       this.div = div;
114       this.updatedCode = '';
115       this.callbacks = {};
116       this.requests = {};
117
118       // hook up noop event handlers
119       this.on('load', function(){});
120       this.on('update', function(code){});
121       this.on('startExecute', function(){});
122       this.on('execute', function(){});
123       this.on('error', function(error){});
124     }
```

    - Now I check for the functions that are left to be defined and defined them. Mostly the types for arguments and types can be inferred from the code itself or the documentation of the library.

    - The type definitions look like this.

- Testing the type definitions - The typings we wrote can be tested by running `python -m scripts.typescript_checks`.

I have found the source codes of all the libraries in the third-party-defs.d.ts and typings for those can be added in a similar way as that of PencilCodeEmbed.

Library	**Source Code**	**Reference**
PencilCodeEmbed	https://pencilcode.net/lib/pencilcodeembed.js	https://github.com/oppia/oppia/blob/develop/extensions/interactions/pencilcode.html
MIDI	/third_party/static/midi-js-a8a842/js/midi/loader.js	https://github.com/oppia/oppia/blob/develop/extensions/interactions/midijs.html
MathExpression	/third_party/static/math-expressions-1.7.0/math-expressions.js	https://github.com/oppia/oppia/blob/develop/extensions/interactions/math_expressions.html
Sk	/third_party/static/skulpt-dist-1.1.0/skulpt.js	https://github.com/oppia/oppia/blob/develop/extensions/interactions/skulpt.html
Guppy	/third_party/static/guppy-b5055b/src/guppy.js	https://github.com/oppia/oppia/blob/develop/extensions/interactions/guppy.html
WaveSurfer	https://unpkg.com/wavesurfer.js@2.2.1/dist/wavesurfer.js	https://github.com/oppia/oppia/blob/develop/manifest.json#L524
BlobBuilder	NA	https://developer.mozilla.org/en-US/docs/Web/API/BlobBuilder

As said the type definitions for **MIDI**, **MathExpression**, **Sk**, **Guppy**, **WaveSurfer** can be added in a similar way as that of **PencilCodeEmbed**.

However **BlobBuilder** is not exactly a third party library but a web API which is obsolete. (Actually, we use an updated API, we use this in case browser doesn't have that Updated API)

But the type definitions for BlobBuilder can still be added which look something like

```
You, 5 hours ago | 1 author (You)
// Reference - https://developer.mozilla.org/en-US/docs/Web/API/BlobBuilder
You, 5 hours ago | 1 author (You)
class BlobBuilder {
  append: (data: ArrayBuffer) => void;
  append: (data: Blob) => void;
  append: (data: String, endings?: String) => void;
  getBlob: (contentType?: String) => Blob;
  getFile: (name: String, contentType?: String) => File;
  constructor();
}
```

These definitions were written by taking this doc as a reference.

Remove all "any" types from the codebase

I have prepared a **google sheet** classifying the files which have any defined according to their types.

Ones with issue #7176

File	Status
core/templates/services/state-top-answers-stats.service.spec.ts	To be done
core/templates/services/improvements.service.ts	To be done
core/templates/services/site-analytics.service.ts	To be done
core/templates/services/compute-graph.service.ts	To be done
core/templates/services/local-storage.service.ts	To be done
core/templates/services/exploration-features.service.ts	To be done
core/templates/services/extension-tag-assembler.service.ts	To be done
core/templates/pages/exploration-player-page/services/state-classifier-mapping.service.ts	To be done
core/templates/pages/exploration-editor-page/editor-tab/services/answer-groups-cache.service.ts	To be done
core/templates/pages/exploration-editor-page/services/exploration-diff.service.ts	To be done
core/templates/pages/exploration-editor-page/services/changes-in-human-readable-form.service.ts	To be done
core/templates/pages/exploration-editor-page/history-tab/services/version-tree.ser	To be done

vice.ts	
core/templates/pages/email-dashboard-pages/email-dashboard-data.service.ts	To be done
core/templates/domain/classifier/ClassifierObjectFactory.ts	To be done
core/templates/domain/collection/CollectionNodeObjectFactory.ts	To be done
core/templates/domain/collection/CollectionObjectFactory.ts	To be done
core/templates/domain/collection/CollectionPlaythroughObjectFactory.ts	To be done
core/templates/domain/collection/CollectionRightsObjectFactory.ts	To be done
core/templates/domain/editor/undo_redo/ChangeObjectFactory.ts	To be done
core/templates/domain/exploration/AnswerGroupObjectFactory.ts	To be done
core/templates/domain/exploration/AnswerStatsObjectFactory.ts	To be done
core/templates/domain/exploration/HintObjectFactory.ts	To be done
core/templates/domain/story_viewer/story-viewer-backend-api.service.ts	To be done
core/templates/domain/classroom/classroom-backend-api.service.ts	In Progress
core/templates/domain/exploration/OutcomeObjectFactory.ts	To be done
core/templates/domain/exploration/ParamChangeObjectFactory.ts	To be done
core/templates/domain/exploration/ParamChangesObjectFactory.ts	To be done
core/templates/domain/exploration/ParamSpecObjectFactory.ts	To be done
core/templates/domain/exploration/ParamSpecsObjectFactory.ts	To be done
core/templates/domain/exploration/ParamTypeObjectFactory.ts	To be done
core/templates/domain/exploration/RecordedVoiceoversObjectFactory.ts	To be done
core/templates/domain/exploration/RuleObjectFactory.ts	To be done
core/templates/domain/exploration/SubtitledHtmlObjectFactory.ts	To be done
core/templates/domain/exploration/VoiceoverObjectFactory.ts	To be done
core/templates/domain/exploration/WrittenTranslationObjectFactory.ts	To be done
core/templates/domain/exploration/WrittenTranslationsObjectFactory.ts	To be done
core/templates/domain/feedback_message/FeedbackMessageSummaryObjectFactory.ts	To be done
core/templates/domain/feedback_thread/FeedbackThreadObjectFactory.ts	To be done
core/templates/domain/learner_dashboard/LearnerDashboardActivityIdsObjectFactory.ts	To be done
core/templates/domain/skill/SkillRightsObjectFactory.ts	To be done
core/templates/domain/statistics/LearnerActionObjectFactory.ts	To be done
core/templates/domain/statistics/PlaythroughIssueObjectFactory.ts	To be done
core/templates/domain/story_viewer/ReadOnlyStoryNodeObjectFactory.ts	To be done
core/templates/domain/story_viewer/StoryPlaythroughObjectFactory.ts	To be done

core/templates/domain/story/StoryContentsObjectFactory.ts	To be done
core/templates/domain/story/StoryNodeObjectFactory.ts	To be done
core/templates/domain/subtopic_viewer/ReadOnlySubtopicPageObjectFactory.ts	To be done
core/templates/domain/suggestion/SuggestionObjectFactory.ts	To be done
core/templates/domain/topic/StoryReferenceObjectFactory.ts	To be done
core/templates/domain/topic/SubtopicPageContentsObjectFactory.ts	To be done
core/templates/domain/topic/SubtopicPageObjectFactory.ts	To be done
core/templates/domain/topic/TopicRightsObjectFactory.ts	To be done
core/templates/domain/topic/TopicSummaryObjectFactory.ts	In Progress
core/templates/domain/user/UserInfoObjectFactory.ts	To be done
core/templates/domain/utilities/AudioLanguageObjectFactory.ts	To be done
core/templates/components/state-editor/state-editor-properties-services/state-editor.service.ts	To be done
extensions/classifiers/svm-prediction.service.ts	To be done
extensions/interactions/NumericInput/directives/numeric-input-validation.service.ts	To be done
extensions/interactions/PencilCodeEditor/directives/pencil-code-editor-validation.service.ts	To be done
extensions/interactions/DragAndDropSortInput/directives/drag-and-drop-sort-input-validation.service.ts	To be done
extensions/interactions/CodeRepl/directives/code-repl-validation.service.ts	To be done
extensions/interactions/MusicNotesInput/directives/music-notes-input-validation.service.ts	To be done
extensions/interactions/MultipleChoiceInput/directives/multiple-choice-input-validation.service.spec.ts	To be done
extensions/interactions/MultipleChoiceInput/directives/multiple-choice-input-validation.service.ts	To be done
extensions/interactions/LogicProof/directives/logic-proof-validation.service.ts	To be done
extensions/interactions/GraphInput/directives/graph-input-validation.service.ts	To be done
extensions/interactions/ImageClickInput/directives/image-click-input-validation.service.spec.ts	To be done
extensions/interactions/ImageClickInput/directives/image-click-input-validation.service.ts	To be done
extensions/interactions/InteractiveMap/directives/interactive-map-validation.service.ts	To be done
extensions/interactions/Continue/directives/continue-validation.service.ts	To be done
extensions/interactions/SetInput/directives/set-input-validation.service.ts	To be done
extensions/interactions/ItemSelectionInput/directives/item-selection-input-validatio	To be done

n.service.ts	
extensions/interactions/MathExpressionInput/directives/math-expression-input-validation.service.ts	To be done
extensions/interactions/EndExploration/directives/end-exploration-validation.service.ts	To be done
extensions/interactions/EndExploration/directives/end-exploration-validation.service.spec.ts	To be done
core/templates/domain/feedback_thread/FeedbackThreadSummaryObjectFactory.ts	To be done

As we can see in the screenshot

```
// TODO(#7176): Replace 'any' with the exact type. This has been kept as
// 'any' because 'topicSummaryBackendDict' is a dict with  underscore_cased
// keys which give tslint errors against underscore_casing in favor of
// camelCasing.
```

This issue is caused because the backend responds with a dict that is in snake_case and we can't define an interface for the dict because tslint favours camelCasing instead of snake_casing. So, the solution to this issue was to change the case of the keys of the dict before sending the response.

To convert the case of the keys of the dicts before sending the response to camelCase so that an interface can be defined for the objects. Now camelizing the response for only some data handlers would make the code messy. So, to generalize things, I would camelize the response of every endpoint. So, I would like to divide this section in further two parts.

- **Camelizing the response of each endpoint** - The main idea is to create a generalized function for camelzing the dicts in the BaseHandler which looks something like

```
def camelize_string(string):
    """Changes the case of string from snake_case to camelCase.

    Args:
        string: str. The string whose case is to changed to camelCase
        from snake_case.

    Returns:
```

```
        The string after changing the case of input string to camelCase.
    """

    snake_case_re = re.compile(r'([^\-_\s])[\-_\s]+([^\-_\s])')

    return ''.join([
        string[0].lower() if not string[:2].isupper() else string[0],
        snake_case_re.sub(
            lambda m: m.group(1) + m.group(2).upper(), string[1:]),
    ])


def camelize(obj):
    """Changes the case of the keys of dict from snake_case to camelCase.

    Args:
        obj: any object. If the object is a dict it changes
        the case of it's dicts else if it is a list it iterates over all
        the items in the list and changes the case of keys to camelCase
        if it finds another dict. However, if the object is neither a dict
nor
        a list it returns the object.

    Returns:
        The object after changing the case of keys in the dictionaries
        in the object to camelCase.
    """

    if isinstance(obj, dict):
        return {camelize_string(k): camelize(v) for k, v in obj.items()}
    elif isinstance(obj, list):
        return [camelize(v) for v in obj]

    return obj
```

The tests for these functions would look something like

```
class CheckCaseChangeServiceTests(test_utils.GenericTestBase):

    def test_camelize_string(self):
```

```
        test_strings = [
            'test_string',
            'testString',
            ]
        expected_results = [
            'testString',
            'testString'
            ]

        for i in python_utils.RANGE(len(test_strings)):
            string = test_strings[i]
            expected_result = expected_results[i]
            result = case_change_service.camelize_string(string)
            self.assertEqual(result, expected_result)

    def test_camelize(self):
        test_objects = [
            [
                {
                    'key_one': 0,
                    'key_two': {
                        'key_three': 0,
                        'key_four': [
                            {
                                'key_five': 0
                            }
                        ]
                    }
                },
                {
                    'keySix': 0
                }
            ],
            {
                'key_one': 'value_one',
                'key_two': {
                    'key_Three': 'value_two'
                }
            },
            [
                'string_one',
                'string_two'
```

```
        ],
        {
            'element_one',
            'element_two'
        },
        {
            'key_one': {
                'element_one'
            },
            'key_two': [
                {
                    'key_three': 0,
                    'key_four': {
                        'element_two'
                    }
                },
                {
                    'element_three'
                }
            ]
        }
    ]

    expected_results = [
        [
            {
                'keyOne': 0,
                'keyTwo': {
                    'keyThree': 0,
                    'keyFour': [
                        {
                            'keyFive': 0
                        }
                    ]
                }
            },
            {
                'keySix': 0
            }
        ],
        {
            'keyOne': 'value_one',
```

```
                'keyTwo': {
                    'keyThree': 'value_two'
                }
            },
            [
                'string_one',
                'string_two'
            ],
            {
                'element_one',
                'element_two'
            },
            {
                'keyOne': {
                    'element_one'
                },
                'keyTwo': [
                    {
                        'keyThree': 0,
                        'keyFour': {
                            'element_two'
                        }
                    },
                    {
                        'element_three'
                    }
                ]
            }
        ]

        for i in python_utils.RANGE(len(test_objects)):
            test_object = test_objects[i]
            expected_result = expected_results[i]
            result = case_change_service.camelize(test_object)
            self.assertEqual(result, expected_result)
```

And use this function in the *render_json* like

```
values_with_camelized keys = self.camelize(values)
json_output = json.dumps(values_with_camelized_keys,
    cls=utils.JSONEncoderForHTML)
```

```
self.response.write('%s%s' % (feconf.XSSI_PREFIX, json_output))
```

And then making the necessary changes in the frontend. These are the files that contain the endpoints that need to be updated.

File
core/controllers/editor.py
core/controllers/feedback.py
core/controllers/suggestion.py
core/controllers/question_editor.py
core/controllers/library.py
core/controllers/topic_viewer.py
core/controllers/skill_mastery.py
core/controllers/collection_editor.py
core/controllers/review_tests.py
core/controllers/practice_sessions.py
core/controllers/subtopic_viewer.py
core/controllers/concept_card_viewer.py
core/controllers/learner_dashboard.py
core/controllers/topic_editor.py
core/controllers/reader.py
core/controllers/profile.py
core/controllers/learner_playlist.py
core/controllers/story_editor.py
core/controllers/questions_list.py
core/controllers/skill_editor.py
core/controllers/collection_viewer.py
core/controllers/creator_dashboard.py
core/controllers/classroom.py
core/controllers/topics_and_skills_dashboard.py
core/controllers/email_dashboard.py
core/controllers/story_viewer.py

- **Writing the type definitions** - As now the responses are camel cased interfaces can be defined for the variables in the files listed above.

In this way all the any types which come under the 7176 issue can be defined. Related PR(#8759).

Ones with issue #7165

File	Type	Status
core/templates/services/autogenerated-audio-player.service.ts	complex_dict	To be done
core/templates/domain/exploration/RuleObjectFactory.ts	complex_dict	To be done
core/templates/domain/exploration/AnswerGroupObjectFactory.ts	complex_dict	To be done
core/templates/services/schema-default-value.service.ts	complex_dict	To be done
core/templates/services/schema-undefined-last-element.service.ts	complex_dict	To be done
core/templates/services/speech-synthesis-chunker.service.ts	complex_dict	To be done
core/templates/pages/exploration-editor-page/editor-tab/services/interaction-details-cache.service.ts	complex_dict	To be done
core/templates/domain/exploration/OutcomeObjectFactory.ts	complex_dict	To be done
core/templates/domain/exploration/RecordedVoiceoversObjectFactory.ts	complex_dict	To be done
core/templates/domain/exploration/WrittenTranslationsObjectFactory.ts	complex_dict	To be done
core/templates/domain/exploration/ExplorationDraftObjectFactory.ts	complex_dict	To be done
core/templates/domain/collection/CollectionNodeObjectFactory.ts	complex_dict	To be done
core/templates/domain/collection/collection-validation.service.ts	complex_dict	To be done
core/templates/domain/story/StoryContentsObjectFactory.ts	complex_dict	To be done

core/templates/domain/story/StoryObjectFactory.ts	complex_dict	To be done
core/templates/domain/objects/UnitsObjectFactory.ts	complex_dict	To be done
core/templates/components/graph-services/graph-layout.service.ts	complex_dict	To be done
core/templates/components/state-editor/state-editor-properties-services/state-property.service.ts	complex_dict	To be done
extensions/interactions/ImageClickInput/directives/image-click-input-rules.service.ts	complex_dict	To be done
extensions/interactions/rules.spec.ts	complex_dict	To be done
extensions/interactions/MathExpressionInput/directives/math-expression-input-rules.service.ts	complex_dict	To be done
core/templates/domain/exploration/ParamSpecsObjectFactory.ts	complex_dict	To be done
core/templates/pages/exploration-player-page/services/audio-translation-language.service.ts	Miscellaneous	To be done
core/templates/pages/exploration-player-page/services/player-transcript.service.ts	Miscellaneous	To be done
core/templates/pages/exploration-player-page/services/extract-image-filenames-from-state.service.ts	Miscellaneous	To be done
core/templates/domain/topic/TopicObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/state/StateObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/exploration/ExplorationObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/exploration/SolutionObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/exploration/ParamTypeObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/exploration/InteractionObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/utilities/language-util.service.ts	Miscellaneous	To be done
core/templates/domain/state_card/StateCardObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/objects/NumberWithUnitsObjectFactory.ts	Miscellaneous	To be done

core/templates/domain/suggestion/SuggestionThreadObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/suggestion/SuggestionObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/skill/ConceptCardObjectFactory.ts	Miscellaneous	To be done
extensions/interactions/CodeRepl/code-repl-prediction.service.ts	Miscellaneous	To be done
extensions/interactions/TextInput/directives/text-input-validation.service.ts	Miscellaneous	To be done
extensions/interactions/GraphInput/directives/graph-input-rules.service.spec.ts	Miscellaneous	To be done
extensions/interactions/GraphInput/directives/graph-input-rules.service.ts	Miscellaneous	To be done
extensions/interactions/NumberWithUnits/directives/number-with-units-rules.service.ts	Miscellaneous	To be done
core/templates/domain/topic_viewer/topic-viewer-backend-api.service.ts	Miscellaneous	To be done
core/templates/domain/topic/SubtopicObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/statistics/PlaythroughObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/statistics/LearnerAnswerDetailsObjectFactory.ts	Miscellaneous	To be done
extensions/interactions/NumberWithUnits/directives/number-with-units-validation.service.ts	Miscellaneous	To be done
core/templates/domain/exploration/StatesObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/question/QuestionSummaryForOneSkillObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/question/QuestionSummaryForOneSkillObjectFactorySpec.ts	Miscellaneous	To be done
core/templates/domain/question/QuestionSummaryObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/question/pretest-question-backend-api.service.ts	Miscellaneous	To be done
core/templates/domain/question/QuestionSummaryObjectFactorySpec.ts	Miscellaneous	To be done
core/templates/domain/collection/collection-validation.service.spec.ts	Miscellaneous	To be done

core/templates/domain/learner_dashboard/LearnerDashboardActivityIdsObjectFactorySpec.ts	Miscellaneous	To be done
core/templates/domain/opportunity/SkillOpportunityObjectFactorySpec.ts	Miscellaneous	To be done
core/templates/domain/opportunity/ExplorationOpportunitySummaryObjectFactorySpec.ts	Miscellaneous	To be done
core/templates/domain/opportunity/ExplorationOpportunitySummaryObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/opportunity/SkillOpportunityObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/skill/skill-mastery-backend-api.service.ts	Miscellaneous	To be done
core/templates/domain/skill/MisconceptionObjectFactory.ts	Miscellaneous	To be done
core/templates/domain/skill/MisconceptionObjectFactorySpec.ts	Miscellaneous	To be done
core/templates/domain/classifier/AnswerClassificationResultObjectFactorySpec.ts	Miscellaneous	To be done
core/templates/components/state-editor/state-editor-properties-services/state-hints.service.ts	Miscellaneous	To be done
extensions/interactions/FractionInput/directives/fraction-input-validation.service.ts	Miscellaneous	To be done
core/templates/pages/exploration-player-page/services/learner-params.service.ts	object_factory_upgrades	To be done
core/templates/domain/statistics/LearnerActionObjectFactory.ts	object_factory_upgrades	To be done
core/templates/domain/classifier/AnswerClassificationResultObjectFactory.ts	object_factory_upgrades	To be done
core/templates/components/state-editor/state-editor-properties-services/state-editor.service.ts	object_factory_upgrades	To be done
extensions/interactions/base-interaction-validation.service.ts	object_factory_upgrades	To be done
extensions/interactions/GraphInput/directives/graph-detail.service.ts	object_factory_upgrades	To be done
extensions/interactions/GraphInput/directives/graph-utils.service.ts	object_factory_upgrades	To be done
extensions/interactions/DragAndDropSortInput/directives/drag-and-drop-sort-input-validation.service.spec.ts	WARNING TYPES constant definition needed	To be done
extensions/interactions/CodeRepl/directives/code-repl-validation.service.spec.ts	WARNING TYPES constant definition needed	To be done

extensions/interactions/MultipleChoiceInput/directives/multiple-choice-input-validation.service.spec.ts	WARNING TYPES constant definition needed	To be done
extensions/interactions/GraphInput/directives/graph-input-validation.service.spec.ts	WARNING TYPES constant definition needed	To be done
extensions/interactions/ImageClickInput/directives/image-click-input-validation.service.spec.ts	WARNING TYPES constant definition needed	To be done
extensions/interactions/InteractiveMap/directives/interactive-map-validation.service.spec.ts	WARNING TYPES constant definition needed	To be done
extensions/interactions/Continue/directives/continue-validation.service.spec.ts	WARNING TYPES constant definition needed	To be done
extensions/interactions/ItemSelectionInput/directives/item-selection-input-validation.service.spec.ts	WARNING TYPES constant definition needed	To be done
extensions/interactions/NumericInput/directives/numeric-input-validation.service.spec.ts	WARNING TYPES constant definition needed	Done
extensions/interactions/EndExploration/directives/end-exploration-validation.service.spec.ts	WARNING TYPES constant definition needed	To be done

The any types in the codebase under issue #7165 can be further divided into multiple categories

- **WARNING_TYPES CONSTANT** - These any types look something like this

```
// TODO(#7165): Replace 'any' with the exact type. This has been kept as
// 'any' because 'WARNING_TYPES' is a constant and its type needs to be
// preferably in the constants file itself.
```

  For assigning a type to these objects just a type for WARNING_TYPES constant had to be declared. I have opened a PR(#8770) doing that. Once this PR gets merged the type of the variable can simply be defined as WARNING_TYPES_CONSTANT like

```
import { Rule, RuleObjectFactory } from 'domain/exploration/RuleObjectFactory';

import { AppConstants } from 'app.constants';
+ import { WARNING_TYPES_CONSTANT } from 'app-type.constants';

describe('NumericInputValidationService', () => {
-   // TODO(#7165): Replace 'any' with the exact type. This has been kept as
-   // 'any' because 'WARNING_TYPES' is a constant and its type needs to be
-   // preferably in the constants file itself.
-   let validatorService: NumericInputValidationService, WARNING_TYPES: any;
+   let validatorService: NumericInputValidationService;
+   let WARNING_TYPES: WARNING_TYPES_CONSTANT;

    let currentState: string;
    let answerGroups: AnswerGroup[], goodDefaultOutcome: Outcome;
```

- **Waiting for object factory to be upgraded** - These are the files in which variables were awaiting angular upgradation of some ObjectFactories before their type could be defined but now those objectFactories have been upgraded and their types can be declared straight away or is similar to the *Ones with issue #7176*. In case there is any Object Factory that is not upgraded and is blocking the type defining I will first upgrade it and then add the type definitions.

```
// TODO(#7165): Replace 'any' with the exact type. This has been kept as
// 'any' because 'outcome' is an outcome domain object and this can be
// directly typed to 'Outcome' type once 'OutcomeObjectFactory' is upgraded.
```

- **Are Complex Objects or have varying types of keys**- Here by complex dict, I mean that there are some keys that may or may not be present or there are some keys whose values can be of multiple types. The any types of this kind have this message

```
// TODO(#7165): Replace 'any' with the exact type. This has been typed
// as 'any' since 'inputs' is a complex object having varying types. A general
// type needs to be found.
```

When I tried to look into files having this comment, I found that this comment is **not always accurate.** In Fact files having this comment can just have simple types or interfaces like *string*, *number* etc. For example in UnitsObjectFactory the type definitions for any types with this comment can be added by type as simple as *string*

```
// TODO(#7165): Replace 'any' with the exact type. This has been kept as
// 'any' because 'units' is a list with varying element types. An exact
// type needs to be found for it.
toMathjsCompatibleString(units: any): string {
toMathjsCompatibleString(units: string): string {
  // Makes the units compatible with the math.js allowed format.
  units = units.replace(/per/g, '/');
```

or simple interfaces like

```
export interface Unit {
  unit: string;
  exponent: number;
}

export interface UnitDict {
  [unit: string]: number
}
```

```
export interface IUnitsDict {
  // TODO(#7165): Replace 'any' with the exact type. This has been kept as
  // 'any' because 'units' is a list with varying element types. An exact
  // type needs to be found for it.
  units: any;
  units: Array<Unit>;
}
```

The complete type definitions I added for the UnitsObjectFactory can be seen here.

But sometimes the comment can be true and the variables may be of varying types. For example in RuleObjectFactory, when I researched about the type of **inputs** I found that it is a dict having variables like *x, y, z* as keys and the values could be of many types like *Graphs, Number With Expressions, Strings etc.*

For cases like these the interface can be added in the following way

```
export interface RuleInputs {
  [variable: string]: (string | number | IFractionDict | NumberWithUnitsDict
    | string[] | Note[] | number[] | GraphDict);
}
```

This is a complete list of values that can be taken by the RuleInputs dict values. It is the only possible way we could define an interface for the RuleInputs. But doing this introduces a new problem i.e.

```
extensions/interactions/GraphInput/directives/graph-input-validation.servic
e.ts(106,29): error TS2339: Property 'vertices' does not exist on type
'string | number | string[] | IFractionDict | number[] | Note[] |
NumberWithUnitsDict | GraphDict'.
Property 'vertices' does not exist on type 'string'.
```

The problem now is that some properties would exist only in a particular type say *GraphDict*, and the code may be logically correct but typescript shows a error here because the type of the variable is not *GraphDict* but actually *(string | number | IFractionDict | NumberWithUnitsDict| string[] | Note[] | number[] | GraphDict)*.

This problem can be solved by adding our **custom type guards** for determining the exact types when we access the variable property of the RuleInputs. **Note** that this has to be done as we can't have runtime checks for comparing the types with interfaces. (Reference) So, the best way to do it would be to add a TypeService for that which would look something like

```
export class RuleInputsTypeFactory {
  _isGraphDict(variable: Object): variable is GraphDict {
    return 'isDirected' in variable;
  }

  graphDictInstance(variable: (string | number | IFractionDict | NumberWithUnitsDict
    | string[] | Note[] | number[] | GraphDict)): GraphDict {
    if (this._isGraphDict(variable)) {
      return variable;
    }
  }
}
```

The above code defines a custom guard for the graphDict, we know the variable is a graphDict if there is a property called 'isDirected' present in it. The same thing could be repeated for the rest of the types.

And this TypeFactory can be used in the other services like this

```
-     answerGroups[0].rules[0].inputs.g.vertices = new Array(11);
-     answerGroups[0].rules[1].inputs.g.vertices = new Array(11);
-     answerGroups[1].rules[0].inputs.g.vertices = new Array(11);
+     ritf.graphDictInstance(
+       answerGroups[0].rules[0].inputs.g).vertices = new Array(11);
+     ritf.graphDictInstance(
+       answerGroups[0].rules[1].inputs.g).vertices = new Array(11);
+     ritf.graphDictInstance(
+       answerGroups[1].rules[0].inputs.g).vertices = new Array(11);
```

In this way we can write the type definitions for the any types really having varying types. The example commit for this kind of type definitions can be seen here.

The type definitions for the rest of the files that fall to this category can be added similarly.

- **Miscellaneous** - These are the any typed variables that don't have a descriptive comment message.

```
getAllAudioLanguageCodes(): Array<string> {
  // TODO(#7165): Replace any with exact type.
  var allAudioLanguageCodes = (
    this.supportedAudioLanguageList.map(function(audioLanguage: any) {
      return audioLanguage.id;
    }));
  return allAudioLanguageCodes;
}
```

Seeing these files, I found that they were similar to the files having *complex_dict* comment that is the type for these variables can be as simple as a string or really a dict with variable types. The type definitions in these files can be declared in the same way discussed in the *complex_dict* part.

Other files

Other files include the files where

- Angular is defined as any
- Where variables are angular js native objects

- The places where the type should be any

File	Type	Status	
core/templates/tests/console_errors.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/terms-page/terms-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/profile-page/profile-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/get-started-page/get-started-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/topic-viewer-page/topic-viewer-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/pending-account-deletion-page/pending-account-deletion-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/exploration-player-page/exploration-player-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/practice-session-page/practice-session-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/creator-dashboard-page/creator-dashboard-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/thanks-page/thanks-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/learner-dashboard-page/learner-dashboard-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/story-editor-page/story-editor-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/donate-page/donate-page.module.ts	angular defined as any	Done	PR #878

			5
core/templates/pages/about-page/about-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/contact-page/contact-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/story-viewer-page/story-viewer-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/topic-editor-page/topic-editor-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/exploration-editor-page/exploration-editor-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/subtopic-viewer-page/subtopic-viewer-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/collection-editor-page/collection-editor-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/topics-and-skills-dashboard-page/topics-and-skills-dashboard-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/notifications-dashboard-page/notifications-dashboard-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/privacy-page/privacy-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/teach-page/teach-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/review-test-page/review-test-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/signup-page/signup-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/delete-account-page/delete-account-page.module.ts	angular defined as any	Done	PR #878

			5
core/templates/pages/library-page/library-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/error-pages/error-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/admin-page/admin-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/community-dashboard-page/community-dashboard-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/maintenance-page/maintenance-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/landing-pages/topic-landing-page/topic-landing-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/landing-pages/stewards-landing-page/stewards-landing-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/moderator-page/moderator-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/collection-player-page/collection-player-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/splash-page/splash-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/email-dashboard-pages/email-dashboard-result.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/email-dashboard-pages/email-dashboard-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/skill-editor-page/skill-editor-page.module.ts	angular defined as any	Done	PR #8785
core/templates/pages/classroom-page/classroom-page.module.ts	angular defined as any	Done	PR #878

			5
core/templates/pages/preferences-page/preferences-page.module.ts	angular defined as any	Done	PR #8785
core/templates/services/csrf-token.service.ts		In Progress	PR #8823
core/templates/services/request-interceptor.service.ts		In Progress	PR #8823
core/templates/services/utils.service.ts		In Progress	PR #8823
core/templates/services/contextual/url.service.ts		In Progress	PR #8823
core/templates/services/debug-info-tracker.service.ts		In Progress	PR #8823
core/templates/domain/utilities/url-interpolation.service.ts		In Progress	PR #8823
extensions/classifiers/python-program.tokenizer.ts		In Progress	PR #8823
core/templates/domain/utilities/browser-checker.service.ts		In Progress	PR #8823
core/templates/services/suggestion-modal.service.ts	possible after angular upgradation	To be done	
core/templates/domain/utilities/FileDownloadRequestObjectFactory.ts	possible after angular upgradation	To be done	
core/templates/filters/remove-duplicates-in-array.pipe.ts	meant to be any	To be done	
core/templates/filters/string-utility-filters/normalize-whitespace.pipe.ts	meant to be any	To be done	
core/templates/filters/string-utility-filters/normalize-whitespace-punctuation-and-case.pipe.ts	meant to be any	To be done	

These are the type of files that don't have a 7165 or 7176 issue comment related to them. These files can also be divided into multiple other categories

- **Angular defined as any** - In almost all the modules the *angular* variable is declared as any.

  ```
  declare var angular: any;
  ```

  On researching a bit about it's type I found out it's exact type by the type of angular in various other directives and controllers. I have made a PR(#8785) already declaring the types for all those angular modules.

- **The variables that should be any** - There are a few files in which some variables should be any and thus an type definition for them should not be created for them. For example In *remove-duplicates-in-array.pipe.ts*

  ```
  export class RemoveDuplicatesInArrayPipe implements PipeTransform {
    transform(input: any[]): any[] {
      if (!input.filter) {
        throw Error(
          'Bad input for removeDuplicatesInArray: ' + JSON.stringify(input));
      }
      return input.filter((val: any, pos: number) => {
        return input.indexOf(val) === pos;      Ankita Saxena, a year ago • Fix
      });
    }
  }
  ```

  Here the work of this function is to remove the duplicates from the array regardless of what is in the array. So, a type definition can't be added to such files.

  Just a comment can be added to these files describing that those variables should be any.

- **Possible after angular migration** - There are some angularJs variables whose type can't be declared right now, but can be after the codebase is migrated to angular.

```
// TODO(YashJipkate): Replace 'any' with the exact type. This has been kept as
// 'any' since '$uibModalInstance' is a AngularJS native object and does not
// have a TS interface.
acceptSuggestion($uibModalInstance: any, paramDict: IParamDict): void {
  $uibModalInstance.close(paramDict);
}
```

- **Other files** - I have created a PR for the files that don't fall to any of the above categories i.e. PR(#8823).

Ones in the typings directory

File	Status
typings/custom-jqueryui-defs.d.ts	To be done
typings/custom-scope-defs.d.ts	To be done
typings/custom-window-defs.d.ts	To be done
typings/third-party-defs.d.ts	To be done

There are four files in the typings directory that have any defined, they are

- **typings/third-party-defs.d.ts** - The any from these files will be removed after we add the custom type definitions for the third party libraries as explained in the "Remove all custom typings for typescript" section.

- **typings/custom-window-defs.d.ts** - This file looks like

```
interface Window {
    BlobBuilder?: any;
    CodeMirror?: any;
    Date?: any;
    GLOBALS?: any;
    HTMLElement?: any;
    MSBlobBuilder?: any;
    Math?: any;
    MathJax?: any;
    MozBlobBuilder?: any;
    WebKitBlobBuilder?: any;
    WaveSurfer?: any;
    __fixtures__?: any;
    decodeURIComponent?: any;
    encodeURIComponent?: any;
    opera?: any;
    safari?: any;
    ga?: any;
}
```

As it can be seen that some variables like *BlobBuilder, WaveSurfer* that can be defined as soon as the custom type definitions for them are added in the previous file.

CodeMirror can be defined in the similar way as the third party libraries as it is also a third party library and it's source code can be found in the *static/code-mirror-5.17.0/lib/codemirror.js* file as defined in the *codemirrorRequires.ts* file.

```
const CodeMirror = require('static/code-mirror-5.17.0/lib/codemirror.js');
Object.defineProperty(window, 'CodeMirror', {
  value: CodeMirror,
  writable: false
});
```

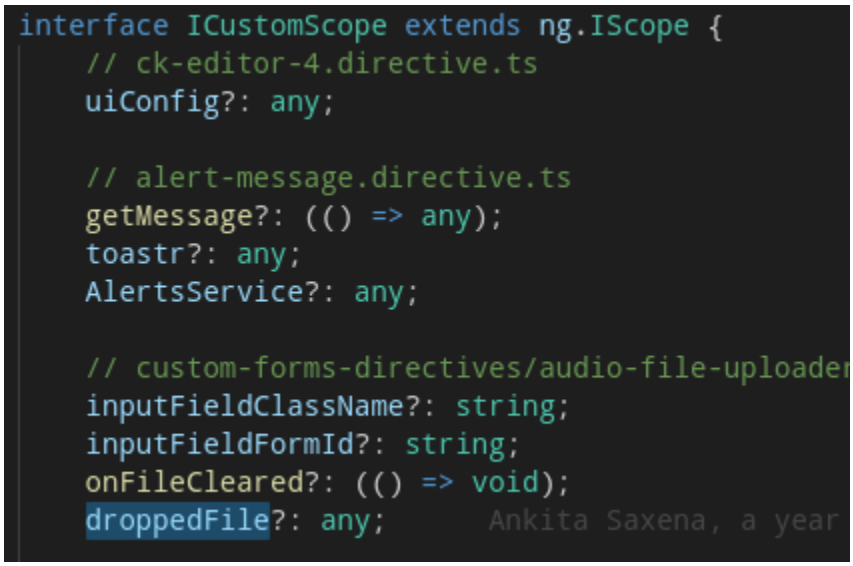
The type definitions for **rest** of the variables can be defined by **tracing** the places where those variables are used. For example the *decodeURIComponent* is used in *signup-page.controller.ts* as follows

```
var defaultDashboard = DASHBOARD_TYPE_LEARNER;
var returnUrl = window.decodeURIComponent(
  UrlService.getUrlParams().return_url);
```

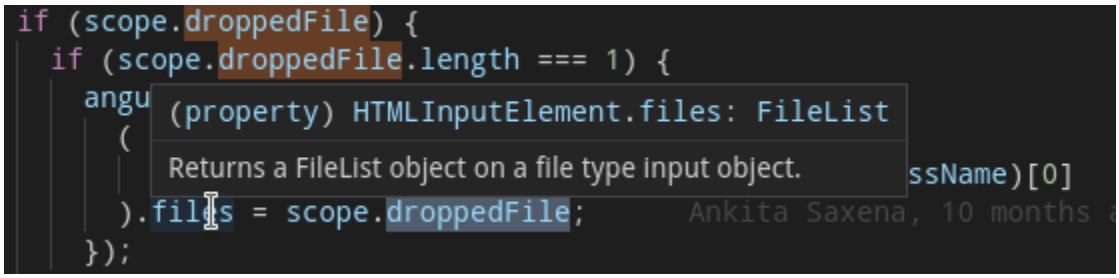
And from here it can be **deduced** that the **decodeURIComponent** is a **function** that takes an encoded url that is a string as a parameter and returns a decoded url which is also a string.

Similarly the types of other variables can be found.

- **typings/custom-scope-defs.d.ts** - There are some variables in this file that are declared as any as seen in the screenshot below



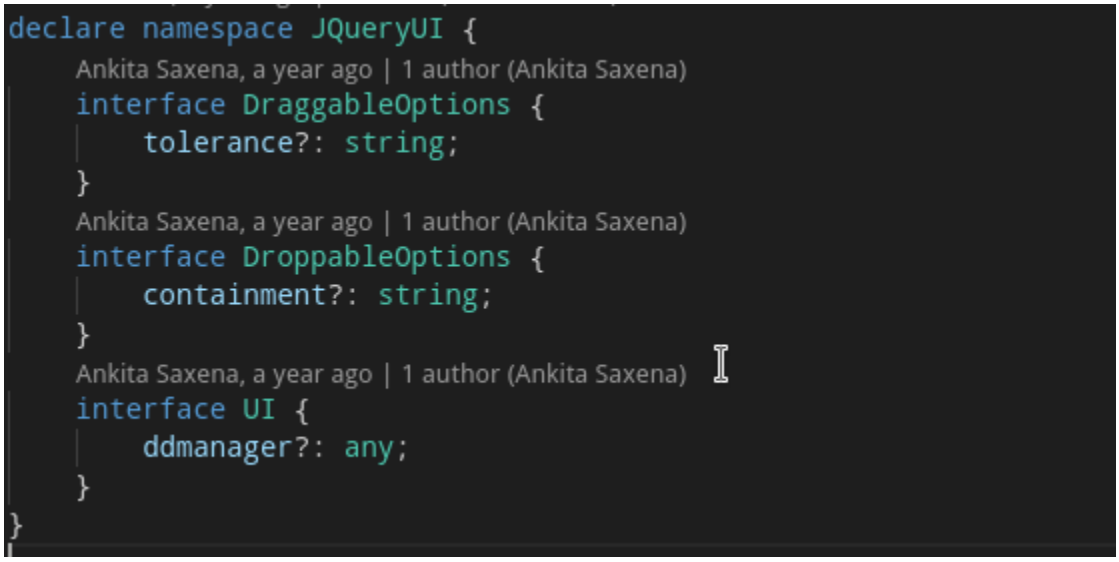
The types of these variables can also be found by tracing the places where they are being used. For ex *droppedFile* in used in *audio-file-uploader.directive.ts* file as follows



As it can be seen we just found the type of droppedFile i.e. FileList.

The types of all the other variables in the file can be found similarly.

- **typings/custom-jqueryui-defs.d.ts** - Currently ddManager is defined as any in this file



  I found the source code for jquery ui and tried to add type definitions for *ddmanager.* After I tried adding an interface for this variable using the type definitions in Definitely Typed I found out those type definitions are obsolete(maybe they were for an older version). So it needs to be **discussed** with the mentors whether we should wait for the authors to update the type definitions or add them ourselves.I plan to add the type definitions for JqueryUI at least for the part we are using in the codebase.

Now any from the codebase would be removed wherever possible. Some places where the any could be present after doing the above steps are

- Third-party js files like third_party.js
- The places where the type should be any as discussed above

Note - It may be possible that some files would be missing in the google sheet. (Some any types introduced afterwards, finally I would make a PR to solve the new files).

As mentioned in the product design, I would also like to introduce a check for the any types in the lint checks itself to ensure that the type of any variable is not declared as any unless it's necessary. So, for that in the pre_commit_linter file,

Firstly we will declare a list of files that need to be excluded from the check at the top of the file

```
+ EXCLUDED_FILES_ANY_TYPE_CHECK = [
+ ]
+
```

Then, we make make a function that checks the any type in the changed files that would look something like

```
def _check_any_type(verbose_mode_enabled):
      """Checks if the type of any variable is declared as any
      in typescript files.
      """

      if verbose_mode_enabled:
      python_utils.PRINT('Starting any type check')
      python_utils.PRINT('----------------------------------------')

      files_to_check = _FILES['.ts']
      patterns_to_match = [r':\ *any', r'^\ *any']

      with _redirect_stdout(_TARGET_STDOUT):
      failed = False
      summary_messages = []

      for file_path in files_to_check:
            if file_path in EXCLUDED_FILES_ANY_TYPE_CHECK:
                  continue

            file_content = FILE_CACHE.read(file_path)
            starts_with_type = False

            for line_number, line in enumerate(file_content.split('\n')):
                  if ((starts_with_type and re.findall(
                        patterns_to_match[1], line)) or re.findall(
                              patterns_to_match[0], line)):
                  failed = True
                  python_utils.PRINT(
                        '%s --> ANY type found in this file. Line no. %s' %
(
                              file_path, line_number + 1))
                  python_utils.PRINT('')

                  if line:
                  starts_with_type = line[len(line) - 1] == ':'
```

```
    if failed:
        summary_message = (
            '%s ANY type check failed' % _MESSAGE_TYPE_FAILED)
    else:
        summary_message = (
            '%s ANY type check passed' % _MESSAGE_TYPE_SUCCESS)

    summary_messages.append(summary_message)
    python_utils.PRINT(summary_message)
    python_utils.PRINT('')

    return summary_messages
```

And finally make some changes to the main function to run this check

```
  def main(args=None):
      """Main method for pre commit linter script that lints Python, JavaScript,
      HTML, and CSS files.
@@ -3131,13 +3175,14 @@ def main(args=None):
          _FILES['.js'], _FILES['.ts'], _FILES['.py'], _FILES['.html'],
          _FILES['.css'], verbose_mode_enabled)
      code_owner_message = _check_codeowner_file(verbose_mode_enabled)
+     any_type_message = _check_any_type(verbose_mode_enabled)
      # Pylint requires to provide paramter "this_bases" and "d", guess due to
      # meta class.
      js_ts_lint_checks_manager = JsTsLintChecksManager( # pylint: disable=no-value
          verbose_mode_enabled)
      other_lint_checks_manager = OtherLintChecksManager(   # pylint: disable=no-va
          verbose_mode_enabled)
-     all_messages = code_owner_message
+     all_messages = code_owner_message + any_type_message
```

The code for these changes can be seen here. After these checks are added they would look something like this

```
Summary of Errors:
----------------------------------------
SUCCESS  CODEOWNERS file check passed

SUCCESS ANY type check passed

SUCCESS  HTML tag and attribute check passed


SUCCESS   HTML linting passed
HTML linting finished.




SUCCESS   1 Python files linted (7.0 secs)
SUCCESS   1 Python files linted for Python 3 compatibility (4.0 secs)


There are no files to be checked.
SUCCESS  Mandatory pattern check passed
SUCCESS  Extra JS files check passed

SUCCESS  JS and TS Component name and count check passed

SUCCESS  Directive scope check passed
```

Also, we can disable the any declaration from the **eslint** by just adding this line at the *.eslintrc* file

```
"@typescript-eslint/no-explicit-any": ["error"]
```

I also plan to **add documentation** on writing type definitions wherever **necessary**. So it would help in the future. As mentioned in the product design section, I plan on adding this documentation as a separate page in Oppia Wiki Page. And also adding a link to that page in the Contributing Code to Oppia page. So, the developers would know about this page. The documentation about the type definitions would need mainly the following three sections

- Adding Type Definitions for Third Party Libraries - This would be similar to the documentation that is already added by me in the typings/README.md file via PR(#8712).
- Adding the type definitions when the backend object has snake_cased

  keys - This would contain the instructions on adding the type definitions when we fetch an object from the backend and it's keys are snake_cased. Basically it will explain how to use the case change service to convert the keys to camelCase.

- Adding type guards when the type can be more than one thing - This will explain how to add custom type guard service when the type can be more than one thing like the one here.

Replace third_party and remaining <script> imports with webpack

Currently we use some script imports in our codebase to import some third party libraries. The goal of this mini project is to load all these third party libs using webpack.

Below are the files that use script imports from the libraries.

File
header_js_libs.html
codemirror.html
guppy.html
math_expressions.html
midijs.html
skulpt.html
pencilcode.html
ui_leaflet.html

Apart from these files there are two libraries that are imported using script tags in almost all the pages. They are -

Libraries
ckeditor
mathjax

Also there are some third party libraries included in the *third_party.js* file. They are -

Included in third_party.js	**Place where the library is used**
angular-animate.js	all modules
angular-resource.js	all modules
angular-sanitize.js	all modules
angular-touch.js	all modules
angular-aria.js	all modules
app/angular.audio.js	all modules
angular-translate.min.js	I18nFooter.ts
angular-translate-loader-static-files.min.js	I18nFooter.ts
angular-drag-and-drop-lists.min.js	(dndLists) all modules
angular-translate-storage-cookie.min.js	I18nFooter.ts
messageformat.js	I18nFooter.ts
angular-translate-interpolation-messageformat.min.js	I18nFooter.ts
dist/angular-toastr.tpls.min.js	all modules
dist/umd/popper.js	all modules
js/bootstrap.js	all modules
angular-translate-loader-partial.min.js	I18nFooter.ts
angular-material.js	all modules
math.js	all modules
ng-joyride.js	all modules
compile/minified/ng-img-crop.js	all modules
ng-infinite-scroll.min.js	all modules
dist/js/select2.full.js	select2-dropdown.directive.ts
ui-bootstrap-tpls-2.5.0.js	all modules

src/sortable.js	all modules
dist/angular-ui-tree.js	all modules
ui-utils.js	all modules
wavesurfer.min.js	audio-translation-bar.directive.ts

Let's talk about the libraries first. Here is an example of ckeditor imported in the *admin-page.mainpage.html*

```
<script src="/third_party/static/ckeditor-4.12.1/ckeditor.js"></script>
```

Now, this dependency can be loaded using webpack by just adding a **require()** or **import** statement. So, now we can add a require statement in the file where the main code of these libraries is implemented instead of importing this library in every page. In case of ckeditor it was ck-editor-4-widgets.initializer.ts and ck-editor-4-rte.directive.ts.

But now we can't just write
`require('/third_party/static/ckeditor-4.12.1/ckeditor.js');`

in each of those files because loading a library with webpack may require some additional lines of code to work. For example, ckeditor required these two lines to work

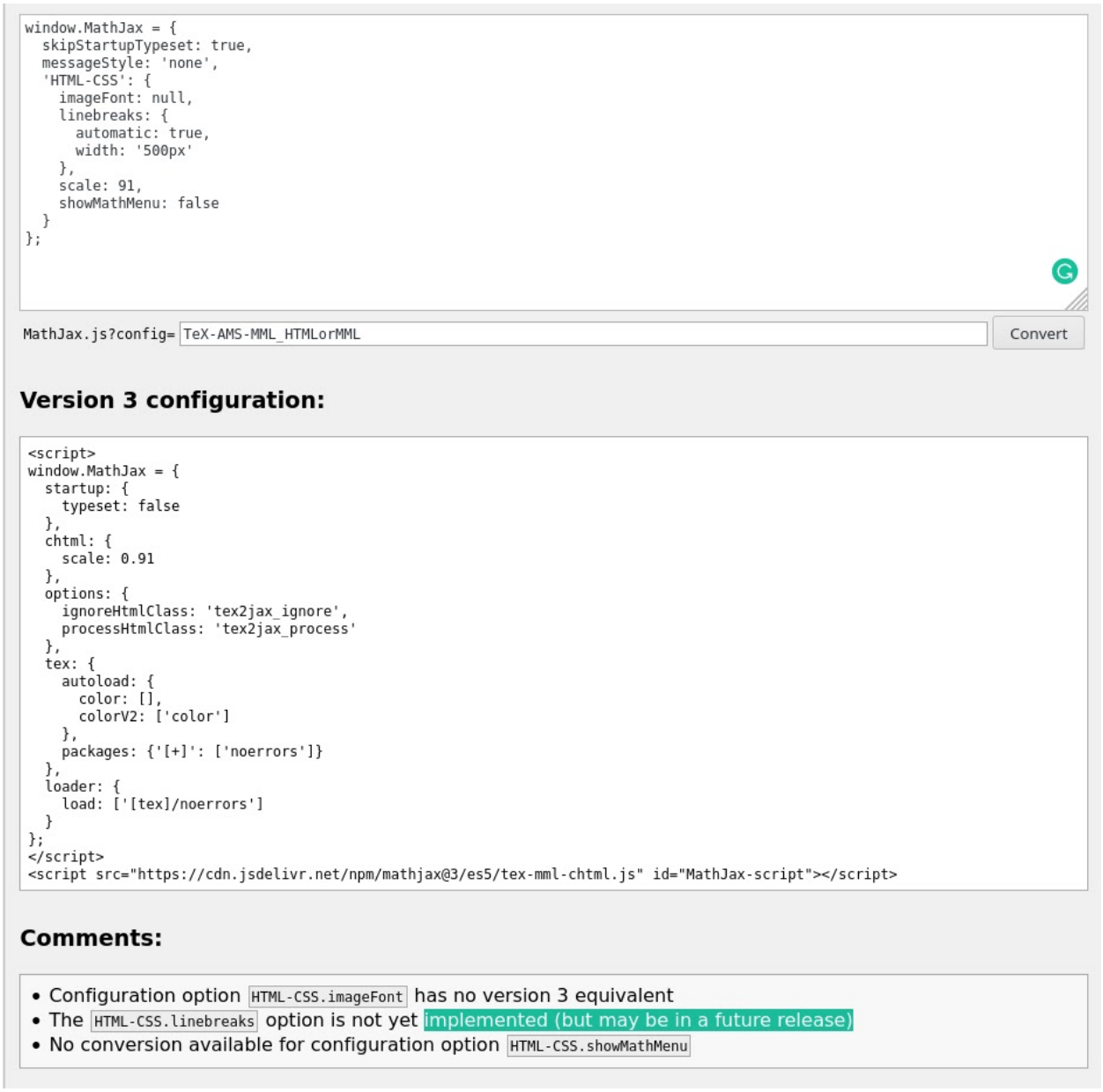
```
window.CKEDITOR_BASEPATH = '/third_party/static/ckeditor-4.12.1/';
require('../../third_party/static/ckeditor-4.12.1/ckeditor.js');
```

So, it might not be a good idea to write this line everywhere. So what we can do is create a file named say *ckeditor.import.ts* that contains the code that would contain that additional config. And that file can be imported in the respective places like

```
require('ckeditor.import.ts');
```

And this concludes our task of loading a library with webpack instead of script imports. I have also created a PR(#8866) for the *ckeditor*.

Now we use the **2.7.5** version of the **mathjax** library which is not compatible with webpack. So, we can just upgrade the version of **MathJax** that we use to **3.0** but there is a problem with that if we put our config in the official v2 to v3 migration tool and read it's documentation, I came to know that there are many features that we use that are not yet available with the v3



So, I think we should wait until the latest version of MathJax is compatible with all the config we use.

Guppy.html, math_expressions.html, skulpt.html

We can make these dependencies to be loaded by webpack similarly as of *ckeditor* library. I.e.

- Creating a file named *library-name.import.ts*, Writing all the necessary config required for loading that file in that file.
- Finding where the library is actually used or implemented.

- Adding a require statement in those files.

Example Code Changes for Guppy

Example Code Changes for MathExpression

Example Code Changes for Skulpt

Codemirror.html
This file contains a script import addon for mergeView addon, There is a file named codemirrorRequires.ts already that contains the webpack requires for the codemirror dependencies. Tis script import can also be shifted to that file as a webpack require.

Ui-leaflet.html
The script imports in this file are unnecessary because there is a file uiLeafletRequires.ts that contains all the libraries for these script imports. Those script imports can be removed simply.

MIDI.html
Loading MIDI using webpack caused the following error

```
Could not resolve 'fs'
```

I found a hacky fix for it but that didn't work out either because it generated more of these similar errors. While searching for a fix, I found out that this library must be kept out of the bundle as changing the config for this library broke other modules. Reference

Pencil Code.html

But the **pencilcode.html** has a script import that loads the file from a **remote** source. So, to include it in the webpack we have to host the library code should be hosted from the oppia codebase itself and then it can be loaded similarly.

.

Now the third party libraries included in the **third_party.js** file can be imported using webpack using the similar steps.

I analysed all the libraries we use in **third_party.js** file and found out we can completely **eliminate** the use of **third_party.js** file. As we can see the third_party libs can be categorized in three parts on the basis of the place they are used

- All modules - These are mainly the libraries that are injected in all the modules like

```
angular.module('oppia', [
  'dndLists', 'headroom', 'infinite-scroll', 'ngAnimate',
  'ngAudio', require('angular-cookies'), 'ngImgCrop', 'ngJoyRide', 'ngMaterial',
  'ngResource', 'ngSanitize', 'ngTouch', 'pascalprecht.translate',
  'toastr', 'ui.bootstrap', 'ui.codemirror', 'ui.sortable', 'ui.tree',
  'ui.validate', 'ui-leaflet', downgradedModule
])
```

  These can be loaded using webpack following the similar steps. For these files we can create a file like *common-libs.import.ts* and import all these libs using a require statement in the App.ts file (as it is imported in all the module.ts files).

- I18nFooter.ts - These are the libraries we use for translation purposes. These can be placed in a separate *translate.import.ts* file and can this file can be imported in the *I18nFooter.ts* file.

- Select2-dropdown.directive.ts and audio-translation-bar.directive.ts - The libraries can be imported in separate files like *select2.import.ts* and *wave-surfer.import.ts* and a require statement can be placed in the respective files where they are used. I also created a PR(#8922) moving the wave-surfer library to webpack.

The code changes for loading these libraries using webpack would look something like this. Now that the generated third_party.js file is empty. We can **remove** all the code related to it in the **build** script.

Now we have a file named **header_js_libs.html** which have some libraries imported using script tags they are

```
<!-- jquery.js, angular.js and jquery-ui.js are removed from bundled js
because
they need to be at the header. Including bundled js at the header will
block
rendering.-->
<script src="/third_party/static/jquery-3.4.1/jquery.min.js"></script>
<script
src="/third_party/static/jqueryui-1.12.1/jquery-ui.min.js"></script>
<script src="/third_party/static/angularjs-1.7.9/angular.min.js"></script>
<script
src="/third_party/static/jquery-ui-touch-punch-0.3.1/jquery.ui.touch-punch
-improved.js"></script>
<script
src="/third_party/static/headroom-js-0.9.4/headroom.min.js"></script>
```

```
<script
src="/third_party/static/headroom-js-0.9.4/angular.headroom.min.js"></scri
pt>
<script
src="/third_party/static/angular-drag-and-drop-lists-2.1.0/angular-drag-an
d-drop-lists.min.js"></script>
```

angular-drag-and-drop-lists.min.js
Here the last script tag can be removed as this library is also included in the **third_party.js** bundle. I have created a PR(#8929) for it.

Headroom.min.js and angular.headroom.min.js
These libraries can be loaded using webpack using similar steps i.e. create a file like *headroom.import.js* with the following contents

```
window.Headroom = require('static/headroom-js-0.9.4/headroom.min.js');
require('static/headroom-js-0.9.4/angular.headroom.min.js');
```

This file can now be imported in the *common-libs.import.ts* file we created above as these libraries are also used by all the modules.
Example Code Changes

angular.min.js
This file can also be loaded using webpack in a slightly different manner. First we create a file like *angular.import.ts* with the following contents

```
module.exports = require('static/angularjs-1.7.9/angular.min.js');
```

Now these line has to be written in top of all the module.ts files

```
import 'angular.import.ts';
```

Example Code Changes

jquery-ui.min.js and jquery.ui.touch-punch-improved.js
These files can be loaded using webpack using similar steps as of third-party-libs that are create a file like *jquery-ui.import.ts* and require() these both files and include these files in *common-libs.import.ts*
Example Code Changes

Jquery
I tried to use webpack ProvidePlugin for importing jQuery. It works to an extent but it introduces new errors because of the third party libraries we use like select2 and jquery-ui etc. So, I think it should be left as it is.

Reduce the overall time for webpack compilation

To reduce the webpack compilation time, first we should know what are the loaders and plugins that consume the time. So I used speed-measure-webpack-plugin for that purpose, We can do that by adding these lines

```
const SpeedMeasurePlugin = require("speed-measure-webpack-plugin");

const smp = new SpeedMeasurePlugin();
```

And replacing

```
module.exports = {
// webpack.config
};
```

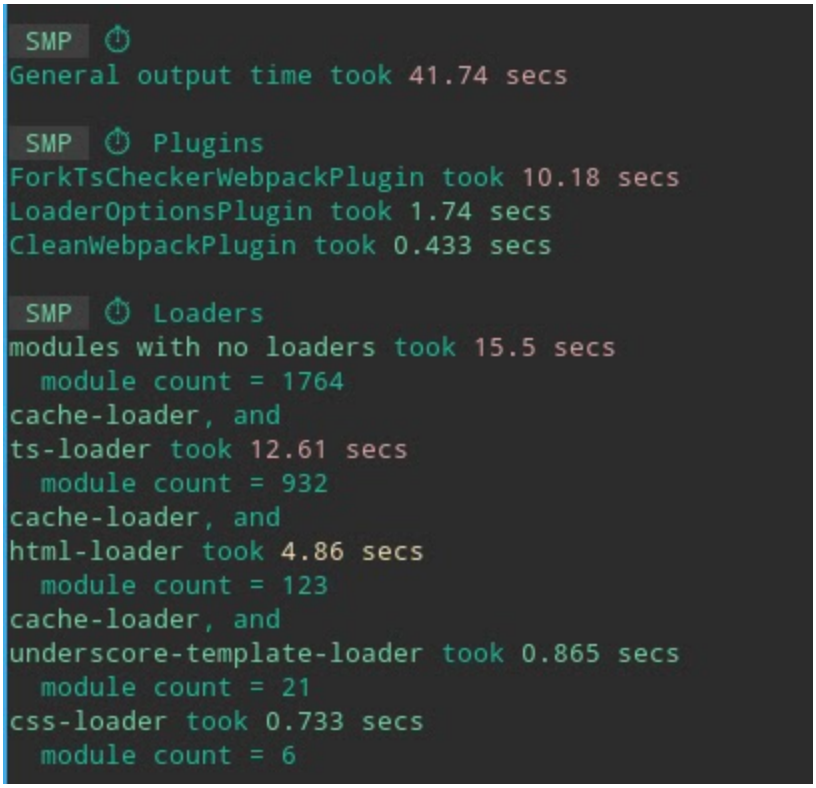
With

```
module.exports = smp.wrap({
// webpack.config
});
```

And running the webpack build using dev config using this command

```
node node_modules/webpack/bin/webpack.js --config webpack.dev.config.ts
```

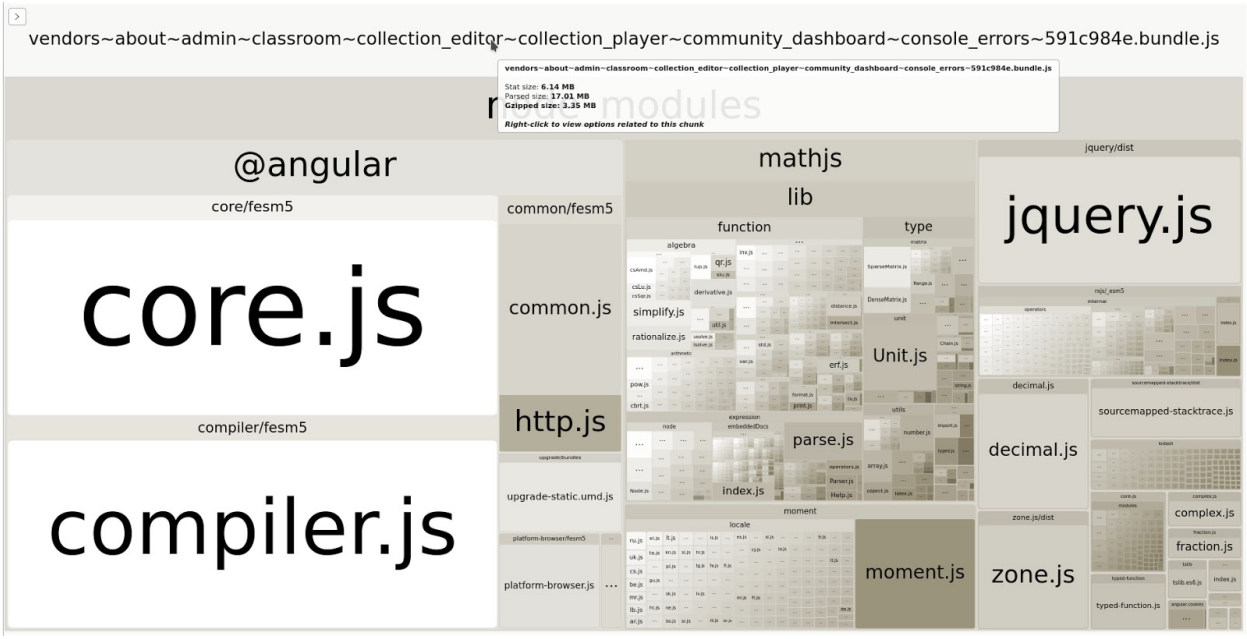
Gave the following output



Actually, the compile time was longer than this, it was reduced due to the use of a cache-loader. Anyways, we can notice that the **ForkTsCheckerWebpackPlugin** is taking a large amount of the compilation time.

Now, the second thing we notice is the *module with no loaders took 15.5 sec*. It is the most time consuming thing in the compilation. These are actually the *js* files, mostly present in the *node_modules* and *third_party* directory. For that I used the webpack-bundle-analyzer, for that purpose.

Here is the output from the plugin



This is the analysis of the most sized bundle after the compilation. As we can see the bundle mainly consists of the angular packages used, and some libraries like zone.js that are important for oppia. We can't remove them from the bundle. So, we can't do much to reduce the time for these modules with no loaders.

Now, we are left with the **ts-loader**, I noticed that the ts-loader is already optimized using the *transpileOnly* flag to true. However, we can optimize it more by using a plugin to divide the work among various threads, like HappyPackPlugin or ThreadLoader.

So, here is a list of things I plan to do to reduce the webpack build time

- **Use cache-loader** - The cache-loader caches the result of the loaders for a faster rebuild time. It can be done adding this block of code before every loader,

```
{
      loader: 'cache-loader'
}
```

- **Remove the use of ForkTsCheckerWebpackPlugin** - As we saw above this type checking plugin consumes a lot of time. So, we have to remove type checking from the webpack compilation for increasing the speed for webpack compilation. We already have CI typescript_checks, so we need not worry about the PR's that have type errors being merged. As far as I know the outcome of the webpacks type checks is not used anywhere. That is a dev is able to push even if the typescript checks fail. That's why I

included a mini project in my proposal to add typescript checks to pre_push hook script.

- **Changing the devTool we use** - Currently we use *inline-source-map* and *source-map* as devTools, if we look at this official documentation on devTools. We see that they are the slowest devTools we use. We can try to use a faster devTool like *eval* or *cheap-module-eval-source-map* for dev and *none* or *cheap-source-map* for production. But with this, we loose some of the control on debugging. But then, again we have to loose something to gain the speed in webpack compilation.

  This can be done by altering the *tsconfig.json* to not generate sourceMaps and changing the value of devTool in *webpack.dev.config.ts* and *webpack.prod.config.ts.*

- **Using the latest version** - It is recommended to use the latest version of webpack. Currently the version 5.0 is in beta. If it is released till that time, I'll upgrade the webpack to the latest version we are using.

- **Using Thread-Loader** - Thread loader can be used with ts-loader to speedup this process. Thread loader distributes the work among various cpu cores and is maintained by the official webpack team. It can be done adding this block of code

```
{
      loader: 'thread-loader'
}
```

These are compile times for different configs I have suggested above. These are recorded on my system and may vary system to system and time to time.

Config	**Dev**	**Prod**
Current	35s	2m 20s
Removing TS Checker	30s	1m 50s
Removing TS Check + (eval) for dev and (none) for prod	22s	28s
Removing TS Check + (eval) for dev and (none) for prod + cache-loader	14s	18s
Removing TS Check + (eval) for dev and (none) for prod + cache-loader + thread-loader	14s	18s
Removing TS Check + (eval-cheap-module-source-map) for	16s	33s

dev and (cheap-source-map) for prod + cache-loader + thread-loader		
Removing TS Check + (eval-cheap-module-source-map) for dev and (source-map) for prod + cache-loader + thread-loader	16s	1m 7s

As we can see the devtools we use currently (*inline-source-map* for dev and *source-map* for prod) and changing them to the fatest devtools brings down the webpack compile time for prod even under 30s. But doing this has a demerit i.e. debuggability is lost.

So, I suggest these devtools as they have a fair balance between debuggability and speed. They are *eval-cheap-module-source-map* for dev and *cheap-source-map* for prod.

Note - As the previous mini project aims to move the script imports to webpack, it **will** increase the webpack build time **drastically**. Therefore, the exact webpack compilation time after completion of this project can't be predicted beforehand.

Add documentation on all typescript and webpack errors and solutions on how to fix them

A part of this project includes adding type definitions for the any types. So, adding the instructions for them regarding the conventions I will follow is very important for the future devs.

As also said above,

I plan to **add documentation** on writing type definitions wherever **necessary**. So it would help in the future. As mentioned in the product design section, I plan on adding this documentation as a separate page in Oppia Wiki Page. And also adding a link to that page in the Contributing Code to Oppia page. So, the developers would know about this page. The documentation about the type definitions would need mainly the following three sections

- Adding Type Definitions for Third Party Libraries - This would be similar to the documentation that is already added by me in the typings/README.md file via PR(#8712).
- Adding the type definitions when the backend object has snake_cased keys - This would contain the instructions on adding the type definitions when we fetch an object from the backend and it's keys are snake_cased. Basically it will explain how to use the case change service to convert the keys to camelCase.
- Adding type guards when the type can be more than one thing - This will explain how to add custom type guard service when the type can be more than one thing like the one here.

Apart from them, I plan to add documentation about some common **typescript** and **webpack** errors and solutions on how to fix them.

- **Typescript** - Here is a list of some of the errors that I have encountered and plan to add a documentation on. Note that this list is **not** the only errors I will add documentation on, I will update the documentation as some dev or me encounter the errors, as I couldn't find a list of all the error codes or something. The documentation would begin with the common steps to **debug** the error. Then how to deal with other specific error messages. Examples -
    - Type 'x' is not assignable to type 'y'.
    - Could not find a declaration file for module 'moduleName'.
    - Overload signature is not compatible with function definition
    - Object literal may only specify known properties
    - external module XYZ cannot be resolved
    - error TS1005: '=>' expected.
    - error TS1068: Unexpected token. A constructor, method, accessor, or property was expected.
    - error TS1128: Declaration or statement expected.
    - Object doesn't have a property 'y'

- **Webpack** - The devs won't encounter much errors from webpack. But we can add a **FAQ** section. That is the goal of this mini project would be to add more content to this page. We can begin by **explaining the complete webpack config** file that would be helpful for the future devs who wish to contribute to the webpack part.

I also plan to **create a google sheet** similar to this sheet for reporting the typescript and webpack errors faced by devs and So, the documentation can be updated accordingly.

Adding typescript checks to the pre_push_hooks

Currently there are no typescript checks in thepre_push hooks. The goal of this mini project is to add the typescript checks in both of this script.

The implementation would be pretty simple,

In *pre_push_hooks* we have a function called *start_python_script* we can use that and add typescript checks by adding a block of code that would look something like this

```
# Typescript checks
typescript_check_status = start_python_script('typescript_checks')
if typescript_check_status != 0:
    sys.exit(1)
```

Milestones

I would like to solve the Remove any types marked under issue #7176, Remove any types marked under issue #7165, Add Custom Type Defs For Third Party Libs, Move some third party libs to webpack in 4, 4, 2, 3 parts respectively. The division into these parts isn't fixed and will be a random distribution.

Milestone 1

Key Objective:

- All third-party libs have type definitions
- All API endpoints have been camelized
- There are less than 20 files in the codebase with 'any' under the issue # 7176

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
1.1	Add Custom Type Defs For Third Party Libs - I		3 Jun	9 Jun
1.2	Add Custom Type Defs For Third Party Libs - II		5 Jun	9 Jun
1.3	Remove any from entire typings directory		10 Jun	14 Jun
1.4	Camelize the response of each endpoint		15 Jun	18 Jun
1.5	Remove any types marked under issue #7176 - I	1.4	19 Jun	21Jun
1.6	Remove any types marked under issue #7176 - II	1.4	23 Jun	25 Jun
1.7	Remove any types marked under issue #7176 - III	1.4	27 Jun	29 Jun

Milestone 2

Key Objective:

- There are no unnecessary occurrences of any in the codebase.
- No PRs that reach the review stage have any defined.
- There is a proper documentation on adding type definitions.

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
2.1	Remove any types marked under issue #7176 - IV	1.4	1 Jul	4 Jul
2.2	Remove any types marked under issue #7165 - I		5 Jul	8 Jul
2.3	Remove any types marked under issue #7165 - II		9 Jul	12 Jul
2.4	Remove any types marked under issue #7165 - III		13 Jul	16 Jul
2.5	Remove any types marked under issue #7165 - IV		17 Jul	20 Jul
2.6	Define all other any types		23 Jul	27 Jul
2.7	Introduce an eslint and lint check for restricting the use of any		25 Jul	27 Jul
	Add Documentation for instructions on adding type defs		27 Jul	NA

Milestone 3

Key Objective:

- There will be no generated *third_party.js* file.
- All *scripts.ts files would be renamed to *import.ts
- There will be no script imports. (Except for 2-3 libs, reason explained above in technical design)
- The webpack compile speed will be reduced to under 20s for dev and under 50s for production from 35s for dev and 2m 20s for production.
- No PRs that reach the review stage have any typescript errors.
- We may be using Webpack 5.0(if it is released in time. Currently released in beta)

- There will be a documentation on dealing with different types of typescript errors.
- There will be a documentation that explains the working of our webpack config.
- There will be a documentation that explains how to deal with some common errors in webpack.

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
3.1	Rename all script.ts files to import.ts		30 Jul	31 Jul
3.2	Move some third party libs to webpack - I (libs in the third_party.js file)		1 Aug	10 Aug
3.3	Move some third party libs to webpack - II (interaction libraries)		4 Aug	10 Aug
3.4	Move some third party libs to webpack - III (Files in header_js.html)		8 Aug	10 Aug
3.5	Add cache loader, thread_loader in webpack		11 Aug	13 Aug
3.6	Change the webpack devTool for faster build		14 Aug	24 Aug
3.7	Remove ForkTsCheckerPlugin from webpack and add typescript checks in pre_push_hook script		16 Aug	24 Aug
3.8	Update the webpack to the latest version.(If released)		20 Aug	24 Aug
	Add documentation on all the typescript errors and webpack and explanation of the webpack config		24 Aug	NA

Optional Sections

Future Work

- Upgrade the mathjax to version 3 after it is compatible with all our requirements. Move the Mathjax to webpack to webpack after that.
- Disabling the declaration of implicit any after the angular migration is complete.

Additional Project-Specific Considerations

Security

No, this feature does not provide any new opportunities for users to gain unauthorized access to user data or otherwise impact other users' experience on the site in a negative way.

Documentation Changes

The project itself has a mini-project called “Add documentation on all typescript and webpack errors and solutions on how to fix them” which includes the documentation changes necessary for this project.