Contributing to Oppia

Joshua Lusk and Scott White, Department of Computer Science

[0] Introduction

FOSS

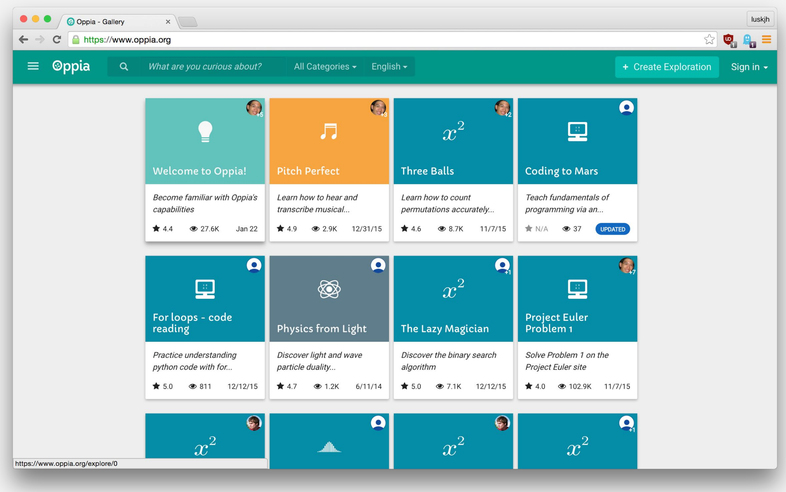
Free and open-source software (FOSS) projects are markedly different in their methods compared to closed-source, proprietary software development. Any individual can submit fixes and improvements to the project. This is made possible by the project's source code being released under an open license; this permits the code to be viewed and modified freely by anyone.

Contributing

Beyond just releasing a project's source code publicly, many FOSS communities conduct their project's development in a public forum as well. The Oppia project uses the Github™ platform for its issue tracking and contributing processes. With this platform, individuals can submit both types of queries: if they find a bug or think of a feature improvement, they can post it to issues; if they program a fix to a bug or a new feature, they can go through the proper channels to become a verified code contributor and work toward getting their new changes accepted into the project's code base.

Our Focus

Within Oppia (see the upper-right section of the poster for information), we targeted issues that affected Oppia's rich-text editor, an online editor that allows users to add (beyond text) links, images, videos, other formatting features (such as tabs), and more.



The live front page of the Oppia project. Oppia is a free and open-source learning platform that allows users to create and partake in 'explorations' - Oppia's tutorials on topics ranging from music to computer science.

[1] Auto-Prepend Protocol to URI

Issue

Within the Oppia exploration editor, when a user attempts to add a hyperlink a warning message is shown if a URI is given, informing the user that a protocol must be prepended to the URI (in oppia, the protocols `http://` or `https://` can be used.). This turns the URI (Uniform Resource *Identifier*) into a URL (Uniform Resource *Locator*).

Proposed New Functionality

It was proposed by one of the project maintainers to instead prepend the `https://` protocol automatically if no protocol was given.

Codified Solution

Our solution to this Issue is presented in *Figure 1* below.

[2] Infinite Loop via encodeURI

Issue

An error is raised when spaces are typed into the hyperlink editor.

Reason

The function callback that is run when a change is made to the link field in the hyperlink editor contains a call to the JavaScript API function `encodeURI`, which escapes many special characters in URIs, such as spaces (changing them to `%20`). By replacing such characters, the URI is changed, which triggers the callback (and thus `encodeURI`) again.

Figure 2 presents more information and our solution to this issue.

[3] Videos Break Out of Container

Issue

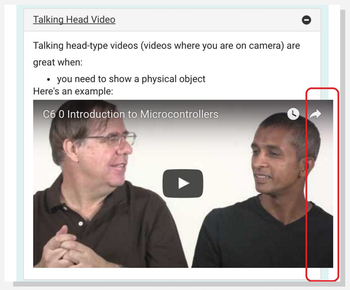
Oppia allows a myriad of different elements in their explorations besides links. These include both videos and accordion containers. When a video is put inside an accordion container, it does not stay within the bounds of the accordion.

Reason

The video container does not dynamically resize based on the container its in.

Codified Solution

Our solution to this Issue is presented in *Figure 3* below.



Videos do not stay within an enclosing accordion container.

Deletions in `extensions/objects/templates/SanitizedUrlEditor.js`

```
-       if (currentValue.indexOf('http://') !== 0 &&
-           currentValue.indexOf('https://') !== 0) {
-         warningsData.addWarning(
-           'Please enter a URL that starts with http:// or https://. ' +
-           'Your changes to the URL were not saved.');
-       } else {
-         if ($scope.active) {
-           $scope.replaceValue(currentValue);
-           // The $scope.$apply() call is needed to propagate the replaced
-           // value.
-           $scope.$apply();
-         }
```

Change in `extensions/rich_text_components/Link/Link.js`

```
          if (untrustedUrl.indexOf('http://') !== 0 &&
              untrustedUrl.indexOf('https://') !== 0) {
-           return;
+           untrustedUrl = 'https://' + untrustedUrl;
          }
```

Change in `extensions/rich_text_components/Link/Link.py`

```
      _customization_arg_specs = [{
          'name': 'url',
          'description': (
-             'The link URL. It must start with http:// or https://'),
+             'The link URL. If no protocol is specified, HTTPS will be used.'),
          'schema': {
              'type': 'custom',
```

Figure 1: *Code solution to Issue `[1]`.*

The call to `encodeURI` was removed from the $watch callback in `extensions/objects/templates/SanitizedUrlEditor.js`...

```
          $scope.$watch('localValue.label', function(newValue) {
-           $scope.$parent.value = encodeURI(newValue);
+           $scope.$parent.value = newValue;
          });
```

...and added to code in `extensions/rich_text_components/Link/Link.js` that is only called once, when the link is being saved.

```
-           var untrustedUrl = oppiaHtmlEscaper.escapedJsonToObj(
-             $attrs.urlWithValue);
+           var untrustedUrl = encodeURI(oppiaHtmlEscaper.escapedJsonToObj(
+             $attrs.urlWithValue));
```

An illustration of the issue:

1. **`http://example.com abc`** is entered into the link field
2. The url is encoded to **`http://example.com%20abc`** (spaces escape to 20)
3. The new url (which was changed by encoding it) is encoded to be **`http://example.com%2520abc`** (percent symbols escape to 25)
4. The new url (which was changed by encoding it) is encoded to be **`http://example.com%252520abc`** (percent symbols escape to 25)
5. The new url (which was changed by encoding it) is encoded to be **`http://example.com%25252520abc`** (percent symbols escape to 25)

Figure 2: *Code solution and illustration for Issue `[2]`.*

Additional styling in `core/templates/dev/head/player/conversation_skin_directive.html`

```
+   .conversation-skin-tutor-card-top-content {
+     width: 100%;
+   }
```

Addition of a wrapper class...

```
-   <iframe width="500" height="280"
-           ng-src="<[videoUrl]>" frameborder="0" allowfullscreen tabindex="<[tabIndexVal]>">
-   </iframe>
+   <div class="oppia-noninteractive-video-iframe-container">
+     <iframe class="oppia-noninteractive-video-iframe" ng-src="<[videoUrl]>"
+             frameborder="0" allowfullscreen tabindex="<[tabIndexVal]>"></iframe>
+   </div>
```

...and styling in `extensions/rich_text_components/Video/Video.html`

```
+ .oppia-noninteractive-video-iframe-container {
+   padding-bottom: 60.9%;
+   position: relative;
+ }
+
+ .oppia-noninteractive-video-iframe {
+   height: 100%;
+   left: 0;
+   position: absolute;
+   top: 0;
+   width: 100%;
+ }
```

Figure 3: *Code solution to Issue `[3]` (note: <style> tags were and comments were left off this representation of the commit).*

[4] Conclusion

Contributing to the Oppia project over the course of this semester has given all team members a greater grasp and appreciation for the open-source contribution and development process.

Takeaways:

1. A standout takeaway from our experience has been how much planning goes into fixing issues before the actual coding starts. Even after the early stages and versions of a project are completed and released, the big picture ideas that inform the organization and rationale behind coding changes are just as relevant in the maintenance stage of the software development process as they are in the analysis and design stages.
2. The open-source development methodology naturally arises as a robust solution for different domain problems because planning and design are so important to keeping the organization and vision of a project consistent through its lifetime - a lifetime that may see many different personel contribute to the project.

College of Charleston
School of Sciences and Mathematics
1770

Team members	Joshua Lusk, Scott White
Class	CSCI 462, Software Engineering Practicum
Professor	Dr. Jim Bowring
Project	Oppia Project http://www.oppia.org
Poster Session Date	Thursday April 14, 2016