# Certificate of Approval

## Summer Internship 2018 Project

### Approval Certificate

Department of Computer Science & Engineering, IIT Bombay

The project entitled "Advanced HTML XBlock for Open EdX platform" submitted by Mr. Ashutosh Sathe, Mr. Srijan Roy Choudhury and Ms. Pravallika Podugu is approved for Summer Internship 2018 programme for 16th May 2018 to 6th July 2018, at Department of Computer Science and Engineering, IIT Bombay

_____

*Prof. (Dr.) D.B.Phatak*

_____

*Dr. Kalpana Kannan*

Place : IIT Bombay, Powai
Date : 6th July, 2018

# Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

*Ashutosh Sathe*

(College of Engineering, Pune)

_____

*Srijan Roy Choudhury*

(NIT, Agartala)

_____

*Pravallika Podugu*

(Kakinada Institute Of Engineering and Technology)

Date : _____

# Abstract

The project focuses on enhancing the open edx html component. Current html component does not allow a fully fledged html, css and javascript content to be added in course. Also the html editor of the current component is not very user-friendly and lacks some fundamental functionalities such as code indentation, code folding etc. Our first approach was to modify Open edX system itself and thus changing the html editor for all users. Later on we decided to move to developing an xblock for this and hence making it optional. We also added extra features to enhance the functionality of xblock. These extra features include live-previewing the html content along with editor.

* * *

# Acknowledgement

We, the summer interns of this group, are overwhelmed in all humbleness and gratefulness to acknowledge our deep gratitude to all those who have helped us put our ideas to perfection and have assigned tasks well above the level of simplicity and into something concrete and unique. We, wholeheartedly thank **Prof. D.B. Phatak** for having faith in us, selecting us to be a part of his valuable project and for constantly motivating us to do better. We thank **Dr. Kalpana Kannan** Senior Project Manager, Content team for providing us with the opportunity to work on this project.We are also very thankful to our mentors **Mr. Hitesh Murkute** and **Mr. Sachin Shirsat** for their valuable suggestions. They were always there to show us the right track when needed help. With help of their brilliant guidance and encouragement, we all were able to complete our tasks properly and were up to the mark in all the tasks assigned. During the process, we got a chance to see the stronger side of our technical and nontechnical aspects and also strengthen our concepts. Last but not the least, we wholeheartedly thank all our other colleagues working on different projects under **Prof. D.B. Phatak** for helping us evolve better with their critical advice.

* * *

# Contents

# *One*

# Introduction

Open edX is a widely used and very popular MOOC platform. The Open edX software is a opensource technology that makes learning easier and studying faster. It was created by MIT and Harvard University, and was quick to gain support of universities such as UC Berkeley, Georgetown, and Stanford. The software platform is designed to engage students and teachers in an interactive, modular way. It promotes active learning by video snippets, interactive components and game-like experience. The course content is presented through learning sequences: a set of interwoven videos, readings, discussions, wikis, collaborative and social media tools, exercises and materials with automatic assessments and instant feedback.

## 1.1 Purpose

"Advanced HTML XBlock for Open edX" is a project undertaken by the Content team at IIT Bombay, Summer Internship 2018, which focusses towards providing additional features and functionalities to the Raw HTML editor component in Open edX in the form of an XBlock. The project's aim is to provide course authors with an Advanced HTML editor XBlock, so that can structure and style their courses better using the advanced and easy to use functionalities of the new editor. Open edX in itself is a huge learning management system with an unparalleled feature list. It is among the top MOOCs platforms in the world. As it is the way of the world, there is nothing perfect here and Open edX is not an exception. All applications/platforms have a scope to enhance its features and abilities of its components. In this project, we worked on enhancing the HTML component which is used for content creation in courses.

## 1.2 Motivation

One of the major features and quality of the edx-platform is the interactivity and the layout of the courses. While current HTML editor does the job well but it has got its own mind and behaves with uncertainty sometimes. This advanced HTML Xblock will boost the user experience of the course creator. Through this advanced HTML XBlock Course authors who prefer editing Raw HTML over using WYSIWYG(What you see is what you get) editors can do so with relative ease. They will be having full control over the code that constitutes the course chapters and section and edit them as they seem relevant. They can even add their custom styling and other third-party style features thus making the course layout more readable and interactive. This will also make the contents interesting for the students to view and learn from.

## 1.3 Scope

This product aims at providing course authors with a full-fledged HTML editor for course content creation. It is an additional feature, which means that anyone who has a background in the fields of HTML, CSS etc can add our XBlock into their course settings and use it as they find relevant. Its functionality is basically in the hands of course content creators who may use it to hard code their entire course and use their own custom styles and attributes.

# *Two*

# Open edX Architecture

## 2.1 What is Open EdX

Open edX is the open source platform software developed by **EdX** and made freely available to other institutions of higher learning that want to make similar offerings. The Open edX project is a web-based platform for creating, delivering, and analysing online courses. It is the software that powers edx.org and many other online education sites.

Main components of Open edX are :

- edX-platform
- Catalog
- Analytics
- Ecommerce
- Notes API

## 2.2 Overview

There are a handful of major components to the Open edX project. These components generally communicate using stable, properly documented APIs.
The centrepiece of the Open edX architecture is the edx-platform, which contains the learning management and course authoring applications (LMS and Studio, respectively). The edx-platform in turn is a very complex service which is supported by a collection of other autonomous web services called independently deployed applications (IDAs).

## 2.3 Key Components

### Learning Management System

The Learning Management System or the LMS is the most visible part of the Open edX project. Learners and students access their courses through the LMS and its effective functionalities makes Open edX a efficient MOOC platform. The LMS also provides an instructor dashboard that users who have the Admin or Staff role can access by selecting Instructor.
The LMS uses a number of data stores. Courses are stored in MongoDB which is a NoSQL Database, with videos served from YouTube or Amazon S3. Per-learner data is stored in MySQL. As learners move through courses and interact with them, events are published to the analytics pipeline for collection, analysis, and reporting.

### Studio

Studio is the course authoring environment. Course teams use it to create and update courses. Studio writes its courses to the same Mongo database that the LMS uses.

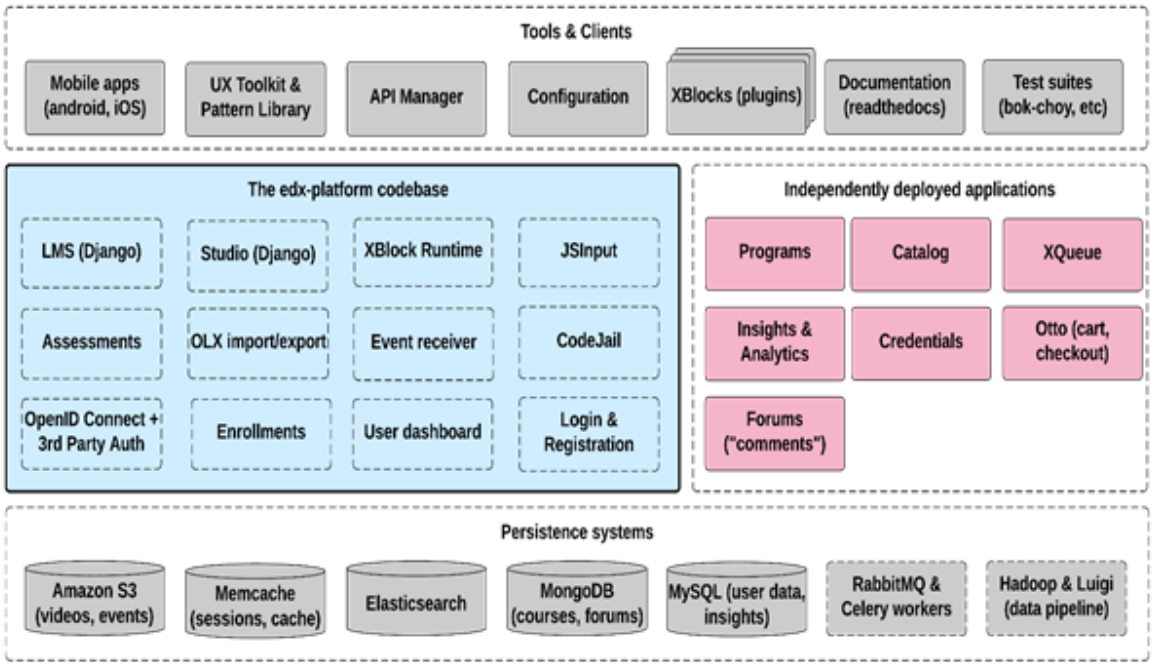


Figure 2.1: Open EdX Architecture

### Discussions

Course discussions are managed by an **IDA** called comments (also called forums) comments is one of the few non-Python components, written in **Ruby** using the **Sinatra** framework. The LMS uses an API provided by the comments service to integrate discussions into the learners' course experience. The comments service includes a notifier process that sends learners notifications about updates in topics of interest.

### Mobile Apps

The Open edX project includes a mobile application, available for iOS and Android, that allows learners to watch course videos and more. EdX is actively enhancing the mobile app.

### Analytics

Events describing learner behavior are captured by the Open edX analytics pipeline. The events are stored as **JSON** in S3, processed using **Hadoop**, and then digested, aggregated results are published to **MySQL**. Results are made available via a **REST API** to Insights, an IDA that instructors and administrators use to explore data that lets them know what their learners are doing and how their courses are being used.

### Background Work

A number of tasks are large enough that they are performed by separate background workers, rather than in the web applications themselves. This work is queued and distributed using **Celery** and **RabbitMQ**. Examples of queued work include:

- Grading entire courses
- Sending bulk emails(with Amazon SES)
- Generating answer distribution reports

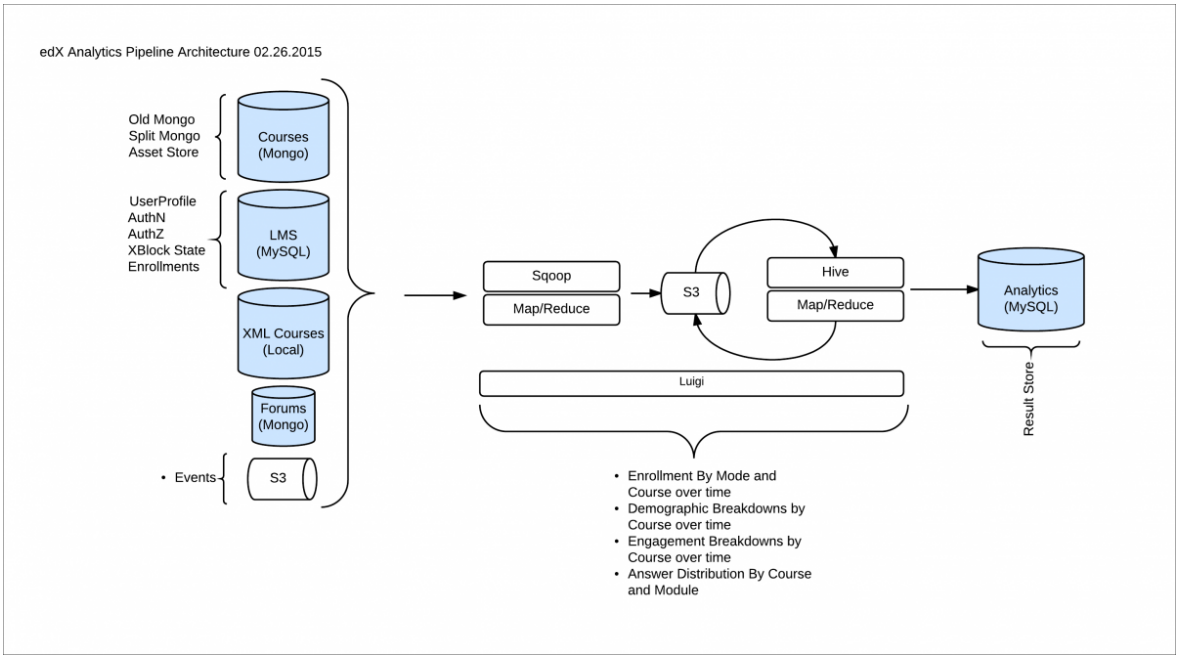


Figure 2.2: Open edX Pipeline

- Producing end-of-course certificates

The Open edX project includes an IDA called **XQueue** that can run custom graders. These are separate processes that run compute-intensive assessments of learners' work.

### Search

The Open edX project uses **ElasticSearch** for searching in multiple contexts, including course search and the comments service.

# *Three*

# XBlocks - The component architecture of Open edX

## 3.1 What is an XBlock ?

The XBlock specification is a component architecture designed to make it easier to create new online educational experiences. An XBlock is the basic building block of any edX course. It controls its own data structure (**model**), its own HTML rendering (**view**), and its own business logic (**controller**). An XBlock executes in a **runtime**, which provides common services/utilities, user/request context, and storage. While the course structure data is primarily stored in a database infrastucture called **modulestore**, an XBlocks **persistence** is configured on a field-by-field basis by designating the scope or each of its **fields**. Some scopes support user-specific data, which results in different XBlock content for different target users.
An XBlock has a unique identifier known as **block-id**. Its instantiation within a context of a course is identified by its **usage-id**. A common class of XBlocks have the same defined **type** (or **category**) that indicate its common behavior and interface. Some examples of XBlock types are: "course_info" (shared by Handouts and Announcements), "problem" (shared by all CAPA-based assessments), "video", "chapter" (a.k.a Section), and "sequential" (a.k.a. Subsection).

## 3.2 XBlock Concepts

XBlocks are build such that course teams use them to create independent course components that work seamlessly with other components in an online course. These include the following

- **XBlock Fields :**
  These are used to store state data for your XBlock.
- **XBlock Methods :**
  These are used in the XBlock Python file to define the behavior of your XBlock.
- **XBlock Fragments :**
  A fragment is a part of a web page returned by an XBlock view method. A fragment typically contains all the resources needed to display the XBlock in a web page, including HTML content, JavaScript, and CSS resources.
- **XBlock children:**
  An XBlock can have child XBlocks.
- **XBlock runtime :**
  An XBlock runtime is the application that hosts XBlock.

## 3.3 XBlock & edX Platform

### EdX Studio

EdX Studio is the application in the edX platform that instructors use to build courseware. Because instructors use Studio to add and configure XBlocks, Studio is also an Xblock runtime application.

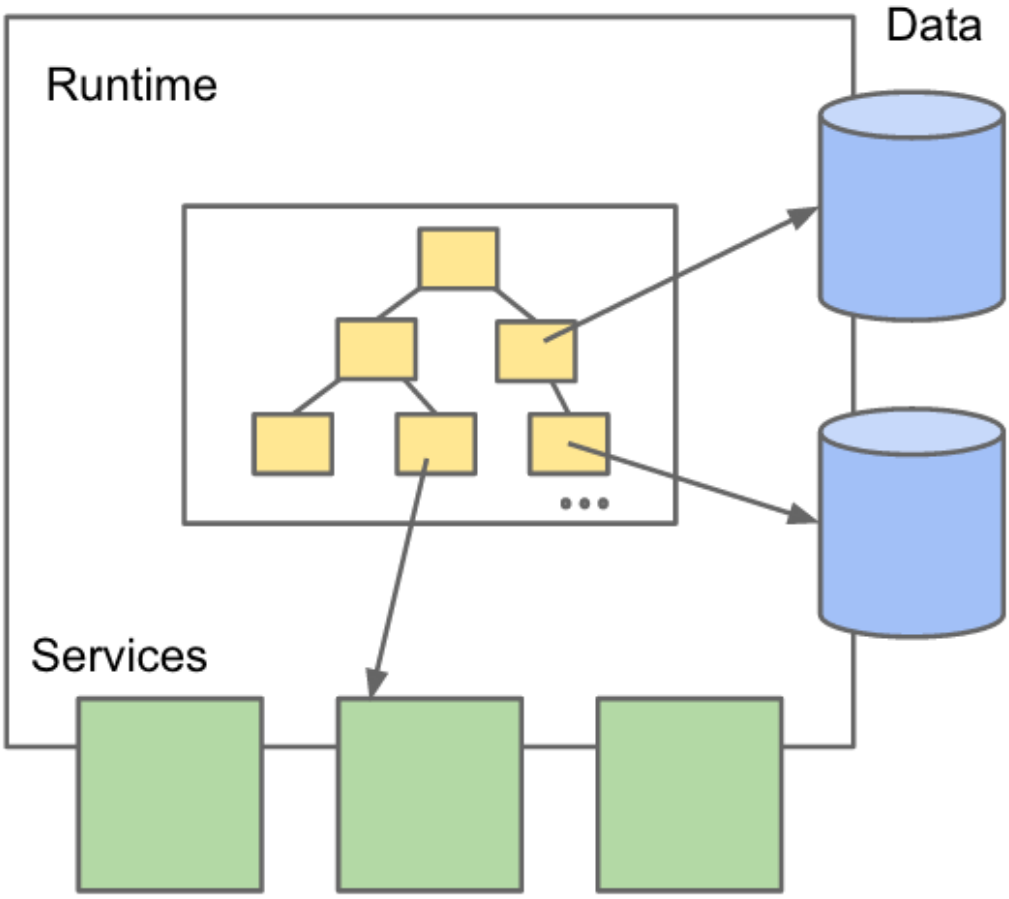


Figure 3.1: XBlock Runtime

### EdX LMS

The EdX Learning Management System (LMS) is the application in the edX Platform that learners use to view and interact with courseware. Because it presents XBlocks to learners and records their interactions, the LMS is also an Xblock runtime application.

## 3.4 XBlock Tree Structure

An XBlock does not refer directly to its children. Instead, the structure of a tree of XBlocks is maintained by the runtime application, and is made available to the XBlock through a runtime service.
This allows the runtime to store, access, and modify the structure of a course without incurring the overhead of the XBlock code itself.
XBlock children are not implicitly available to their parents. The runtime provides the parent XBlock with a list of child XBlock IDs. The child XBlock can then be loaded with the `get_child()` function. Therefore the runtime can defer loading child XBlocks until they are actually required.

## 3.5 Data Access & Persistence

**Data Access** refers to software and activities related to storing, retrieving, or acting on data housed in a database or other repository. Two fundamental types of data access exist:

1. Sequential Access

2. Random Access

**Persistence** refers to object and process characteristics that continue to exist even after the process that created it ceases or the machine it is running on is powered off.

### Data Access and Persistence in edX platform

This part describes how the edx-platform does CRUD (Create, Read, Update, Delete) operations on xblocks and other models and backs those operations with persistence. Currently Open edX uses Django's ORM backed by a SQL DB as the persistence layer for user-relative data. The user history data is moved to a non-SQL db(Mongo db).

### XMLModuleStore

- The original course data is in xml which is stored on a file system
- When the server launches, it scans the file systen and loads every such course into memory
- It then serves the courseware from memory
- The **XMLModuleStore** is the data access layer which finds, loads and serves up the course data to the application. It has only read access. NO create, update NOR delete.

### MongoModuleStore

- When Studio is launched, a read-write storage mechanism and data access layer is needed so, **MongoModuleStore** with CRUD methods is added.
- Here the persistence technology (Mongo) with the DAO CRUD functionality has been convoluted
- **MongoModuleStore** stores each Xmodule as a separate document in a non-SQL (Mongo) database.
- The db connection and pymongo usages are needed to be abstracted out. Hence it uses the xblock's old style location (tag, org, courseid, type, blockid, draft or None) as the key.
- All changes (add, remove, move child) immediately impact the published course.
- **MongoModuleStore** handles inheritance by loading the whole course on each access.

### SplitMongoModuleStore

- It is created to fix the expense and fragility of inheritance.
- It offers full versioning of all edits, the ability to share the same content and settings between courses, as many named branches of a course or named-subcourse structure as one needs.
- Once again the persistence technology (Mongo) with the DAO functionality (CRUD operations on xblocks) is being convoluted.

# *Four*

# HTML Component in Open edX

## 4.1 What is an HTML editor

An **HTML editor** is a program for editing HTML, the markup of a webpage. HTML can be written with any text editor but specialized HTML editors can offer convenience and added functionality. For example, many HTML editors handle not only HTML, but also related technologies such as CSS , XML and JavaScript. In some cases they also manage communication with remote web servers via FTP and WebDAV, and version control system such as Subversion or Git.

## 4.2 HTML component in Open edX

- HTML components are the basic building blocks of the course content.
- These are used for adding and formatting text, links, images and much more.
- One can work with the HTML components in a "visual " or WYSIWYG editor that hides the HTML code details, or in a "raw" editor that is required to mark up the content.

The Open edX platform uses the following three JavaScript editors presently :-

1. TinyMCE
2. CodeMirror
3. WMD

## 4.3 Options for editing HTML components

To work with HTML component, two different editing interfaces can be used.

- **Visual Editor (TinyMCE)**
    1. This interface is almost similar to the interface of "MS Word".
    2. Using the visual editor one can create, edit, add links & images and format content without using HTML markup directly.
    3. The visual editor includes an HTML option for reviewing the HTML markup and can make any changes to the content if we want
- **Raw HTML Editor (CodeMirror)**
    1. It is a text editor and it does not offer a toolbar with formatting options.
    2. This is used to markup content directly with HTML markup.
    3. To include custom formatting or scripts in the course content, a raw HTML editor is needed.

## 4.4 Current scenario of Open edX HTML Editors

### TinyMCE

- Version 4.0.20(from March 2013). Version 4.7.13 is available as of May 2018
- Where is it used :

1. WYSIWYG editing of HTML course content in studio
2. Bulk email in the instructor dashboard

- Plugins used :
  1. "CodeMirror-For-TinyMCE" which allows TinyMCE to interoperate with CodeMirror
  2. "Spell checker"
  3. A studio skin for TinyMCE has also been written.
- Challenges involved in updating :
  1. The installed version of TinyMCE is very old
  2. It is not accessible (Accessible here means accessibility to browser screen readers.)
  3. There are large number of edX specific modifications that will be harder to migrate

### CodeMirror

- Version 3.15(from February 2014). Version 5.38.0 is available as of May 2018
- Where is it used :
  1. Code editing
  2. Raw HTML editor in studio
  3. XML code for authoring studio's advanced editor for problems
  4. JSON in studio's advanced settings
  5. Python (and other languages) code for externally graded assessments in the LMS
- Challenges involved in updating :
  1. The installed version of CodeMirror is very old
  2. It may not be accessible
  3. CodeMirror has issues with right-to-left layouts, and even the most recent version doesn't seem to address it.

### WMD

- Version unknown(edX forked the code in 2012)
- Where is it used ? It is used as markdown editor for discussion posts
- Challenges involed in updating :
  1. WMD/PageDown is a dead project
  2. WMD is not fully accessible
  3. There are a lot of custom edX modifications that will be hard to support going forward

# *Five*

# Getting started

## 5.1 Limitations of the present HTML editor in Open edX

Open edX in itself is a humongous learning management system with unparalleled feature list. It is among the top MOOCs platforms in the world. As it is the way of the world, there is nothing perfect here and Open edX is not an exception. All applications/platforms have a scope to enhance its features and abilities of its components. In this project we worked on enhancing the HTML component which is used for content creation in courses.

While current HTML editor does the job well but it has got its own mind and behaves with uncertainty sometimes. Enhancing the editor with internal CSS and ironing out the indentation issues will boost the user experience of the course creator. Our aim was to develop and integrate a full-fledged HTML editor like Notepad++ or Sublime for Open edX.

Here are some issues that we have encountered in the HTML editor used by the Open edX platform

1. **Indentation in the current HTML editor :-**

   The purpose of code indentation and style is to make the program more readable and understandable. It saves lots of time when a developer revisits the code and uses it. Code indentaion in an important part towards increasing the aesthetic of any editor, as it increases readability and makes understanding of the code easier.

   In Open edX however the code indentation feature is not implemented in the raw HTML editor, which is the CodeMirror v3.21.0. This is mainly because the version of CodeMirror used in Open edX has been implemented with a minimalistic use case and has not been updated since October 2015. The problem is that even if we manually indent our code, once we save our progress and close the raw editor, the indentation is lost i.e it doesn't gets saved.

2. **Code Folding in the current HTML editor :-**

   Code folding is a feature of some text editors, source code editors, and IDEs that allows the user to selectively hide and display – "fold" – sections of a currently edited file as a part of routine edit operations. This allows the user to manage large amounts of text while viewing only those subsections of the text that are specifically relevant at any given time.

   This is a pretty useful feature for any editor and presents the user with various use patterns, primarily organizing code or hiding less useful information so one can focus on more important information. Since the CodeMirror version implemented in Open edX is very old this feature is not available and thus was added to our List of possible enhancements.

3. **Internal CSS in HTML Code :-**

   Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML CSS is a cornerstone technology of the World Wide Web alongside HTML and JavaScript.

   The three main types of CSS styles are inline, internal and external. Currently in Open edX platform in the raw HTML editor only inline or embedded CSS support is available. Inline

styles are CSS styles that are applied directly in the page's HTML. Because inline styles they are the most specific in the cascade, they can over-ride things we didn't intend them to. They also negate one of the most powerful aspects of CSS - the ability to style lots and lots of web pages from one central CSS file to make future updates and style changes much easier to manage.

If we had to only use inline styles, our documents would quickly become bloated and very hard to maintain. This is because inline styles must be applied to every element we want them on. That is why internal CSS is preferred whenever exhaustive use of CSS is required as opposed to inline CSS. Internal CSS is Open edX raw HTML editor is not supported as internal CSS must be defined inside the style tag in the head element of the html body. However the raw HTML editor of Open edX stips down style tags and the effects are not reflected as wanted. Styles of other components in the web page also gets changed which we do not want. Thus allowing internal CSS is an important enhancement criterion which should be addressed.

To understand the limitations better we have provided the screenshots as mentioned below:

**[See Figures 1-4 in the "Figures and Screenshots" section.]**

## 5.2 Objectives

Here are the list of enhancements we were aiming for in the html component of Open edX:

1. Improvise the raw HTML editor.
   Expected features:
   - Pretty indentation
   - Code folding
   - Auto completion of tags
2. Check if we can add a separate CSS section in raw HTML editor to enable working of internal CSS.
3. Fix the cursor placement issue.
4. Enable inernal JavaScript.
5. Adding support for all HTML tags.

## 5.3 Observations

Regarding the Open edX paltform and the editors used in the Open edX one might come up with the following kind of questions

- Is Open edX using a raw editor or visual editor or both ?
- Did Open edX write their own editor ? If yes, then how is it working ? If no, then from where they have implemented their text editors
- What kind of editor are they using ?
- Where do we find the editors source code ?
- How many editors are they using ?
- Their method of working ?
- Their current scenario ?
- On what basis is this editor working along with the Open edX platform ?

To answer all these queries, here are some facts that would be helpful:

1. Open edX uses both the visual editor as well as the raw editor.
2. Open edX has not written their own text editor.
3. The visual editor is tinyMCE editor and the raw editor is CodeMirror Editor.
4. These editors are located in edx source code, in github.
5. CodeMirror is made to work in the custom mode configuration. Open edX is using a old version of CodeMirror.
6. The tinymce editor folder of Open edX consists of a plugin called "codemirror-for-tinymce".
7. The good thing about this plugin is that any version of codemirror can be used in the code and it will work perfectly fine.
8. The "codemirror-for-tinymce" plugin is already in use in the Open edX repository.
9. We can use the "codemirror-for-tinymce" plugin for tinymce 4.x.x versions.
10. There is an xblock module or simply an xblock called html-module.py
11. This particular xblock module handles the operation of the html editor.
12. This is the module that loads the html editor widget and configures both (tinymce and codemirror) editors to work hand in hand with the help of "codemirror" plugin for tinymce.

## 5.4 Our plan of action

Depending on the facts that we uncovered regarding the html editors we had come up with two possible approaches initially, that would fulfil our required objectives.

1. **Approach I**
   Updating the present version of WYSIWYG and raw HTML editors in Open edX
2. **Approach II**
   Integrating other third party open source text editors into Open edX platform.

# *Six*

# Approach I - Local integration

## 6.1 Updating the current version of Editors present in Open edX

Our first approach was to download the latest versions of TinyMCE, CodeMirror and the plugin "codemirror-for-tinymce" from their respective git repositories into our local Machine.
Next step was to integrate the two editors i.e., CodeMirror and TinyMCE using the plugin and test it on our local machine, and later integrate it with Open edX using one of the Open edX developing environment.
The result would be TinyMCE loading the instance of latest version of CodeMirror as its raw HTML editor. However this implementation could only solve the indentation and code folding problem, but still the internal css problem wouldn't be solved just with this.

For enabling the internal CSS we thought of two possible implementations:

1. **Using iframes**
   For this, we were planning to wrap the entire html content inside an iframe and then add theming to the iframe.

   - Set the width of iframe to 100%
   - Set the height of iframe equal to the height of content of iframe
   - Set the border of iframe to zero/none
   - Open all hyperlinks within the iframe in new tab

   Hence the expected result is an invisible iframe holding our HTML content.

2. **Using JavaScript**
   This is a slightly tough and critical approach, which was to apply CSS using JavaScript. The plan was to separate out CSS part from HTML content and wrap the entire content in a html div tag with an unique id. Next we would have to manually select each and every child of the div by parsing the DOM (Document Object Model) tree and apply styles to it.

We opted for the first i.e the iframe approach because of the following reasons:

1. This method allows the course content creator to include third party CSS such as bootstrap, w3css which will make the course even more beautiful
2. Implementaion is easier
3. Wrapping data inside the iframes containerizes the data ensuring no mixing of CSS between two HTML blocks in LMS.
4. The overhead of the second approach will be very large, since parsing the DOM tree would take considerable amount of time and effort.

## 6.2 Implementation of our Local integration

In the first phase of implementing our approach of updating the present editors, we first downloaded both the editors onto our local machine and then integrated the latest version of CodeMirror and TinyMCE in our local machine. Later we integrated it with Django framework since Open edX uses Django framework in the back end as their Web Application Framework. This step was done to give

us a better insight into the interaction of the present editors in Open edX with Django. The steps of our implementation are mentioned in brief below:

1. We had downloaded the latest version of CodeMirror and TinyMCE from their github repositores and then tested them individually to check whether all the desired features are working or not.

2. We downloaded the latest version of the "codemirror-for-tinymce" plugin from its git repo.

3. Next we had replaced the default raw HTML editor of TinyMCE with our instance of the latest version of CodeMirror. This was done by configuring the **init-tinymce.js** file to use CodeMirror as an external plugin and thus load it as its default raw HTML editor.

4. Later, for enabling the internal CSS we had encapsulated the entire html content of the editor into an iframe.

5. We have then set the width of the iframe to 100% and set the height of the iframe equal to the height of the html content.

6. We had made sure that the borders of iframe was set to zero to make it invisible so that the user experience is better and smooth.

7. After completing our stand alone integration we merged it with a Django project to run our editor as an app in the Django server. This integration was quite useful as Open edX itself uses Django framework in its backend and it helped us in understanding about the Django back end of Open edX.

## 6.3 Summary of our local integration

After completing our local integration, firstly we had tested our local instances of TinyMCE and CodeMirror on the Django server. A brief summary of our observations are listed below.

1. Most of the HTML tags and their atrributes are working fine. Some of the exceptions being script tag , audio tag etc.

2. Almost every CSS style and its attributes works fine. The exceptions are those style features which by default require a script tag to be implemented.

3. CSS animations can be implemented and other third party HTML-and CSS-based design templates such as Bootstrap can be linked and used without any error.

4. We prepared a detailed Test Report which can be found here:
[https://docs.google.com/spreadsheets/d/1AJQWfAG2NIjpxFZt2fnLPlKdhP2-q35KZ0DOmdb-F0w/edit#gid=0]

5. Source code is available in our github repository here:
[https://github.com/ashutoshbsathe/codemirror-in-tinymce]

## 6.4 Issues faced and solutions

1. **AJAX:**

   We had no idea about how to pass data between RESTful APIs at first. Then later we found out that this can be done via a request called as AJAX which basically passes the data in JSON format from one API to another API. And hence we used the same. On a button click in our html, we would send the AJAX query from webpage to Django backend. While this seems easy at first, it is a bit tricky when we are using only a single text area and not entire form element. So the main problem was our csrf_token was not being set correctly since we were not using a form. So we decided to completely skip the csrf authentication by adding @csrf_exempt decorator to our view

2. **Stripping of style tags:**

   So tinyMCE does not allow you to directly add CSS in your html. It will constantly filter out your own `<style>` and `<link>` tags out of it's display content. The only way to get around this is by setting valid_children, valid_elements and extended_valid_elements properly in TinyMCE config.

3. **Indentation in the "CodeMirror with TinyMCE setup" :**

   Now because the tinymce keeps filtering out content, there's no actual way to show a nice indented code in CodeMirror. For this we found a script that will indent our html code nicely and we called it before setting the content in CodeMirror.
   **Note :** Whatever indentation that was done in CodeMirror was a pseudo indentation and not true indentation as the course creator wants. The code is surely nicely indented, but if somewhere content creator specifically wanted a specific tag on multiple line such as

   ```
   <a href="https://www.google.com">
   A SearchEngine
   </a>
   ```

   It won't be rendered in that way because the script treats anchor tag as a single-line tag and it will be rendered as follows:

   ```
   <a href="https://www.google.com">ASearchEngine</a>
   ```

   Sadly enough, there is no way for tinyMCE to shut down it's content filtering and hence this issue remains unsolved.

4. **Unnecessary modification of html code:**

   This issue specifically occurs when you try to edit something that was added in raw mode. For example, let's hope you add a nice CSS animation and then return back to tinyMCE for saving the content, while returning back, you accidently clicked in the div of your animation and you pressed enter. Then tinymce will try to split your div in half breaking it where you pressed enter, and will also add some bad html code which you definitely don't want. Again this issue is unavoidable because of content filtering algorithms used in tinyMCE.

# *Seven*

# Research about Approach II

## 7.1 Integration of other third party editors

Just like CodeMirror and TinyMCE there are other third party open source text editors which meet all the prerequisites of our enhancement criteria. Some of them are Notepad++, Quill, Brackets, ContentTools, Grapejs,etc. As an alternative to our first approach we thought of another possibility of reaching our objective. Just like how Open edX has merged two text editors for their HTML editor, similarly even we can try to integrate WYSWYG editor using the same methodology. In this approach we develop an entire new editor itself. Moreover since all these editors are open source their github repositories is easily accessible. So we also thought of trying to import the logic they used in their source code to add features such as code indentation ,code folding etc. We put this approach as our backup for our approach I because, approach II is a bit tough and complicated to implement compared to approach I

## 7.2 What kind of editors do we want?

Before integrating here are some of the questions that we asked ourselves.

- What kind of editor do we want ?
- What features should the editor have ?
- How does these editors help us with our motive ?

So here are few points that helped us clear our doubts:

- We need an editor that makes our work easy, simple, and has got much better working properties as compared to the editor that is currently being used in the Open edX platform.
- The editor should have inbuilt features(code folding, pretty indentation,languages,etc) that could just wipe out the issues that are being faced by the Open edX html editor.
- Since our motive is to add features such as indendation, cold folding, internal css,etc, so considering an editor that has these features inbuilt, and integrating it by making some required changes might help us in achieving our motive, and would even resolve the issues faced by the current HTML editor that is being used by Open edX platform.

After a bit of research work what we found out is some pretty good and well built editors that support html language. All these editors are open source and all their source code is available in their respective git repositories. The list of editors that we had researched on are as follows:

- Notepad++
- Brackets
- ContentTools

The detailed explanation about each editor follows.

## 7.3 Notepad++

### About

Notepad++ is a free (as in "free speech" and also as in "free beer") open source code editor and Notepad replacement that supports several languages. Running in the operating system its use is governed by the GPL license.

Current stable verison of Notepad++ is v7.5.6.

### Features

1. Syntax Highlightning
2. Syntax Folding
3. User Desfined Syntax Highligting and Syntax Folding. (images (n++1,2,3,4))
4. PCRE (Perl Compatible Regular Expreession) Search and Replace.
5. GUI entirely customizable.
6. Auto completion: Word completion, Function completion and Function parameter hint.
7. Multi document (Tab interface).
8. Multi view
9. WYSIWYG (printing).
10. Zoom in and Zoom out.
11. Multi language environment supported.
12. Macro recording and playback.
13. Bookmark.

### Observations Made

Here are a list of things that we have found about notepad++ :

- Scintilla is the main component of Notepad++ which is very powerful.
- This editor is written in c++
- All the syntax styling part is done using the Scintilla component.
- Scintilla edits and even is used for debugging the source code of the Notepad++
- Some files where the required changes are to be made and importing those files so they can be used in building an editor for Open edX platform.

### Idea Of Implementation

In a similar way in which the Open edX platform uses an xmodule(html_module.py) that handles the operation of its current html editor, similarly, using the same working principle we can replace our editor and load the html editor and configure it for proper working.

## 7.4 Adobe Brackets

### About

With focused visual tools and preprocessor support, Brackets is a modern text editor that makes it easy to design in the browser. It's crafted from the ground up for web designers and front-end developers.
Brackets is an open source editor written in HTML, CSS, and JavaScript with a primary focus on web development. It was created by Adobe Systems, licensed under the MIT License, and is currently maintained on GitHub by Adobe and other open-sourced developers. Brackets is available for cross-platform download on Mac, Windows, and is compatible with most linux distros. The main purpose of brackets is its live html, css and js editing functionality.

### Features

1. Quick Edit
2. Quick Docs
3. Live element debugging
4. Live preview
5. Inline Editors
6. Split View
7. Thesus Integration
8. LESS support
9. W3C Validation
10. Drag and Drop
11. Auto Prefixer
12. Git Integration for Brackets
13. JS Lint

### Observations Made

Here are a list of things that we have found out about Brackets:

- Brackets is written in JavaScript
- Its an automated WYSIWYG editor
- Mainly designed for web developers and front end developers
- Applies quick edit for HTML elements which helps in displaying corresponding CSS properties for that particular element.
- When a color is typed it shows an inline color picker for our better convienience in searching.
- While writing the code it enables live preview , that works only for Google Chrome, and if there is any html syntaxtical error then it forbiddens opening of the live prview.
- Using this Thesus integration feature that enables inspecting an element in the real time and debug any extension in brackets.
- This uses CodeMirror text raw html editor for Code Formatting.
- Brackets has got more than 20 Dependencies .
- It consists of a bracket (which is a root module) that pulls in other modules as dependencies.
- Some files where the required changes are to be made and importing those files so they can be used in building an editor for Open edX platform

### Idea Of Implementation

In a similar way in which the Open edX platform uses an xmodule(html_module.py) that handles the operation of its current html editor, similarly, using the same working principle we can replace our editor and load the html editor and configure it for proper working.

## 7.5 ContentTools

### About

ContentTools is a JavaScript/CoffeeScript library aimed at building WYSIWYG editors for HTML content. ContentTools aims to provide both a fully-functional editor that can be used out of box and a toolkit of classes, a set of tools for performing common editing tasks, and a history stack for managing undo/redo. Whilst the components provided by the toolkit work well together, they can also be used or replaced as required.

### Features

1. Easily integrated with any HTML document.
2. Drag and Drop directly within the page.
3. Floating context-sensitive toolbar.
4. Compatible with all major web browsers and operating systems.
5. The javaScript library can transform any HTML page into an WYSIWYG editor.
6. Countless possibilities for building wondrous interactive apps and services.
7. Media Resizing.

### Observations Made

- Allows text content, images, embedded videos, tables and other page content to be edited, resized, or moved using the Drag and Drop property within the page itself.
- Requires bower or npm for installation.
- For building library's and project, requires grunt node modules and SASS.
- Uses an HTML string for formatting rather than using an HTML parser written in JavaScript.
- Its library uses a minimal finite state machine(FSM) for JavaScript.
- It consists of an JS library that provides cross-browser support for content selection.
- It has a javascript library that provides a set of classes for building content editable HTML elements.
- The ContentTools editor has already been implemented into the content management systems of a number of websites.
- This editor is a recent editor which has got wonderful, simplified and an unique UserInterface.
- Its Framework integration includes "Django REST Framework" and "Image Uploads with Cloudinary".
- ContentTools is part of a collection of JavaScript libraries (ContentTools, ContentEdit, ContentSelect, FSM, HTMLParser) which were developed to aid in the creation of HTML WYSIWYG editors.
- It has an ability to configure styles for your content.
- Contenttools can be easily integrated into the CMS.

- It is an opensource web-based HTML editor whose working is quite similar to CodeMirror and TinyMCE, so integrating with the Open edX platform would be quite simpler as all of then belong to the same working methodology.

### Idea Of Implementation

In a similar way in which the Open edX platform uses an xmodule(html_module.py) that handles the operation of its current html editor, similarly, using the same working principle we can replace our editor and load the html editor and configure it for proper working.

Integrating this would be much easier compared to Brackets and Notepad++ as its working methodology is just as similar to the CodeMirror and TinyMCE which are being used as of now in the Open edX platform.

### Installation Issues

Installing Notepad++ and Brackets easy was error free but with ContentTools the only issue that occurred while installing was related to the npm package installer.
The packages were not completely installed and hence npm threw an error while installing.

Here is the log file that shows the errors that would be occurring ,
[https://drive.google.com/open?id=1KUe3jcAPg8DpxmhshPUBYPKya_B-oRSt]

**SOLUTION :** To resolve this issue here are the things that are to be followed:

1. Open your terminal and perform the following operations

```
sudo apt-get update synaptic
```

2. A dailog box opensSearch for npm and node modules

3. Delete those folders manually

4. Save the changes and close the dialog.

5. Open the terminal and type in the following commands

```
npm install --save ContentTools
```

## 7.6 Advantages Of Approach I Compared To Approach II

1. Upgrading something is always better than integrating something new altogether. As per our research and observation the versions of CodeMirror and TinyMCE in Open edX are outdated. Since the latest versions of these two editors fullfill our requirements updating the editors is much straightforward and sensible than incorporating two completely new ediitors into the Open edX platform, since it can lead to a lot of dependency issues.

2. The iframe approach also allows the course content creator to include third party CSS such as bootstrap, w3css which will make the course even more interactive and boost student experience. Course creators can style their courses with a variety of such third party CSS features and make it a lot more interactive for the students.

3. Wrapping data inside the iframes containerizes the data ensuring no mixing of CSS between two HTML blocks in LMS. This is very important in order to provide an error free user experience.

## 7.7 References for approach II

The following links would be helpful in learning more about the working of the editors discussed previoulsy

1. **Notepad++**
    - https://github.com/notepad-plus-plus/notepad-plus-plus
    - https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/lexlib/Accessor.cxx
    - https://github.com/notepad-plus-plus/notepad-plus-plus/blob/master/scintilla/src/AutoComplete.cxx
    - https://github.com/notepad-plus-plus/notepad-plus-plus/tree/master/scintilla/lexers
    - https://github.com/notepad-plus-plus/notepad-plus-plus/tree/master/scintilla/lexlib
2. **Adobe Brackets**
    - https://github.com/adobe/brackets
    - https://github.com/adobe/brackets/blob/master/src/styles/brackets_codemirror_override.less
    - https://github.com/adobe/brackets/tree/master/src/LiveDevelopment
    - https://github.com/adobe/brackets/blob/master/src/LiveDevelopment/Agents/CSSAgent.js
    - https://github.com/adobe/brackets/blob/master/src/editor/
    - https://github.com/adobe/brackets/tree/master/src/htmlContent
3. **ContentTools**
    - https://github.com/GetmeUK/ContentTools
    - https://github.com/Cotidia/django-contenttools-demo
    - http://getcontenttools.com/api/content-select
    - http://getcontenttools.com/tutorials/content-tools-plus-cms

# *Eight*

# Setting up a development environment for Open edX platform

## 8.1 Open edX installation options

In order to test our local integation what we needed was a running instance of Open edX platform on our local machine. Open edX has provided us with different installation options depending on our use. They are described in brief below.

- **Devstack:** useful if we want to modify the Open edX code. The code will be in directories shared between the host and the guest operating systems. We should run the code in the guest, but can edit it in the host, where we can use our own tools.

  1. For Ginkgo and earlier, devstack was a Vagrant installation.
  2. For Hawthorn, devstack is based on Docker. An unsupported pre-release is available.

- If we want to run a production installation and want a production like environemnt on our local machne then we can use **Native** or **Manual** installation. The Native installation installs the Open edX software on our own Ubuntu 16.04 machine in a production-like configuration.

- If we want a production-like installation for testing, we can use **Fullstack** or **Native**. Fullstack is a Vagrant instance designed for installing all Open edX services on a single server in a production-like configuration. Fullstack is a pre-packaged Native installation running in a Vagrant virtual machine.

Since our project deals with making changes in Open edX source code we naturally opted for Devstack, which is designed for developers

## 8.2 Open edX Devstack

Devstack is traditionally a Vagrant instance designed for local development. Devstack has the same system requirements as Fullstack. This allows us to discover and fix system configuration issues early in development. Devstack simplifies certain production settings to make development more convenient. For example, nginx and gunicron are disabled in devstack; devstack uses Django's runserver instead in conformation with Open edX platform.
Devstack is maintained as a set of docker containers lately and hence we chose that way as well

## 8.3 Installing a docker based devstack

### Prerequisites

- This installation requires Docker 17.06+ CE. It is recommended to use docker stable, but docker edge should work as well

- GNU-Make

- pip

**Note :**
Linux users should not be using the `overlay` storage driver. `overlay2` is tested and supported, but requires kernel version 4.0+. We can check which storage driver our `docker-daemon` is configured to

use through following command :

```
docker info | grep -i 'storage driver'
```

## 8.4 Installing docker and docker compose

Please note that, docker based devstack needs a fairly powerful computer. Our computer was using quad core core i5 7500u CPU with 8 GB RAM. Also, give more space to / partition because docker needs more space for storing images.If you already have configured docker on your distro, you can skip this section. For following this guide, we recommend using any debian based distro where you would be able to use `apt` package manager.

### Getting docker for ubuntu

Before we install Docker CE for the first time on a new host machine, we need to set up the Docker repository. Afterward, we can install and update Docker from the repository. The steps are mentioned below :

- Update the `apt` package index:

```
$ sudo apt-get update
```

- Install the packages to allow `apt` to use a repository over **HTTPS**

```
$ sudo apt-get install apt-transport-https \

ca-certificates curl software-properties-common
```

- Add docker's official GPG key :

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

  Verify that you now have the key with the fingerprint `9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88` by searching the last 8 characters of the fingerprint

```
$ sudo apt-key fingerprint 0EBFCD88
```

```
>> pub 4096R/0EBFCD88 2017-02-22
Key fingerprint = 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88
uid Docker Release (CE deb) <docker@docker.com>
sub 4096R/F273FCD8 2017-02-22
```

- Use the following command to set up the **stable** repository. You always need the **stable** repository, even if you want to install builds from the **edge** or **test** repositories as well. To add the edge or test repository, add the word **edge** or **test** (or both) after the word `stable` in the commands below.

```
$ sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \ stable"
```

### Installing Docker CE

1. Update the `apt` package index:

```
$ sudo apt-get update
```

2. Install the latest version of Docker CE

```
$ sudo apt-get install docker-ce
```

3. Docker should now be installed, the daemon started, and the process enabled to start on boot. To check that its running, we can use the following command

```
$ sudo systemctl status docker
```

### Post installation steps for ubuntu

1. **Manage docker as a non root user**
   The docker daemon binds to a Unix socket instead of a TCP port. By default that Unix socket is owned by the user root and other users can only access it using sudo. The docker daemon always runs as the root user. If we don't want to use sudo when you use the docker command, create a Unix group called docker and add users to it. When the docker daemon starts, it makes the ownership of the Unix socket read/writable by the docker group. To create the docker group and add our user:

   a) Create the docker group

   ```
   $ sudo groupadd docker
   ```

   b) Add our user to the docker group

   ```
   $ sudo usermod -aG docker $USER
   ```

   c) Log out and log back in so that the group membership is re-evaluated. If testing on a virtual machine, it may be necessary to restart the virtual machine for changes to take effect.
   On a desktop Linux environment such as X Windows, we need to log out of our session completely and then log back in.

   d) Verify that you can run docker commands without sudo.

   ```
   $ docker run hello-world
   ```

   This command downloads a test image and runs it in a coontainer. When the container runs, it prints an informational message and exits.

2. **Configure Docker to start on boot**
   Most current Linux distributions (RHEL, CentOS, Fedora, Ubuntu 16.04 and higher) use `systemd` to manage which services start when the system boots. Ubuntu 14.10 and below use `upstart`.

   a) `systemd`

   ```
   $ sudo systemctl enable docker
   ```

   To disable this behavior, use `disable` instead

   ```
   $ sudo systemctl disable docker
   ```

b) `upstart`
Docker is automatically configured to start on boot using `upstart`. To disable this behavior, use the following command

```
$ echo manual | sudo tee /etc/init/docker/override
```

`chkconfig`

```
$ sudo chkconfig docker on
```

### Getting Docker compose for ubuntu

On **Linux**, you can download the Docker Compose binary from the Compose repository page on git hub. These step by step instructions are also included below.

1. Run this command to download latest version of docker compose:

   ```
   sudo curl -L
   https://github.com/docker/compose/releases/download/1.21.2/docker-compose-
   $(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
   ```

   We should use the lates compose release number in download command. The above command is an example and it may become out-of-date. To ensure you have the latest version, check the compose repository on release page on GitHub https://github.com/docker/compose/releases If you have problems installing with `curl`, you can find alternative options in docker documentation.

2. Apply the executable permissions to the binary

   ```
   $ sudo chmod +x /usr/local/bin/docker-compose
   ```

3. Test the installation

   ```
   $ docker-compose {version
   >>docker-compose version 1.21.2, build 1719ceb
   ```

## 8.5 Installing docker based devstack

1. Clone the git repository of devstack

   ```
   $ git clone https://github.com/edx/devstack && cd devstack
   ```

2. Install the requirements inside of a python virtualenv

   ```
   $ make requirements
   ```

3. The Docker Compose file mounts a host volume for each service's executing code. The host directory defaults to be a sibling of this directory. For example, if this repo is cloned to `~/workspace/devstack`, host volumes will be expected in `~/workspace/coursediscovery`, `~/workspace/ecommerce`, etc. These repos can be cloned with the command below.

   ```
   make dev.clone
   ```

   We may customize where the local repositories are found by setting the `DEVSTACK_WORKSPACE` environment variable

4. We must Run the provision command, if we haven't already, to configure the various services with superusers (for development without the auth service) and tenants (for multitenancy). When running the provision command, databases for ecommerce and edxapp will be dropped and recreated. The username and password for the superusers are both edx. We can access the services directly via Django admin at the `/admin/` path, or login via single sign-on at `/login/`.

   Default: `make dev.provision`

5. Start the services. This command will mount the repositories under the `DEVSTACK_WORKSPACE` directory. It may take up to 60 seconds for the LMS to start, even after the `make dev.up` command outputs done.

   Default: `make dev.up`

Please note that all the above commands are to be executed in `devstack` directory and `DEVSTACK_WORKSPACE` MUST be exported correctly throughout the process.

After the services have started, if we need shell access to one of the services, we can run

```
make <service>-shell
```

```
make discovery-shell
```

To see logs from containers running in detached mode, we can either use "Kitematic" (available from the "Docker for Mac" menu), or by running the following:

```
make logs
```

The screenshots of the running instance of our Devstack along with the LMS and CMS section are provided as mentioned below:

**[See Figures 5-8 in the "Figures and Screenshots" section.]**

### Service URLs

Each service is accessible at `localhost` on a specific port. The table below provides links to the homepage of each service. Since some services are not meant to be user-facing, the "homepage" may be the API root

| Service | URLs |
|---|---|
| Credentials | http://localhost:18150/api/v2/ |
| Catalog/Discovery | http://localhost:18381/api-docs/ |
| E-commerce/Otto | http://localhost:18130/dashboard/ |
| LMS | http://localhost:18000/ |
| Notes/edx-notes-api | http://localhost:18150/api/v1/ |
| Studio/CMS | http://localhost:18010 |

## 8.6 Issues and solutions

During the course of our Docker based installation and configuration of Open edX Devstack, we faced multiple issues such as proxy settings etc. The issues are mentioned below along with the solutions we used to overcome them.

1. **Proxy configuration of the system :**

   - Initially we tried installing our instance of Open edX Devstack behind a proxy server. But many packages mainly the ones which were being installed using pip were failing.

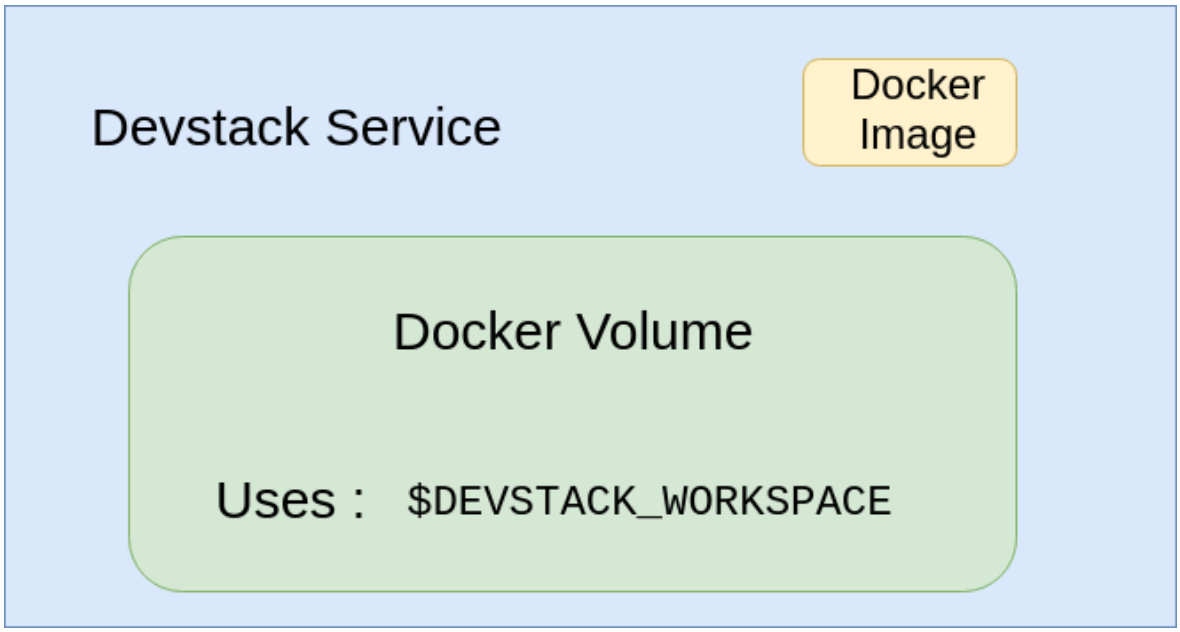


Figure 8.1: Block diagram of devstack service

- We thought of trying to enable proxy inside the docker container, but due to the large amount of time it took to resolve several layers of proxy authentication the reuqest was getting timed out.
- Thus, the docker container was unable to fetch the required packages from Open edX.

**Solution :**After trying several times, we decided to discard the installation using a proxy. We used normal mobile data to install all the require services and get our Devstack up and running. With the help of mobile data the installation was completed error free and all the packages were downloaded correctly.

2. **Issue with docker installation :**
   Docker isntallation can create some problems if the steps are not followed correctly and in order. After installing Docker it is important to perform the post installation steps , because it happens sometimes that the system is unable to connect to the docker daemon.

3. **Issues compiling static assets :**
   This is a package related issue. This can happen with any of the service randomly. The issue here is the script that is building static assets for a particular service, needs only a specific version of npm package "babel-loader". Sometimes durirng the provisioning, it updates the package "babel-loader" via npm. We could not exactly find out when the update was happening, it was random most of the times. If you ever encounter issue with newer "babel-loader", just modify the webpack to explicitly tell the "babel-loader" to build JSX as ReactNative assets. Also it might give error about Object Spread, all you need to do it to add plugin into your webpack
   The modified webpack can be found on our git
   Also check out :

   https://stackoverflow.com/questions/33460420/babel-loader-jsx-syntaxerror-unexpected-token
   &
   https://babeljs.io/docs/en/babel-plugin-transform-object-rest-spread

4. **Docker environment issues :**
   As we can see in the diagram, every devstack service is made up of docker image and docker volume. Docker images are pulled from openedx repos whereas volumes are mounted on your host device. While mounting these docker volumes, script uses an environment variable called `DEVSTACK_WORKSPACE` as shown in figure. It is highly probable that you set this variable once and when retrying we forget to set this variable again. If we donot set this variable correctly

the mount silently fails and mounts a new volume instead. This is why it is necessary to set this variable everytime you login into your host machine. For doing so, add the

```
"export DEVSTACK_WORKSPACE=/path/to/your/workspace"
```

line at the bottom of your `.bashrc.` After doing "`source .bashrc`" will export the variable correctly. To see whether the variable is set correctly or not, type "`printenv`"
Important note for "`sudo`". If you are doing any command with sudo, make sure to preserve the current environment variables. For doing so, use `-E` flag with sudo to preserve them.

# *Nine*

# Approach I - Open edX integration

After testing our hybrid local integration and setting up our development environment for Open edX, the next phase was to integrate it with the running instance of Open edX on our Devstack.

## 9.1 Plan of action

The idea and plan behind integrating our running hybrid instance of CodeMirror in TinyMCE with Open edX is mentioned below:

1. **Indentation and code folding features**
   - Add the latest version of CodeMirror v5.38.0 to the source code of Open edX leaving the older version untouched.
   - Configure TinyMCE to open the updated version of CodeMirror as its raw HTML editor.
   - Test the updated CodeMirror in TinyMCE and see if it's working as expected.
2. **Enabling internal CSS**
   - Understand the working of the XModule html_module.py which is the widget which loads the WYSIWYG editor in Open edX.
   - Learn how the html content is getting stored in the MongoDB .
   - The idea was to wrap the html content inside an inframe before it gets stored in the Database. However this was to be done only when the course authors used internal CSS in their courses.
   - Similarly, while loading the content in LMS, it would be wrapped inside an iframe first and then rendered in order to produce the desired effect.

## 9.2 Implementation

### Configuring TinyMCE to use updated CodeMirror v5.38.0

- The CodeMirror and TinyMCE source files are located in the edx-platfrom, can be found on the following path : ***edx-platform/common/statis/js/vendor***
- Inside the tinymce source code, we placed the updated version of CodeMirror in the plugins folder for TinyMCE.
  The path was : ***edx-platform/common/static/js/vendor/tinymce/js/tinymce/plugins***
- Next we configured TinyMCE to use the updated CodeMirror as its raw HTML editor. This was done through making changes in the TinyMCE configuration file **edit.js** located here : ***edx-platform/common/lib/xmodule/xmodule/js/src/html/edit.js***

After performing the following steps we were able to configure TinyMCE to open our updated instance of CodeMirror v5.38.0 as its raw HTML editor succesfully. Features such as indentation, code-folding etc were working fine.

### Using iframes to enable internal CSS

#### Issues faced

Once we were able to integrate updated version of CodeMirror with TinyMCE, we tried testing internal CSS in our new editor. Our observations are mentioned below:

- The effect of the internal CSS is only visible inside the WYSIWYG editor i.e inside TinyMCE.
- Once we save our progress the Open edX backend automatically strips the style tag and removes it. As a result of this, no effect of the CSS is visible in the actual course content.

## 9.3 Change in Approach

### What happened to the issues ?

We tried to look into the issues by looking into Open edX backend by logging at several places in several different xmodules and we failed. We discussed this issue with Aparna Ma'am(Mentor, IITBombayX team). She said that we were on a right track and we should explore the backend through logs itself. Strangely enough, we were unable to see any logs generated. This lead us to a roadblock situation where we needed to switch to another approach for checking out the backend. However, all of the ways went at least once through logging step which was not working on our devstack very well. We also discovered that we were unable to see logs in the R&D server(10.129.27.44) which was provided to us.

### Need for change in approach

Because of these issues, we were facing a major roadblock ahead of us and hence we needed to switch approaches. Now we had two more options, Xblock or integrating entirely a new editor. Integrating a new editor in Open edX platform would also require accesing the log files frequently which weren't able to do. Another option was to reinstall devstack and see what causes devstack to not generate logs, but the clock was ticking and we only hand two and a half weeks left for the internship period. We could have pursued approach I to the end and maybe end the internship without completing the project but we chose to change approach.

### Why Xblock approach and not integrating editors such as contenttools/grapejs ?

Two main reasons:

1. If we wanted to integrate any of these with Open edX, we would need the logger to be working perfectly. As described in the issues above, that is what kept us from pursuing approach I itself
2. Xblocks are modular. If we decide to modify the system, we're forcing the editor changes to all users. Some of these users don't need this advanced editor and features. On the other hand, if we make this into an xblock, it will be used by people who actually need it and know how to use it. A hidden advantage of xblock is that they are largely platform independent. Most of the xblocks do not need to be updated on every single version of Open edX since the core Xblock API remains unchanged.

# *Ten*

# Basics of developing an XBlock

## 10.1 Introduction

XBlock is a way to create rich engaging courses. XBlock is a component architecture of edX, and a course can be build from the different Blocks as pieces. On the other hand, XBlock will offer APIs and structure for other modules in the edX to comunicate with the course content since all other web applications in a online ecosystem needs to access the course content.
XBlock is the software development kit for edX-MOOC and it is written in python2. XBlock generally contains few python classes which create a small web application. So, to create new xblocks, a new module called xblock-sdk has been open sourced. Creating a new xblock means creating a simple python installable python package with a class derived from the XBlock.

## 10.2 XBlock API and Runtimes

Any web application can be an XBlock runtime by implementing the XBlock API. The XBlock API is not a RESTful API. XBlock runtimes can compose web pages out of XBlocks that were developed by programmers who do not need to know anything about the other components that a web page might be using or displaying.

## 10.3 XBlocks for Developers

Developers can select from functionality developed by the Open edX community by installing an XBlock on their instance of Open edX. Developers can integrate new or propriety functionality for use in XBlock runtimes by developing a new XBlock using the supported XBlock API.

XBlocks are like miniature web applications: they maintain state in a storage layer, render themselves through views, and process user actions through handlers. XBlocks differ from web applications in that they render only a small piece of a complete web page. Like HTML `<div>` tags, XBlocks can represent components as small as a paragraph of text, a video, or a multiple choice input field, or as large as a section, a chapter, or an entire course.

## 10.4 Criteria

One must design a XBlock to meet two criteria:-

- The XBlock must be independent of other XBlocks. Course teams must be able to use the XBlock without using other Xblocks.
- The XBlock must work together with other XBlocks. Course teams must be able to combine different XBlocks in flexible ways.

## 10.5 Building an Xblock

### Instaling XBlock Prerequisites

To build an XBlock, we must have the following tools on our computer.

- Python 2.7
- Git

- A Virtual Environment

#### Python 2.7

To run the a virtual environment and the XBlock SDK, and to build an XBlock, we must have Python 2.7 installed on our computer.

#### Git

EdX repositories, including XBlock and the XBlock SDK, are stored on GitHub. To build our own XBlock, and to deploy it later, we must use Git for source control.

#### A Virtual Environment

It is recommended that we develop our XBlock using a Python virtual environment. A virtual environment is a tool to keep the dependencies required by different projects in separate places. With a virtual environment we can manage the requirements of our XBlock in a separate location so they do not conflict with requirements of other Python applications we might need.

### Setting Up the XBlock Software Development Kit

After installing all prerequisites, we are ready to set up the XBlock SDK in a virtual environment. To do this, we must complete the following steps.

- Create a Directory for XBlock Work
- Create and Activate the Virtual Environment
- Clone the XBlock Software Development Kit

#### Creating a Directory for XBlock Work

It is recommended that we create a directory in to store all our XBlock work, including a virtual environment, the XBlock SDK, and the XBlocks we develop.

1. At the command prompt,we can run the following command to create the directory.

```
$ mkdir xblock_development
```

2. Change directories to the xblock_development directory.

```
$ cd xblock_development
```

The rest of our work will be from this directory.

#### Creating and activating an Virtual Environment

We must have a virtual environment tool installed on our computer. Then we can create the virtual environment in our xblock_development directory.

1. At the command prompt in xblock_development,we can run the following command to create the virtual environment.

```
$ virtualenv venv
```

2. Next Run the following command to activate the virtual environment.

```
$ source venv/bin/activate
```

3. When the virtual environment is activated, the command prompt shows the name of the virtual directory in parentheses.

```
(venv) $
```

### Cloning the XBlock Software Development kit

The XBlock SDK is a Python application that helps us build new XBlocks. The XBlock SDK contains three main components:

1. An XBlock creation tool that builds the skeleton of a new XBlock.
2. An XBlock runtime for viewing and testing our XBlocks during development.
3. Sample XBlocks that we can use as the starting point for new XBlocks, and for our own learning.

After we create and activate the virtual environment, we must clone the XBlock SDK and install its requirements. To do this,we must complete the following steps at a command prompt.

1. In the xblock_development directory,we must run the following command to clone the XBlock SDK repository from GitHub.

```
(venv) $ git clone https://github.com/edx/xblock-sdk.git
```

2. Next Run the following command to change to the xblock-sdk directory.

```
(venv) $ cd xblock-sdk
```

3. Run the following command to install the XBlock SDK requirements.

```
(venv) $ pip install -r requirements/base.txt
```

4. Run the following command to return to the xblock_development directory, where we will perform the rest of our work.

```
(venv) $ cd ..
```

When the requirements are installed, we are in the xblock_development directory, which contains the venv and xblock-sdk subdirectories. We can now create our Xblock.

## Creating an Xblock

Before we can continue,we must make sure that we have set up the XBlock SDK. We then create the XBlock and deploy it in the XBlock SDK.

- Create an XBlock
- Install the XBlock
- Create the SQLite Database
- Run the XBlock SDK Server

### Creating an XBlock

We use the XBlock SDK to create skeleton files for an XBlock. To do this,we can follow these steps at a command prompt.

1. Change to the xblock_development directory, which contains the venv and xblocksdk subdirectories.
2. Run the following command to create the skeleton files for the XBlock.

```
(venv) $ xblock-sdk/bin/workbench-make-xblock
```

3. At the command prompt, we must enter the Short Name we selected for our XBlock.

```
$ Short name: myxblock
```

4. At the command prompt,we must enter the Class name we selected for our XBlock.

```
$ Class name: MyXBlock
```

The skeleton files for the XBlock are created in the myxblock directory

#### Installing the Xblock

After we create the XBlock, we can install it in the XBlock SDK. In the xblock_development directory,we can use pip to install our XBlock.

```
(venv) $ pip install -e myxblock
```

We can then test our XBlock in the XBlock SDK

#### Running the XBlock SDK Server

To see the web interface of the XBlock SDK, we must run the SDK server. In the xblock_development directory, we can run the following command to start the server.

```
(venv) $ python xblock-sdk/manage.py runserver
```

#### Getting Help for the XBlock SDK Server

To get help for the XBlock SDK runserver command, we can run the following command.

```
(venv) $ python xblock-sdk/manage.py help
```

The command window lists and describes the available commands.

## 10.6 Customizing our XBlock

1. **Define Fields:**
   The first step in this process is defining field, so, the data of these field types will be stored by the xblock.

2. **Define Views:**
   The Views are needed to create HTML pages to display blocks in the course page. It is simple to render html pages but its difficult to show what we want on the course page. The view also takes the help of CSS and JavaScript to help the html pages.

3. **Define Handlers:**
   When course web pages are interactive, we will get events from the Javascript to handle. So, handler is a function to respond to a specific URL. Here we can use the URL to communicate with the server too.

## 10.7 Testing and Installing Created XBlock

### For Workbench

We can test a xblock, while running with edX or workbench. We can install the xblock using following commands and start as a local server, and open the browser to find the link for new xblock

1. Go to the root directory of new created xblock.

2. Type in the following commands

```
pip install -e name of xblock
```

3. run the workbench as a local server using command python manage.py runserver

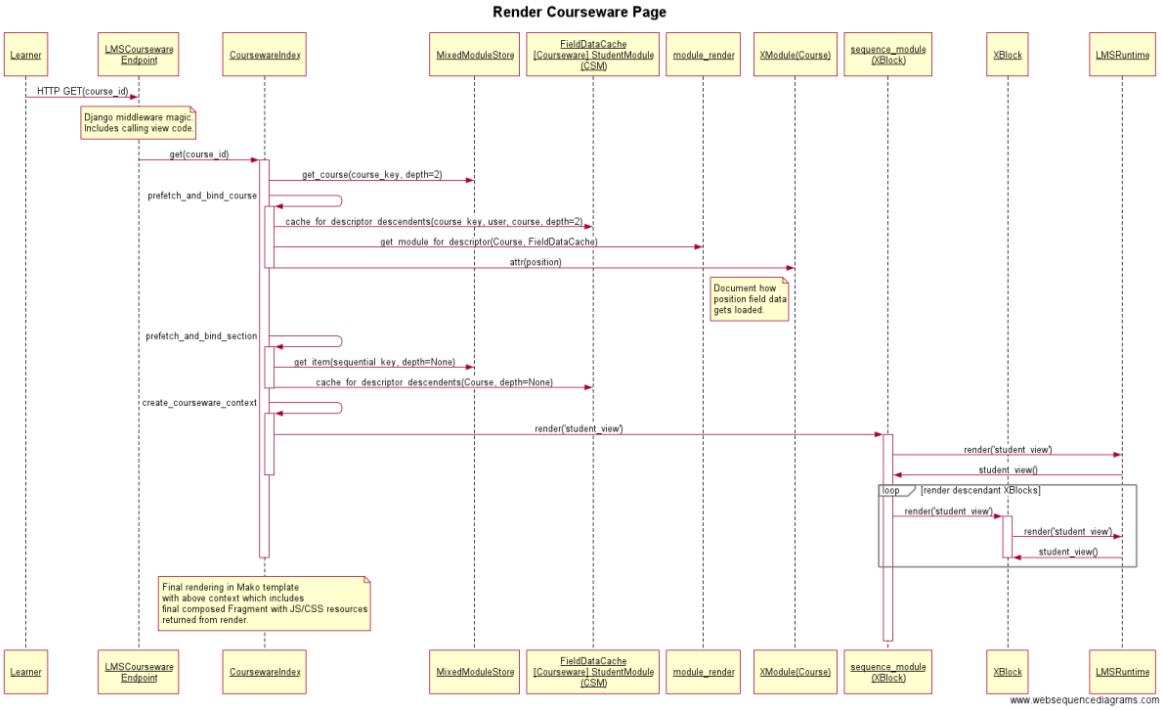


Figure 10.1: Rendering of courseware using XBlocks

### For Open edX instance

Clone the repository on our local machine. Testing this first on devstack is always recommended

1. Enter the shell of your devstack and execute :

   ```
   sudo -u edxapp /edx/bin/pip.edxapp install /path/to/cloned/directory
   ```

   **Note:** You need to point pip to the directory containing setup.py of our project.
   For example: If we cloned this repo in directory called **/home/edx/advhtmlxblock** and **/home/edx/advhtmlxblock/setup.py**is present then **/home/edx/advhtmlxblock** is your required path

2. (Re)start your LMS and CMS.

3. Login to studio as staff

4. Go to "Advanced Settings" in your course

5. Add word advancedhtml to list of "Advanced Modules"

6. Save changes

7. Advanced HTML component should be present in "Advanced" section in your course.

# *Eleven*

# Advanced HTML XBlock for Open edX platform

## 11.1 Features

- Full CSS Support
- JavaScript Suppport
- Live Preview HTML
- Code Indentation
- Code Folding
- Autocomplete Tags
- Autocomplete Brackets
- Supports all HTML tags

## 11.2 Installation

Installation instructions are already covered in previous chapters(Creating an Xblock)
**Special Note for Docker based devstack :**
Since the docker based devstack has two separate containers for lms and studio, it is required that you install this xblock in both the containers and then [re]start both lms and studio to see changes. The workflow would look something like this :

1. `make studio-shell`
2. Install xblock as mentioned before
3. `exit` from studio-shell
4. `make lms-shell`
5. Install xblock as mentioned before
6. `exit` from lms-shell
7. `make studio-restart && make lms-restart`

## 11.3 Working

AdvancedHTMLXBlock essentially extends raw HTML component of Open edX. This Xblock uses the latest version of CodeMirror(5.38 as of June 2018). The Editor is configured to enable code folding/ code indentation etc. All the html content received from the editor is then put into an iframe. The height of the iframe is changed on changes in html content and iframe is styled so that it looks virtually absent.

## 11.4 Technical Details

Fields :

1. **display_name** : This is the name shown to user(course creator) in "Advanced" component list in studio

2. **htmlcontent** : This is the actual htmlcontent that will be rendered to student in student_view

3. **live_preview**: This stores the live preview preference of each xblock

4. **unique_id** : This is the id used in student_view's html to differentiate from other xblocks. This is NOT the unique id generated by Open edX platform(also known as locator). This unique_id is used as id of iframe in html template. Since this is unnique to each advancedhtmlblock, you can add multiple advanced html xblocks on same page.

5. Rest fields are just the required fields for the xblock to be used successful in lms and studio and do not hold much of significance

Functions :

1. **student_view** :
   This is the function called by LmsRuntime to render the xblock to students. Since the xblock does not explicitly define author_view, student_view is used as author_view. On the first run, this will generate unique_id using uuid library. This id is passed to student_view template everytime student_view is called. The student_view is nothing but a simple iframe with NO content inside it initially. Javascript will ask for htmlcontent once it is initialized. Javascript queries the htmlcontent via AJAX and the response is JSON. This htmlcontent is then written into iframe by javascript. Once the content loads inside the iframe, height required for iframe is calculated and then that height is set to the iframe rendering as if the iframe is absent.

2. **studio_view** :
   This is the editor panel that is shown to course creator in studio after "edit" button is clicked. The function adds all the requrired CSS and JavaScript required for CodeMirror to work properly. The interface of studio_view also consists of a live preview panel and Advanced Settings panel. Advanced Settings panel allows course creator to hide live preview panel and change display name of xblock.

## 11.5 Screenshots and Source Code

- The screenshots of our Workbench along with the editor are provided as mentioned below

**[See Figures 9-11 in the "Figures and Screenshots" section.]**

- The screenshots of our Advanced HTML XBlock in our running instance of Open edX are provided as mentioned below:

**[See Figures 12-20 in the "Figures and Screenshots" section.]**

- The source code of our Advanced HTML XBlock is available on our github repository as mentioned below:

[https://github.com/ashutoshbsathe/AdvancedHTMLXBlock]

## 11.6 Testing

We have tested our xblock to see if we can include multiple xblock in a unit, move/copy an xblock from one unit to another unit, importing/exporting a course with our xblock
We have also tested if all HTML tags/CSS styling and JS is working
JavaScript has one restriction that it cannot access the topmost window for security reasons. This is tested and it works good so far.

Full test report can be found here :
https://docs.google.com/spreadsheets/d/11XM9PDOGZzXspLcwYCWpo9gYTPaiNUo_MKtTmz873sM/edit#gid=0

## 11.7 Issues and solutions

- **Broken codemirror features :**
  The codemirror source was stripped down to include only the features we need. So it was tricky to include them in the xblock. However, understanding codemirror code helped us include it nicely

- **CodeMirror conflicts :**
  This issue was not faced on local workbench, we faced this issue when we integrated our xblock in devstack. The problem was studio and lms already have a version of CodeMirror loaded with them, so loading our version alongside with it was going to be a problem. We modified a bit of codemirror source code to make it work in "browser only" mode. This also brings us to the important conclusion about fragment API. The xblock's html, CSS and JavaScript is rendered into a fragment which is then rendered by studio/lms. The catch is the fragments are not separate from each other. That means one fragment's CSS/JS can interfere with another fragment's CSS/JS and in usual, can interfere with main CSS and JS used by page. This is potentially dangerous as it can execute some bad code which can break the entire page.

- **JavaScript Security concerns :**
  While our xblock can allow you to add html and css to beautify your course, it will also allow you to add javascript to your course content to make it interactive. The security concern was scope of javascript. The scope of javascript must be limited to iframe and iframe ONLY. Xblock's javascript in no way should be allowed to access toplevel window. This was fixed by "sandboxing" the iframe and allowing limited javascript functionalities. Check out this for more info :

[Commit hash : 3a9d7e3890aa3da834be1d335f4ab799b6629cf3]

OR Click below

https://github.com/ashutoshbsathe/AdvancedHTMLXBlock/commit/3a9d7e3890aa3da834be1d335f4ab799b6629cf3

# *Twelve*

# Conclusion

Delivering the course content to learners in better ways is most important in any educational system. In present online education system, the successful completion rate of a course is very less. XBlocks can be used to provide the interface to deliver the course content in better ways to attract the learners and deliver courses in a modular way.

Our Advanced HTML Xblock provides a great platform to increase the interactivity of the courses and enhance the learning experience of the students. It is a great tool for text-intent course content creators as it provides them with total control over the course content. In contrast to the present text editor in edx-platform, functionalities of our Xblock are merely limited to the imagination of the course author. It will be a great addition to the current working lists of Xblocks present in Open edX and will considerably boost the user experience.

# *Thirteen*

# Future Work

1. **Auto-complete feature in the editor :**
   Currently, this feature is absent in our version of CodeMirror implemented in the Advanced HTML Xblock . It can be implemented in the future where a user will be shown a list of options of the probable keywords on the basis of a keystroke.

   **Reference:** [https://codemirror.net/demo/complete.html]

2. **Setting a common CSS for an entire course :**
   Presently, if a course author wants to have a unified visual theme for all the content in a particular course, he/she would have to manually apply the CSS style to each and every unit in the html code. To make this easier we can have a common CSS file which will be automatically fetched and imported for every unit in that course as per the settings of the course. One possible way of implementing it would be making changes on the Django backend of that particular course to fetch that CSS file from the **"Files & uploads"** section present for every course in Open edX platform.

3. **Additional settings in the Xblock :**
   At present our Xblock has two options in the settings tab. Display name and Live preview toggle. Some other features that can be added are :

   - An option for selecting a dark/light theme for the editor to enhance the user experience even more.
   - Enable a draggable partition interface so that users can adjust the size of the editor window and the preview windows as desired.

# *Fourteen*

# Figures and Screenshots

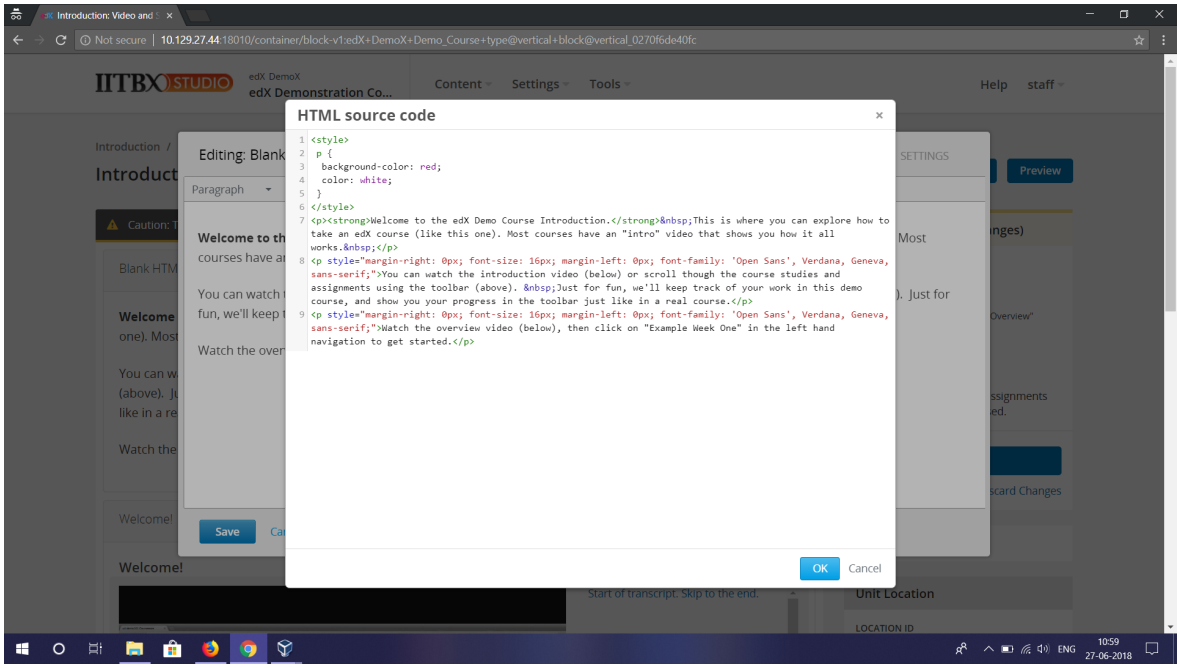


Figure 14.1: Adding CSS style tag manually

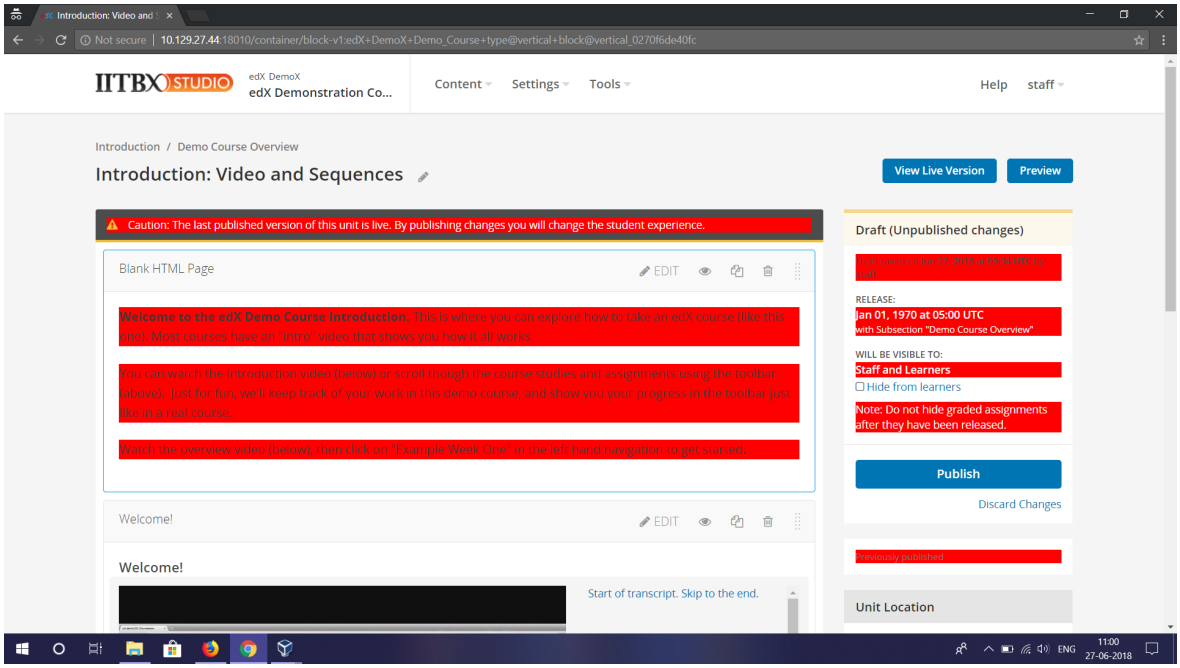


Figure 14.2: CSS gets spilled all over page

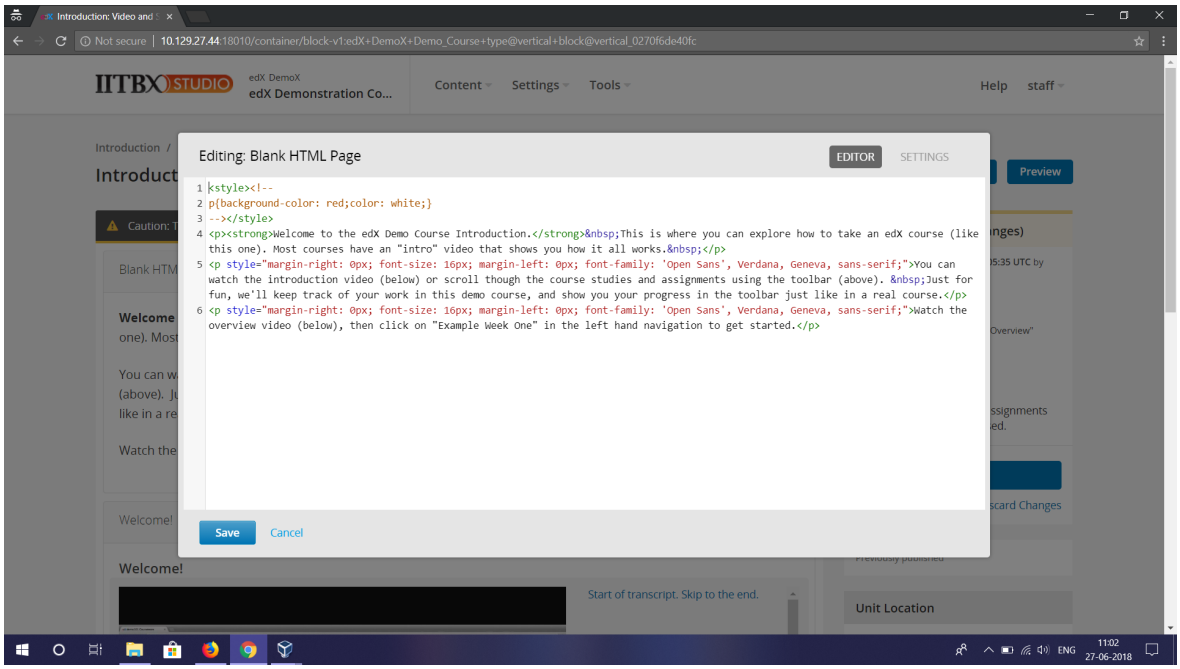


Figure 14.3: Indentation not preserved on re-editing

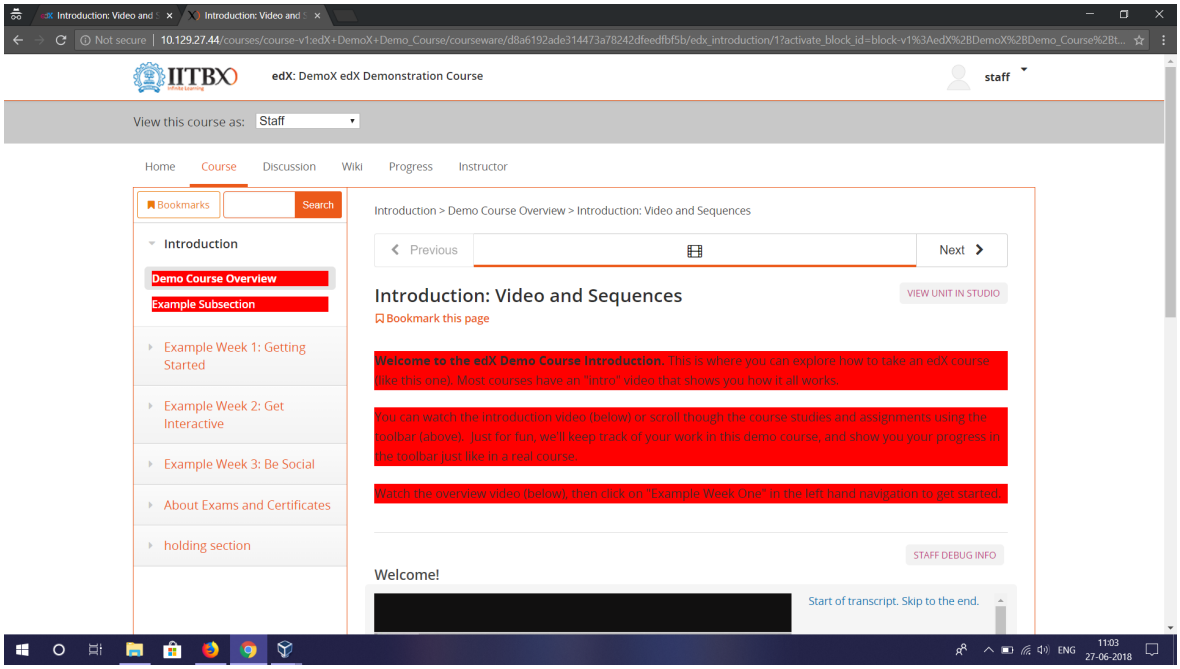


Figure 14.4: CSS spilled all over page in LMS

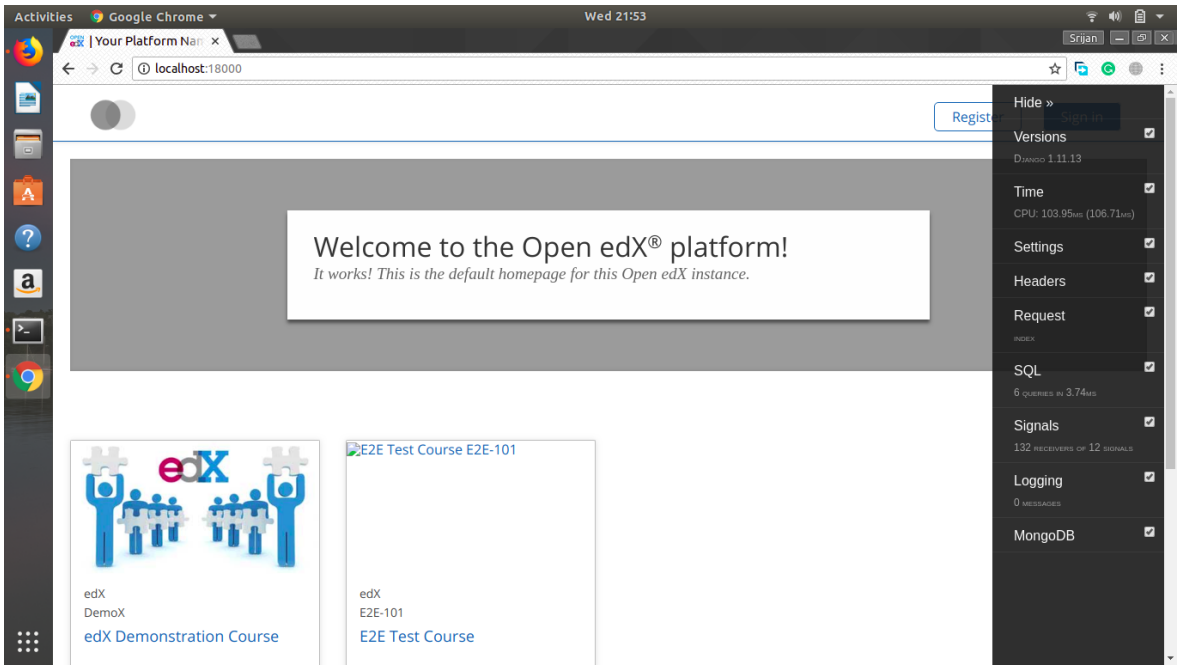


Figure 14.5: LMS landing page

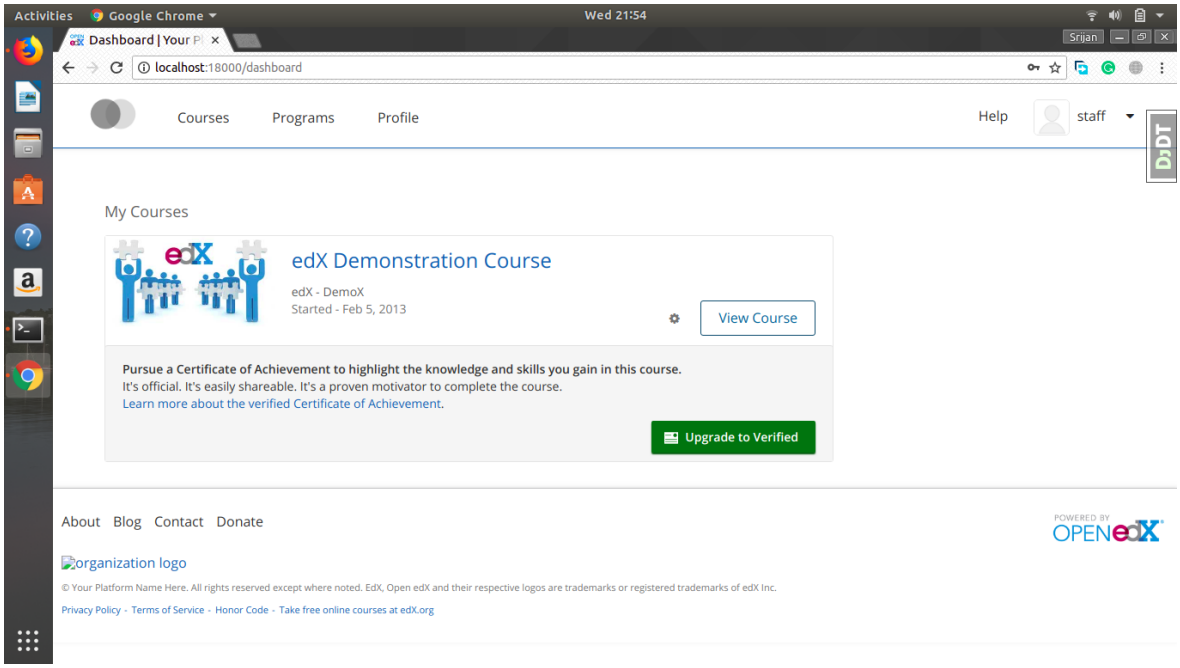


Figure 14.6: Staff login in CMS

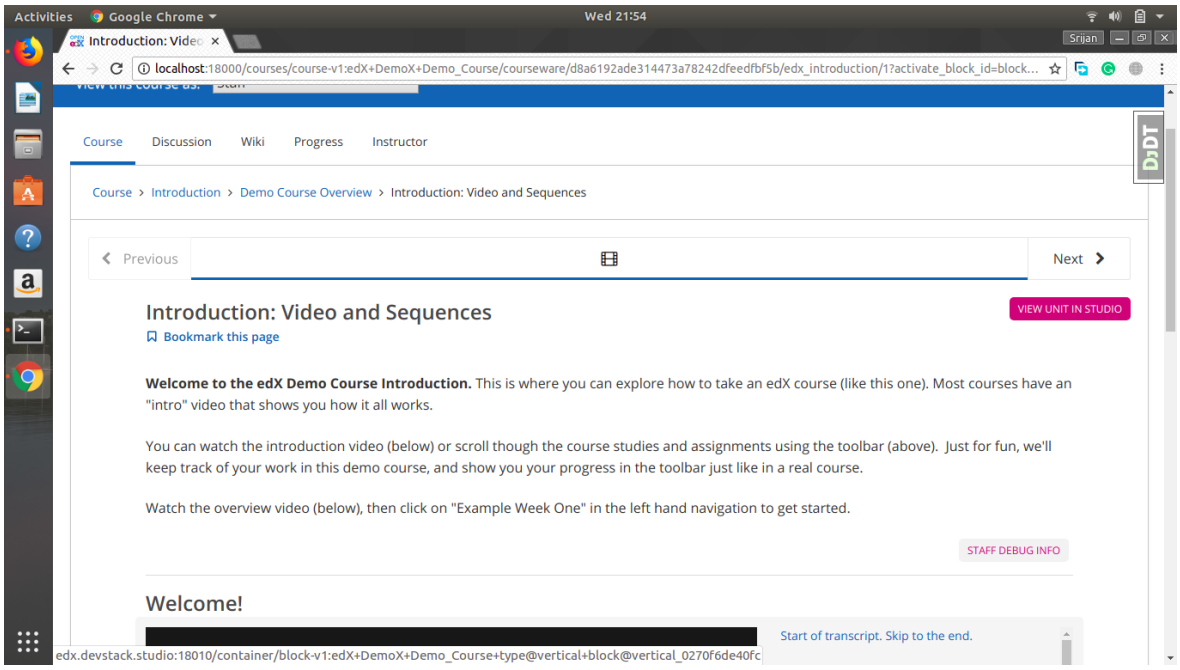


Figure 14.7: Viewing course in LMS

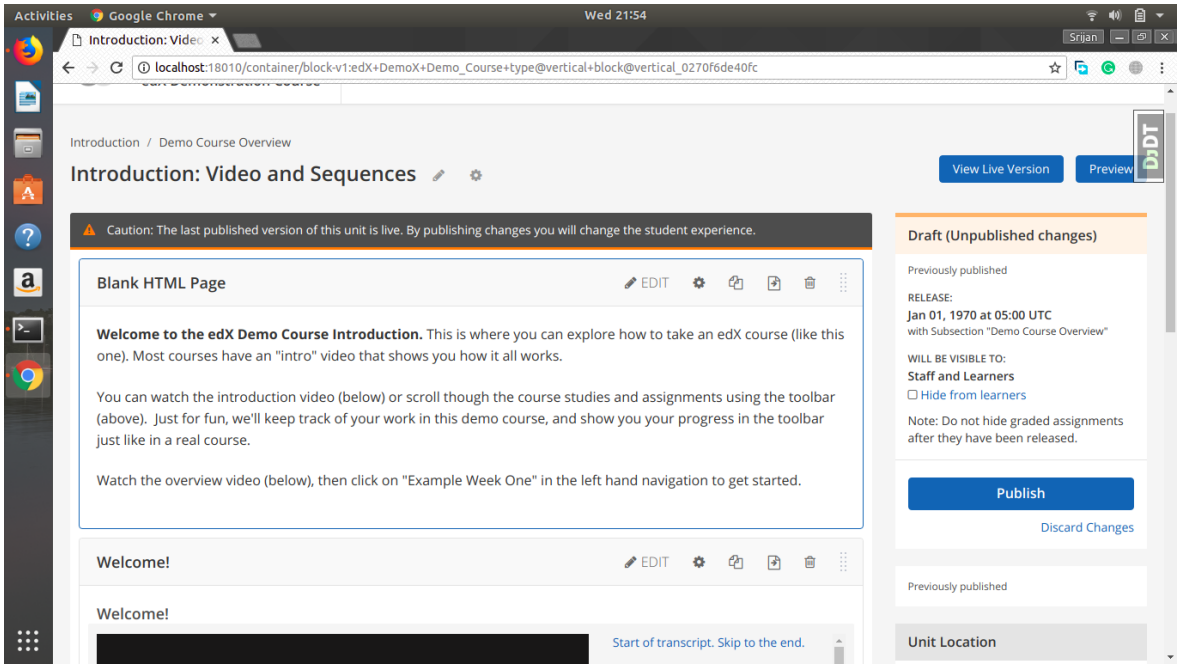


Figure 14.8: Viewing course in CMS

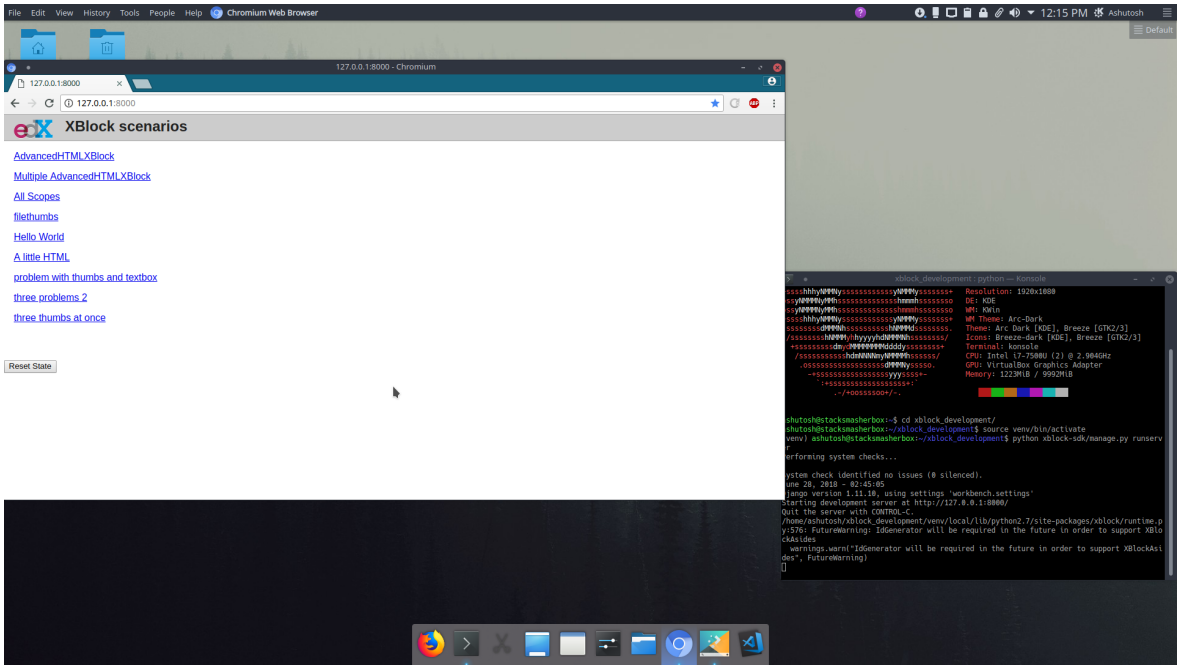


Figure 14.9: Landing page of workbench

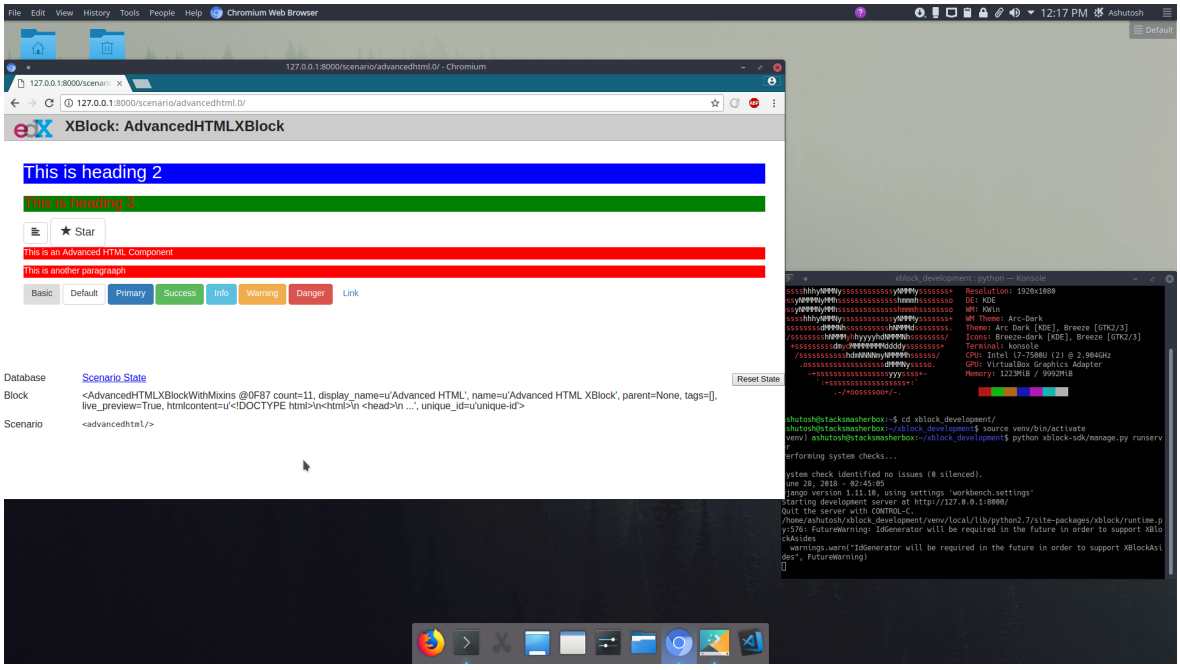


Figure 14.10: Viewing single xblock scenario

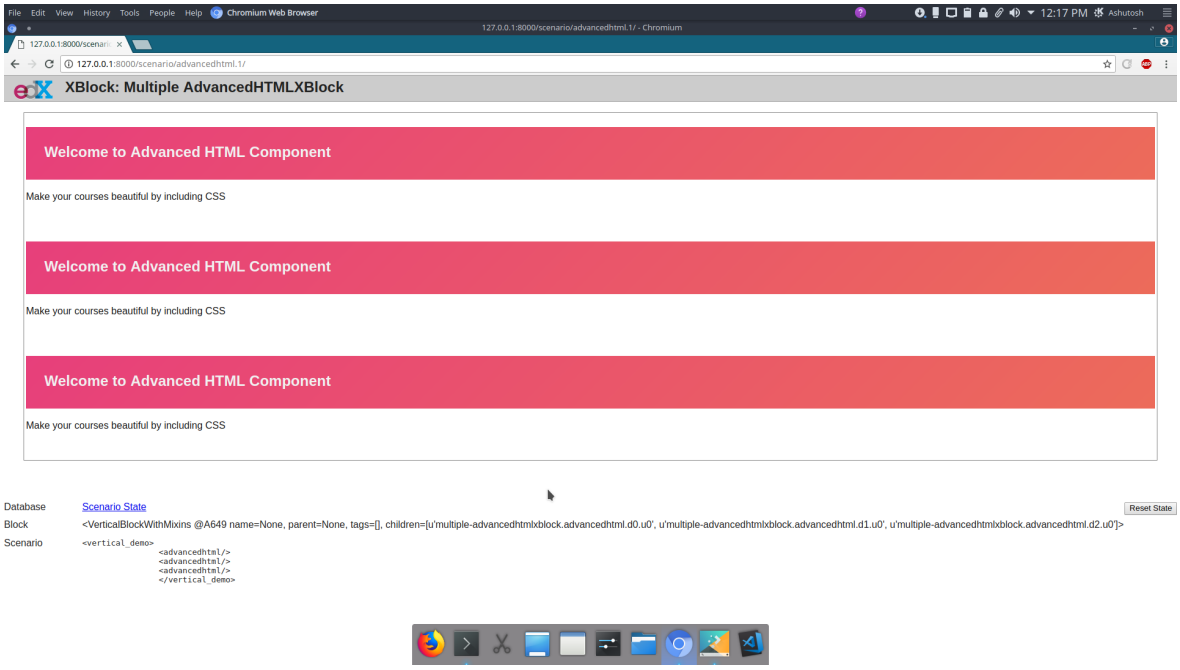


Figure 14.11: Viewing multiple xblocks senario

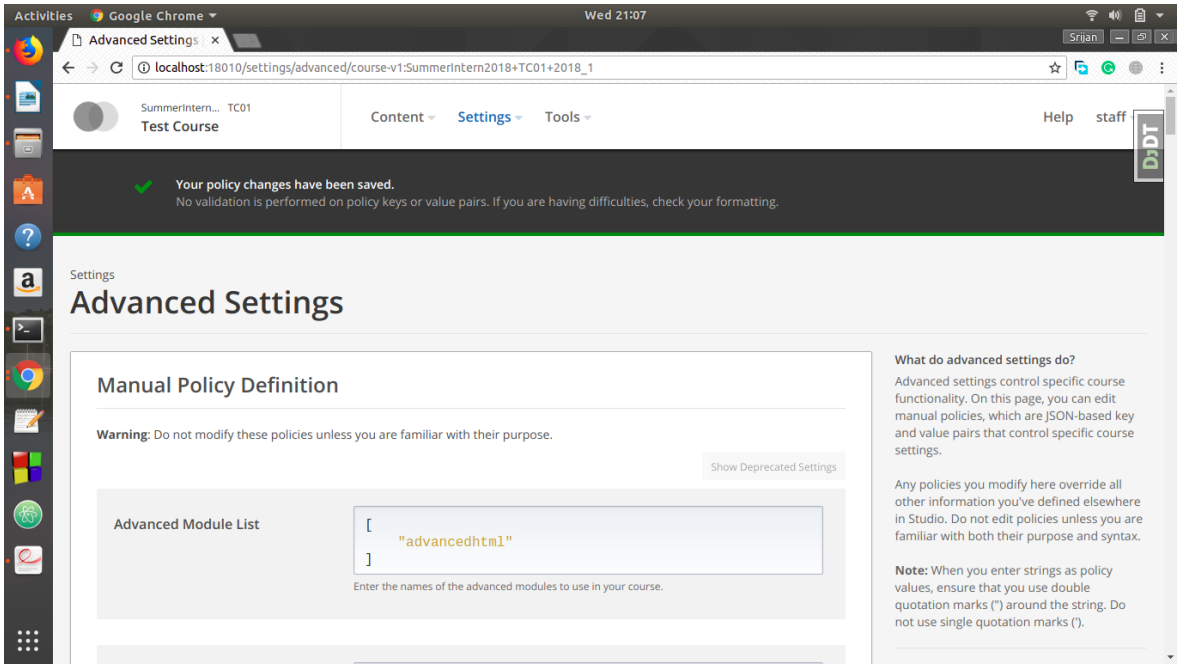


Figure 14.12: Adding xblock name in advanced settings

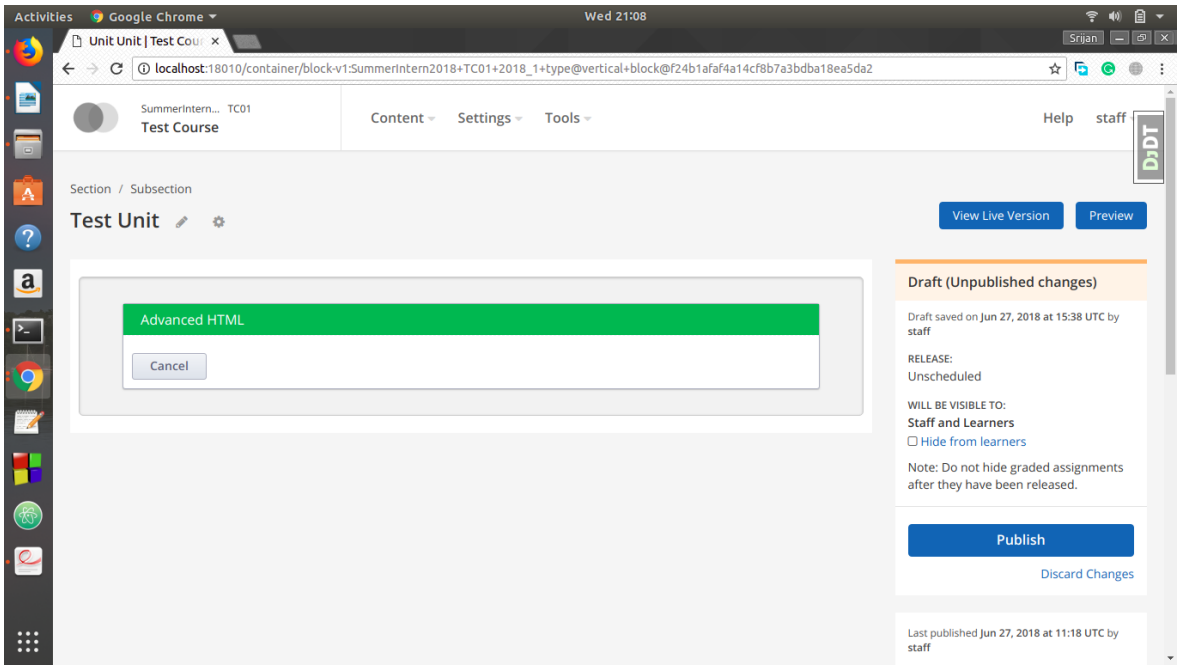


Figure 14.13: Xblock available in advanced components

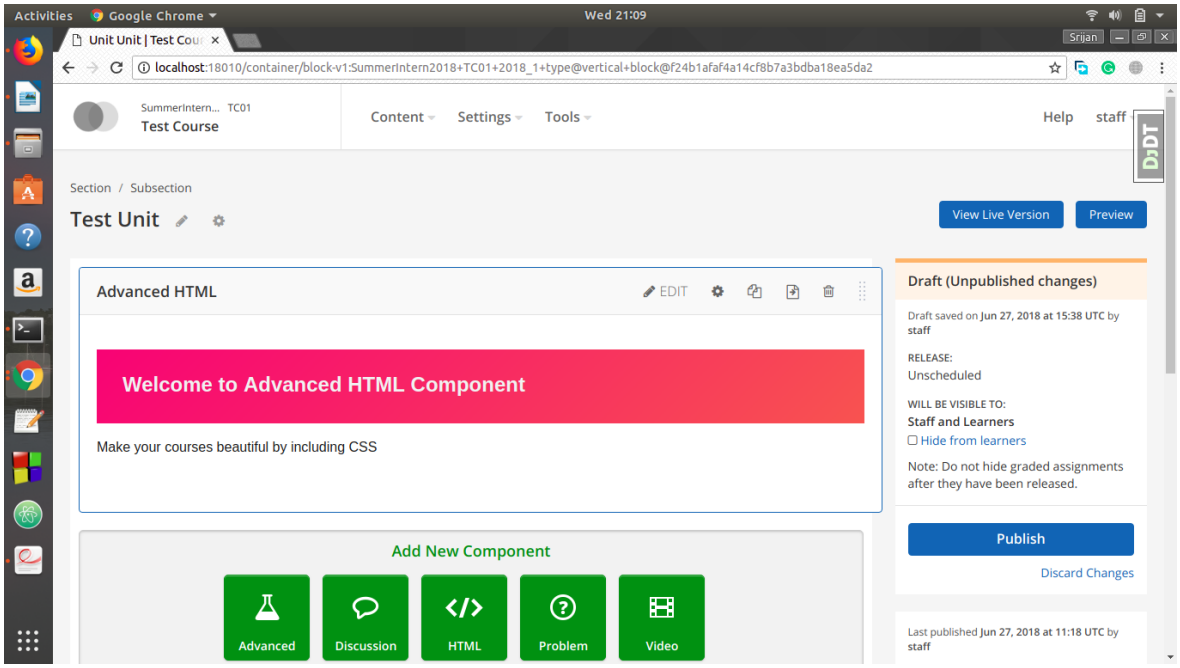


Figure 14.14: Default content of xblock

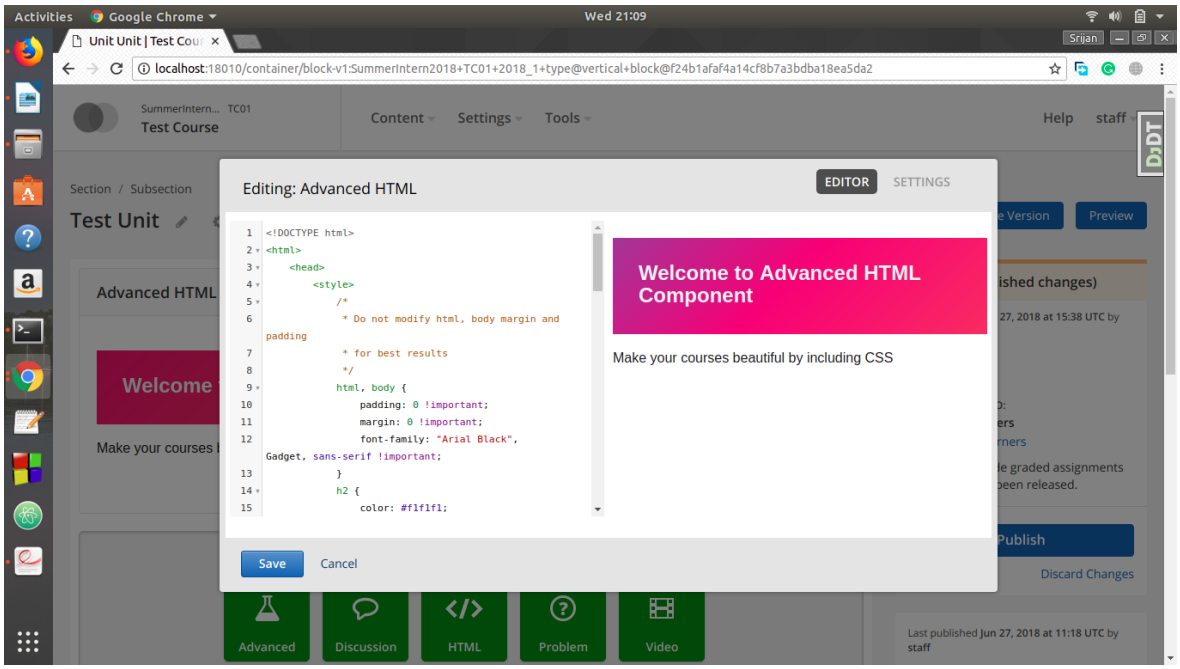


Figure 14.15: Editor with live preview

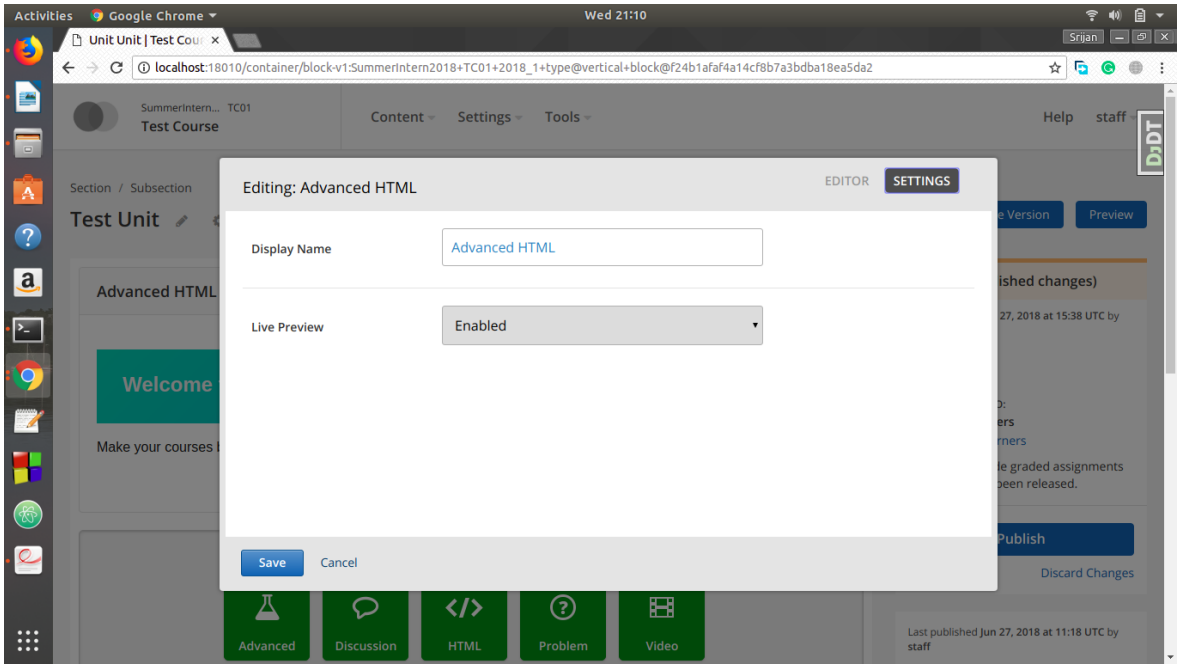


Figure 14.16: Settings tab

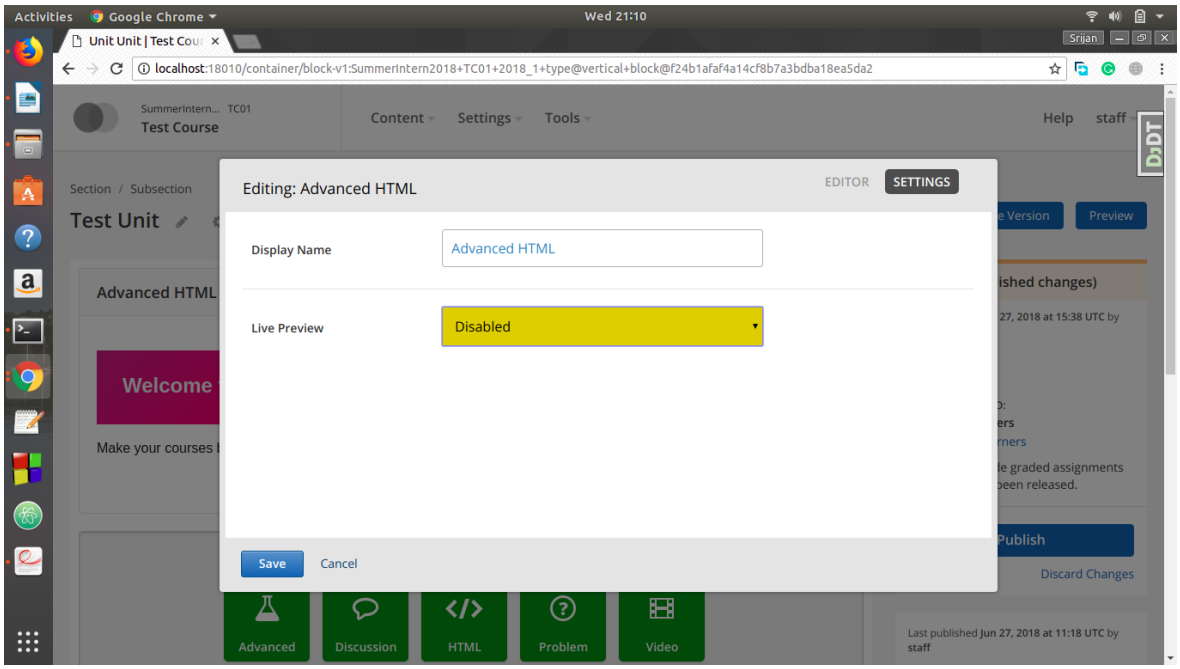


Figure 14.17: Disabling live preview

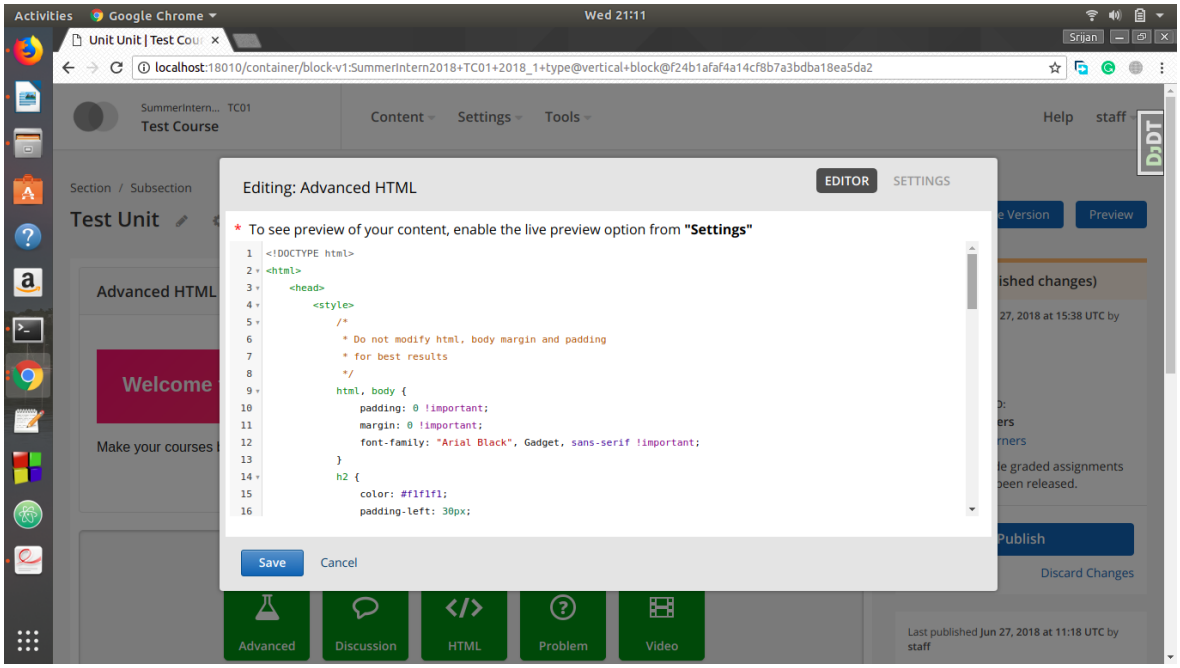


Figure 14.18: Full width editor available

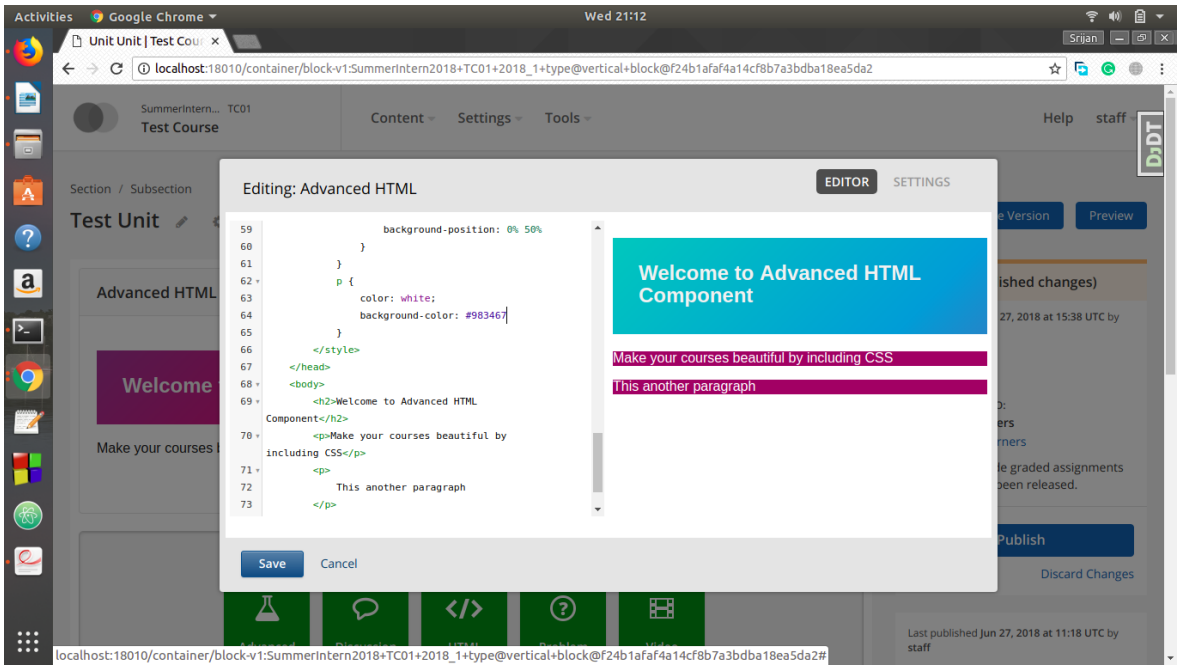


Figure 14.19: Live preview in action

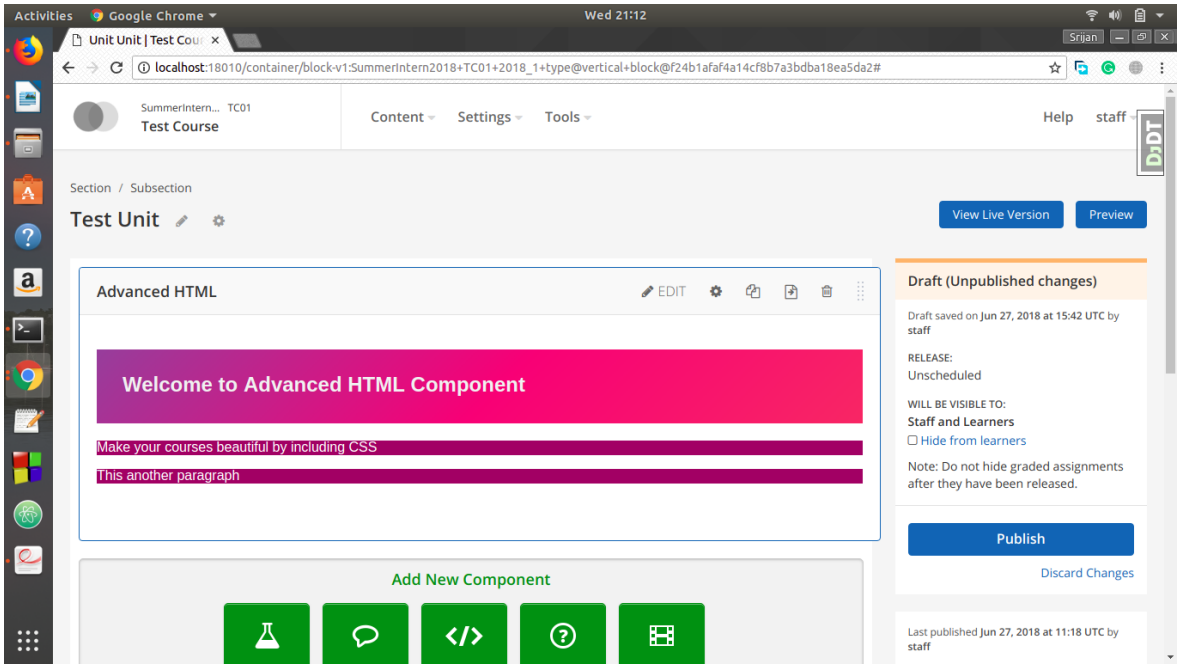


Figure 14.20: Final component after editing content