# Enhancing Course Page Usability on a Standard Moodle Site Using Generic JavaScript and Bootstrap CSS

**Gordon Bateson**
*Kochi University of Technology, Japan*

## Introduction

The emergence of Moodle as the most widely used learning management system (LMS) in the world is largely due to the fact that it is open-source, which makes it not only free to download, free to use, and free to modify, but also supported by a multitude of active developers and users. However, although the appearance and performance of the software has the potential to be modified in minute detail through settings and 3rd-party plugins which extend the functionality of the core system, the reality is that many Moodle sites are left to run in a largely standard state, thus failing to utilize the full potential of Moodle to be adapted to particular teaching and learning situations.

There are several reasons for this under-utilization. Firstly, the size and complexity of Moodle means that one person, or even one department of people, can never comprehend all of it, and as a result, unexplored and little used settings remain unchanged. The plethora of settings can also lead to a fear of changing anything, as changing a setting on one page can have significant but unexpected consequences in other areas of the Moodle site. This problem is exacerbated when there are several people who are assigned as Moodle administrators, each of whom may be making changes to the site without fully informing the others.

As a result, teachers who use Moodle, are often faced with the task of creating an online course using only standard Moodle functionality. In essence, this requires building a "course page" which consists of links to "resources" and "activities". The former contain static content in the form of text, audio or video, whereas the latter are tasks that students attempt, and for which they receive a grade. Access to these materials is granted only to users who are registered as managers, teachers or students of the course. A simple course, that contains only 10 or so resources and activities, will fit easily onto a single computer screen, even on a handheld device. However, as the length of the course page grows, so does the need to add navigation and other visual aids to help users grasp the structure of the page and navigate around it easily.

To address these shortcomings, the current study introduces several modifications that were made to improve the usability of course pages on a standard Moodle installation. The modifications

ISSN 2758-5700
*TEFL Praxis Journal*, 2022, Vol. 1, pp.56–64
DOI:10.5281/zenodo.7455740

were designed to emphasize the course structure and schedule, reduce vertical and horizontal scrolling, and clarify students' progress through the materials. They were implemented using two ubiquitous web technologies, namely JavaScript and CSS, the details of which are described in the next section.

## Implementation

As stated earlier, this study focusses on ways in which the appearance and behavior of a standard Moodle course page was modified using JavaScript and CSS, which are both popular and freely-available web technologies. JavaScript is a programming language that runs in the browser once a web page has loaded. It can add, modify, move or delete elements in the webpage, as well as respond to user actions such as clicks and taps. CSS (Cascading Style Sheets) is a formatting language that can be used to specify how particular elements on a webpage should be displayed. For example, it can be used to specify the color of backgrounds and text, the width of borders, margins and padding, and the layout of items in rows or columns.

CSS settings that are common to several webpage elements can be shared by grouping them together into a "class". Any element that uses a particular CSS "class" will automatically be assigned all the CSS settings in that class. Furthermore, rather than create their own classes, many website builders use one of several open-source and popular CSS libraries, because they include many useful CSS classes. In the case of Moodle, the Bootstrap CSS library is included with the standard Moodle install.

One important feature of the Bootstrap library is that it supports "responsive" page layout. In a "responsive" page, the layout changes depending on the size of the display area. Thus, on wide screens, videos, images and audio players can be full size, and tables of data can extend horizontally to utilize the full screen width. However, on smaller screens, the media players and images are reduced in size and the tabulated data is reorganized to be displayed vertically.

In Moodle, the Bootstrap CSS classes are commonly added to webpage elements when they are created by the Moodle scripts running on the server, but they can also be added to elements in the browser by JavaScript that runs after the webpage has loaded. In such a case, the JavaScript first searches for particular elements and then adds the appropriate Bootstrap classes to these elements. At that point, the browser recognizes the new classes and adjusts the display of the elements accordingly.

Teachers using Moodle can insert JavaScript and Bootstrap CSS into pages in their Moodle courses using the same webpage editor that is used to add text, audio and video. However, as the standard editors offer no graphical interface to add JavaScript and CSS, these technologies must be added directly to the HTML source code of the page. The source code can be made accessible either by selecting the "Plain text" editor on a user's editor preferences page, or by clicking the "Show source code" icon, "<>", in the HTML editor. Once the source code is accessible, the JavaScript code can be inserted as plain text.

When deciding where to insert the JavaScript and CSS into a Moodle course page, it is important to consider that, depending on the display settings, a Moodle course page will sometimes display only one section at a time. However, even in such a case, the first section of the page is always displayed. Therefore, the optimal place to insert the JavaScript is in the description area for the first section of the Moodle page. By default, this section is called the "General" section, and its

description can be edited by first enabling "Edit mode" on the course page, and then clicking the edit menu for the "General" section of the course.

## Modifications

A standard format Moodle page is shown in Figure 1. As can be seen, each section contains a list of links to its resources and activities followed by a horizontal line to separate it from the following section. It is not immediately obvious which of the items are resources and which are graded activities. Furthermore, there is no indication of what grades have been awarded for which activities, and thus, it is not clear how far a student has progressed through the course and what should be accessed next.


**Figure 1.** *Standard Moodle Course Page*

As mentioned earlier, an additional difficulty with this layout is that as the number of links to materials increases, so the page grows longer, and it becomes necessary to scroll vertically, particularly on devices with small, narrow screens. This situation worsens as the course progresses and it becomes necessary to scroll further and further down the page to access the materials for the "current" lesson.

To remedy these issues, the implementation technique described earlier was employed to make the following modifications to the course page:

1. Add formatting to section headings
2. Add dates to each section
3. Add a button to update dates automatically
4. Add a section navigation menu
5. Add activity grades

Each of these modifications is described in detail below.

***1. Add formatting to section headings***

To improve the visibility of the section titles, and at the same time replace the horizontal lines that separate the course sections, the section headings were reformatted to act as separators (Figure 2). This could be achieved using JavaScript to first locate all the <h3> HTML elements containing a section title, and then adding the following Bootstrap classes to each title:

`bg-secondary`: add the grey background
`text-dark`: use a dark text color
`rounded`: round the corners
`mt-0`: remove the top margin
`py-2` and `px-3`: set the vertical and horizontal padding respectively.


**Figure 2.** *Emphasized Section Titles*

***2. Add dates to each section***

Although there does exist a Moodle course format that automatically inserts dates for each section of the course, the dates cannot be changed or rescheduled. As a result this "weekly" format is of limited use in a real teaching situation. Instead, the most practical solution is to insert the dates either into the section title, the section description, or as a "label" within the section. Of these options, inserting the date into the title is preferable, because it takes up less vertical space on the page.

One small issue with doing this is that the title and the date appear in the same large text, making it hard to quickly differentiate the two types of information. As a solution, the dates were inserted within square brackets (Figure 3), which could later be identified by the JavaScript and replaced with <small> HTML tags with the following Bootstrap classes:

`ml-3`: add a margin to the left of the date to separate it from the title
`text-info`: set the text color to a light blue, signifying "information"

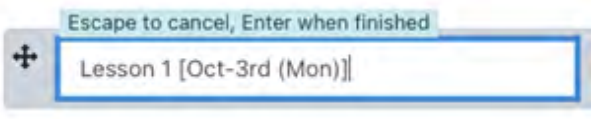


**Figure 3.** *Specifying a Date Within Square Brackets*

As a result of inserting dates into the section titles, the course page itself became the "schedule" for the course, and therefore it was not necessary to create a separate page showing the schedule. This not only removed the need to navigate away from the course page to see the schedule, but

also served to gently and continuously remind students of the lesson dates. When it was necessary to changes dates, for example because of school holidays or teacher absence, the new dates were immediately available to all course participants.


**Figure 4.** *Formatted Dates in Section Titles*

***3. Add a button to update dates automatically***

Adding dates to the section titles is effective, but it can be a laborious and error-prone task if done manually at the beginning of a semester or quarter term by a teacher who modifies the dates one at a time. A more efficient and accurate method is to update the dates programmatically using JavaScript, but the algorithm to do this must be sophisticated enough to account for national holidays, school holidays and rescheduled lessons.

In Japan, there are 16 national holidays, of which some occur on the same date every year, for example "Children's Day" is always on May 5th, while others occur on the same weekday within a certain month, for example "Marine Day" is held on the third Monday in July. School holidays are less predictable and cannot be reliably calculated in the same way that national holidays can. Instead, they must be added to the algorithm as a set of explicit values. The same is true for rescheduled classes, which are classes that are cancelled unexpectedly and then moved to a later date. This can happen due to natural disasters or teacher sickness. One further complication is the frequency with which classes are held. The most common frequencies for classes held at universities in Japan are probably once a week, twice a week or every weekday, but other frequencies are also possible.


**Figure 5.** *Button to Update Dates*

To initiate the algorithm, an red button is added to the page when it is in Edit mode. The button is displayed at the top of the page, to the left of the blue button that disables Edit mode. By limiting the display of the red button to Edit mode, we can firstly be sure that the button is only displayed

to teachers, and secondly reduce the likelihood of teachers pressing the button unintentionally when they are viewing the page in ordinary mode.

When the red button is clicked, some settings are displayed in a popup box (Figure 6). The settings allow the teacher to specify the start date, the number of lessons, the days on which lessons are held and the required format for the dates.

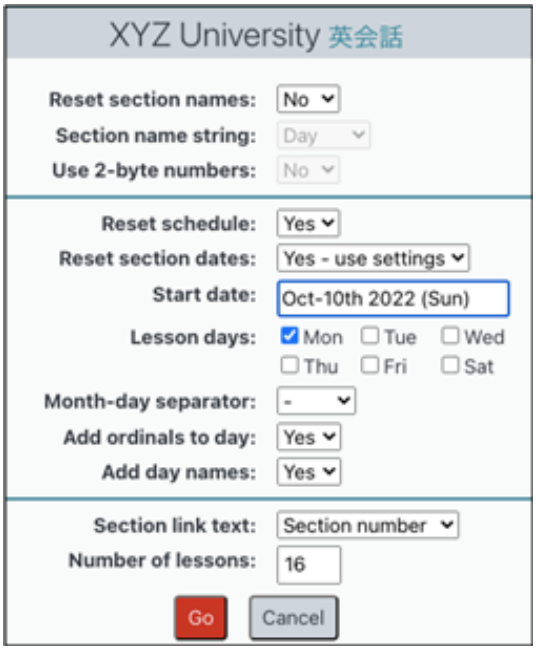


**Figure 6.** *Settings to Recalculate Dates*

Using these settings and the information about holidays discussed earlier, the JavaScript algorithm first calculates the lesson dates, and then inserts them into the appropriate section titles. Of course, merely inserting the new dates into the webpage does not update the information on the Moodle database held on the server. To do that, the JavaScript mimics the functionality that allows a section title to be updated directly from the course page, as shown in Figure 3. This functionality is enacted using one of the APIs (Advanced Programming Interfaces) that Moodle offers to JavaScript. The new section titles are sent to this API, which then updates them in the Moodle database, thus ensuring that the changes are permanent, and the new dates will be used the next time the course page is displayed.

***4. Add a section navigation menu***

One way to reduce vertical scrolling is to add a navigation menu to the top of the "General" section of the course. The menu is actually just a series of links, one to each section of the course. By labelling each link only with the number of the section that it refers to, the menu becomes compact enough to fit neatly on a single line when viewed on a desktop or laptop computer, or two lines when viewed on a smartphone. When a link is clicked, the browser scrolls down automatically, immediately and precisely to the required section. Furthermore, as the page scrolls down, a return

arrow becomes visible. When this arrow is clicked, the page scrolls back to the "General" section at the top of the page.


**Figure 7.** *Navigation Menu Containing Links to Each Section*

Because the section titles contain the dates of each lesson, it is possible for the JavaScript to calculate which section contains materials for the "current lesson". The "current lesson" is simply the one that is nearest in time to when the page is viewed. It may be in the past, i.e. the "most recent" lesson, or in the future, i.e. the "soonest upcoming" lesson, but it will always be closest to the time that the course page is viewed.

Having established which section contains the "current lesson", the JavaScript first highlights that section's link in the navigation menu, in the way that lesson "6" is highlighted in Figure 7, and then sets the background color of that section to stand out from the other sections of the course. Thus, students are immediately alerted to what they are expected to study next. As a result, students need only access the page and click on the highlighted link in the navigation menu to arrive quickly at where they are expected to be.

***5. Add activity grades***

University students are typically curious to know what grades they are receiving for assignments and homework, but in Moodle this information is not readily available on the main course page. Instead, students must view the gradebook page for the course. This page shows the grades for individual tasks and how they are combined in grade categories to eventually calculate the course grade. However, the gradebook structure does not always match the lesson structure, so it can be hard for students to merge these two views of the course.

It is beneficial therefore to extract the activity grades and display them on the course page. In JavaScript, this can be done by requesting the grade book page in the background, i.e. while the main course page is being displayed. The grades can then be extracted from the gradebook page in the background and inserted alongside the corresponding activities in the course page. The grades are color coded, so that high grades appear in green, borderline grades appear in orange, failing grades appear in red, and empty grades appear in grey. Furthermore, when a teacher views the course page, the grade that is displayed for each activity is the average grade, along with a count of how many students have attempted that activity, as shown in Figure 8.


**Figure 8.** *Activity Grades Displayed on the Course Page*

## Discussion

The previous sections of this paper have described why and how, with innovative use of JavaScript and Bootstrap CSS, it was possible to significantly modify the course page of a standard Moodle site in order to improve the usability and user experience. Now that the solution has been devised, implemented, tested and optimized, it runs efficiently and effectively, allowing the teacher to periodically update the dates for each new semester, and allowing students and teachers alike to quickly access the most relevant section of the Moodle course.

However, despite all the convenience that these improvements bring, it should be noted that adding and editing large amounts of JavaScript code in a Moodle course page is not only awkward but also has several significant risks and drawbacks.

Firstly, it is not possible to share the JavaScript code between courses. As a result, changes made to the code in one course must be manually copied and pasted into other courses which use it. In the future, this could be alleviated by storing the JavaScript in a centrally accessible file and sharing it somehow between the courses, perhaps by importing it as a JavaScript module.

Secondly, to prevent HTML editors from tampering with the JavaScript, care should be taken to use the "Plain text" setting in the "Editor preferences" page when editing the JavaScript. The Atto and TinyMCE editors that are supplied with the latest versions of Moodle will generally not reformat Javascript, but even so, the safest editing tool to use is the "Plain text" editor. Furthermore, using the Atto and TinyMCE editors requires two extra clicks before the source code can be edited: one click to show the "Advanced" editing icons, including the "Show source" icon; and a second click to show the source.

A third drawback is that there is no "version control" built in to Moodle, so you cannot rely on Moodle to be able to revert to a previous version of the JavaScript, if it stops working for some reason. Instead, it is necessary to create a workflow that includes some measure of backup and version control of the JavaScript code as it is developed. For example, the code in this project was edited in a text editor on a laptop computer. The text editor allows multiple levels of "undo", so that recent changes can be revoked. When closing down the editor, the code was updated in a file that was stored in an online location from where it was accessible to other computers used by the developer. In this way, it was possible to keep both the latest version of the code, and also recent revisions, accessible, and thus prevent the code being erased or lost accidentally by a

human error, such as pasting into the wrong text box in a webform, or hitting the wrong button on the computer keyboard.

By the end of this project, the length of the JavaScript code was just under 3300 lines. To develop it any further, the code should probably be split into smaller modules, in order to make it easier to maintain, and possibly reusable in other similar projects. Ideally, these modules should be made available from a folder with the Moodle scripts area of the server. This could be done if they were included as part of a Moodle plugin such as a theme or block.

## Conclusion

This research has shown that the usability of a course page on a standard Moodle site can be significantly improved using Javascript and Bootstrap CSS. Specifically, it was possible to emphasize the course structure by adding background colors to the section headings, thus converting them to separators that clearly delineated the sections. Also, formatted dates could be added to the course sections showing when the lessons would be held. These dates made it possible to identify the section for the current lesson and highlight that by changing the background color. At the top of the page, the JavaScript inserted a navigation menu that allowed users to go to specific sections without the need to scroll vertically. In addition, student progress through the course was emphasized by displaying an activity's grade on the course page alongside the link to that activity. The resulting software is stable, versatile and reusable on both current and future courses.