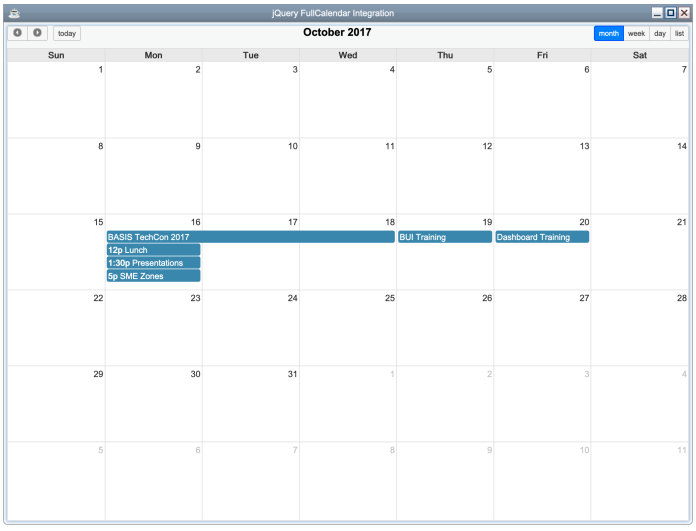
Introducing the BBjCalendarWidget Plug-In

By Nick Decker

BASIS Advantage 2022

Background

A few years back at TechCon17, our Language and Interpreter session had a section on "Native JavaScript Events in the BBjHtmlView." During this presentation, we demonstrated integrating the FullCalendar library into a BBj program (see **Figure 1a** below). The demo was pretty exciting at the time, despite its limitations, as it showed that it is possible to leverage a mature JavaScript library from within a BBj application. That demo, along with the desire to have a robust calendar control available in BBj, prompted BASIS to write a calendar plug-in for BBj 21 (see **Figure 1b** below).



Figures 1a and 1b. The TechCon '17 demo (left) vs. a BBjCalendarWidget demo (right)

Calendar Origins

The TechCon demo program was a "proof of concept" of sorts since it only exercised a couple of the FullCalendar's capabilities and largely ignored all of the library's configuration options and event callbacks. There was also quite a bit of low-level code required to get the calendar up and running, covered by this YouTube video of the presentation. For example, we had to inject the jQuery, FullCalendar, and Moment.js JavaScript libraries into a BBjHtmlView control before we could even think of creating the calendar. And once we got the calendar up and running, we had to write custom JavaScript code to notify our BBj application whenever the user dragged an entry from one day to another to reschedule the appointment.

BASIS soon realized that despite the FullCalendar library's power and flexibility, integrating it into a new or existing BBj application might require too much up-front development effort for many application developers. Thus, the idea was born to create a plug-in that takes care of all the setup, custom JavaScript code, and event information communication for the application developer. We can think of it as the second in a series of plug-ins that use pre-existing JavaScript libraries in the BBjHtmlView control, where the first was the powerful new Grid control, the BBjGridExWidget. The BBjCalendarWidget, installed with BBj 21, relieves the application developer from having to learn and work with JavaScript by providing a plug-in that handles all the low-level details, saving weeks or even months of development time. Now adding a calendar control to a BBj GUI, BUI, or DWC application isn't very different from adding a traditional control, and we can do it with a single line of code. And the best part? You don't have to worry about learning JavaScript!

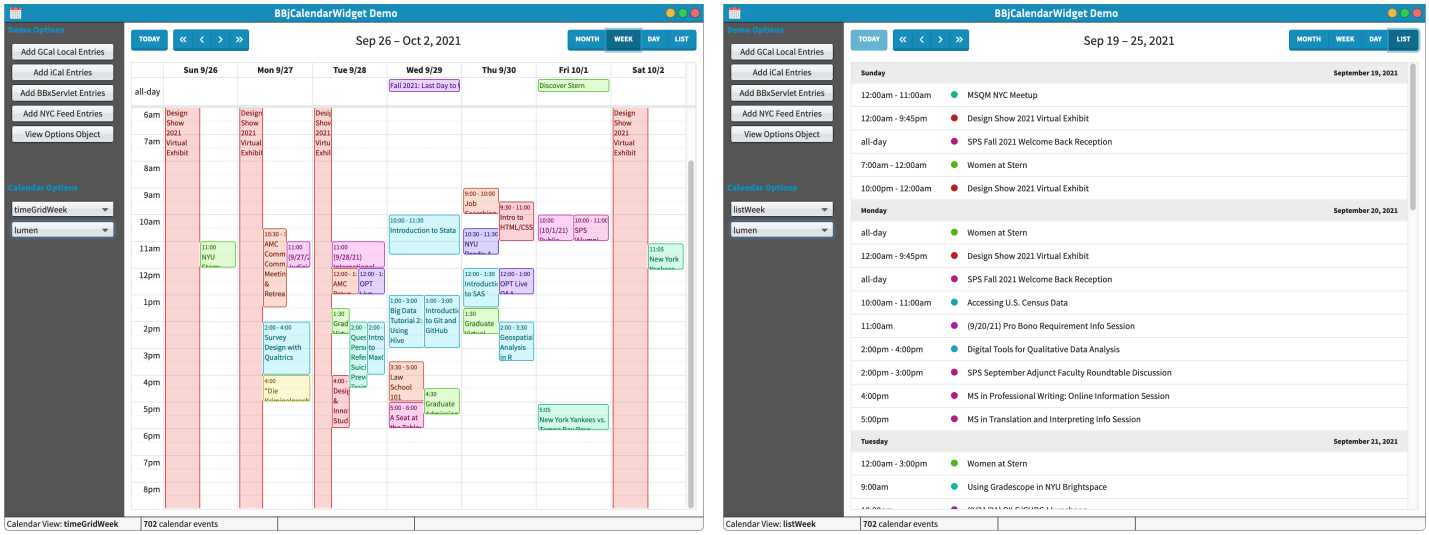
The FullCalendar Library

Under the covers, the BBjCalendarWidget uses the FullCalendar JavaScript library. This library is open-source and free to use, even for commercial products. It's actively developed and maintained with over 100 contributors and still enjoys constant improvements. This is particularly impressive given that it was first created in 2009. Since our TechCon presentation, the FullCalendar library has continued to improve. It has eliminated its dependence on the jQuery library, and its vanilla JavaScript API is now blazingly fast and uses a small embedded Virtual DOM library. The upshot of the inclusion of the library is that the code executes more quickly and efficiently compared to traditional methods of modifying the DOM directly. Besides the size and speed improvements, the FullCalendar library has continued to add new features and options over the years, making it one of the industry leaders. It's used by several prominent companies (**Figure 2** below), including Amazon, Netflix, Samsung, PayPal, and even NASA. And if you're still not convinced, the library currently boasts over 20 million CDN downloads every month!

Figure 2. A few of the companies that use the FullCalendar library

Calendar Views

The FullCalendar library provides eight view types, and users can select their desired view by clicking on a button in the calendar's toolbar. In general, the views are broken down into three categories: (1) a grid of days showing a full week (**Figure 3a** below) or month,(2) a time grid showing a single day or week, and (3) list views. List views (**Figure 3b** below) are similar to an agenda, where it's a listing of all the entries in the selected day, week, month, or even year. It's also possible to programmatically set the initial view or switch to a specific view type.



Figures 3a and 3b. The BBjCalendarWidget's week view (left) and list view (right)

The BBjCalendarWidget's Custom Classes

The BBj plug-in is a set of BBj custom classes that make it easy to add a functional calendar to a BBj program — in as little as one line of code (**Figure 4** below)! It takes over all the grunt work associated with injecting the JavaScript libraries into the HtmlView control and even simplifies the two-way communication between the BBj application and the FullCalendar library. BASIS added the BBjNativeJavaScriptEvent object in BBj 17, and this capability made it possible for JavaScript code executing in a BBjHtmlView to pass JavaScript events back to the BBj program. In BBj 17.10 this was further enhanced to allow for DOM elements and arbitrary data to be passed back to the BBj program. This capability is critical to the BBjCalendarWidget, as that's how the plug-in 'translates' the FullCalendar's JavaScript events to BBjCustomEvent callbacks.

You can think of the BBjCalendarWidget as the middleman between your application and the FullCalendar library. Despite the differences in computer languages on either end, it acts as a translator so that you can focus on your application code and don't need to know or write any JavaScript code.

```
rem Add the calendar widget to our window
myCal! = CalendarAPI.createBBjCalendarWidget(myWindow!)
```

Figure 4. The BBj code that creates a calendar widget and adds it to a window

Customizing the BBjCalendarWidget

The plug-in offers other BBj classes that let you get the most out of the calendar. For example, the **`CalendarOptions`** class gives you access to dozens of calendar configuration parameters, such as whether to display weekends and week numbers and defining the first day of the week (Sunday, Monday, or whatever day fits best for your application). If your application needs to limit the timeframes for an appointment, you can provide a starting and/or ending date to set a valid modification range. That way, the calendar displays the invalid dates differently, and the user may only move an entry within the valid range. If they attempt to move an entry outside of the valid range, the entry will zip back to its original position, and the calendar cancels the user's modification. This has several benefits, including:

- The calendar's date navigation buttons are intelligently hooked into the valid date range, so the user can only navigate to days, weeks, or months within the valid range.
- The calendar grays out the invalid dates, making it easy to tell which dates are available and which are off-limits.
- The calendar takes care of handling the case where the user tries to reschedule an entry to an invalid date by repositioning it to its original time and date. The calendar handles this scenario entirely, so your application doesn't have to provide any code to deal with invalid dates.

Figure 5's animation illustrates the concept of valid date ranges as implemented in the fourth calendar demo. The calendar's valid date range has been set to September 22nd through 23rd by providing a valid starting and ending date. Notice that the invalid dates are grayed out, providing a visual indication of the allowed range. If the user attempts to reschedule an event outside of the permissible range, the calendar automatically cancels the modification and moves the entry back to its last position.

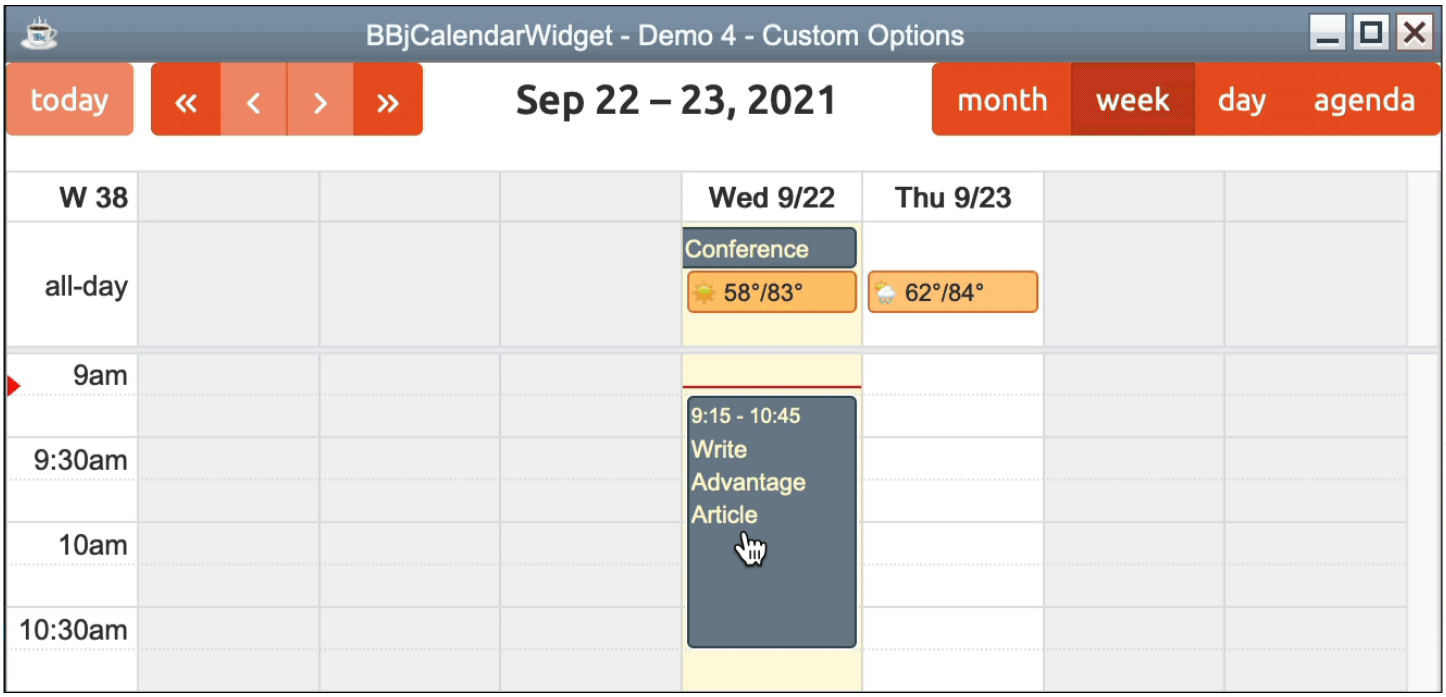


Figure 5. The calendar prevents the user from moving an entry to an invalid date

There are also a host of configuration options that deal with how the calendar and its entries are displayed, and you can set the text, background, and border colors for individual events. You can set whether entries are allowed to overlap one another for more efficient use of space, define the starting and ending times for the time grid views, and even make the dates clickable to drill down to an agenda view for the selected day. The calendar itself can be themed by selecting one of the built-in Bootstrap CSS theme files, and there's extensive CSS support so that you can set the font, color, and more for elements such as entries, weekends, weekdays, etc.

Calendar Entries

You can create an instance of the **`CalendarEntry`** class to create a single entry or add several of those to a BBjVector to add multiple entries to the calendar more efficiently. There are also classes that define entry sources, so it's easy to add entries to the calendar from web services in iCalendar or JSON format. These also make it possible to add entries from public Google calendars in a read-only format with very little code. And you can still add entries programmatically — all you need is an entry title and start time to create a basic entry. This means that you can read appointment data from many sources, including traditional BASIS KEYED files or any database table, create one or more **`CalendarEntry`** objects, then add them to the calendar.

Calendar entries offer several optional property fields, making it easy to set and get commonly-used information such as whether it's an all-day event, a description that goes beyond the entry's title, a location, and a URL. If you need to add more properties to entries, you can add as many custom fields as needed by accessing the entry's extended properties that are stored in a HashMap. Entries also support GroupIDs, which

lets the user manage like entries in the calendar as a group. For example, it's possible to set all workout entries to the same GroupID, which means that we can reschedule all of them simultaneously by dragging and dropping or resizing one of the entries, as shown below in **Figure 6**.

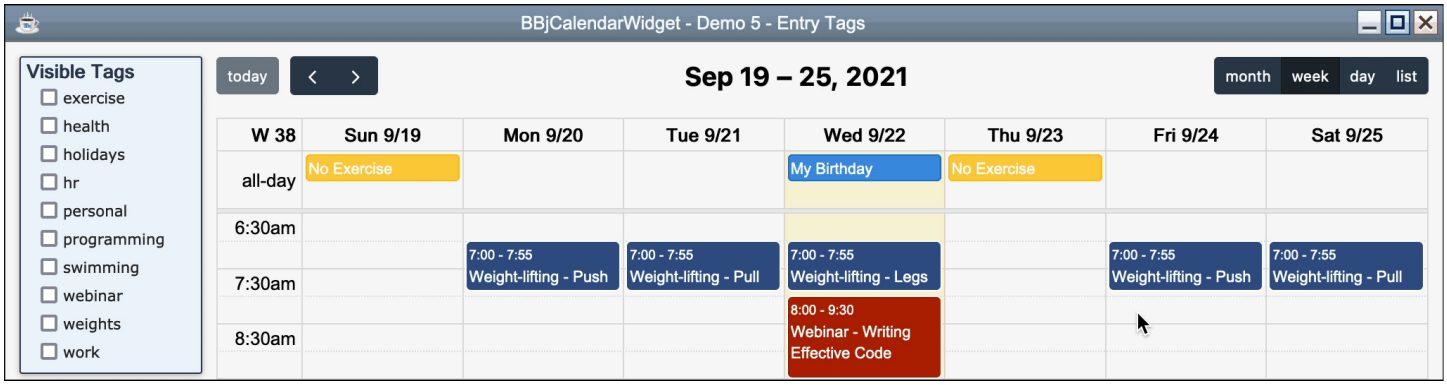


Figure 6. Modifying all weight-lifting entries at once

The entries also support any number of custom tags, and you can call methods on the calendar widget to hide or show entries that contain a particular tag. The CalendarEntry object has several methods to facilitate tag management, such as adding and removing tags, setting and getting tags, and methods to see if an entry contains a particular tag, one out of several, or contains all the queried tags. For example, you can easily toggle between showing work and/or personal appointments by attaching tags to the calendar's entries. **Figure 7** below demonstrates this concept by alternately hiding, then showing all entries that have been tagged as "exercise." Toggling the visibility of this tag affects multiple exercise-related entries, including the "Weight-lifting," "Swimming," and "No Exercise" entries, as they all contain the "exercise" tag.

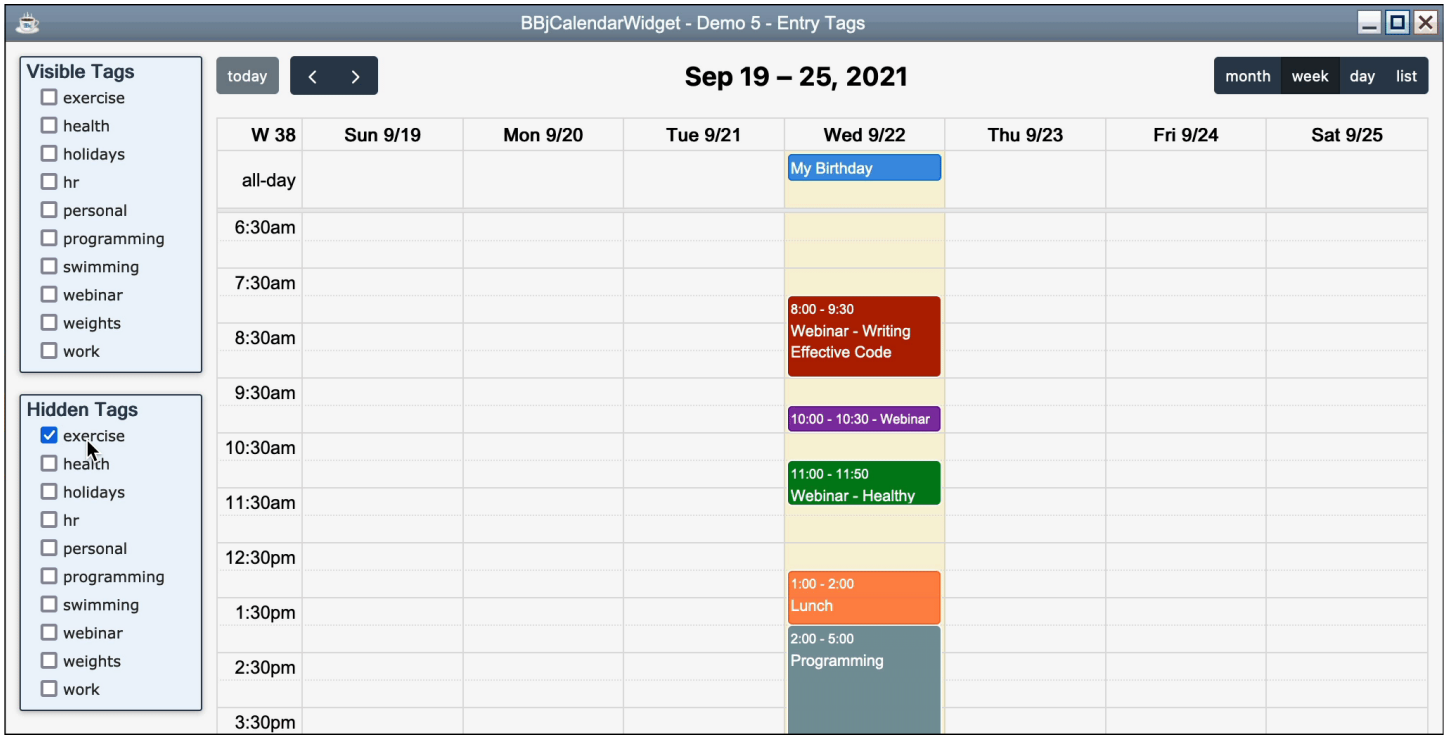


Figure 7. Alternately hiding and showing all entries that contain the "exercise" tag

Editable Entries

The BBjCalendarWidget wasn't meant to be a static display of appointments, although it's possible to lock it down and prevent users from modifying any of the entries on the calendar. Its real power comes into play when you configure it to be editable by the user. There are a few parameters that determine how editable it should be, and you can empower users to reschedule appointments by dragging and dropping an entry from one day or time slot to another. Users can also intuitively change an entry's duration by dragging its starting or ending edge, as shown in **Figure 8** below. It's even possible to move an entry into the "all-day" header, converting it from a timed entry into an all-day entry, as **Figure 8** demonstrates with the "Experiment with the BBjCalendarWidget" entry. And modifying events in this manner is possible in the month, week, and day grid views — basically any view except for the list-style views, which present the calendar appointments in a read-only agenda format.

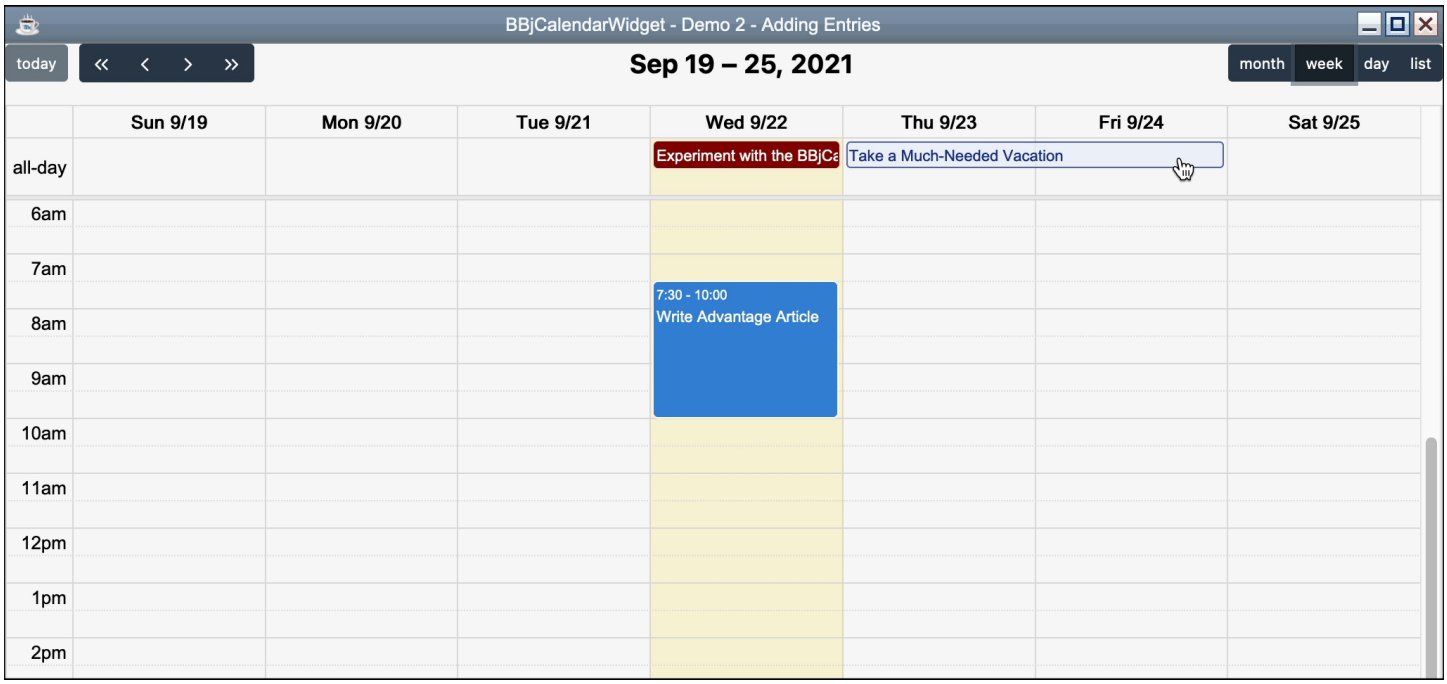


Figure 8. The user can easily modify entries by dragging and dropping or resizing them

User Events and Callbacks

Your application can register for the user's events with traditional callback syntax, which allows you to execute custom-tailored code after the user interacts with the calendar and its entries. This means that you can load the calendar with entries from a database and let the user change them. Whenever the user reschedules an entry or adjusts its duration, the BBjCalendarWidget executes your callback routine and provides a custom calendar event object as a **`CalendarEntryChangeEvent`**. When the user edits an entry in that manner, your application has access to two copies of the calendar entry from the event — the original version and the updated version that includes the user's changes. Your callback routine can then update the appointment data in the source database or file, persisting the user's modifications. There are many other callback events available,

so you can execute custom code whenever the user clicks on a date or appointment, modifies an entry or group of entries, resizes an entry, mouses in or out of an entry, etc.

Programmatic Navigation

By default, the calendar displays a toolbar on the top with navigation buttons on the left, as shown below in **Figure 9**. The user may click these buttons to move to the previous or next day, week, month, or year depending on the current view type. But it's also possible to navigate the calendar programmatically, and the widget offers methods to accomplish the same user navigation selections. By calling methods like **`navigatePrev()`** and **`navigateNext()`**, you can tell the calendar to display the next or previous month, week, or day. Other navigation methods instruct the calendar to go to today's date or any specified date in the past or future. Additionally, you can set the initial date to be displayed in the calendar and even show a live "now indicator." This draws a line in the time grid views that advances over time, so it's easy to tell which appointments have already passed and which are coming up soon. **Figure 9** shows the now indicator red arrow and line just before the 3:30 p.m. slot.

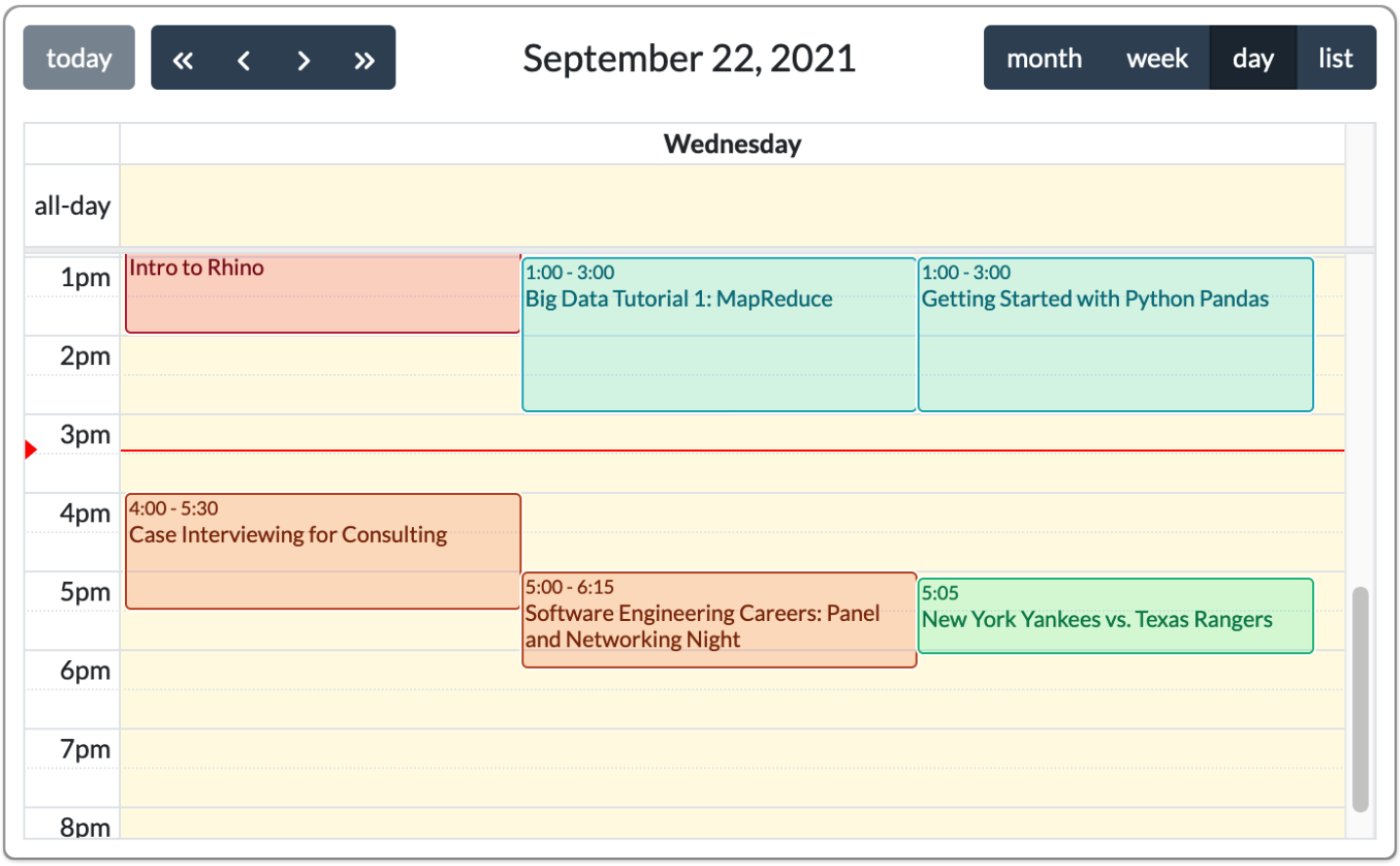


Figure 9. The now indicator at 3:30 pm separates past and future calendar entries

Summary

The concept of integrating the FullCalendar JavaScript library into a BBjHtmlView originated with a demo from TechCon 2017. The demo was somewhat primitive, focusing

on the potential of integrating third-party JavaScript libraries into a BBj application via the BBjHtmlView control without spending much time or code on a full-scale integration. Because the demo was well-received, BASIS created the BBjCalendarWidget — a full-blown plug-in that takes care of all the setup, custom JavaScript code, and event information communication for the application developer. The BBjCalendarWidget relieves the application developers from having to learn any JavaScript and provides a fully functional calendar control that's easy to add to new or existing BBj graphical applications. The widget works in GUI, BUI, and even BASIS' new Dynamic Web Client. With the release of the BBjCalendarWidget, BASIS delivers another powerful application Building Block component to your toolkit. Stay tuned for more articles in the series that will include a tutorial, cover the demos, and discuss some advanced topics.