Using fullCalendar

Edwood Ocasio

Using fullCalendar

Edwood Ocasio

Contents

Introduction

I believe `fullCalendar`[1] is the best JavaScript based calendar solution a developer could wish for. The features are compelling and the documentation is excellent. If you are reading this I guess you also agree. ;)

This book follows the same style as my `pyFPDF` book[2]. I share many code snippets and advice that you may find helpful while working with `fullCalendar`. Although the documentation is excellent and Google and StackOverflow are a developer's best friends, I think that by centralizing and organizing our knowledge in one place, a book for example, we can learn better our tools. This book will be of help to you as it has already been for me. Trying to teach a subject is one of the best ways to master it.

This book will be updated every time a learn something new and valuable about `fullCalendar`. The updates will be available to you at no additional cost.

I hope this book makes your life easier and helps you make money faster in those client projects that require a calendar solution.

All code samples are based on `fullCalendar` version 2.6.1 (2016-02-17)

> When you buy the book all code samples and resources will be available in a ZIP package linked from right here ("Introduction").

Regards,

Edwood Ocasio

[1]http://fullcalendar.io/
[2]https://leanpub.com/pythondoespdfpyfpdf

The basics of fullCalendar

A plain instance of fullCalendar

This sample has just enough code to have a working fullCalendar instance.

This is the minimal required JavaScript.

Listing fcbasic.js

```
// Create calendar when document is ready
$(document).ready(function() {

    // We will refer to $calendar in future code
    var $calendar = $("#calendar").fullCalendar();
    }
);
```

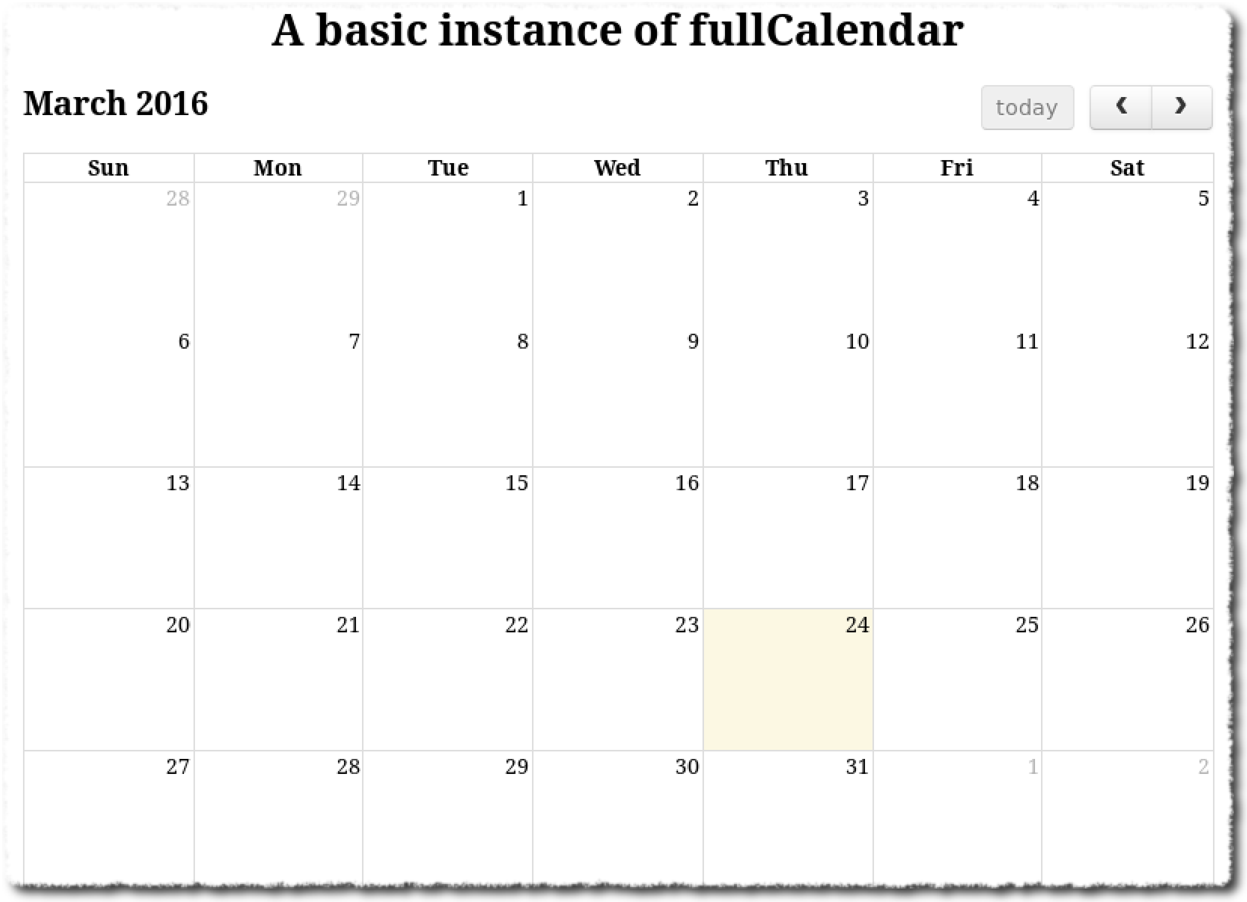
This is the HTML page:

Listing fcbasic.html

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
        <title>Using fullCalendar</title>

        <!--
        All CSS and JS files needed for fullCalendar. Change you
        files locations accordingly.
        -->

        <link rel='stylesheet' href='fullcalendar/fullcalendar.css' />
        <script src='fullcalendar/lib/jquery.min.js'></script>
        <script src='fullcalendar/lib/moment.min.js'></script>
        <script src='fullcalendar/fullcalendar.min.js'></script>

        <!-- Some styles for the calendar and other elements -->
        <style>
        #calendar {
            width: 80%;
            display: block;
            margin-left: auto;
            margin-right: auto;
            }

        .centered {
            text-align: center;
        }

        </style>
    </head>

<body>
<h1 class="centered">A basic instance of fullCalendar</h1>

<!-- The calendar container -->
<div id="calendar"></div>

<!-- Calendar creation script -->
<script src="fcbasic.js"></script>

</body>
</html>
```

We could have included the JavaScript code inside the HTML page but further in the book the code will get more involved and the HTML will just get in the way.

This gives us a month view calendar with simple navigation and plain appearance:



More navigation buttons

The following example demonstrates all the navigation buttons available in `fullCalendar`:

Listing fc-full-navigation.js

```
// Create calendar when document is ready
$(document).ready(function() {

    // We will refer to $calendar in future code
    var $calendar = $("#calendar").fullCalendar(

        { // Start of options
            header: {
                left: 'prevYear,nextYear',
                center: 'title',
                right: 'today,month,agendaDay,agendaWeek prev,next'
            },
        } // End of options

    );
});
```

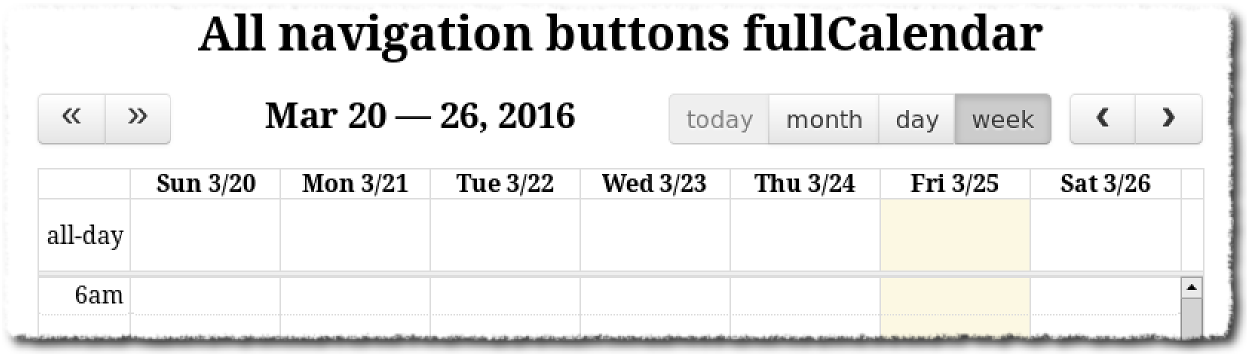
This is the HTML page:

Listing fc-full-navigation.html

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
        <title>Using fullCalendar</title>

        <!--
        All CSS and JS files needed for fullCalendar. Change you
        files locations accordingly.
        -->

        <link rel='stylesheet' href='fullcalendar/fullcalendar.css' />
        <script src='fullcalendar/lib/jquery.min.js'></script>
        <script src='fullcalendar/lib/moment.min.js'></script>
        <script src='fullcalendar/fullcalendar.js'></script>

        <!-- Some styles for the calendar and other elements -->
        <style>
        #calendar {
            width: 80%;
            display: block;
            margin-left: auto;
            margin-right: auto;
            }

        .centered {
            text-align: center;
        }

        </style>
    </head>

<body>
<h1 class="centered">All navigation buttons fullCalendar</h1>

```

```
<!-- The calendar container -->
<div id="calendar"></div>

<!-- Calendar creation script -->
<script src="fc-full-navigation.js"></script>

</body>
</html>
```

Now the calendar looks like this:



To change the position of the title and buttons just move them in the `header` object from or within the `left`, `right` or `center` properties. Use a comma to join buttons that should be adjacent and put spaces between those that should be separated by a gap.

Hiding days

In some applications you need to hide days because they are not needed. For example, weekends (Saturday/Sunday) can be hidden in an appointment calendar to avoid been used or to make clear they are not service days.

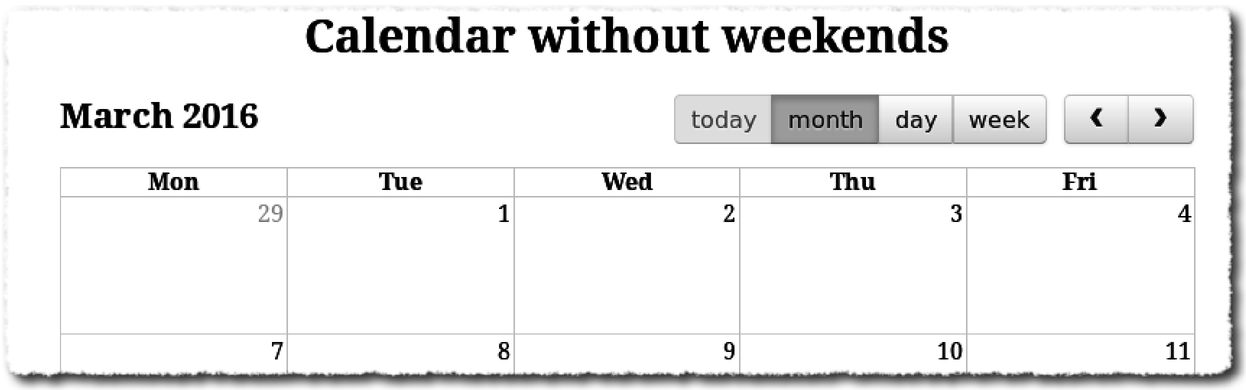
In the case of weekends `fullCalendar` has already a boolean property named `weekends`. If you set it to `false` weekends will not be shown:

Listing fc-no-weekends.js

```
// Create calendar when document is ready

$(document).ready(function() {
    // We will refer to $calendar in future code
    var $calendar = $("#calendar").fullCalendar(

            { // Start of options
                // Do not show Saturday/Sunday
                weekends : false,

                // Calendar header and navigation buttons
                header: {
                    left:   'title',
                    center: '',
                    right:  'today,month,agendaDay,agendaWeek prev,next'
                },

            } // End of options

        );
    }
);
```

Th HTML page follows the same pattern as in the previous sections. Basically we just changed the source of the calendar script. It is included in the resource package for the book.

Now the calendar looks like this:

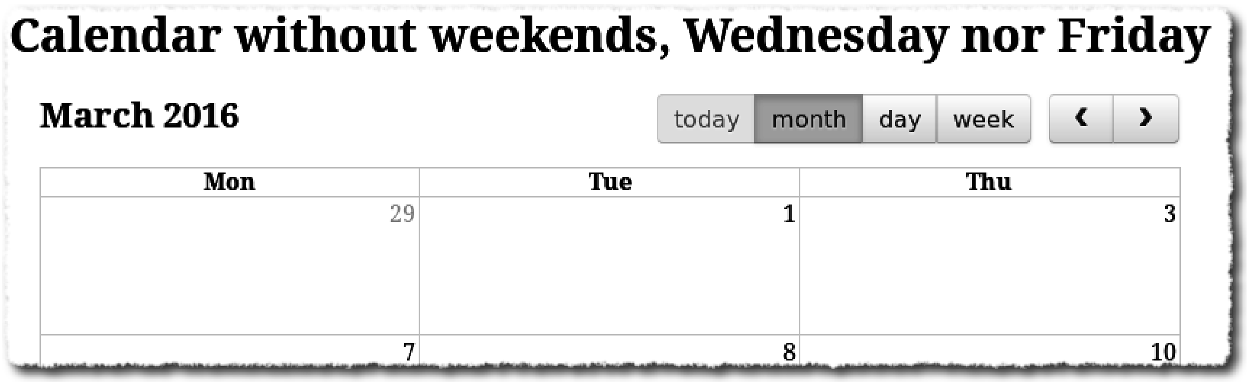


Now let us hide also Wednesdays and Fridays:

Listing fc-no-wend-friday.js

```
// Create calendar when document is ready
$(document).ready(function() {
    // We will refer to $calendar in future code
    var $calendar = $("#calendar").fullCalendar(

        { // Start of options
            // Do not show Saturday/Sunday
            weekends: false,

            // Hide Wednesday (3) and Friday (5). Week starts with
            // Sunday (0)
            hiddenDays: [3, 5],

            // Calendar header and navigation buttons
            header: {
                left: 'title',
                center: '',
                right: 'today,month,agendaDay,agendaWeek prev,next'
            },

        } // End of options

    );
});
```

Now the calendar looks like this without those days:



Changing locale

`fullCalendar` provides good support for internationalization. We can change the locale of our calendars and display dates and times according to different international regions. There are 52 language files distributed with `fullCalendar` in the folder `lang`:

> ar.js, ar-ma.js, ar-sa.js, ar-tn.js, bg.js, ca.js, cs.js, da.js, de-at.js, de.js, el.js, en-au.js, en-ca.js, en-gb.js, en-ie.js, en-nz.js, es.js, fa.js, fi.js, fr-ca.js, fr-ch.js, fr.js, he.js, hi.js, hr.js, hu.js, id.js,is.js, it.js, ja.js, ko.js, lt.js, lv.js, nb.js, nl.js, pl.js, pt-br.js, pt.js, ro.js, ru.js, sk.js, sl.js, sr-cyrl.js, sr.js, sv.js, th.js, tr.js, uk.js, vi.js, zh-cn.js, zh-tw.js

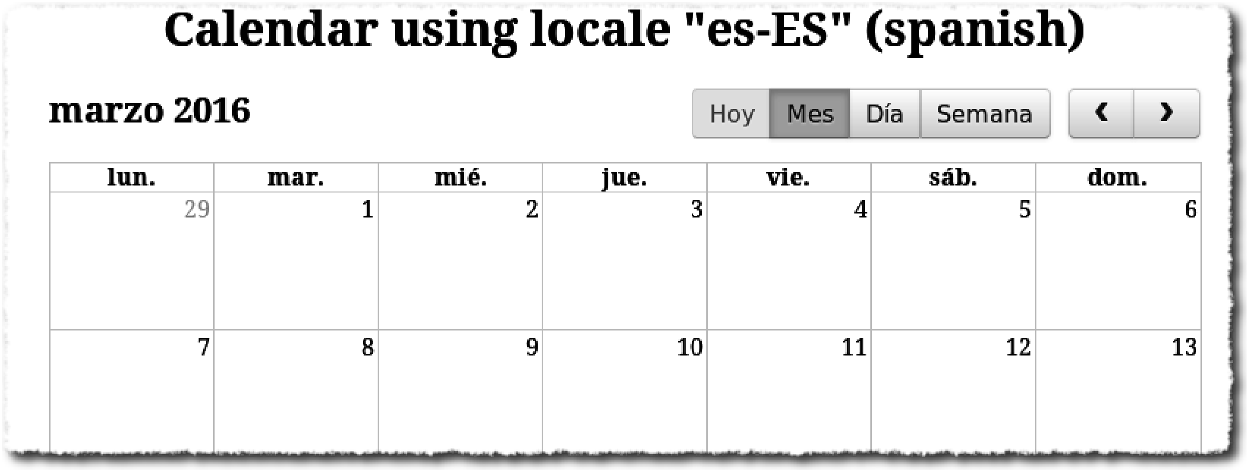
For this example we will change the locale to Spanish ("es-ES"). The change only needs to happen in the HTML page. What we need to do is to include the language files for `fullCalendar` **in the appropriate order**:

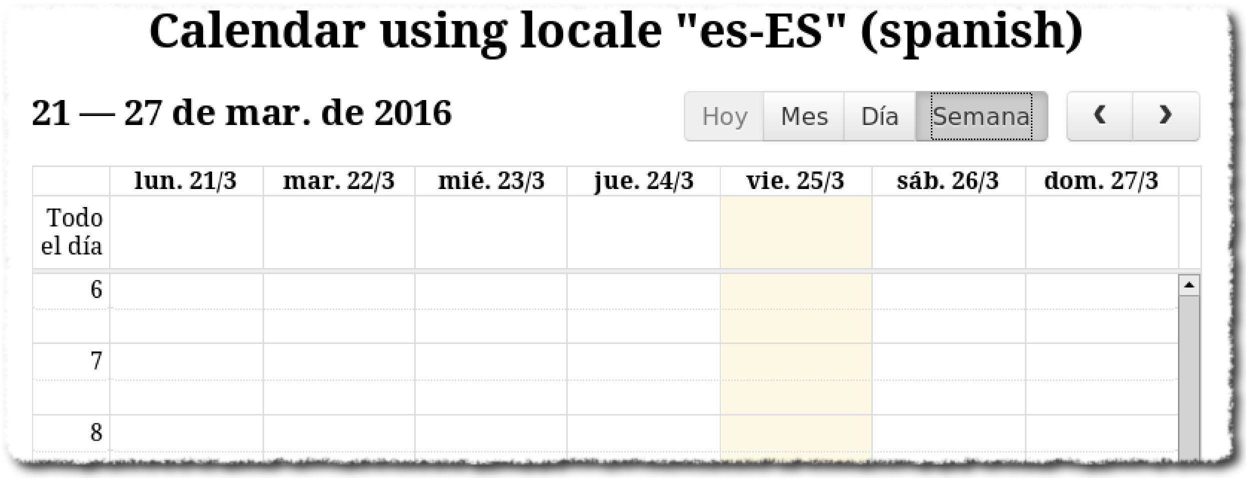
```
<!-- Spanish language file loaded after fullcalendar.js -->
<script src='fullcalendar/lang/es.js'></script>
```

To display correctly all international characters when changing locales it is important that the page charset be declared UTF-8. That is why we have this meta tag on top of every HTML page:

```
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
```

The calendar looks like this in Spanish:





The distribution package of `fullCalendar` includes a more interesting example in the `demo` folder that allows access to all language files.

Managing events

Adding events

`fullCalendar` provides a few ways to add events: event arrays, an event generation function, a JSON feed or a Google Calendar feed.

But before we start filling the calendar with events …

We must talk about date formats

The date format is very important. `fullCalendar` since version 2.0 depends on the awesome MomentJS library. The dates a `moment` object understands by default are in the ISO 8601 format[3]. If they are not, MomentJS falls back to a regular JavaScript Date object, but it issues a warning that the fall back could be removed in the future.

So, it is always recommended to use the ISO 8601 format in the JSON objects we feed `fullCalendar`. For dates use “YYYY-MM-DD” and for combined date and time use one of these: “YYYY-MM-DD hh:mm:ss” or “YYYY-MM-DDThh:mm:ss”, where a space or the letter “T” separates the time from the date. The time uses the **24-hour clock format**.

One of the common errors when formatting dates for `fullCalendar` is forgetting to **zero-pad** the numbers, in the date or time. This is a valid date: “2016-01-04 08:30:00”, but not this one “2016-1-4 8:30:00” because some numbers are not zero-padded. Some times finding this bug in `fullCalendar` is like searching for a sneaky semicolon in JavaScript code.

When the events are generated by a server side application you must take care of formatting the `start` and `end` dates and times. For example, in Python we will always use this formatting strings: “%Y-%m-%d” for dates and “%Y%m-%d %H:%M:%S” for date and time.

I should mention that MomentJS is very capable of parsing[4] almost any reasonable date/time string, but I think we should take responsibility for returning what it expects by default. Of course, this will not always be possible, specially if we do not have control of the application generating the events.

Let’s continue.

Using an event array

In this code we use an event array to create events in the calendar:

[3]http://en.wikipedia.org/wiki/ISO_8601
[4]http://momentjs.com/docs/#/parsing/

Listing fc-events-array.js

```
// Create calendar when document is ready

$(document).ready(function() {

    // Array of events
    var events_array = [{
            title: "Event 1",
            // Set to 1st of the month at 12:00 am
            start: moment().startOf('month'),
            // Set to en the 1st of the month at 1:30 am
            end: moment().startOf('month').add(90, 'minutes'),
            color: "red"
        }, {
            title: "Event 2",
            // Set to 1st of the month at 12:00 am
            start: moment().startOf('month').add(1, 'days'),
            // Set to end the 1st of the month at 3:00 am
            end: moment().startOf('month').add({
                'days': 1,
                'hours': 3
            }),
            color: "green"
        }, {
            title: "Multi-day event",
            // Set to start the 1st of the month
            start: moment().startOf('month'),
            // Set to end one week after the start of the month
            end: moment().startOf('month').add(1, 'weeks'),
            color: "blue",
            // This is an all-day event
            allDay: true
        },

    ];
    // We will refer to $calendar in future code
    var $calendar = $("#calendar").fullCalendar(

        { // Start of options
            header: {
                left: 'title',
                center: '',
                right: 'today,month,agendaDay,agendaWeek prev,next'
            },

            events: events_array
        } // End of options

    );
});
```

Notice the use of the MomentJS library to fabricate the `start` and `end` dates for each event. That is not `fullCalendar`'s functionality, but since `fullCalendar` 2.* requires MomentJS, those utilities are at our disposal.

Let's delve for a "moment" in MomentJS.

Study this line:

```
moment().startOf('month').add(90,'minutes')
```

What do you think it does?

That line creates a date object representing the first day of the current month at 12:00 am and then it adds 90 minutes to it. The final date should be (if we are in march 2016): "2016-03-01 01:30:00".

This other line creates a date also at the first day of the month and then adds one day and 3 hours to that date:

```
moment().startOf('month').add({'days':1, 'hours':3 })
```

The final date should be (if we are in march 2016): "2016-03-02 03:00:00".

I used `startOf()` just to force the events to appear at the top of the calendar for your viewing pleasure.

To learn more about manipulating dates and times with MomentJS check the Moment.js Documentation[5]

This is the HTML page:

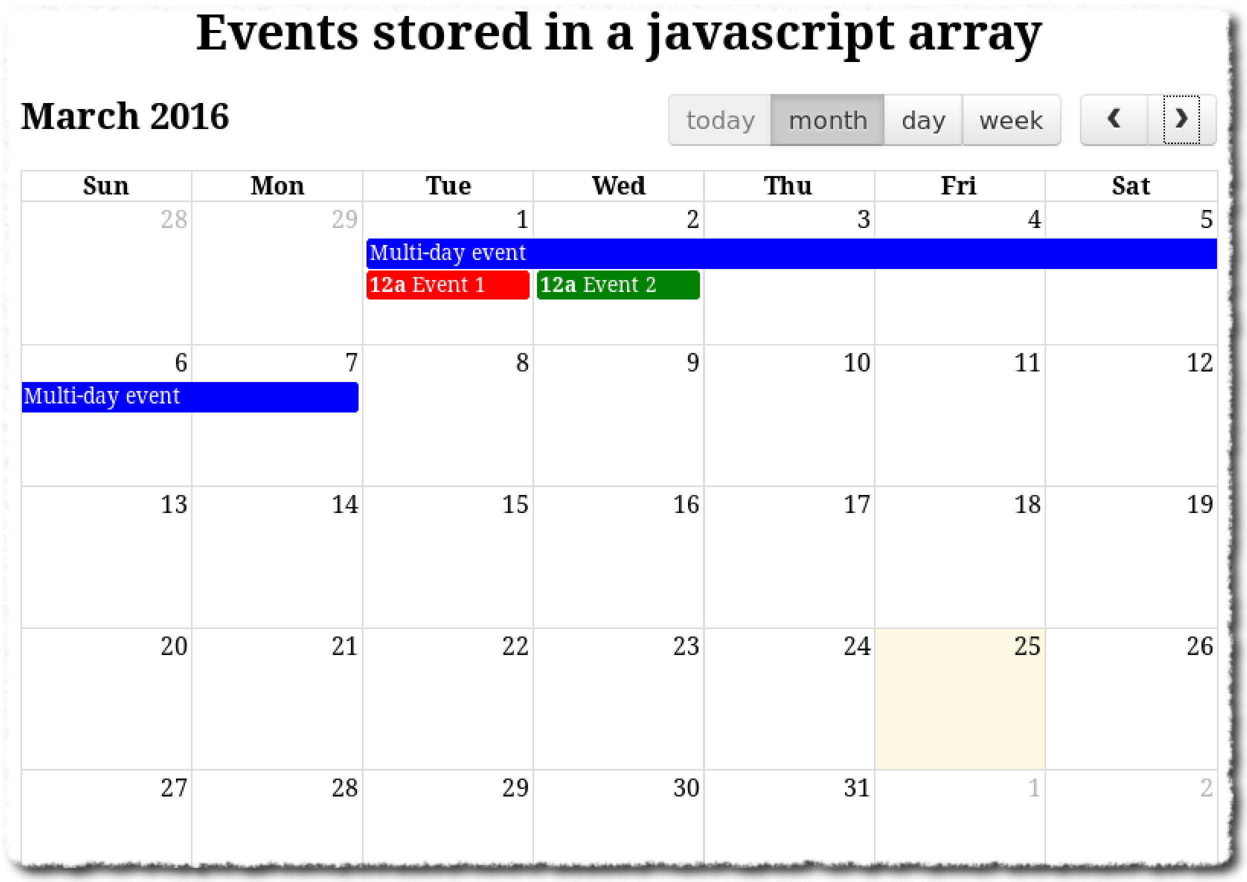
[5] http://momentjs.com/docs/#/manipulating/

Listing fc-events-array.html

```html
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
        <title>Using fullCalendar</title>

        <!--
        All CSS and JS files needed for fullCalendar. Change you
        files locations accordingly.
        -->

        <link rel='stylesheet' href='fullcalendar/fullcalendar.css' />
        <script src='fullcalendar/lib/jquery.min.js'></script>
        <script src='fullcalendar/lib/moment.min.js'></script>
        <script src='fullcalendar/fullcalendar.js'></script>

        <!-- Some styles for the calendar and other elements -->

        <style>
        #calendar {
            width: 80%;
            display: block;
            margin-left: auto;
            margin-right: auto;
            }

        .centered {
            text-align: center;
        }

        </style>
    </head>

<body>
<h1 class="centered">Events stored in a JavaScript array</h1></h1>

<!-- The calendar container -->
<div id="calendar"></div>

<!-- Calendar creation script -->

<script src="fc-events-array.js"></script>
```

```

</body>
</html>
```

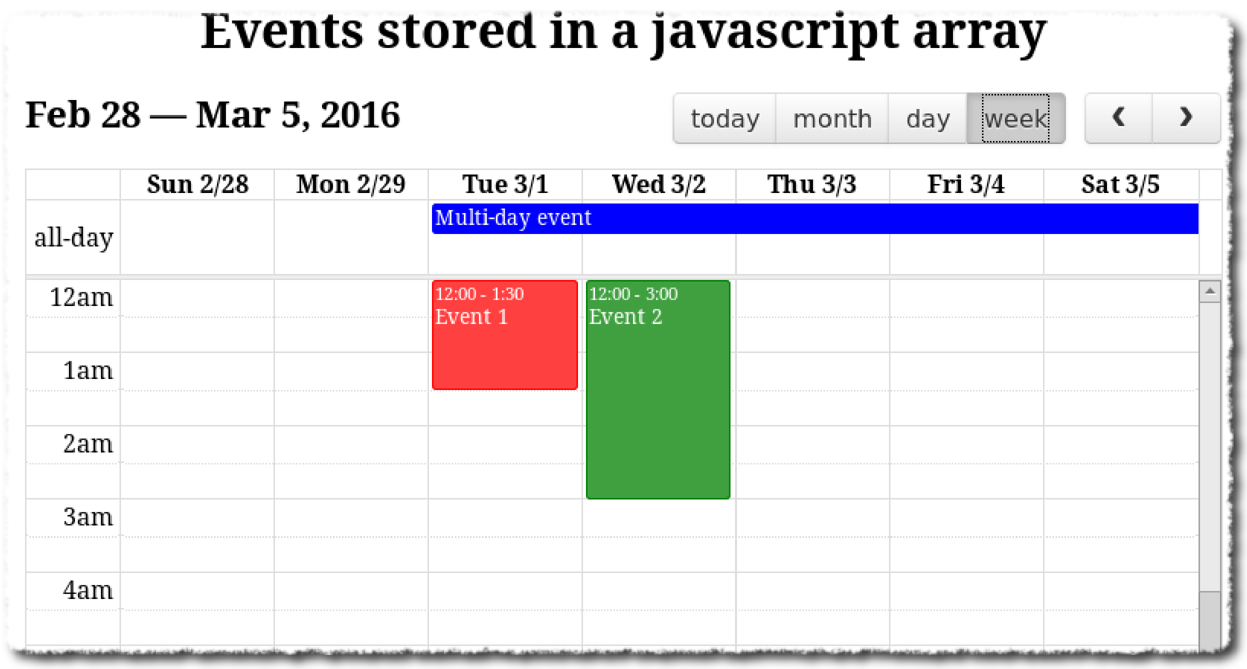
These are the views:

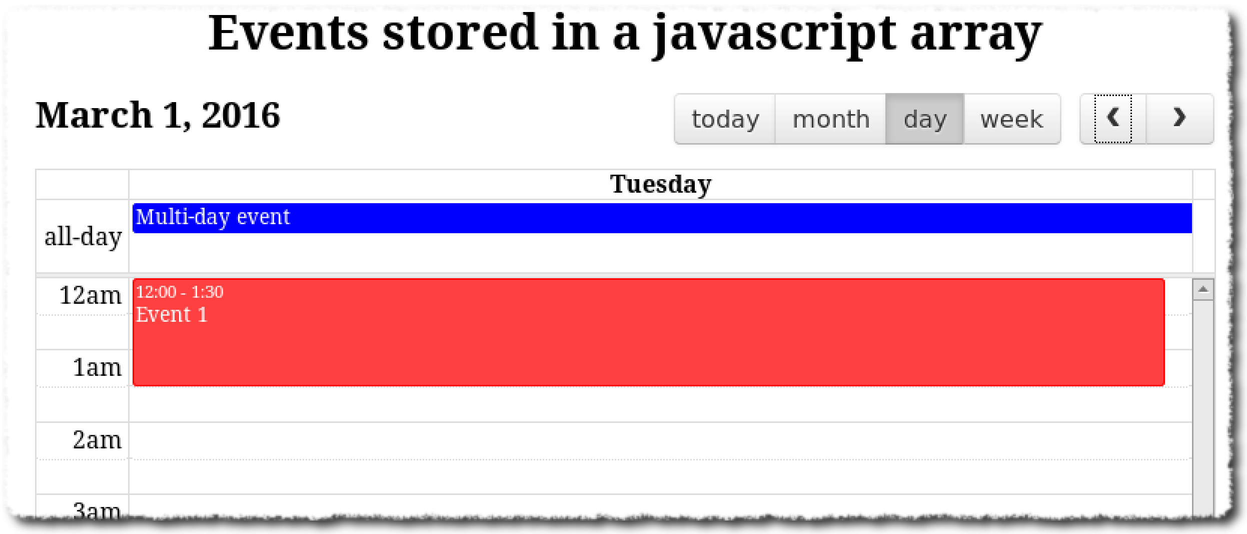


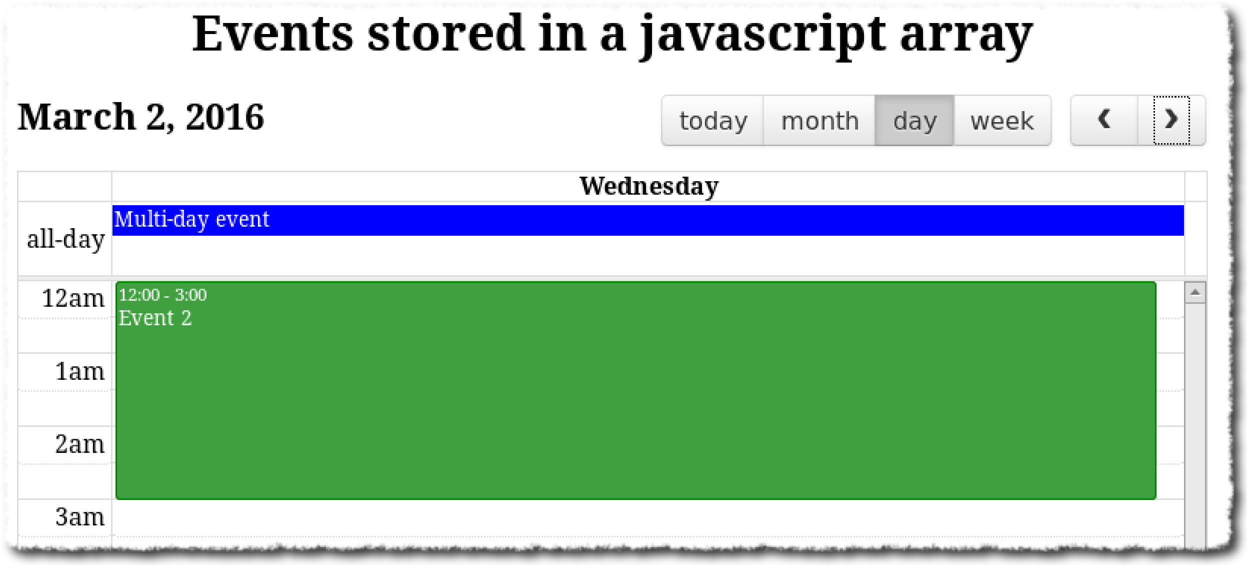
Events stored in a javascript array

March 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	1 Multi-day event 12a Event 1	2 12a Event 2	3	4	5
6 Multi-day event	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2


Events stored in a javascript array
Feb 28 — Mar 5, 2016
today
month
day
week
Sun 2/28
Mon 2/29
Tue 3/1
Wed 3/2
Thu 3/3
Fri 3/4
Sat 3/5
all-day
Multi-day event
12am
1am
2am
3am
4am
12:00 - 1:30
Event 1
12:00 - 3:00
Event 2


Events stored in a javascript array
March 1, 2016
today
month
day
week
Tuesday
all-day
Multi-day event
12am
1am
2am
3am
12:00 - 1:30
Event 1



Selection of date and time slots

By default `fullCalendar` does nothing when we click on a date/time slot or try to select a range of them. To enable selection we must set the property `selectable` to `true`.

Once that is done we can respond to user selections through the `eventClick` callback.

With the following code we respond with a JavaScript alert box to a click or a range selection in any view of the calendar:

Listing fc-select.js

```
// Create calendar when document is ready
$(document).ready(function() {

 // We will refer to $calendar in future code
 var $calendar = $("#calendar").fullCalendar({
   // Start of calendar options
   header: {
    left: 'title',
    center: '',
    right: 'today,month,agendaDay,agendaWeek prev,next'
   },

   // Make possible to respond to clicks and selections
   selectable: true,

   // This is the callback that will be triggered when a selection is made
   select: function(start, end, jsEvent, view) {
    alert(start.format("MM/DD/YYYY hh:mm a") + " to " + end.format("MM/DD/YYYY h\
h:mm a") + " in view " + view.name);

   }
  } // End of calendar options
 );
});
```

The code also shows how to format the date and time objects associated to each calendar slot:

```
start.format("MM/DD/YYYY hh:mm a")
```

To learn more about date and time formatting visit the MomentJS library documentation[6]

Notice also that the current view name is obtained from `view.name`.

This is the HTML page:

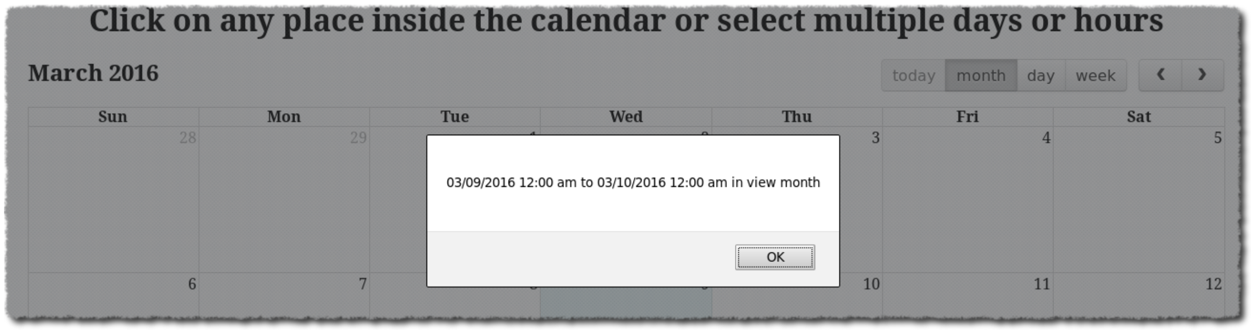
Listing fc-select.html

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
        <title>Using fullCalendar</title>

        <!--
        All CSS and JS files needed for fullCalendar. Change you
        files locations accordingly.
        -->

        <link rel='stylesheet' href='fullcalendar/fullcalendar.css' />
        <script src='fullcalendar/lib/jquery.min.js'></script>
        <script src='fullcalendar/lib/moment.min.js'></script>
        <script src='fullcalendar/fullcalendar.js'></script>

        <!-- Some styles for the calendar and other elements -->

```

[6]http://momentjs.com/docs/#/displaying/

```
        <style>
        #calendar {
            width: 80%;
            display: block;
            margin-left: auto;
            margin-right: auto;
            }

        .centered {
            text-align: center;
        }

        </style>
    </head>

<body>
<h1 class="centered">Click on any place inside the calendar or select multiple d\
ays or hours</h1></h1>


<!-- The calendar container -->
<div id="calendar"></div>

<!-- Calendar creation script -->
<script src="fc-select.js"></script>

</body>
</html>
```

This is what you should see after clicking on a calendar slot or selecting many slots by dragging the mouse (Firefox web browser on Linux):

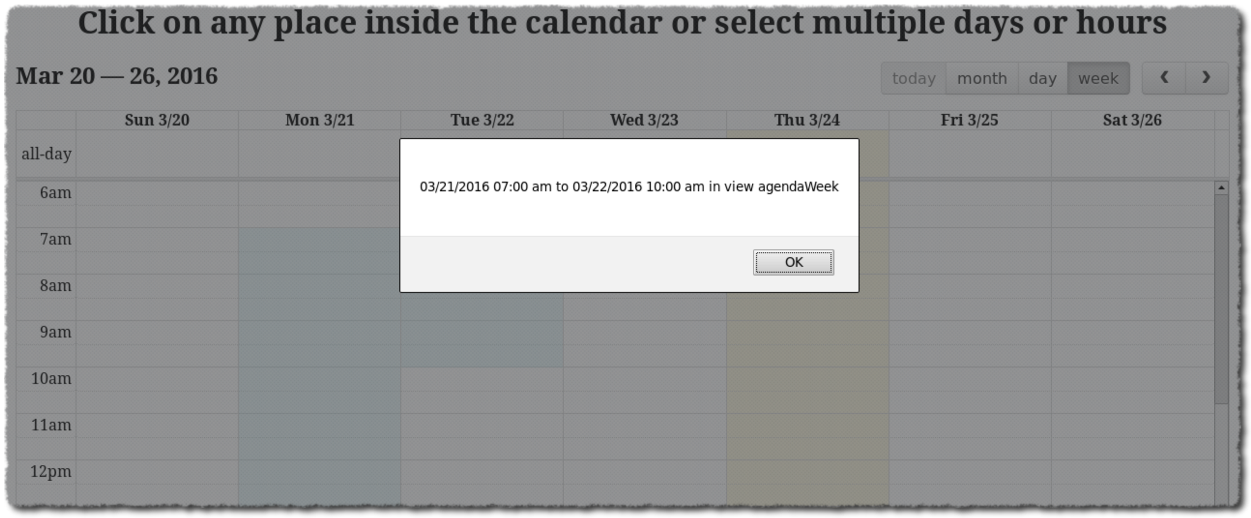


Click on any place inside the calendar or select multiple days or hours

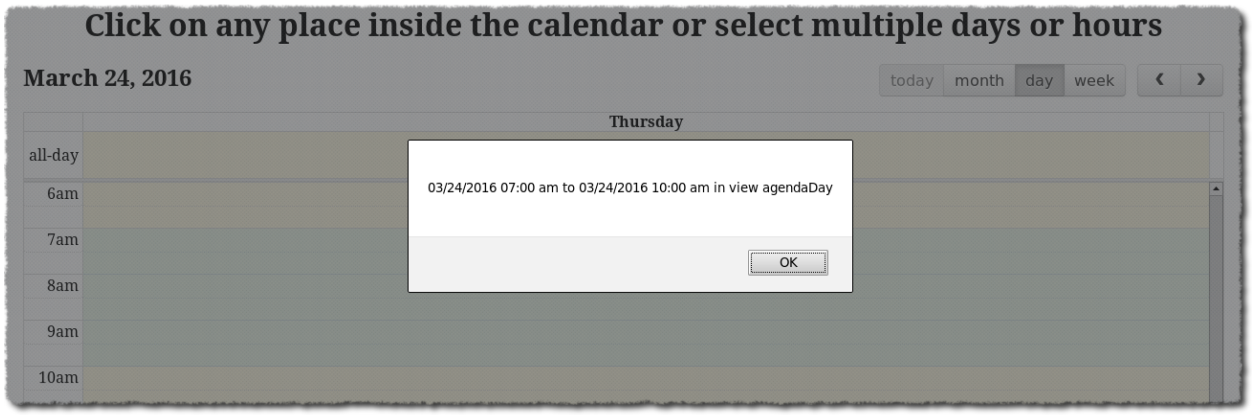
March 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	1	2	3	4	5
6	7	8	9	10	11	12

03/09/2016 12:00 am to 03/10/2016 12:00 am in view month



03/21/2016 07:00 am to 03/22/2016 10:00 am in view agendaWeek


Click on any place inside the calendar or select multiple days or hours
March 24, 2016
today
month
day
week
Thursday
all-day
6am
7am
8am
9am
10am
OK

03/24/2016 07:00 am to 03/24/2016 10:00 am in view agendaDay

Creating an event interactively

The following sample code will allow us to ask for a title for an event, create the event and then store it in the `fullCalendar` internal array of events. You can navigate away from the day of the event to other month, week or year and it will be kept in memory, unless you refresh the page.

Listing fc-interactive-event.js

```
// Create calendar when document is ready
$(document).ready(function() {

 // We will refer to $calendar in future code
 var $calendar = $("#calendar").fullCalendar({
   // Start of calendar options
   header: {
    left: 'title',
    center: '',
    right: 'today,month,agendaDay,agendaWeek prev,next'
   },

   // Make possible to respond to clicks and selections
   selectable: true,

   // This is the callback that will be triggered when a selection is made.
   // It gets start and end date/time as part of its arguments
   select: function(start, end, jsEvent, view) {

     // Ask for a title. If empty it will default to "New event"
     var title = prompt("Enter a title for this event", "New event");

     // If did not pressed Cancel button
     if (title != null) {
      // Create event
      var event = {
       title: title.trim() != "" ? title : "New event",
       start: start,
       end: end
      };


      // Push event into fullCalendar's array of events
      // and displays it. The last argument is the
      // "stick" value. If set to true the event
      // will "stick" even after you move to other
      // year, month, day or week.

      $calendar.fullCalendar("renderEvent", event, true);
     };
     // Whatever happens, unselect selection
     $calendar.fullCalendar("unselect");

    } // End select callback
  } // End of calendar options
 );
});
```

Notice the use of the following methods:

```
$calendar.fullCalendar("renderEvent", event, true);
```

That call to `renderEvent` displays the event in the calendar and sets the `stick` argument value to `true` to keep the new events from disappearing when we navigate away from them.

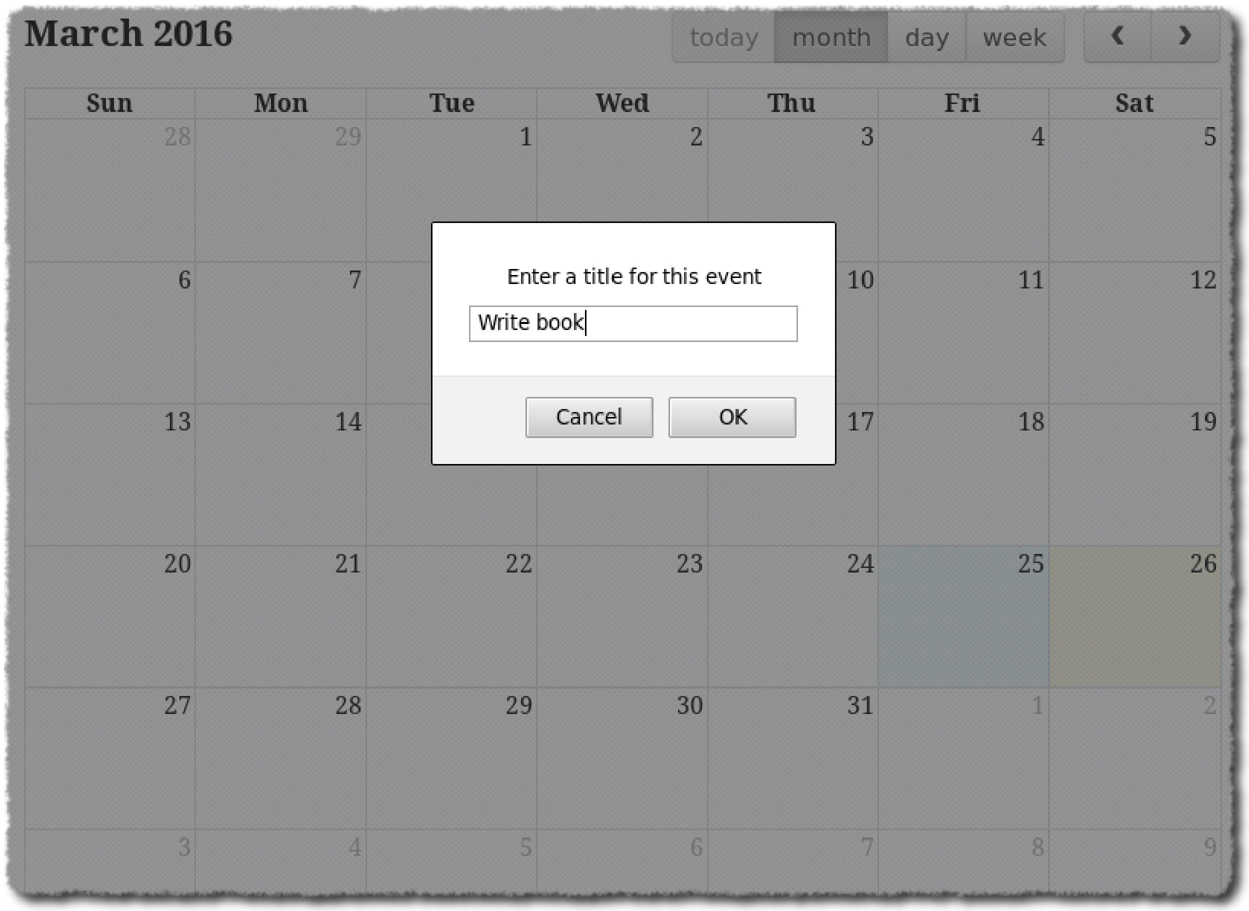
This call to the `unselect` method removes the selection after the event is created or after the action is canceled:

```
$calendar.fullCalendar("unselect");
```

In the HTML page just change the source of the script tag:

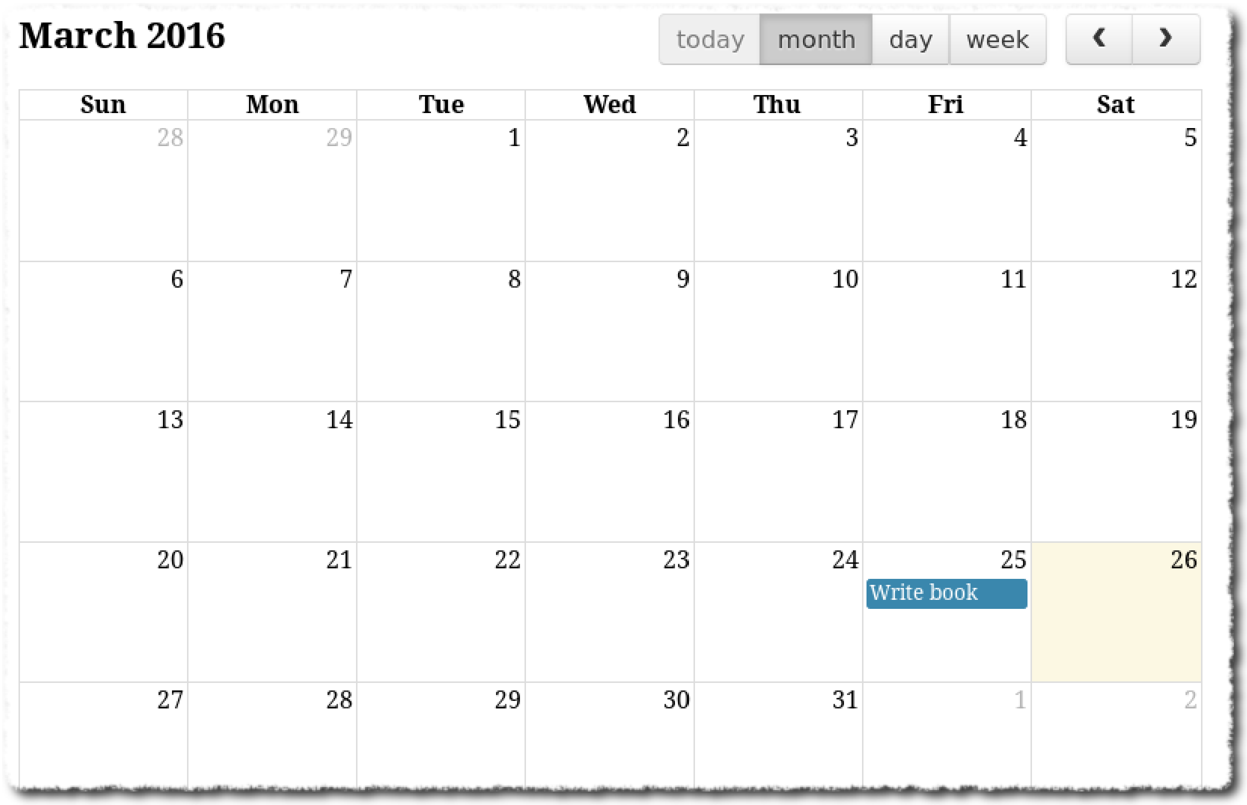
```
<script src="fc-interactive-event.js"></script>
```

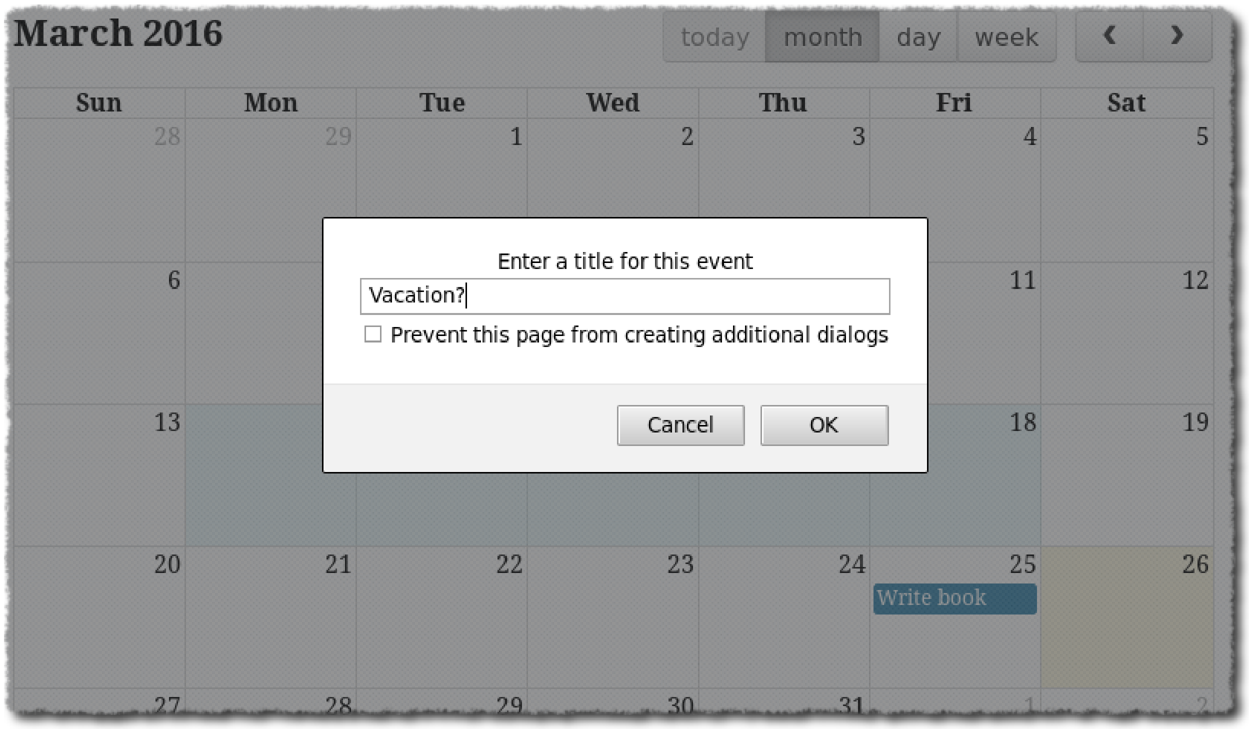
Some views of the action:



Enter a title for this event

Write book



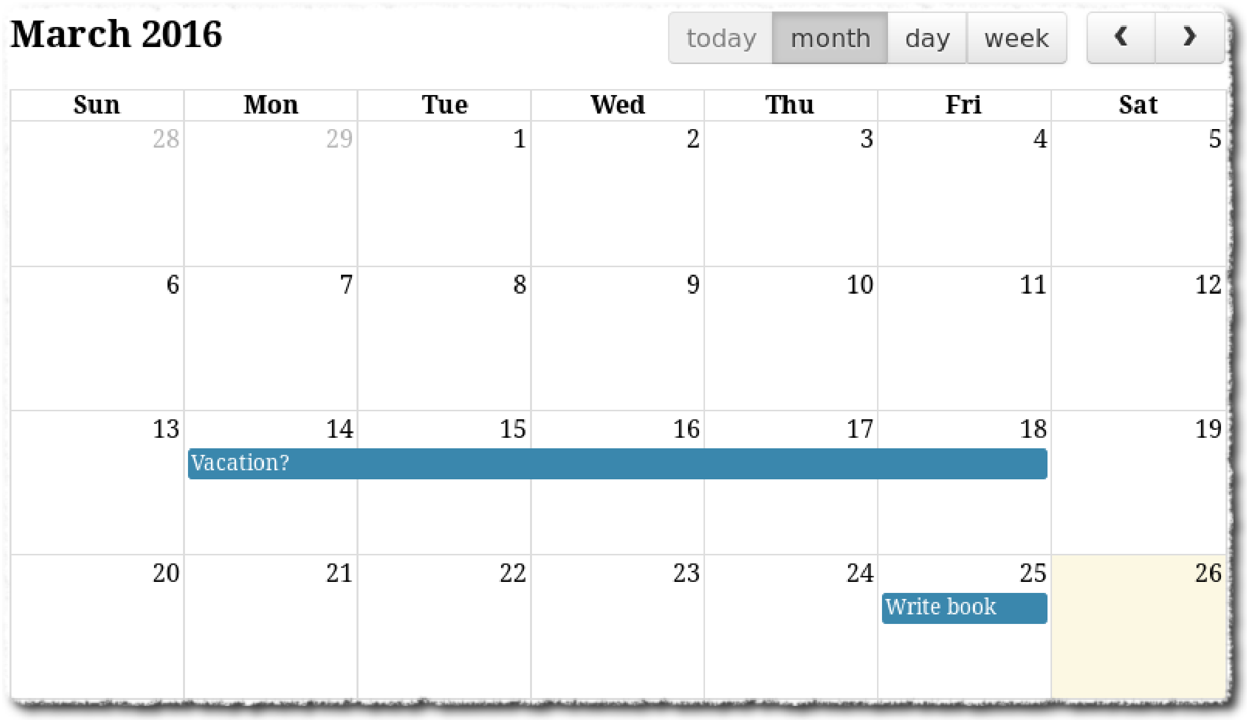


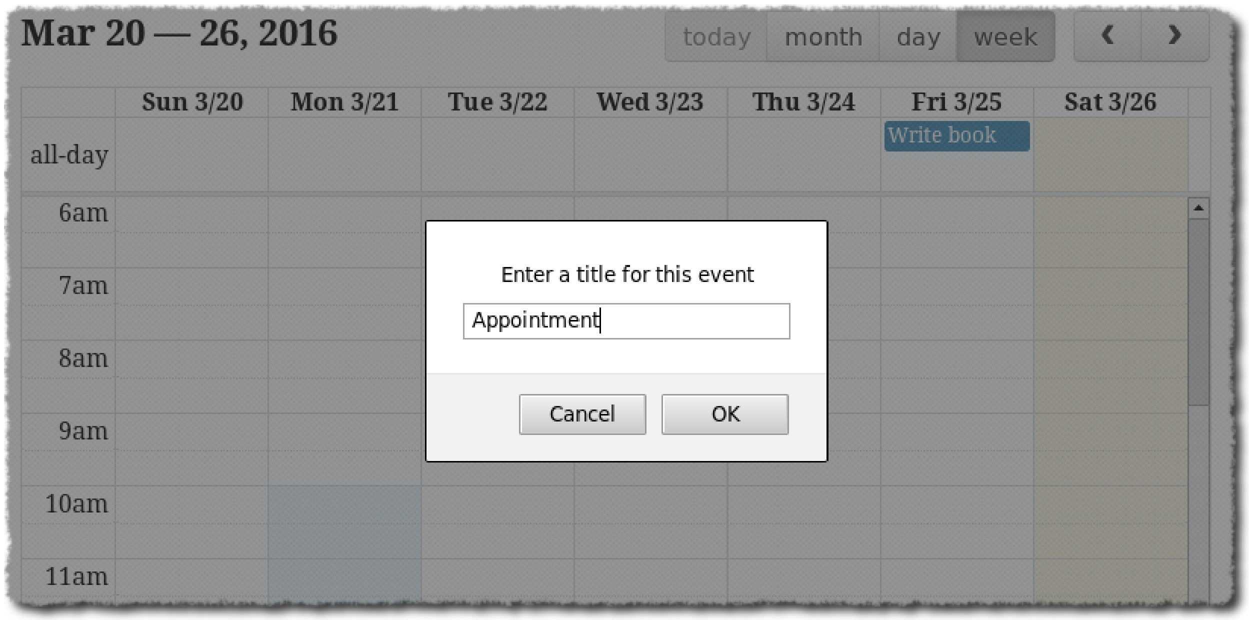
Enter a title for this event

Vacation?

☐ Prevent this page from creating additional dialogs

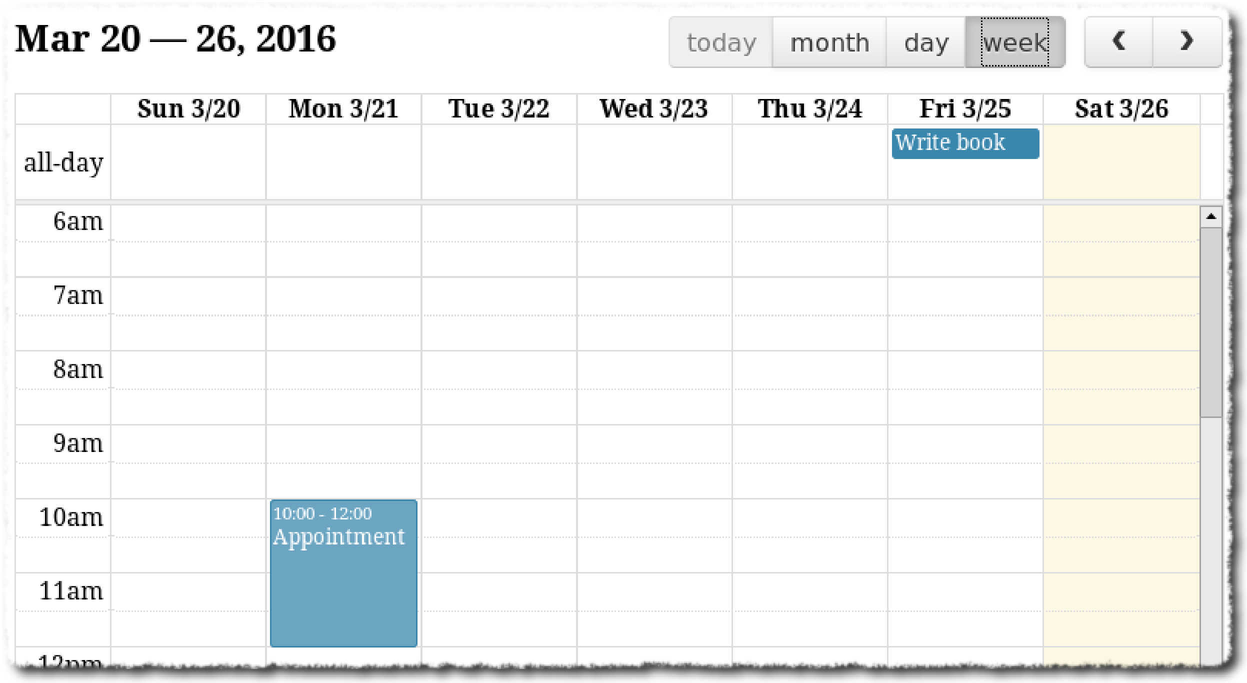
Cancel OK





Enter a title for this event

Appointment



Modifying events

There is not much to be done here, basically just tell `fullCalendar` that it should allow events to be `editable`.

Really. That's it.

To make the event responsive to clicks and to mouse dragging we must set the `editable` flag to `true`. The `editable` flag can be set globally for all events or locally for each event. We will go for the global flag in this example.

The code also shows how to edit the title of the event when we click on it.

Listing fc-modify-event-simple.js

```
// Create calendar when document is ready
$(document).ready(function() {

 // We will refer to $calendar in future code
 var $calendar = $("#calendar").fullCalendar({
   // Start of calendar options
   header: {
    left: 'title',
    right: 'today,month,agendaDay,agendaWeek prev,next'
   },

   // Make possible to respond to clicks and selections
   selectable: true,

   // This is the callback that will be triggered when a selection is made.
   // It gets start and end date/time as part of its arguments
   select: function(start, end, jsEvent, view) {

     // Ask for a title. If empty it will default to "New event"
     var title = prompt("Enter a title for this event", "New event");

     // If did not pressed Cancel button
     if (title != null) {
      // Create event
      var event = {
       title: title.trim() != "" ? title : "New event",
       start: start,
       end: end
      };


      // Push event into fullCalendar's array of events
      // and displays it. The last argument is the
      // "stick" value. If set to true the event
      // will "stick" even after you move to other
      // year, month, day or week.

      $calendar.fullCalendar("renderEvent", event, true);
     };
     // Whatever happens, unselect selection
     $calendar.fullCalendar("unselect");

    }, // End select callback

    // Make events editable, globally
    editable : true,

    // Callback triggered when we click on an event

    eventClick: function(event, jsEvent, view){
     // Ask for a title. If empty it will default to "New event"
     var newTitle = prompt("Enter a new title for this event", event.title);

    // If did not pressed Cancel button
    if (newTitle != null) {
        // Update event
        event.title = newTitle.trim() != "" ? newTitle : event.title;

        // Call the "updateEvent" method
        $calendar.fullCalendar("updateEvent", event);

       }
     } // End callback eventClick
   } // End of calendar options
 );
});
```

In the HTML page just change the source of the script tag:

```
<script src="fc-modify-event-simple.js"></script>
```

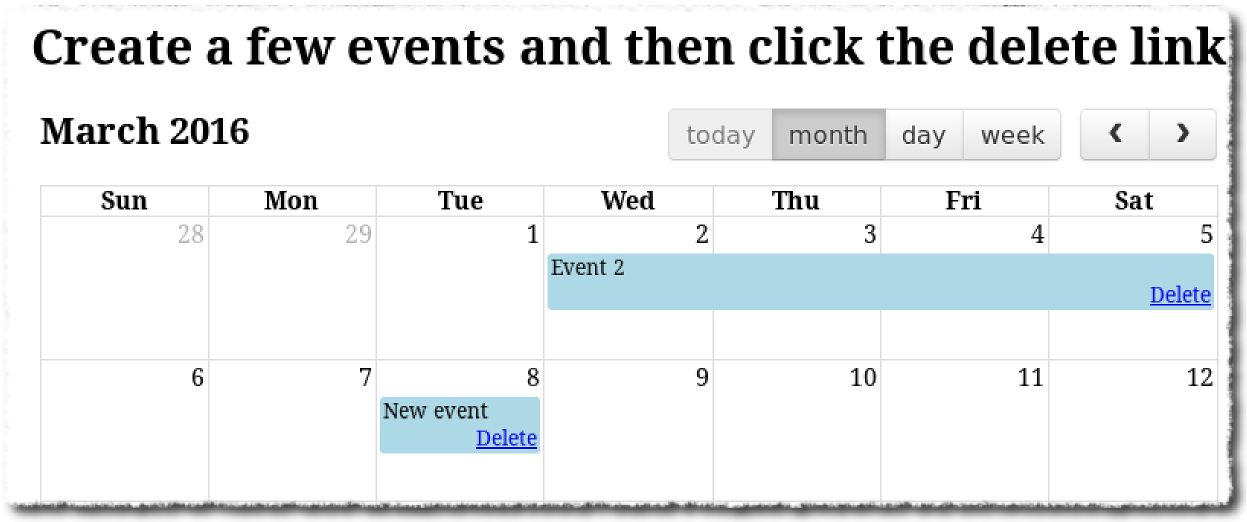
Now create some events and then then click, extend, contract, drag or drop them.

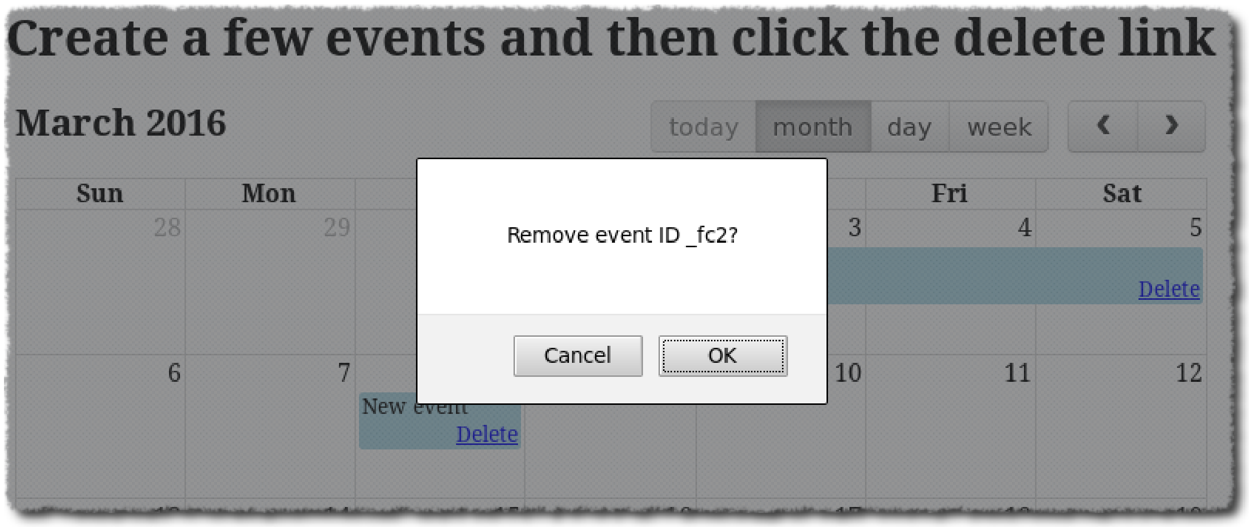
Removing events

The challenge of removing events is mostly how to make it easier for the user. There are many ways to expose this functionality. It is a matter of creativity.

In this example we will show a “Delete” link inside the event. When clicked, a confirm dialog will appear and the code will delete the event if the user answers “OK”. It sounds simple enough. Unfortunately, there will be some unexpected behavior. But there will also be a fix.

Some screenshots:





Remove event ID _fc2?

Thanks for reading. This is the end of the sample. I hope this fraction of the book has already been useful to you. After buying the book all updates to the content will be free.

Other topics I will cover are: using JQuery dialogs to manage events, using multiple event sources, recurrent events, persisting events in a database and controlling a calendar with external widgets like datepickers, checkboxes and dropdown lists. Get the book[a].

[a]https://usingfullcalendar.wordpress.com/using-fullcalendar-book/